mDIS User Documentation

mDIS - mobile Drilling Information System

Start reading →

The software

mDIS is a data entry system for geological samples and drilling engineering data.

Contact Information

Documentation: Knut Behrends, k.behrends@icdp-online.org

What is mDIS?

Brief introduction to mDIS for the complete newbie (1 minute read)

mDIS online

A few **mDIS instances** are available online for testing (sign-in required)

Full Documentation: https://data.icdp-online.org/mdis-docs/guide/

Private repository:

`git clone https://gitlab.informationsgesellschaft.com/dis/dis-docs/` - (sign-in required)

Ask for access if you want to contribute to this documentation.

mDIS Source Code

Private repository:

`git clone` **https://gitlab.informationsgesellschaft.com/dis/dis** - (sign-in required)

The repository will remain private while mDIS development is ongoing. mDIS contains only open-source components though.

Documentation as PDF

This Documentation is available as a single PDF File. It is okay for offline reading: **v20200621.pdf** .

What is ICDP?

ICDP is the International Continental Scientific Drilling Program. See **www.icdp-online.org** and **Wikipedia**.

ICDP is the sponsor of the mDIS software development project and the copyright owner.

Table of Contents

> mDIS at a Glance
>
> 😃 🎉 **mDIS for Data Entry and Data Management:** - **Regular mDIS Usage**
>
> 😃 🎉 **mDIS for Administrators:** - mDIS Installation and Configuration
>
> 😃 🎉 **mDIS for Developers:** - mDIS Advanced Customization

- **Getting Started**
  - **Technical Requirements** (Operating Systems/Supported Browsers)
- 1. 😃 🎉 **mDIS for Data Entry and Data Management** - **Regular mDIS Usage**
  - **mDIS Basics**
  - **Data Entry Forms**
  - **The Filter Bar**
  - **Hierarchical Filters**
  - **Data Input**
  - **File Upload**
  - **Reports**
  - **Data Export**
  - **Backup**
- 2. 😃 🎉 **mDIS for Administrators** - mDIS Installation and Configuration
  - **SA Roles and Permissions**
  - **mDIS Installation with VirtualBox** (extra page)
  - **Native Installation on Linux** (extra page)
  - **mDIS User Management**
  - **Using the Templates Manager**
    - See also "Developers" pages: **Short Intro**, (**extra page**)
  - **Backup/Restore** (extra page)
  - **Console Commands, *npm***
  - **Updating Third-Party Code**
  - **Further System Administration Tasks** (extra page)
  - **PHP Code dependencies** (extra page)
- 3. 😃 🎉 **mDIS for Developers** - mDIS Advanced Customization
  - **Roles and Permissions**
  - **Advanced System Setup and Configuration**
  - **Templates Manager** (extra page)

    - **The Yii console runners**
    - **Code Generation Principles**
    - **mDIS MVC Architecture**
    - **REST API** (extra page)
    - **DisForm.vue** (extra page)
    - **Form Validation**
    - **Yii Behaviors**
    - **File Importers** (see **Importing data**)
    - **Reports**
    - **Tutorials and Screencasts** (TBC)
- 4. For developers: **ICDP Internal**
    - **Testsuites**
    - **Extra Content, Design Tweaks**
    - **Applications and Extensions**
    - **Tooling**
    - **Best Practices**
    - **References**
    - **Scratch Area**

This page is online here: **https://data.icdp-online.org/mdis-docs/guide/** (internet) and **http://wb31.gfz-potsdam.de/mdis-docs/guide/** (ICDP/GFZ-Sek42 Intranet).

What is mDIS

Brief introduction for the complete newcomer

mDIS - the mobile Drilling Information System is a data entry system for describing geological data.

Such data can be about rock samples, or about any data gathered during field work.

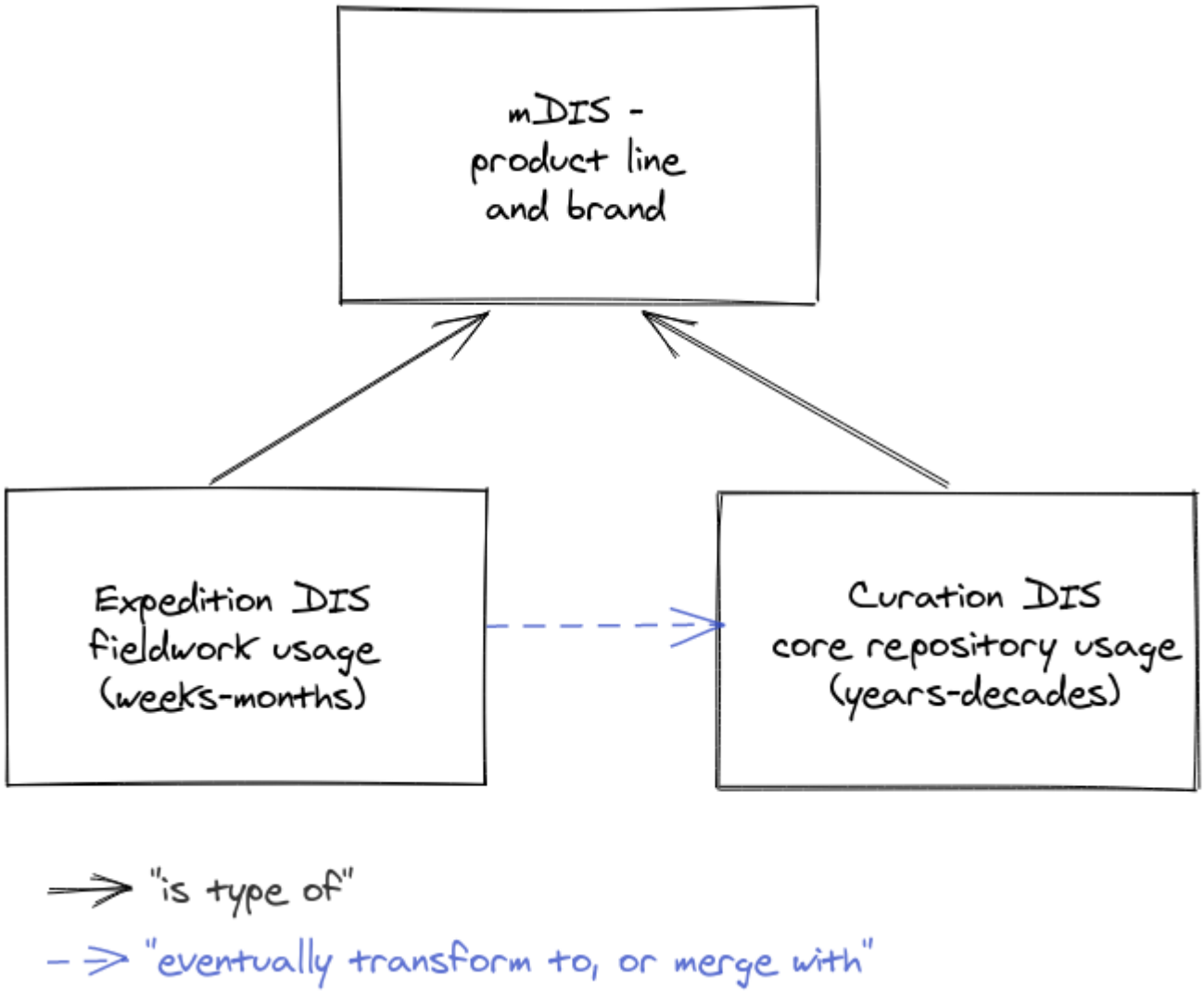
The mDIS software is designed for use by professionals, notably Earth Scientists and Drilling Engineers. They can use mDIS to manage datasets important to them.

Typically mDIS contains information about rock samples. However, at a larger scale, mDIS also contains information about the drilling process and wellholes drilled. At a smaller scale, mDIS contains datasets about core sections and tiny subsamples taken from the rocks, chosen for scientific study.

The mDIS software is available on laptops or PCs, where it runs as a VirtualBox virtual machine. The mDIS data-entry system is also available as a website, see **list**.

mDIS variants

mDIS is available in two variants: *Expedition DIS* and *Curation DIS*. Typically mDIS is distributed as an *Expedition DIS*; this variant is suitable for field work. Larger installations are called *Curation DIS*. They are used in larger research institutions and core repositories, which maintain long-term archives for geological samples.



More Information

If you are interested in more details, continue with the **"Getting started"** page.

Please see also the **TOC - Table of Contents** of this documentation.

Getting Started

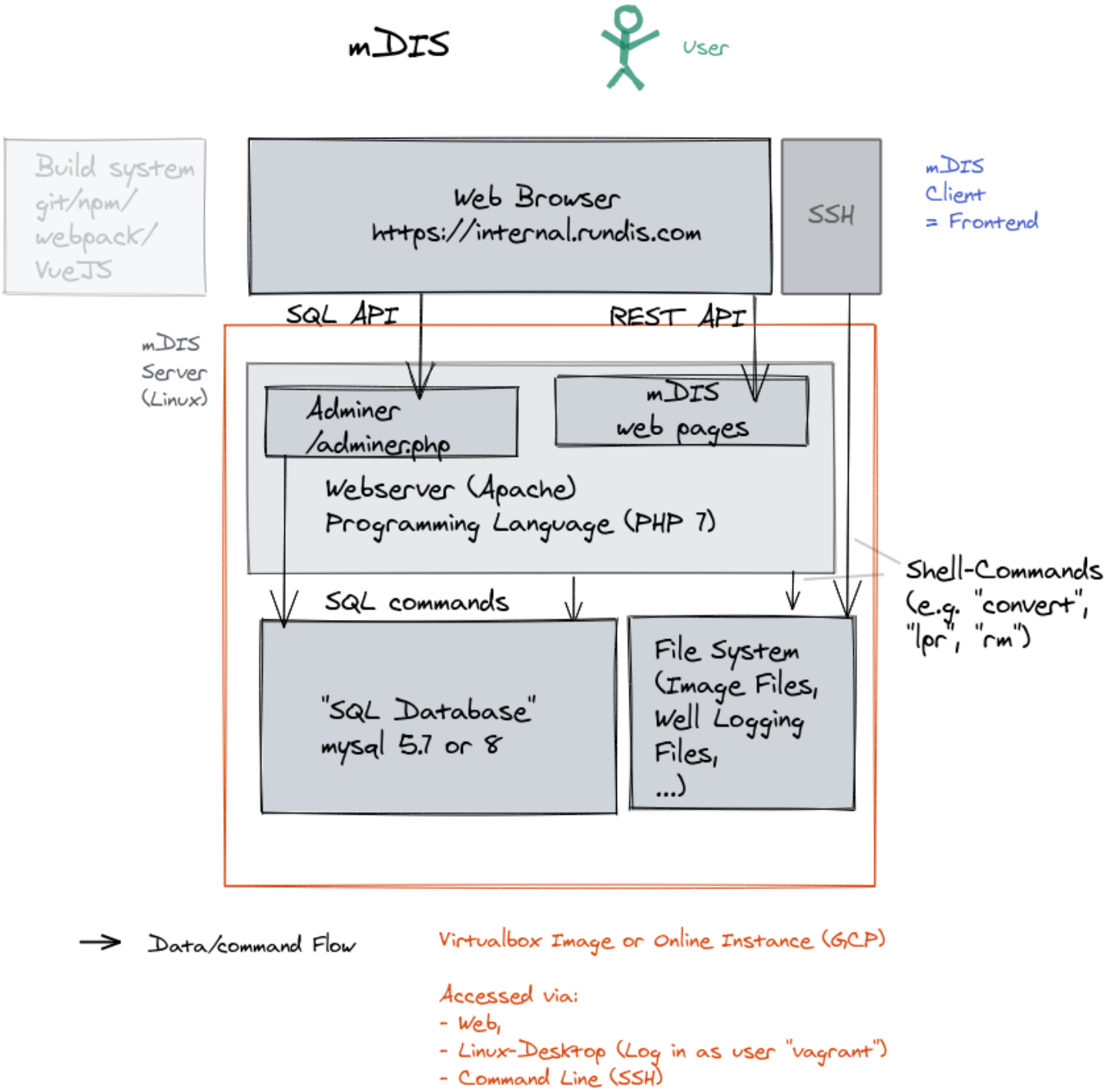
developer page

Technical requirements

> **Introduction**
>
> All topics mentioned on this page can all become relevant to any mDIS user, when first encountering the software. Most likely, the content on this page needs to be mentioned at least once, perhaps at the beginning of any prospective ICDP mDIS training course.
>
> Contents are related to the **"setup and configuration" Documentation** written for administrators.

mDIS Client / Web Browser Frontend

The general architecture is shown here. The large orange box is known as a **"LAMP web service stack"**.



The most important way to access MDIS is via the web. However other access mechanisms exist. The database can be accessed via an SQL API which is conveniently accessible via the *adminer.php* web page. Command line access via SSH is also possible for system administration purposes.

Logging In and Out

To use even the most basic features of mDIS, you need to log in to the mDIS web page.

At this time password management is done by administrators.

> **Password management**
>
> Password management (e.g. password reset, password reminder) *by users themselves* is currently not implemented and will be supported in a future release.

Installation and Configuration

For end users and data curators and editors, mDIS is a web application. This means, mDIS runs in a modern web browser.

The mDIS software is available on laptops or PCs, where it runs as a VirtualBox virtual machine. The mDIS data-entry system is also available as a website, see **list**. For technical details, see "**For Administrators**" and **Installation with VirtualBox** pages.

mDIS Supported Browsers

Browser support implied by user-interface components library **VuetifyJS**:

- Chrome: Supported
- Firefox: Supported
- Edge: Supported
- Safari 10+: Supported
- IE11 or older / Safari 9: Not supported
- For data entry, we support the English language version of the browsers. Form labels, status/error messages, and other system output will be displayed in English only. Of course data entry can occur in any language.
- Keyboard shortcuts specific to localized browsers, are not supported.
- The *display format* of input fields for date/time values (i.e. "Core on Deck") depends on the language of the browser. However, for data entry with Month names e.g. "Dec 21 2019", English month abbreviations are assumed (German "Dez 21 2019" will not work).
- Mobile devices are generally supported: You should preferably use Google Chrome, Firefox or a new version of Safari as a browser. On mobile devices, read-only access works reasonably well. *Entering* data on a small display like a smartphone can be quite awkward, but should technically work.

Note on Browser Extensions and Operating System Settings

- Browser must be configured/permitted to open New Tabs (Dependent tables, e.g. Core -> Sections) and Popup Windows ("Exported Records") - otherwise a warning appears "Firefox prevented this site from opening up a popup window".
- Internet Access (for mDIS users entering data): Not Needed.
  (This question was asked quite a few time during our demo setup at EGU 2019.)
  Internet is still nice to have, for accessing additional resources such as help files and external reference

materials.

For mDIS Admins and Developers, internet access is needed at least sometimes, e.g., for

- updates of third-party code (OS, PHP, JS/Node)
- uploading data gathered by mDIS to the ICDP website or external websites
- setup of https connectivity.

Runtime performance

How to run mDIS smoothly?

The mDIS *backend* can run well on a small laptop with a Intel Celeron CPU manufactured in 2010, with just 2 GB of RAM. This is really sufficient for simple interactive data entry into forms. However, if you work with larger files, such as high-resolution core-scans embedded in web pages and reports, mDIS will be much slower on such a small computer.

The mDIS *frontend* also runs small systems such as phones just fine, but it can be a bit slow depending on wireless connectivity and available bandwidth.

Performance also depends on the network speed. Cable-bound LAN is faster than Wifi which is usually faster than commercial telecom networks.

For administrators: changes of the PHP code in the backend will be effective immediately. However the development task `npm run build` (for rebuilding the mDIS *frontend* to achieve substantial changes in the input forms) takes about 10 minutes on such a machine, so if you plan to do development and customization, please use a machine quite a bit more powerful than that.

mDIS Server Components / Backend

- see **"For SAs" Documentation** for technical details.

Virtualbox Virtual Machines

In 2020 most mDIS installations were **Virtualbox**-based. The mDIS server component was distributed as a Virtualbox Guest. This setup is one of many. It is not mandatory. mDIS can also be set up on its own server, without Virtualbox.

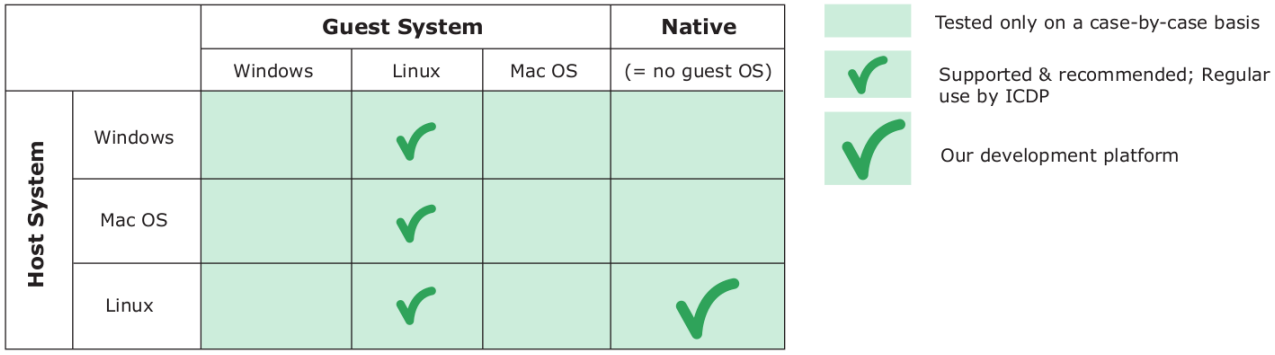
However, if your mDIS instance *is* VirtualBox-based:

Regarding the Guest Operating System, we assume that mDIS admins only use properly configured releases of fairly common operating systems. That means:

- Fairly recent releases, that means
  - no Windows 7. After January 14, 2020, Microsoft will no longer provide security updates or support for PCs running Windows 7.

    - no ancient MacOS (TBC)
    - no "exotic" Linux distributions (with very small market share)
- *Long-Term* releases of the OSes (no beta releases, no rolling releases, self-built Linux distributions or kernels)
- "Fresh" standard installations (no older, crufty Linux releases previously upgraded to newer versions)
- Installations designed for use inside a small subnetwork (no enterprise features such as LDAP- or Active Directory connectivity, security-optimized "hardened" Linux, no cryptofilesystems, no type-1 hypervisors such as KVM.) mDIS might work with these setups but we have not tested them.
- at least 8 GB of free disk space inside the guest VM. Reserve 20 GB for the entire Virtual Machine.

Regarding the VM hosts and VM guests, the supported operating systems are shown in the figure below.



A green background means all versions should work, and we have tried them at least once, briefly. But we cannot test all combinations thoroughly. The best supported combinations are shown with a checkmark.

For details, see **For SAs/Installation with VirtualBox**

Hardware Requirements

mDIS Client

Modern PC, Tablet, or Smartphone as implied by "Supported Browsers" above

- Display Size is important: must be large enough, high resolution. Small laptops are inconvenient to use for complex data entry tasks on large forms.
- For PC, mouse is required:
    - data input forms need mouse events in places, e.g. some modal dialogs can only be closed by mouse events (or touch events)
    - the Template-manager GUI which includes a Table Designer and a Form Designer works with drag-and-drop only.

TBC

mDIS Server / mDIS VirtualBox host

- mDIS needs resources typical for a small instance server, when no GUI is required.
- 2 GB of RAM should succeed.
- On initial install, user must mave Administrator/Superuser rights on the host. On Windows the Windows Firewall (also called "Windows Defender Firewall") will perhaps complain; on Linux the VirtualBox installer will install 4 Kernel modules.
- On Windows Hosts, the Microsoft **Hypervisor** Product called "Hyper-V" must not be enabled. To disable **Hyper-V**, go to "Control Panel"/"Programs and Features"/Left Sidebar/"Turn Windows Features On and Off". A long selection list of optional Windows Features will appear. Check off "Hyper-V". Check the list for more entries containing "Virtualization". Turn them off as well. Reboot the Windows Host. Disabling Hyper-V probably means you cannot run the lightweight virtualization product **Docker** *and* VirtualBox on the same Windows machine at the same time.

Network Pre-requisites

Client

- mDIS User needs to be familiar with Browser
- mDIS Operator, mDIS Admin need some basic familiarity with internet (FTP, Client-Server) and relational database concepts (for tableset design)

Server

Information Technology skills required, by user group

- **mDIS User:** - awareness that mDIS server exists on a local network (accessible via some IP address from a private address range). Users optionally get an SSID and password for Wifi and get the URL to enter in their browser. They need to know that the address entered is in most cases not reachable from the public internet. Instead it is a "local address" or an "intranet address", whatever is more familiar to the user.
- **mDIS Operator, mDIS Admin**: familiar with IPv4, Level 2 internet (Ethernet, WLAN, client/server architecture concepts), basic OS knowledge, familiar with LAMP/MAMP administration (that can be a lot of stuff!). For a permanent mDIS server-installation I expect from the administrator even more, like backup strategies, monitoring, etc. For an mDIS installation on a project laptop I expect the admin to be able to handle the Virtualbox software: create or restore snapshots, export OVA files, shutdown and start a VM, etc. but nothing more.
- **mDIS Developers**: need to know an absurdly rich set of technologies for both client and server. (Thats why they are developers.)

(**Developer page**)

Customizing models and templates with Yii Behaviors

There exists a **shorter introduction** to this topic in the developer documentation that is perhaps easier to read than the text below.

Yii Behaviors for models

Yii Behaviors are related to **PHP traits** and **"mixins"**, and **similar constructs** known from other programming languages.

In mDIS we use Yii Behaviours to change the capabilities of models / linked tables. There are two main situations where Behaviors are used in mDIS model classes:

- We want to use one particular feature in a lot of different but related model classes.
- We want to provide a lot of optional features for a model class.

> **Yii Behaviors**
>
> **For Databases**
>
> With regard to databases, Yii behaviors are similar to **database triggers**. But behaviors allow to implement functionality that database triggers cannot offer.
>
> Just like database triggers, behaviors can be bound to many different events, e.g. the BEFORE_INSERT-event, and the AFTER_INSERT-, BEFORE_UPDATE-, AFTER_UPDATE-, BEFORE_VALIDATE-events. However the events are processed by the **Yii ActiveRecord Object**, not by the database.
>
> **Not only for databases**
>
> Actually, ***every class*** that inherits from the `yii\Component` class has the `behaviors()` method. Class `yii\base\Component` inherits from `yii\BaseObject`, and thus is very high (on level 2) in the **class hierarchy of the Yii framework**.
>
> Thus, you can *attach* some new behaviour not only to ActiveRecords, but also to Controllers, Modules or Components. And you can *create* that new behavior by inheriting from the `yii\base\Behavior` class.

In mDIS, there are a few Yii-Behaviors already available to use. They required custom PHP programming. Examples see below.

For the implementation of mDIS Yii Behaviours see the `*.php` -Files in directoy `backend/behaviors` .

mDIS Custom Behaviors

There are 2 important types of behaviors:

- *"AttributeBehaviours" Behaviors* - also known as "Automatic" in our Documentation: see **below**. They are parameterizable.
- *User-Assigned (Parameterizable) Behaviors*, for example see below, **ChildrenLimitBehavior - Complete code example**).

mDIS makes use of Behaviors to create...

- very specific combined keys (`CombinedIdBehavior` - Automatic)
- IGSNs (`IgsnBehavior` - Automatic)
- autoincremented values that are customizable (`UniqueCombinationAutoIncrementBehavior` - User-defined)
- numeric constraints on some groupings inside a data table (`ChildrenLimitBehavior` - User-defined)
- serialized JSON strings that can be converted back to Javascript Objects (`JsonFieldBehaviour` - User-defined)
- numeric values that are carried over from a previous record to the next (bottom-depth n becomes top-depth of record n+1)

Besides, mDIS makes use of Behaviors to...

- store JSON Arrays as Strings in database columns, and converting back, serializing multivalued entries. This happens in the Dashboard widgets.
- manage Access Control for users in the `cg` Module. This is an internal task.

Automatic Behaviors

For Automatic Behaviors, (Attribute Behaviours) you don't have to do anything except assigning a column name, and little else. The system will attach a behavior to a form or to an item, when you give a column that certain name.

- If you name a column `combined_id` , mDIS will attach a `CombinedIdBehavior` to your input form.
- If you name a column `igsn` , mDIS will attach an `IgsnBehavior` to your input form. However, remember that you must set the correct default value for the IGSN type in the *table designer* of the mDIS Template Manager. Set "C" as default value for Core IGSN, for example. - Similar functionality exists for columns that are named `igsn_ukbgs` , see class `IgsnUkBgsBehavior` .

- `AnalystBehavior` : If you name a column `analyst` , the name of the logged-in user will be inserted into this field after saving the record.

Actually, the correct name for "Automatic" behaviors is `AttributeBehavior` . The custom `UniqueCombinationAutoIncrementBehavior` is also an `AttributeBehavior` but it needs multi-column (=multi-attribute) input.

CombinedIdBehavior

CombinedIdBehavior creates a human-readable ID hierarchy like it was used before, in the Legacy DIS. Strings such as "5063_1_A_3_1" are easier to work with for humans than tuples such as `[5063, 1, "A", 3, 1]` . Every time a record is inserted or updated, the value of the assigned column (`combined_id`) is being updated. This behavior does not have to be assigned manually, it is invoked automatically as soon as there exists a column named `combined_id` in the mysql datatable.

The `combined_id` model definition should have these characteristics when assigned in the table designer of the Templates Manager:

```json
// JSON fragment
"combined_id": {
            "name": "combined_id",
            "importSource": "",
            "type": "string",
            "size": 16,  // use appropriate length
            "required": false,
            "primaryKey": false,
            "autoInc": false,
            "label": "Combined Id",
            "description": "Convenience column to print on labels etc. Filled out automaticall
            "validator": "",
            "validatorMessage": "",
            "unit": "",
            "selectListName": "",
            "calculate": "",
            "defaultValue": ""
        }
```

IgsnBehavior

IgsnBehavior assigns a unique **IGSN** value to a newly inserted record. `IgsnBehavior` uses the algorithm described in Wiki Article **"How to generate igsns (int geo sample nr) in legacy dis"**. Depending on the table type (Core, Section, ...), a different letter (= "Object Tag" in IGSN jargon) has to be used. The different Object Tags are described in an older **mDIS Wiki Article**, or in the PHP source code of the behavior (File `backend/behaviors/IgsnBehavior.php`).

The behavior requires an object tag (one character) in a parameter `objectTag`. This behavior does not have to be assigned manually: If a table has a column `igsn` (type `string`, size 15) and the `defaultValue` attribute for that column is a single letter (H, C, S, X, B, W, U, T, Y, Z, or F), then the behavior will be automatically applied, and this letter will be used as an appropriate object tag by the IGSN algorithm. You should set the default value in the *table designer* of the mDIS Templates Manager.

Table column `igsn` should have the following characteristics when assigned in the table designer of the Templates Manager:

```json
//JSON fragment from backend/dis_templates/models/CoreCore.json
"igsn": {
    "name": "igsn",
    "importSource": "IGSN",
    "type": "string",
    "size": 15,
    "required": false,
    "primaryKey": false,
    "autoInc": false,
    "label": "IGSN",
    "description": "IGSN",
    "validator": "",
    "validatorMessage": "",
    "unit": "",
    "selectListName": "",
    "calculate": "",
    "defaultValue": "C" // C for core. Can be H, C, S, X, B, W, U, T, Y, Z, or F
},
```

Form Field `igsn` could be designed like this:

```json
//JSON fragment from backend/dis_templates/forms/core.json
{
  "name": "igsn",
  "label": "IGSN",
  "description": "IGSN (autom. calculated)",
  "validators": [
    {
      "type": "string",
      "min": null,
      "max": 15
    }
  ],
  "formInput": {
    "type": "text",
    "disabled": false,
```

```
    "calculate": "",
    "jsCalculate": ""
  },
  "group": "Subgroup",
  "order": 3
}
```

Actual values of some fields (such as Label descriptions) are a matter of the form designer's personal taste and preferences.

Parameterizable Behaviors

UniqueCombinationAutoIncrementBehavior

Creates versatile autoincremented values that can be customized (e.g. they can have gaps, can be nonunique, can be given a start value > 1).

UniqueCombinationAutoIncrementBehavior fills a "fake" auto-increment column. It is usually populated with a unique value, but nonunique values may sometimes occur. It might also be necessary that an autoincrement column starts at an unusual value, say 40, because the project requires it. A normal mySQL autoincrement column does not (easily) support such customizations.
Thus, a `UniqueCombinationAutoIncrementBehavior` column is not a "real", production-quality auto-increment column. It is usually used to create unique values for records with the same parent record, like the column `section` in table `core_section`. You have to provide the parameter `fieldToFill`, which specifies the behavior of the column to be filled by the calculated value. The second parameter `searchFields` is an array of the columns in the record, that build the group in which a unique value should be created.
Here is the example of that behavior used for the table `core_section` (from Model file `backend/models/CoreSection.php`):

```php
    [
        'class' => app\behaviors\UniqueCombinationAutoIncrementBehavior::class,
        'searchFields' => ['core_id'],
        'fieldToFill' => 'section'
     ],
```

When a new record is inserted, the behavior looks up the maximum value of all records with the same value in the column `core_id` as the new record, increments that value and writes it into the column `'section'`.

> **Note**
>
> For details you have to look at the source code in file `backend/behaviors/UniqueCombinationAutoIncrementBehavior.php`.

ChildrenLimitBehavior

ChildrenLimitBehavior generally constrains a column value based on the number of records in the same table, or in a different table. Use this when you know in advance: *"The maximum number of child items allowed for this record is n"*. In this respect, `ChildrenLimitBehavior` is a multi-row constraint, similar to a database assertion.

More concretely, we can use `ChildrenLimitBehavior` to set a max-threshold for the section count of a single core. e.g. *"The number of sections in this corebox is 3"*.

This code snippet shows how to use `ChildrenLimitBehavior` to set such a max-threshold on the count of core sections for a core, depending on the value of column `last_section` in the `core` table.

```php
    [
        'class' => app\behaviors\ChildrenLimitBehavior::class,
        // 'parentRefColumn' and 'limit' are the two _parameters_ of this behavior:
        'parentRefColumn' => 'core_id',
        'limit' => function ($model) {
            return $model->core->last_section;
        }
    ]
```

At data entry time, mDIS looks up the value specified in the column `last_section` in the (parent) core. When a new record is inserted, and more sections exist than the value of `last section` (and with the same value of column `core_id`) specifies, then that new record will be rejected by mDIS, and the record cannot be stored.

- The required parameter `parentRefColumn` defines which column is used to group by, and then to determine the current rowcount of that group.
- The required parameter `limit` is a PHP function that returns the maximum count-value for the records. The count-value is known in advance.

JsonFieldBehavior

JsonFieldBehavior automatically converts the specified attributes from JSON string to array and back.

This behavior is needed (... perhaps to store serialized data for some Widgets on the Dashboard?)

TBC

```php
    *
    * To use JsonFieldBehavior, insert the following code to your ActiveRecord class:
    *
    * ```php
    * public function behaviors()
```

```
 * {
 *     return [
 *         [
 *             'class'  => \app\behaviors\JsonFieldBehavior::class,
 *             'fields' => ['foo_data', 'bar_data'],
 *         ],
 *     ];
```

More

There are even more behaviors in `backend/behaviors/template`:

- `DefaultFromParentBehavior`
- `DefaultFromSiblingBehavior`
- `SiblingsLimitBehavior`
- `TemplateManagerBehaviorInterface`

TBC

Tutorial: Using a behavior

This tutorial is designed to teach beginner PHP programmers how to customize PHP files generated by the Template Manager. This is needed when the Templates-Manager GUI for attaching behaviors to tables/models is not sufficient or not flexible enough.

Using `ChildrenLimitBehavior`

You can easily attach a behavior to the model of a table:

- A PHP file for customizing the desired behavior is needed. Fortunately, a good starting point, a basic PHP file with a only very few lines of code, was already generated by the Templates-Manager.
- Locate that PHP file in the directory `backend/models/`. For example, for the data table `core_section` that PHP file name is `backend/models/CoreSection.php`.
  Do *not* modify the files in the subdirectory `backend/models/base`; these files are generated from the template manager and could be overridden.
- Open the file with a text editor.
- First, it is indispensable that we add this line to the top of the file `backend/models/CoreSection.php`:

```php
use app\behaviors\ChildrenLimitBehavior;
```

This imports the behaviour, or just "switches on" the ability to `use` it.

- Locate the following snippet inside the file. If there is *no* such `public function behaviors(){}` in that file, simply add it to the file. Inject the code on the second-to-last line of the file, before the closing `}` on the last line. (The `}` should remain occupying its own line, because it closes the `class { ... }` definition block stated at the top of the file). If there is *no* function `behaviors()`, inject this code snippet:

```php
public function behaviors()
{
    return array_merge(parent::behaviors(), [
      // Enter the behavior(s) here
    ]);
}
```

If there *is* already a function `behaviors()`, add a new item to the end of the `[]` array in the `return array_merge([...])` statement:

- Each behavior is a block written in square brackets. If there are more behaviors in the method, they are separated by commas. The lines in the square brackets are also separated by commas.
- The first line defines the class of the behavior, i.e.

```php
  [
    // This single line is not enough,
    'class' => app\behaviors\ChildrenLimitBehavior::class,
    // ...since ChildrenLimitBehavior requires parameters, see below
  ],
```

- Depending on the behavior, you can or must provide parameters. These parameters are written in additional lines after the class. Each line consists of of the name of the parameter and its value. See the **example below**, how to add the parameters.
- Save the changes and try it out: Insert a new record into the table the `ChildrenLimitBehavior` has been assigned to. You can add as many items in the dependent table as specified in the record, *but not more*.

ChildrenLimitBehavior - Complete code example

The complete function/method might look like this:

```php
// in file: backend/models/CoreSection.php
// add this line at the _top_ of the file, above the line `use Yii;`
use app\behaviors\ChildrenLimitBehavior;

// add this at the _bottom_ of the file before the closing curly bracket of the class.
public function behaviors()
{
```

```
        return array_merge(parent::behaviors(), [
            [
                'class' => ChildrenLimitBehavior::class,
                'parentRefColumn' => 'core_id',
                'limit' => function ($model) {
                    return $model->core->last_section;
                }
            ]
        ]);
    }
```

Here `ChildrenLimitBehavior` was assigned to the `CoreSection` class.

The behavior is *defined* in a different file. Our function `behaviors()` inside the file `backend/models/CoreSection.php` just registers it with mDIS tables/forms, in a *parameterized* manner.

Advanced

Summary and some Hints

- As mentioned above, there are Automatic Behaviors / Attribute Behaviours (`IgsnBehavior` , `CombinedIdBehavior`) and User-Assigned Behaviors (`ChildrenLimitBehavior` , `UniqueCombinationAutoIncrementBehavior` , `JsonFieldBehavior` ,...).
- If you want to register a *completely new type* of Custom Behavior (Automatic or User-defined) with mDIS, you must create a new class in directory `backend/behaviors` , e.g. `backend/behaviors/WaitForCorescannerBehavior.php` (fictitious example).
- If it is a new Automatic behaviour you have invented, you might consider inserting an appropriate block of code in file `backend/models/base/Base.php` , method `behaviors()` . The additional code should be similar to what is already there.
- Behaviors are a server-side feature. On the client-side you can also implement your own "behaviors" to calculate properties on the fly, perform checks, etc. See VueJS / **DisFormTutorial**. For a bigger picture you should learn more about VueJS. Its feature similar to Yii Behaviors is called "Instance Lifecycle Hooks". See the diagram in the **corresponding Vue Documentation**.
- As yet, there is no *central* place in the Templates Manager or in the file system to manage behaviors.
- In the mDIS Templates Manager, there is a GUI tool to parameterize column-wise behaviours, but its capabilities are limited. Editing and postprocessing source code is required.
- Behaviors can also be used to govern the *caching* behavior of controllers, and other Yii Objects. **This topic** is not discussed here, because it is not used in mDIS.

Template Behaviors

New feature

There exists a GUI to customize and parameterize behaviors.

We implemented this in Winter 2019/2020, documentation and examples coming soon.

TBC

(**Developer page**)

Form Validation

Calculated Fields

The target audience for this article is mDIS developers.

This article explains how the "Calculate" feature in the forms-manager works. **It happens both on the server- and on the client-side.** "Calculate" allows to create "calculated" or "derived fields" in an mDIS data entry form using short Javascript expressions. These "derived fields" are not stored in the database. That would be redundant, and also be a potential source of errors and inconsistencies.

> **Reminder**
>
> See "For **Operators/Viewers**" what Calculated Fields are.
>
> **Advanced**
>
> *Server Side/Backend:* See **Yii behaviors**. *Client Side/Frontend:* For *complex* multiline calculations/transformations see **Dis-Form** tutorial. For an alternative mechanism to calculate "derived fields", see also the **Vue Slots part** of that article.

Implementation

This is not a tutorial. Do not change any files mentioned in this article.

The transformation of the formula entered in "Calculate" to JavaScript is performed by

- the PHP method `getJsCalculate()` in File `backend/components/templates/FormTemplateField.php`
- Javascript in Vue Files:
  - `src/components/DisSmartForm.vue`
  - `src/components/DisForm.vue`

Backend

File `backend/components/templates/FormTemplateField.php` :

```php
/**
 * Converts the content of the "calculate" field
```

```
     * into a string appropriate for Javascript
     * @return string
        */
    public function getJsCalculate () {
        if ($this->formInput['calculate'] > "") {
            $calc = $this->formInput['calculate'];
            $calc = trim($calc, "\t\n\r\0\x0B=");
            // replace fields name with form model properties
            $calc = preg_replace('/\[(\S*)\]/m', "this.formModel['$1']", $calc);
            // replace absolute function
            $calc = preg_replace('/ABS/m', "Math.abs", $calc);
            return $calc;
        }
        return "";
    }
```

Note the `trim()` call. Besides removing preceding whitespace and trailing whitespace, it also removes any preceding `=` characters from the user input.

Frontend

For each form input field marked as "calculated", file `src/components/DisSmartForm.vue` takes the string returned from PHP method `getJsCalculate()` and evaluates it as Javascript code, using `eval()` (!):

```
js
calculatedFields () {
      let calculated = {}
      this.formTemplate.fields.filter(item => item.formInput.jsCalculate !== '')
        .map(item => {
          calculated[item.name] = function () {
            return eval(item.formInput.jsCalculate)
          }
        })
      return calculated
    }
```

File `src/components/DisForm.vue` assigns it to the Vue field watcher:

```
this.$watch('formModel', {
      deep: true,
      handler: () => {
        for (const calculatedField in this.calculatedFields) {
          try {
            let cb = this.calculatedFields[calculatedField].bind(this)
            let v = cb()
            if (v) {
```

```
            console.log('set ' + v)
            this.formModel[calculatedField] = v
          }
        } catch (e) {
          console.log('calc err', e)
        }
      }
    }
  }
```

TBC

- For complex expressions that are needed in many forms, you should define a Yii behavior and/or a JS module, put it at the correct place in the filesystem, let Vue import it, and then use it like this... TBC- perhaps as independent article/tutorial.

(**Developer page**)

> **Advanced**
>
> The content of the screenshots below is *not* part of mDIS. It is shown here to demonstrate the "default" look and feel of the Gii code-generator which mDIS uses internally. The mDIS Version of Gii is heavily customized and does not look like this.

mDIS Developer Documentation

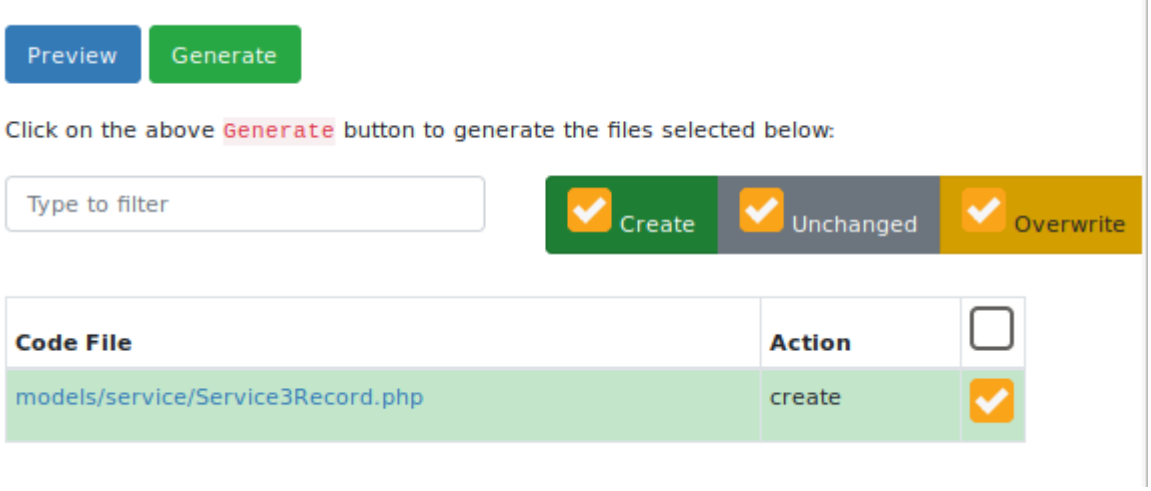
A deep dive into Gii

The technology behind the mDIS Templates Manager

> **Gii is the Yii code-generation extension**
>
> Internally, the mDIS Templates-Manager uses **Gii** quite a bit, therefore it is useful, for comparison, to show here the "native" GUI of Gii that the Yii developers have designed for code generation.
>
> Obviously screenshots cannot show everything Gii can do. For "normal" usage of GII, mDIS developers should set up a new basic Yii project and enable the `/gii` route on their development workstations to see what is displayed below.
>
> A basic Yii project template can be created with command `composer create-project --prefer-dist --stability dev yiisoft/yii2-app-basic basic` .

Preview

Without further ado, you can see that the Preview feature of the Gii GUI is somewhat similar to the mDIS Templates-Manager Preview:



Note the similarities with the mDIS Templates Manager. The look of the "Preview" and "Generate" buttons should look familiar. So should the workflow of with the "Create/Overwrite" alternatives.

What's the magic behind the code generation process?

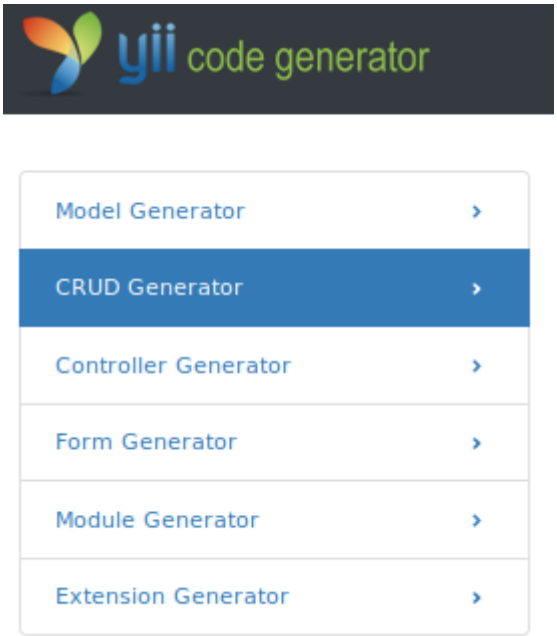
Background on Gii

Gii has a graphical user interface that is shown here in pieces, to reduce complexity.

By default the sidebar looks like this:

Sidebar of Gii

As you can see from the sidebar, there are 5 categories of code generators available.



> **Key Message**
>
> Internally, the mDIS Templates-Manager uses 2 generator classes:
>
> - the Base Generator (in file `backend/modules/cg/generators/DISForm/Generator.php`)
> - the Model Generator (in file `backend/modules/cg/generators/DISModel/Generator.php`)

> Here we only show what these features look like, not how they can be used. That would be out of the scope of this article.

Base Generator

PHP class `Generator` in file `backend/modules/cg/generators/DISForm/Generator.php` inherits from `\yii\gii\Generator`. This is the base class for all generator classes.

A generator instance is responsible for taking user inputs, validating them, and using them to generate the corresponding code based on a set of code template files. A generator class typically needs to implement the following methods:

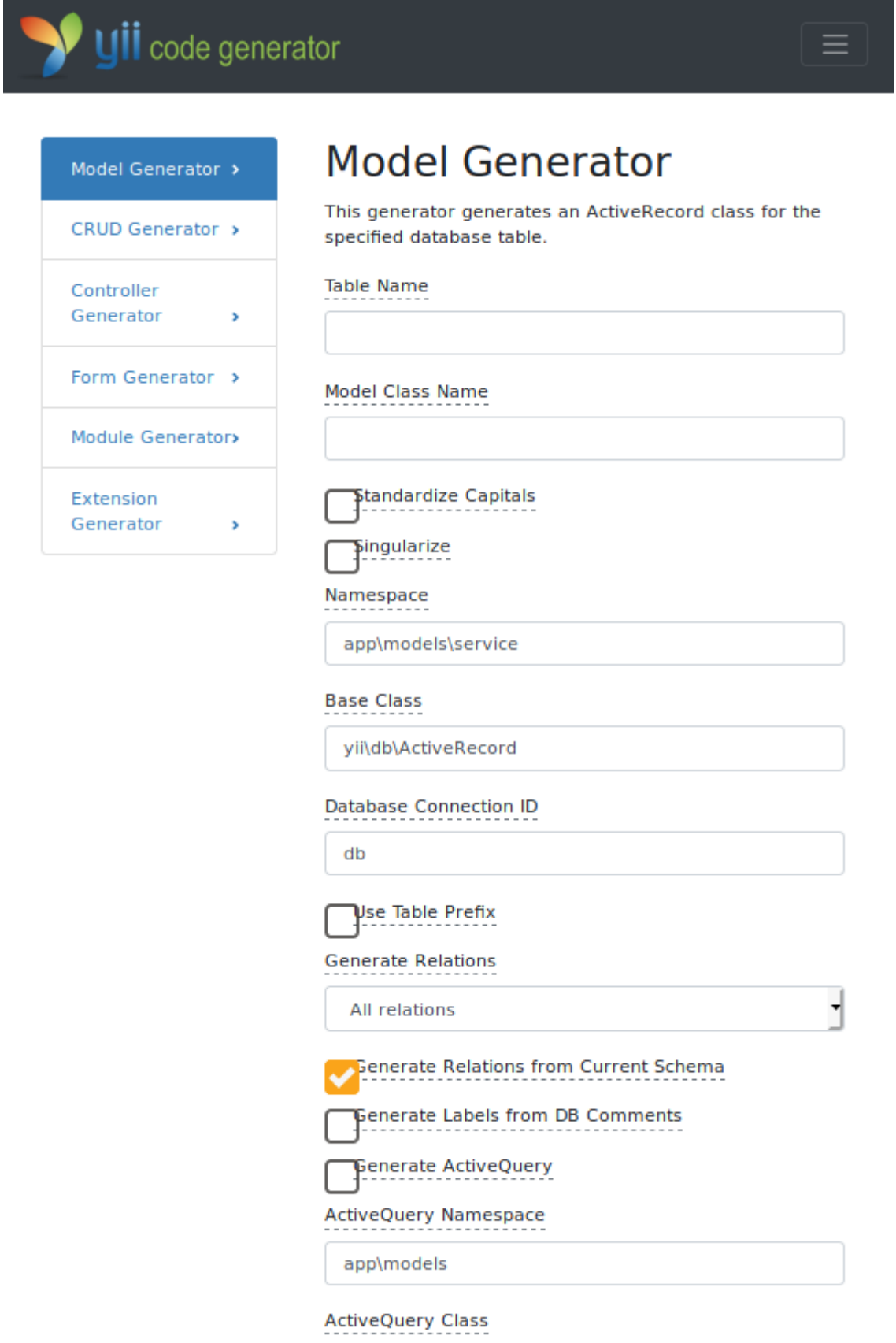
- `getName()`: returns the name of the generator
- `getDescription()`: returns the detailed description of the generator
- `generate()`: generates the code based on the current user input and the specified code template files. This is the place where main code generation code resides.

Model Generator GUI

The default Graphical User Interface of the Model Code-Generator has a lot of helpful tooltips (dashed lines).

This generator generates an ActiveRecord class for the specified database table.

[TBC]



The PHP class in file in file `backend/modules/cg/generators/DISModel/Generator.php` inherits from `\yii\gii\generators\model\Generator` .

> **Illustration purposes only**
>
> **MDIS does not use the default Gii CRUD generator and Gii Form generators** shown below, because they generate code based on the Twitter Bootstrap UI framework plus some Yii-specific JavaScript code. MDIS does not use this framework for the front end. MDIS uses VueJS/Vuetify

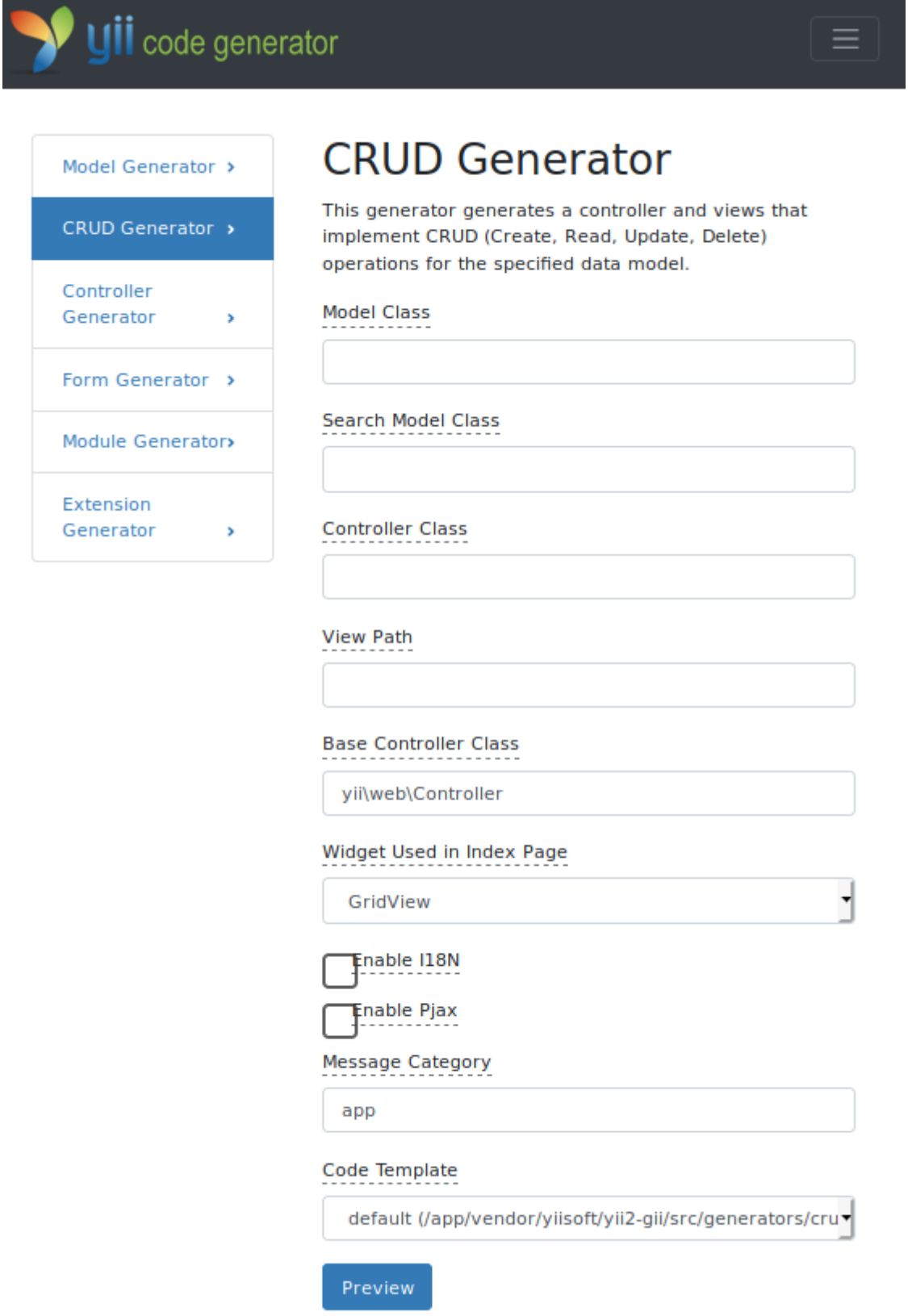
instead. We still include this here, because what the MDIS code generator accomplishes is *conceptually* similar.

CRUD Generator GUI

The default Graphical User Interface of the CRUD Code-Generator. *(not used by MDIS)*

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

[TBC]

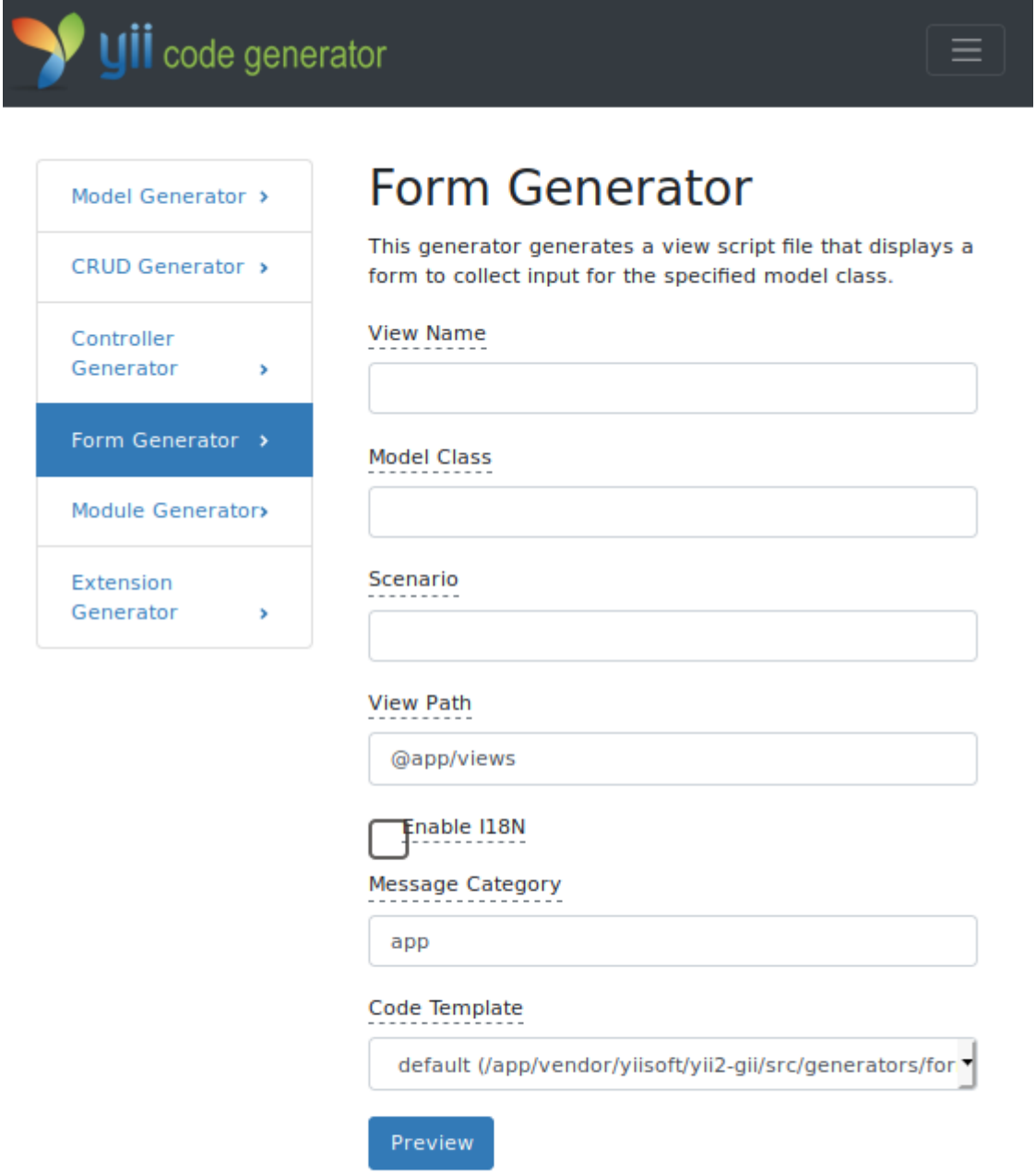


Form Generator GUI

The default Graphical User Interface of the Form Code-Generator. *(not used by MDIS)*

This generator generates a view script file that displays a form to collect input for the specified model class.

[TBC]



The code generation process in Yii is implemented according to the MVC composite pattern.

cg on the mDIS yii console

There is a `yii cg` console runner task. It seems to be unfinished work, though

See also the output of `./yii` which will show this:

```php
The following commands are available:

...
- cg                                This is the command line version of Gii - a code gen
    cg/dis-form                     DIS Form Generator
    cg/index (default)
```

At this time (2019) this results:

```php

./yii cg dis-form
Error: No help for unknown command "gii"

```

mDIS cg API

TBC

(Back to **developer page**)

mDIS For Developers (ICDP internal)

continued from **mDIS For Developers**

ICDP internal

This section is somewhat specific to ICDP internal mDIS-installations. However you might still find some pearls of wisdom here on this page.

Testsuites

- For testing the REST API of the PHP backend, there exists an ICDP-internal **"testsuite"**, available as a collection for the **Postman** test runner. (**Older version** `.json` **file**). It contains many tests of API calls provided by the mDIS PHP controllers. It automates the following actions: create items, insert, update, delete, get, login. This collection is available to all ICDP team members as a Postman Shared Workspace (just ask us for it).
- For testing the web pages and Graphical User Interface, the Yii developers recommend the end-to-end testing tool ***Codeception***. It is installed in the `backend/vendor` directory, because it is a third-party code dev-dependency of Yii2. Although it is available, Codeception is not currently being used for testing mDIS.
- There are a few unit tests for Javascript code, currently for Datetime entry only. Run testsuites on the command line with
  - `npm run test:unit` or
  - `npm run test:unit -- --watchAll`

Debugging

Remote-Debugging PHP with XDebug

Remote-Debugging means going through running code on a step-by-step basis. This is documented in **PHP Debugging**.

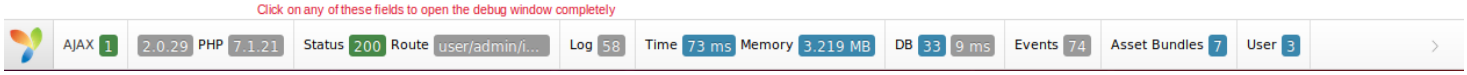
The Yii2 Debug Bar

Getting Diagnostic information

The Yii Debug-Bar is usually invisible. If it is enabled, it will become visible on the mDIS Users-Manager page.

It looks like this, and users must click to expand it from a minimized, collapsed state:

Expanded it looks like this:



Click on any field to expand the debug bar to an even larger debug-*window*. (Not shown here).

Then you can search for many things, e,g, stack traces, errors texts, and even SQL statements sent by Yii to the MySQL database.

To enable the debug bar, open file `backend/config/web.php`. Change this line of code

```
'bootstrap' => ['log', 'api', 'cg'],
```

to this:

```
'bootstrap' => ['log', 'api', 'cg', 'debug'],
```

This line of code is located at the top of the file, (in line 11 or so).

Remove older `sess_*` files from the directory specified in PHP's `session.save_path`, e.g. `/var/lib/php/sessions`. Set the following variables in file `(project_root)/.env` :

```
YII_DEBUG=true
YII_ENV=dev
```

Do not enable this feature in production.

Debugging VueJS

See **this hint** from our "**VueJS DevTools**" article.

Getting into an mDIS Docker Container

wb45 only: Enter the running Docker container (the server-side, the backend) with:

```sh
docker-volume up -d                # start all containers
docker exec -it disapp_php_1 bash  # run bash inside webserver container
```

Logfiles

Viewing mDIS webserver- and PHP logfiles on Linux (inside a Virtualbox instance or inside a Docker container):

```sh
cd /var/www/dis/backend/runtime/logs
tail app.log
tail convert.log
#grc tail convert.log # grc: logfile colorizer
```

Accessible from outside Docker container, too, when `/var/www/dis` is mounted as a Docker volume.

Docs

Offline reading

Offline reading of this documentation is very easy by downloading the **PDF version**. This contains everything as a single PDF file, but the content of the PDF might be slightly older than these web pages you are just reading.

Advanced: git-clone the **Gitlab repository**, and read `*.md` files opened from your local directory. These are text files written in Markdown. The `.md` files are human readable.

Optional, for developers: To generate html files locally on your computer, install **Vuepress** with `npm i -g vuepress`, and inside the repository's directory, run `vuepress build`. Then perhaps run one of the following commands to open the documentation as a web page in your browser:

- Python: `python -m SimpleHTTPServer 8000; firefox localhost:8000`
- PHP: (built-in server)
- Javascript/Node: TBC

Searching this Documentation

See the text field with the looking-glass icon [🔍], located in the title bar at the top of this webpage.

Searching with Vuepress

There is a search engine that comes built-in with Vuepress, but it is not very powerful. It only searches headlines and keywords that appear in the sidebar.

We have replaced vuepress search with an index provided by the commercial indexing service *Algolia*. They have a free tier that we are using.

Searching with Algolia.com

Algolia-based search supports fulltext search, similarity search, and it has a nice user pull-down interface.

The documentation site must be indexed with their scraper and pushed to algolia.com.

The indexing script:

```sh
#!/bin/sh
cd /home/knb/code/git/dis-docs/_work/dis-search/
# on wb45, the actual command to run the algolia indexer.
# details are in the .env file and  prod_mdis.json file
docker run -it --env-file=.env -e "CONFIG=$(cat /home/knb/code/git/dis-docs/_work/dis-search/c
cd -

```

Config file `/home/knb/code/git/dis-docs/_work/dis-search/config/prod_mdis.json` for the Algolia scraper + indexer:

```json
{
  "index_name": "prod_mdis",
  "start_urls": ["https://data.icdp-online.org/mdis-docs/guide/"],
  "selectors": {
    "lvl0": "#app main.page h2",
    "lvl1": "#app main.page h2",
    "lvl2": "#app main.page h3",
    "lvl3": "#app main.page h4",
    "lvl4": "#app main.page table",
    "text": "#app main.page p,#app main.page ul,#app main.page ol, #app main.page td"
  }
}
```

> **TIP**
>
> If the search query yields a 404 error, remove the duplicated section from the URL search result page. Often there is a duplicated path-fragment in there. **Change the URL fragment from** `mdis-docs/mdis-docs/` **to** `mdis-docs/` and reload the 404 page.
>
> Run the indexing script again. It must be run from within directory `dis-docs/_work/dis-search/` to create correct URLs.

The `.env` file mentioned in the script contains an Algolia APP-ID and and API key to give access as a free-tier customer.

Tutorials and Screencasts

- Illustrate regular, trouble-free usage of advanced topics

    - The Frontend Build Process: System Startup
    - Hot module replacement in a development environment
- (see **this page** for more basic Video tutorials)

Extra Content, Design Tweaks

How to add static content

- **How to add static HTML content to the mDIS dashboard with Vue**
- The Vuejs Router software currently permits that you can put any static *.html files with extra content into the `web/` directory and call them directly in your browser. This is demonstrated in the **Javascript tutorial** REST API access.
- You can add HTML fragments to forms via the Vue Slot mechanism. See **Vue Slot Tutorial**, then run `npm run build` .

How to change the mDIS page design

For mDIS reports, the CSS code can be changed easily. See directory `web/css/` and change files there.

For a single Vue page or Vue component, edit the `<style>` element in a .vue file, and run `npm run build` .

For *all* pages (or components), read the **Vuetify theming guide** first. It contains a lot of alternatives to change the design.

If you are adventurous, **read this** next. Open directory `backend/node_modules/vuetify/src/stylus/components/` and change the layout by editing Vuetify's Stylesheets ending in `.styl` . (These are ***Stylus*** Stylesheets, not CSS Stylesheets). Don't do this unless it is absolutely necessary. Test your modifications and then run `npm run build` .

Alternatively, try to edit the minified css files in `web/css` directly. Good luck with that.

Tooling

A new set of **"Tooling"** pages is under construction. **PHP Debugging**

Applications and Extensions

Desktop Apps, Browser extensions, Commandline tools...

in no particular order

- Recommendations concerning VS Code editor: Install some extensions: Vetur, Bracket Pair Colorizer, PHP CodeSniffer, Markdown Preview Enhanced... See this script on mDIS-installer: **vscode-install-**

extensions.sh

- As Mysql administration tool, we recommend: **Adminer**, (**Installation**),. It can do 99% of basic DBA tasks. For more advanced tasks, such as renaming foreign keys, see **phpMyAdmin** or **Mysql Workbench**.
- The VueJS devtools Browser Extension - for visualising the component structure on the mDIS data entry pages. See **HowTo** document.
- Postman: There exists an ICDP-internal "integration testsuite" for REST API calls [, available as a **Postman** collection. (**Older version** `.json` **file**). It can automate create, inserts, updates, deletes, gets, logins. It also contains many tests and a test runner. This collection is available to all ICDP team members.
- NPM: `npm run serve` builds mDIS in a "development mode" which supports Hot Module Reloading, and other developer productivity features. Global NPM Modules needed: `@vue/cli` , `serve` . See package.json file in Gitlab repo.

Best practices, TBC

- If you go on a field trip, take offline documentation with you
- Study mDIS thoroughly, because adapting it in the field requires self-confidence and deep understanding

Contact Information

ICDP - Address is important to 'real' non-ICDP users

For now, see **https://gitlab.informationsgesellschaft.com/dis/dis/-/project_members**

TBC: establish a *real* feedback channel for end-user communication

References

Books and Wikis

- **The Definitive Guide to Yii 2.0** Authors: Many, 2014-today. (**"Living Document"**).
- **Web Application Development with Yii2 and PHP**. Authors: Mark Safronov and Jeffrey Winestett. Packt Publishers, 2014.
- **Pro Git** Authors: Scott Chacon and Ben Straub, Apress, 2014.

Websites

- Frontend: **Vue.js**, **Vuetify**, **JavaScript**

- Frontend Buildsystem: **NPM**, **Webpack**, **Vuepress** (Documentation generator)
- Backend: PHP-Stuff: **Yii2**, **PHP**, **Composer**, **Packagist.org**, **Adminer**, a free Database Management and Query Tool in a single PHP file.
- Backend: Shell Scripting: **BashFAQ Wiki**, **Linux Doc Proj.**,
- Virtualization: **VirtualBox**, **Vagrant**, Base Images used for mDIS: **ubuntu 18.04**, **ubuntu 20.04**, **on Vagrant**

Misc:

- **List of Emojis** available for embedding into Markdown text (Github)
- **Material Icons** - Material icons are symbols for common actions and items.
- **Twitter Bootstrap 3** - Governs look-and-feel for features that are rendered server-side, e.g. the mDIS User Administration page, some Single-Item Reports and Labels
- "Awesome" Link Collections (plentiful but not always up-to-date): **VueJS** Javascript Framework, **Vuetify** component library, **Vuepress** documentation system, **PHP** programming language, **Composer** PHP package manager, **NPM** Node Package Manager, **Vagrant**, **Docker**

Scratch Area

Questions from 2019

> **Older Questions for mDIS training Dec 2019**
>
> Not ordered or ranked
>
> - Q: Templates Manager: "Add Behaviors" feature of the Generated Code is in Base*.php files. Could be overridden? (= Should it be copied/moved into derived classes?)
> - Q: Templates Manager: How to add "Online help" to Templates-manager GUI? (At least links to documentation)
> - Q: Maintenance: What is safe to delete in `web/` directory?
> - Q: Yii Asset compression: Is it useful to us? We do not use it, no plan to do so.
> - Q: How to adapt the Vuetify JS theme. Is it a good idea at all to attempt to change the theme?
> - Q: how to learn and make better use of "Charts.js" JS library?
> - Q: Vue 3: what's new? (no more work on vue2?) . See various Talks and discussions on the internet, e.g. **"Vue 3 Beta"** on Hacker News
> - ~~@vue/cli v3 -> v4 major upgrade possible?~~ yes, I simply did it on my dev workstation

Feature-requests

Not prioritized or ordered

- feature: Export to "GFZ FTP Server" feature
- ~~feature: Export to CoreWall (create exchange file, XML-based interchange format)~~
- feature: Export to PSICAT (create exchange file, XML-based interchange format)
- ~~Barcode scanner integration TBC. At this time only an ICDP feature.~~
- feature: Special-Purpose File-Importers, Complex Device integration TBC Corescanners, Cameras etc ICDP only
- ~~Feature: Delete many rows in batches. "Select All" checkbox as convenience feature for deleting records. At this time, Adminer has to be used, and a "DELETE FROM WHERE" SQL Query~~
- Feature: Support password management by users themselves
- Feature: HTTPS-, Progressive web app (PWA) support- (Is it really important for mobile users?)
- Maintenance: finetune caching settings: see backend/config/web.php. We use `\yii\caching\FileCache` for code generation, for database caching, keys `enableSchemaCache`, `enableQueryCache` should be set to true in a production environment. Browser-Side caching is another can of worms.
- Feature: DB Schema design
  - *renaming* existing forms and models (with low-level DB tools, and IDE) is hard
- using the CSV Importer correctly, customizing it, using it together with Adminer's Importer
- using version control (git) efficiently
  - rapid growth of git-Repo with .php files generated by mDIS Templates-manager
  - Q: what to add to file `.gitignore`? (provide a recommendation or a prepopulated `.gitignore` template), what to add to .gitattributes
  - coupling with other Gitlab features: for future mDIS installations, keep a feedback channel to Main development site open;
  - settings and notable changes in file `.git/config`; removal of `.git` folder (implications)
- Docs: use Continuous-Integration features of gitlab to trigger a rebuild and reindex, publish rendered HTML to webserver

Missing documentation

TBC Need to document this

- Templates Manager: Documentation of `DefaultFromParentBehavior`, `DefaultFromSiblingBehavior`, `SiblingsLimitBehavior`, `TemplateManagerBehaviorInterface`
- Yii routing:
  - general knowledge is important for API calls
- Dashboard: (its configuration and the widgets) - documentation, tips+tricks
- DB-based/Model-based validation vs Form validation: even more fine points
- Browser validation: Input Types (built-In to the Browsers)
- Browser Validation vs Javascript Validation
- Javascript validation: principles and implementation
  - how Vue does it (required fields)
  - (validation of custom listvalues / selection boxes)
  - (autocompleters)
- See also older **wiki page from 2018**
- Data File Upload: Filesystem I/O, Event Processing
- document DB Schema design : Primary Keys, Foreign keys, Relations, Pseudo-Autoincrement cols
- Development: `yii cg` task, APICalls
- Templates manager: Naming convention: when to use Kebab-case, CamelCase, snake_case etc
- Development: Anatomy of a Webpage rendering (walkthrough all APIcalls necessary to render an input form, Step-By-Step, perhaps with the remote debugger)
- Development: Browser Local storage: What it is, how it is used by mDIS, when is cache busting (refresh) necessary
- Documentation See sections marked "TBC" in **viewer-operator**
- Document caching: see backend/config/web.php. Server-side caching, Browser-Side caching is another can of worms.

Testing TO DO

Ideas for Unit Tests and Integration Tests

- use more *assertions* in some of the production code
- Also write some *end-to-end tests* with Codeception
- Document how they can be run
  - to get a quick-look on the current state of the app/stack, (assert that mysql is up, php/apache up, can login)
  - to check important features of the app (assert that data tables are populated)
  - write some *load tests*: many connections, handle large imports, ...
  - write GUI tests for GUI based workflows

ICDP Configuration TO DO

WARNING

Change management

Development best practices for working i n a distributed team

- Periodic Recordings for monitoring, baselining (Nagios?)
- Extra Log for Important configuration changes
- ~~Virtualbox configurations: Reference Implementation (write a Vagrantfile)~~
- ~~specify Vagrant benefits (setting up, etc)~~

Hardware/Device configurations

- same items as Virtualbox configurations (see above)
- but also
  - Database of devices (collect Invoices, Warranty infos...)

Development Environments

- same items as Virtualbox configurations (see above)
- but also
  - specific hacks and setups (coding style - avoid messy diffs due to IDE autoformatting whitespace)
  - .editorconfig file
  - .env files
  - .gitignore files
- git conventions (technical and social) .
- Fringe Ideas
  - Collect ideas about potential premium/open source features of mDIS

- Study Vuetify in depth
- Follow "Material Design guidelines" - needed to understand Vuetify - **Iconset Library** and **how to use it** . We don't use the Guidelines yet but it would be cool if we did.

.

(**Developer page**) (**Templates-Manager page**)

The `backend/models` directory

The following paragraphs explain how an mDIS model object is actually implemented: via *several* PHP classes that are represented as files predominantly located in directory `backend/models` .

PHP Class hierarchy

The Class hierarchy in a nutshell explained using the "Core" table as example.

The class is named `CoreCore` because table "Core" is part of the "Core" tableset. The mySQL table would be named `core_core` .

```php
# mDIS PHP Class Hierarchy
# Model Classes (and 1 Form Class)
  yii\db\ActiveRecord                  # large parent class, provided by Yii
   app\models\base\Base                # "Handwritten" Base Class
    app\models\base\BaseCoreCore       # implements linked table 'core'
     app\models\CoreCore               # add own behaviors here (optional)
       app\forms\CoresForm\            # has only 1 method: scenarios()

# Search Classes
     app\models\base\BaseCoreCoreSearch # inherits from BaseCoreCore (!)
       app\models\CoreCoreSearch        # empty class, no methods on purpose
         app\forms\CoreCoreSearch       # empty class, no methods on purpose
```

In the code block above, the indentation means "extends" or "inherits".

At the top, there is a large base class `yii\db\ActiveRecord` , with lots of methods. It is **provided by Yii.** . See also **Yii Classes** below.

Beginning on line 3, following `ActiveRecord` , are mDIS Model classes, provided by code-generation and by customization.

Class `app\models\base\Base` is hand-coded, the others are code-generated. They can be customized, but this is optional.

Actually, for consistency reasons `app\models\base\Base` should be called `app\models\base\BaseRecord` (just as mDIS Importers and Reports), but the class hierarchy is already pretty deep and names would get unreasonably long, so we prefer the shorter class names.

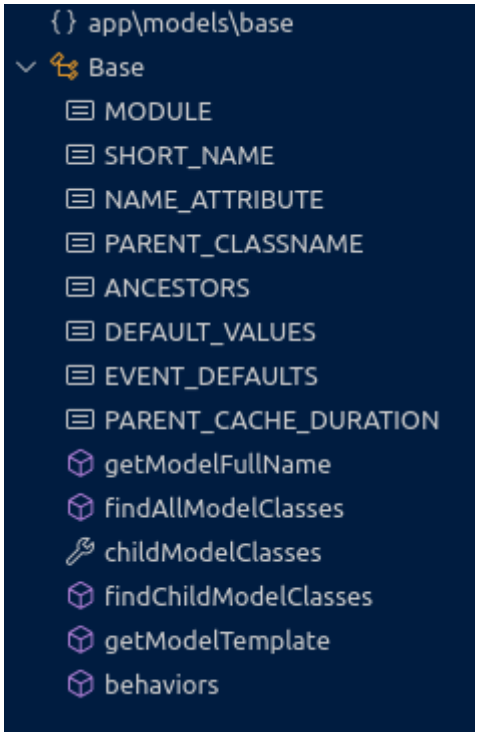
mDISModel classes

> **Background**
>
> An mDIS data model object represents one record in the corresponding data table. Every access to the record in the mDIS data tables is done via the model object. This way, it can be assured that validators are checked, behaviors are executed, etc.

These classes are hand-coded, as well as automatically created by the code-generator of the mDIS **Templates-Manager**.

Class `Base`

***Abstract* base class for all mDIS data models.** Hand-coded.



The `behaviors()` method should be modified if a new AttributeBehavior is added system-wide.

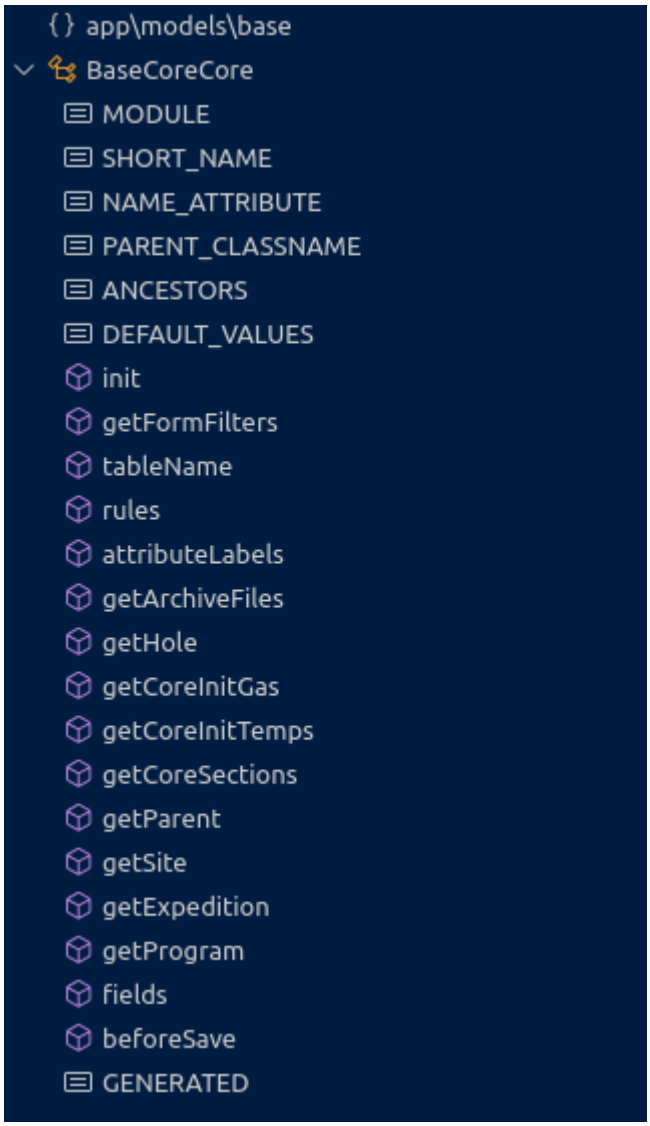
Class `BaseCoreCore`

For every MySQL data table containing science data, a base class (e.g. `base/BaseCoreCore.php`) is generated by the Templates-Manager. This file should not be modified, since it could be overridden by the Templates-Manager.

It implements:

- textfield default values (in the `DEFAULT_VALUES` constant array and in the `init()` method)
- textfield datatypes and validators (in the `rules()` method)
- textfield labels, column-labels (in the `attributeLabels()` method),
- calculations (in lowermost method `beforeSave()`).

- lots of `getXXX()` methods that return `\yii\db\ActiveQuery` objects TBC



The `rules()` method might have some additional value, `'safe'`. This means this attribute is not constrained by some validation rules and *still* marked as 'safe'. We can assign anything to this field. If was *not* marked as 'safe', it would be ignored by the `setAttributes()` method of the `yii\base\Model` class. That feature allows us to pass the entire content of $_POST to `setAttributes()` and be sure that only expected values will be assigned to the model. (See also "**'Unsafe' attributes and Scenarios**" below)

The `GENERATED` constant contains a Unix timestamp.

Class `CoreCore`

Additionally, the Template-Manager has created a mostly empty class (e.g. `CoreCore.php`) that extends the `BaseCoreCore` class above. If you want to modify the data model manually, you should do it here:

You can **add custom behaviours** in the `behaviors()` method for instance.

The `fields()` method modifies the capabilities of the Filter Bar.

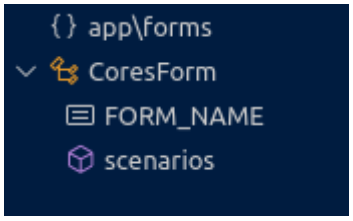
You can add your own `rules()` method here. (Take a look at `ImageOfTheDay.php`, `ListValues.php`, `ListValuesSearch.php`, `MessageOfTheDay.php`, `Post.php`, `Widgets.php` for inspiration.)

Class `CoresForm`

This class is not a Model Class but for completeness we list it here, because this class inherits from `CoreCore`. Class `CoresForm` has only 1 method: `scenarios()`, which returns a list of scenarios and the corresponding active attributes.

The `CoresForm::scenarios()` method overrides a base-class method 6 levels up, the `yii\db\Model::scenarios()` method.

> **Scenarios**
>
> What are Scenarios?
>
> Form Scenarios: Create, edit or view.
>
> ??? (A validation scenario represents a search-, sort- or filter-action requested by the user where the mDIS model objects must validate sets of input parameters or criteria). TBC
>
> Test case Scenarios



It belongs to a different namespace, `app\forms` so the full name is `app\forms\CoresForm\`. The file is located in directory `backend/forms` (not `backend/models`).

> **Unsafe attributes and Scenarios**
>
> **From the PHPdoc of the `yii\db\Model::scenarios()` class:**
>
> By default, an active attribute is considered "safe" and can be "massively assigned". If an attribute should NOT be massively assigned (thus considered unsafe), please prefix the attribute with an exclamation character (e.g. `'!rank'`). (In 2019, we do not use this construct in mDIS PHP code).
>
> The default implementation of this method will return all scenarios found in the `rules()` declaration. A special scenario named `SCENARIO_DEFAULT` will contain *all* attributes found in the `rules()`. Each scenario will be associated with the attributes that are being validated by the validation rules that apply to the scenario.

The returned array should be in the following format:

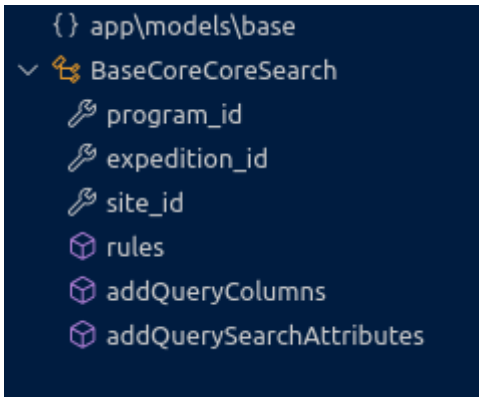
```php
[
   'scenario1' => ['attribute11', 'attribute12', ...],
   'scenario2' => ['attribute21', 'attribute22', ...],
   ...
]
```

Search Classes

These are the PHP classes implementing the **Hierarchical Filter Bar** and the **"Filter by Values"** search feature of the mDIS data entry forms.

Together with the data model, two search classes have been automatically generated, (1) `BaseCoreCoreSearch.php)` with some boilerplate code, and (2) `CoreCoreSearch.php` which is mostly empty (on purpose). These classes are used to find records based on query values entered by the mDIS user.

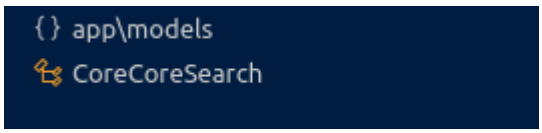
Class `BaseCoreCoreSearch`



The `BaseCoreCoreSearch` class contains a line of generated code for each table attribute. The other code that is needed is highly redundant. Therefore it has been factored out from each `BaseXXXSearch` class to **helper class** `\app\models\SearchModelTrait` , see below.

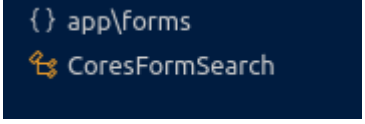
Class `CoreCoreSearch`

This class is empty on purpose. Modifications are almost never needed at this time (2019).



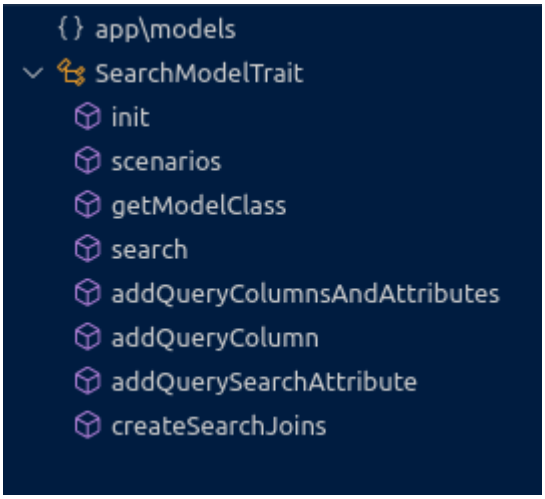
Class `CoresFormSearch`

This class is empty on purpose. Modifications are almost never needed at this time (2019).



Helper Classes

Class `SearchModelTrait`



The `BaseXXXSearch*` classes contain boilerplate code for each table attribute. The classes `use` a hand-coded trait-class `SearchModelTrait.php` that contains methods that would otherwise identical in every base search class.

If you do not like this code and you need to modify *how* records of a data model can be searched for, you should override some of the methods in the derived search model class (i.e. `CoreCoreSearch.php`), which is empty on purpose and can be customized.

(**Developer page**)

Yii Classes

For completeness, we show the Yii base classes from which mDIS class `app\models\base\Base` inherits.

Note that our `app\model\Base` class (in row 7) "stems from" a hierarchy of 5 base classes above it (beginning in row 2).

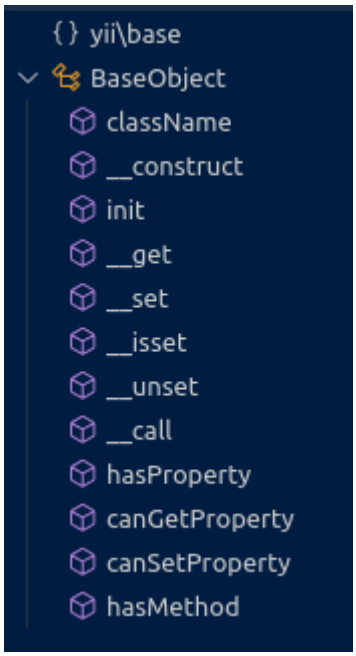
```php
# Yii Classes
yii\BaseObject
  yii\Component
    yii\db\Model
      yii\dbBaseActiveRecord          # implements ActiveRecordInterface
        yii\db\ActiveRecord           # parent class of mDIS 'Base.php' class
          app\models\base\Base        # mDIS class
```

These are the constants and methods that the Yii2 framework provides, from top to bottom:

`yii\base\BaseObject` class

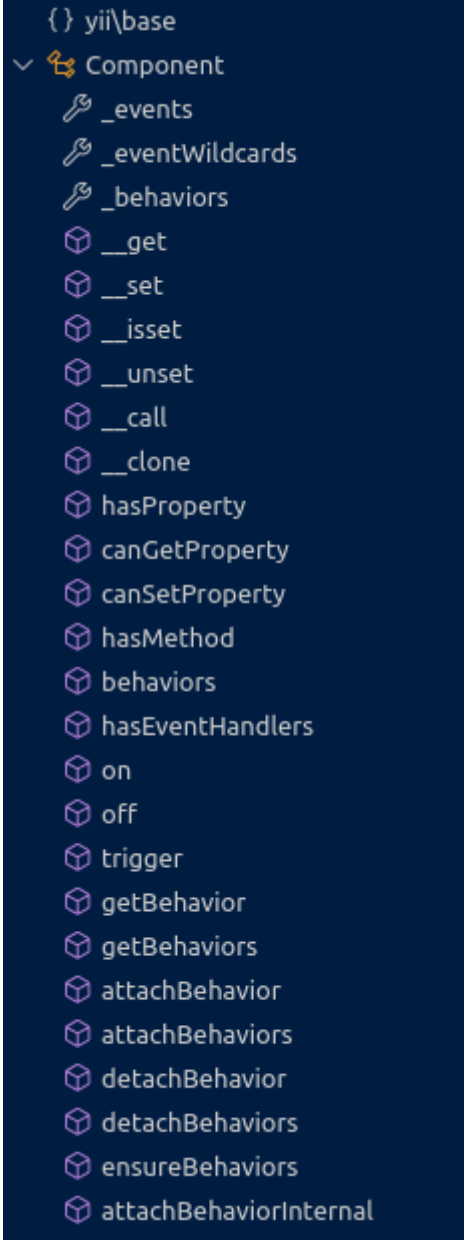
BaseObject is the base class that implements the *property* feature.

class `BaseObject` implements `Configurable`. Methods from this interface are not shown here separately.



`yii\base\Component` class

Component is the base class that implements the *property*, *event* and ***behavior*** features.



`yii\db\Model` class

`yii\db\Model` is the base class for data models.

`yii\db\Model` implements the following commonly used features:

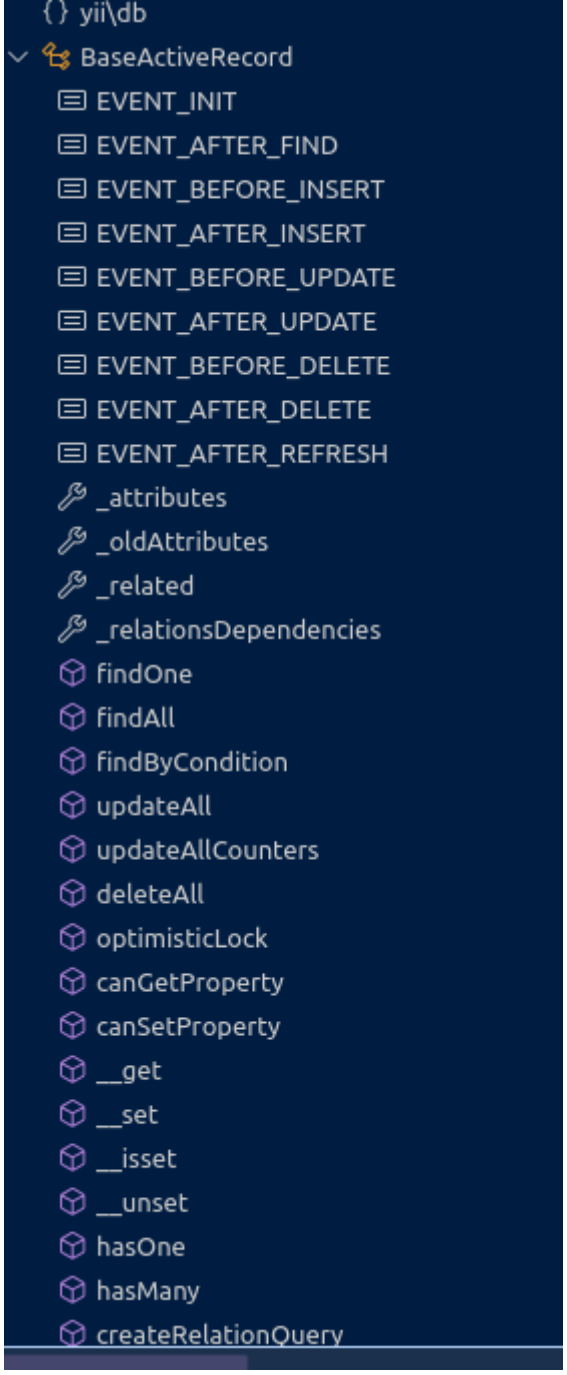
- attribute declaration: by default, every public class member is considered as a model attribute
- attribute labels: each attribute may be associated with a label for display purpose
- "massive" attribute assignment with `setAttributes($_POST)`
- scenario-based validation

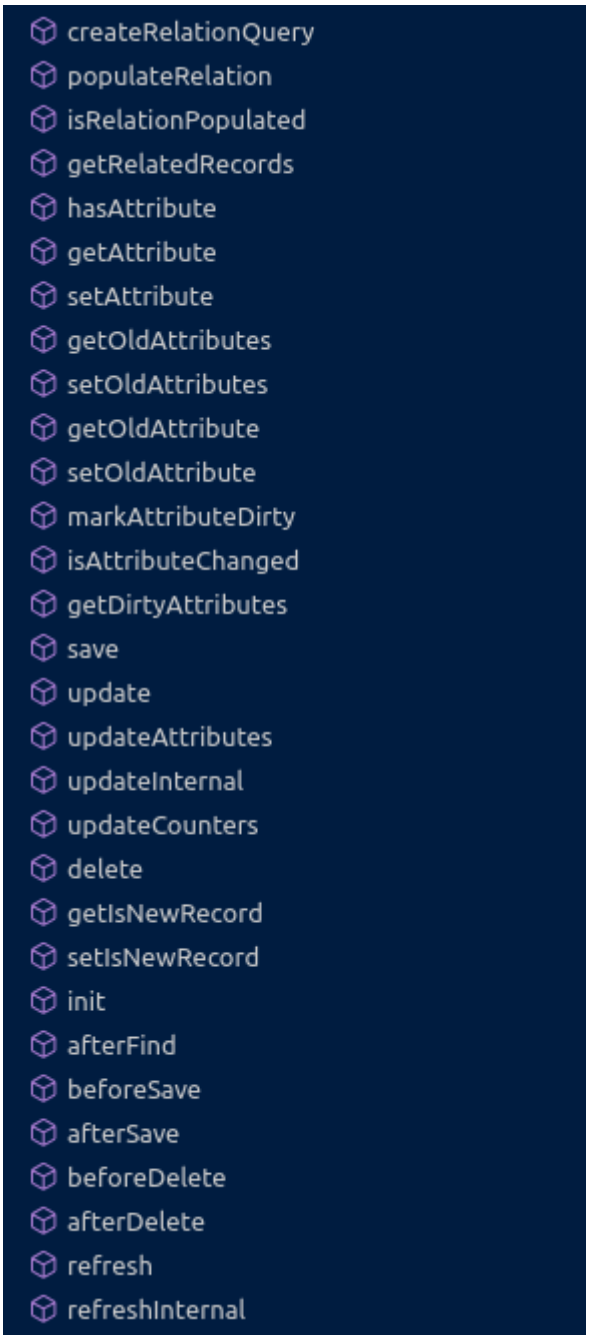
`class Model` extends `Component` implements `StaticInstanceInterface, IteratorAggregate, ArrayAccess, Arrayable`. The respective methods from these interfaces are not shown here separately.

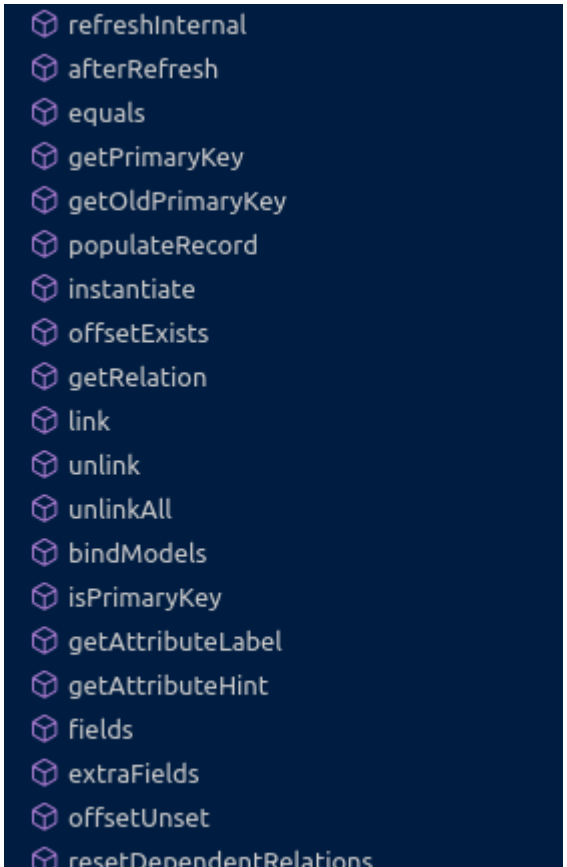
`yii\db\BaseActiveRecord` class

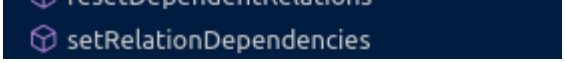
`db\BaseActiveRecord` is the base class for classes representing relational data in terms of objects.

See `\yii\db\ActiveRecord` for a concrete implementation. **Official Documentation**


{} yii\db
BaseActiveRecord
EVENT_INIT
EVENT_AFTER_FIND
EVENT_BEFORE_INSERT
EVENT_AFTER_INSERT
EVENT_BEFORE_UPDATE
EVENT_AFTER_UPDATE
EVENT_BEFORE_DELETE
EVENT_AFTER_DELETE
EVENT_AFTER_REFRESH
_attributes
_oldAttributes
_related
_relationsDependencies
findOne
findAll
findByCondition
updateAll
updateAllCounters
deleteAll
optimisticLock
canGetProperty
canSetProperty
__get
__set
__isset
__unset
hasOne
hasMany
createRelationQuery


createRelationQuery
populateRelation
isRelationPopulated
getRelatedRecords
hasAttribute
getAttribute
setAttribute
getOldAttributes
setOldAttributes
getOldAttribute
setOldAttribute
markAttributeDirty
isAttributeChanged
getDirtyAttributes
save
update
updateAttributes
updateInternal
updateCounters
delete
getIsNewRecord
setIsNewRecord
init
afterFind
beforeSave
afterSave
beforeDelete
afterDelete
refresh
refreshInternal


refreshInternal
afterRefresh
equals
getPrimaryKey
getOldPrimaryKey
populateRecord
instantiate
offsetExists
getRelation
link
unlink
unlinkAll
bindModels
isPrimaryKey
getAttributeLabel
getAttributeHint
fields
extraFields
offsetUnset
resetDependentRelations

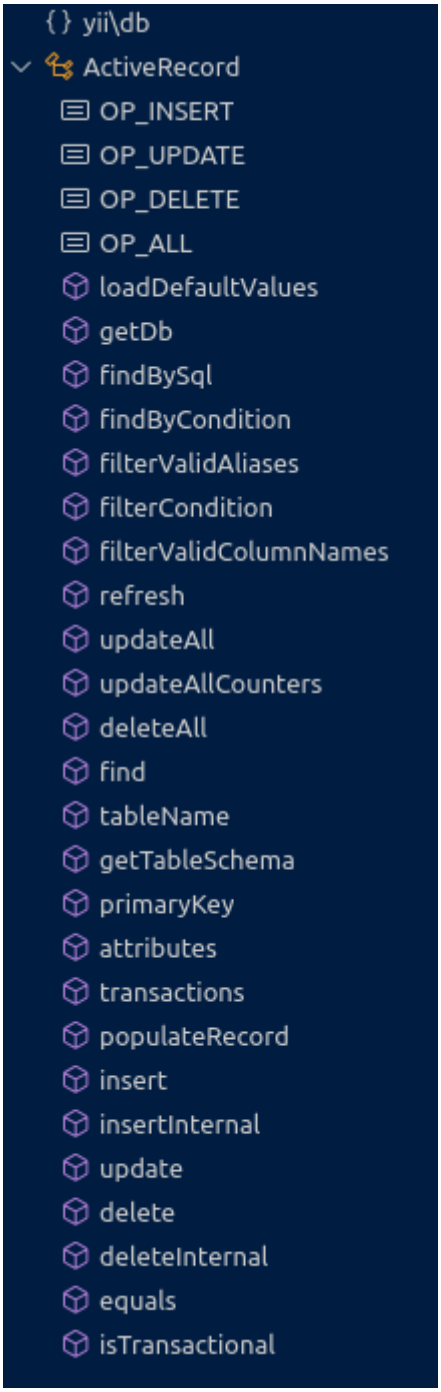


Implements the `save()` method which, if successful, magically sets a new `CoreCore->id` field value as a side-effect. (? is this important?)

`yii\db\ActiveRecord` class

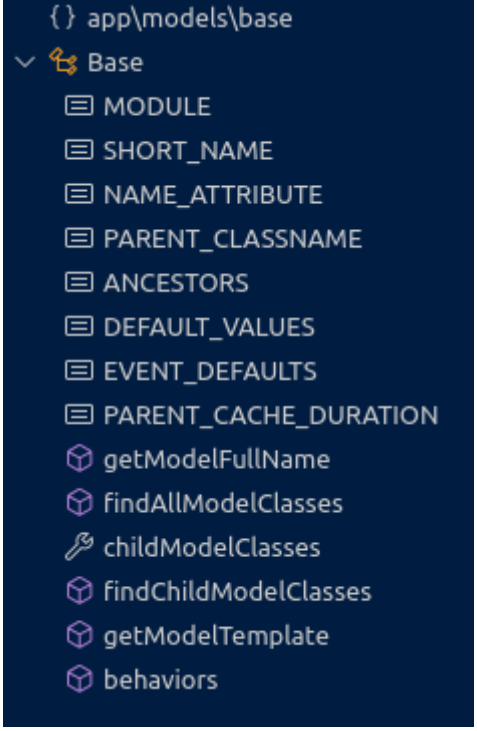
`ActiveRecord` is the concrete implementation of its abstract superclass `db\BaseActiveRecord`, and represents relational data in terms of objects.

Official Documentation



`app\models\base\Base` class

The first mDIS class, located at level 6 in the inheritance hierarchy. (See **above.**)



mDIS adds another 3 hierarchy levels, making it 9 or 10 levels in total. It depends on how you count them.

3. mDIS For Developers

Roles and Permissions

There are two aspects of working as an mDIS *developer*:

- *Software Developer*: Managing an mDIS system, making changes to the mDIS source code, as a software developer with write-access to mDIS sourcecode files and to the mDIS database backend. The user can change everything inside mDIS.
- *Predefined Role 'developer'*: Interacting with mDIS as a user having the predefined role `developer` according to the Role-Based-Access-Control (RBAC) User Management feature. The name of this role was chosen for backward compatibility with Legacy DIS. This role implies a user can design forms and customize an mDIS production system, however with *less* privileges than an mDIS Sysadmin (`sa` role). An mDIS `developer` *role*, as defined in the MySQL `your_mDISdb.auth_assignment` table, cannot create new users or change the permissions of other users. This role also cannot modify tables/models (e.g., add columns).

This section describes both aspects, but focuses mainly on the Software Developer role.

> **Note**
>
> Many tasks and responsibilities overlap with the reponsibilities of an **"mDIS sysadmin"**. A clear separation is not always possible.

Assigned roles

In the server backend, the equivalent to the mDIS Frontend Role `developer` is the operating system account `administrator` or `mdis` . Substitute your preferred user-account name here. Make this user a "sudoer", in other words enable this user to perform commands as the `root` user. MS Windows calls this "Runs as Administrator...".

We do not suggest altering the default permissions (or the `umask`) of the predefined unpriviledged account `www-data` . Instead the `mdis` user should be enabled to change file/directory permissions and file ownerships of files that belong to the "www-data" account.

This is necessary because the webserver process will perform code generation and will *need write access* to several distinct directories on the backend server.

Thus you should give *both* the operating system user `mdis` and the user `www-data` (unpriviledged user) write permissions to certain directories: `backend/dis_templates/forms/` , `backend/dis_templates/models/` ,

`backend/forms/` , `backend/models/` , `src/forms/` . User `www-data` is the owner of the webserver process or service and will need to write in these directories when mDIS users work with the templaes manager.

> **More about Directory Permissions**
>
> It is quite common to put files in these directories under version control (*git*). When you checkout files from version control, you do so via the command line, using a regular user account, e.g. `mdis` . This user must be able to overwrite files that `www-data` owns, and vice versa. You can define such fine-grained file-access rules via Access Control Lists, ACLs.

This makes it possible to...:

- do installation tasks on the host system and within the VirtualBox
- do seamless rebuilds of forms and tables from the command line, within VirtualBox
- work interactively with the templates-manager web pages.

See **the Sysadmin Documentation** for details and operating-system specific issues (user management etc).

Advanced System setup and configuration

For more common tasks see **the Sysadmin Documentation**

> **WARNING**
>
> This page describes advanced tasks that require the mDIS developer to make changes to the mDIS source code. Anyone who does this should know how to use **Git** in order to restore the codebase back to a known good state from any misconfigurations or unwanted changes.

NodeJS, the Javascript runtime, should be installed on an mDIS developer's workstation. The Node module **@vue/cli** should be installed "globally", `-g` flag, for building the mDIS frontend.

Updating Third-party JS Code with `npm`

npm is three things:

1. `npm` is the Node Package Manager, a command line tool.
2. npm is a large software registry on the internet. See **npmjs.com**.
3. (optional): A service on website npmjs.com which can grant you a login to manage your own software packages there. (We have not uploaded mDIS code there)

mDIS relies on the command line tool `npm` to fetch software from the npmjs.com registry. This is necessary during installation of the system. Occasionally it might become necessary during maintenance, e.g. to check for updates. Use `npm` to download and apply these updates.

The `npm` tool uses the `package.json` file in the base directory as a configuration file for the third-party code mDIS needs. It stores the metadata about what it actually downloaded and installed from npmjs in file `package-lock.json` .

Updating Third-party Code needed for Development: *Client side*

The update-commands as such are simple, but verifying that upgrades do not break anything can be quite time consuming. The command-Line tool `npm` , the Node Package Manager, will be used to perform the upgrades.

> **TBC**
>
> See also **this section** from the Sysadmin Documentation which describes *server-side* updates, in particular: other, less critical updates such as monthly security patches for the operating system, for example.

Update the client side on a development environment first, then create a production build with `npm run build` . If your mDIS instance runs on a different machine than your development machine: copy the contents of directory `web/` to the production server directly. If you prefer to upgrade mDIS in a more careful way, perhaps put `web/` on a staging server first, or set up a Continuous Integration Pipeline. It's up to you to control and fine-tune the upgrade process.

For the client side of a development machine, many dependencies are stored in directory `node_modules/` . In March 2020, there are over 1000 subdirectories in `node_modules/` , occupying ~340 MB of disk space. The nodejs runtime and its "global" dependencies reside in `~/.nvm/versions/node/v12.16.1/` . Node will require an extra ~100 MB of storage space for the mDIS production deployment, and ~150 MB for development deployment because module vue-cli gets added.

- Basically, upgrade the mDIS client side with

```sh
cd /var/www/dis     # where file "package.json" resides
npm list --depth 0  # show high-level structure of packages needed
npm outdated        # list upgradable third-party code
npm update          # update third party code (security updates and minor patches only)
```

- Verification:
  - TBC - a cumbersome task. Do this by trial and error 😉, and run test suites.

Command `npm list --depth 0` shows the major Node/Javascript modules that mDIS relies on. In March 2019 there are 38 direct dependencies required. The 1122 directories mentioned above contain the dependencies in the subdirectories of these 38 direct dependencies. The `node_modules/` directory structure still appears relatively flat, because this reduces redundancies; moreover, directory structures that are nested very deeply can cause problems with the operating system.

Command `npm outdated` will check the **npm registry** **(docs)** to see if any (or, specific) installed packages are currently outdated.

Command `npm update` will update all the packages listed to the latest version, respecting semantic versioning. Only "minor version upgrades" and "patch version upgrades" will be applied by `npm update` .

Run the command `npm help update` for details.

> **npm run build**
>
> In a *production environment*, only the essential Javascript dependencies necessary to run the front end are packaged together into directory `web/` . The Webpack Buildsystem merges them into a few files that are highly optimized for disk space. The most important file is called `app.js` and occupies only about 2 MB. Third-party code needed later is stored in files called `chunk_*.js` . The packaging is achieved by command `npm run build` .
>
> At compile time, to build the system, an install size of ~ 400 MB of node modules is needed, but at runtime, the actual javascript code of the MDIS product, loaded by the browser, is ~2MB (publish size).

Again: For more common tasks tasks see **system setup - for SA's**.

For a good explanation of Javascript dependencies for the "Gatsby" Software, read this **blog post**.

Templates Manager

Using the GUI of the Templates Manager, the developer can create, update, duplicate, or delete Data Model Templates and then Forms Templates, in that order. Generating tables and forms is an essential DIS administration task.

> **Note**
>
> This **large topic has been placed in a separate page**, to avoid that this page gets unreasonably long.

For details, see

- the separate **Templates Manager** page.
- PHP Files: the **class hierarchy** explained
- JSON Files: For data **models** templates JSON structure, see **here**
- JSON Files: For **forms** templates JSON structure, see **here**
- mDIS Forms: for basics, see the **introduction** page.

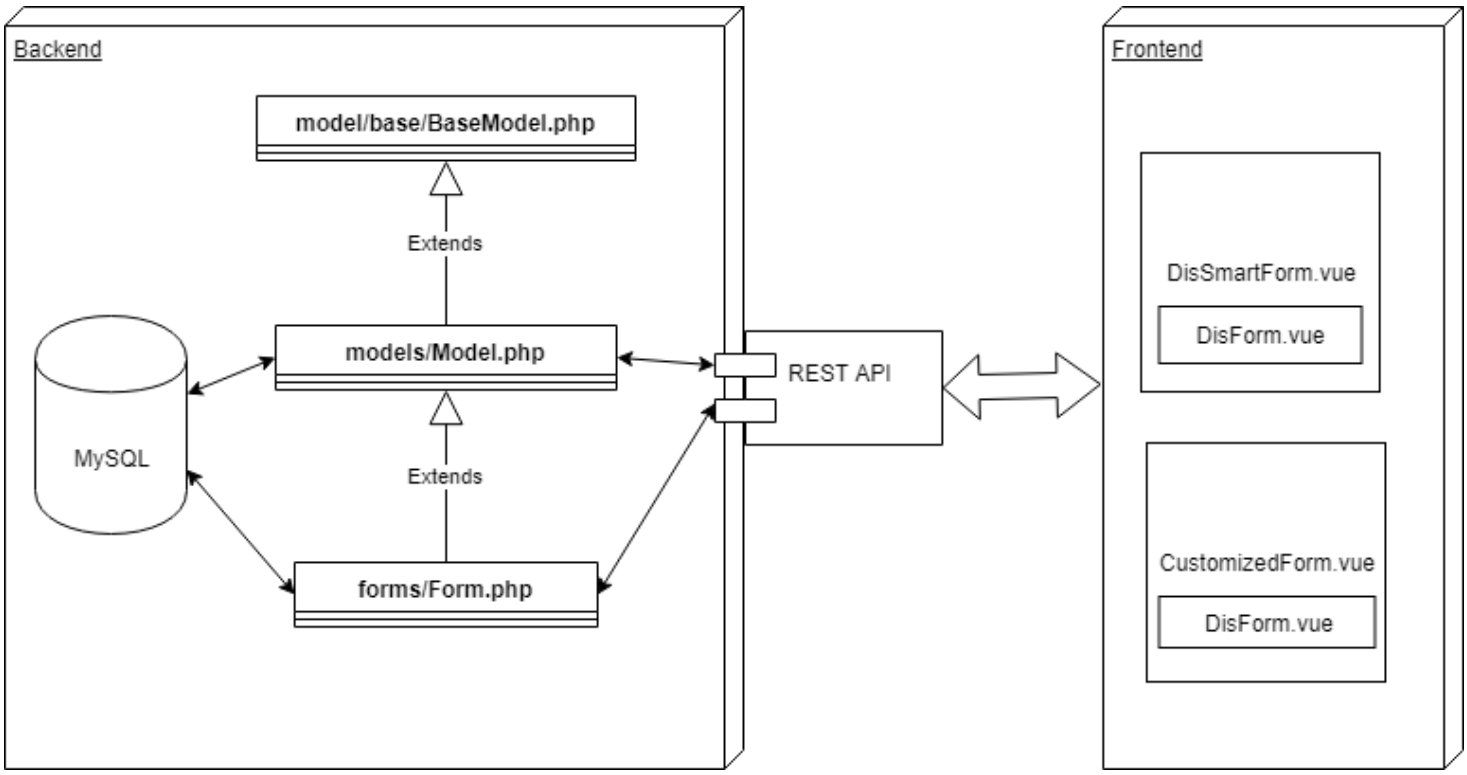
For the output of a single run of the Templates Manager, see section **"Summary"** below.

See also a separate document on **gii** for a deep dive into code generation.

Code generation principles

> **Technical note**
>
> Most PHP code gets generated into **directory** `backend/models` . The details of the Code Generation mechanisms are hidden from mDIS administrators by a GUI and an infrastructure that is explained in simple terms below.
>
> Internally, the mDIS Template manager consists of `yii\base\Module` classes, which make use of **Gii**, the Yii2 extension for code generation. For background on Gii, see also **a separate document**.
>
> For implementation details, see directories `backend/modules/api/` , `backend/modules/cg/` and file `backend/config/web.php` .

These are the main software components that are involved on the server-side (backend) and on the client-side (frontend)**:**



Backend

All this is explained in greater detail in the separate **Templates Manager** page and in this **background document**.

- **Model template** generator output, directories and files:
  - `dis_templates/models/*.json` (not shown in **figure above**): These files contain model definitions and model metadata. These files can be used to transfer table definitions from one mDIS instance to

another (not the data, though).

- `models/base/Base<model name>.php` files. These are classes which inherit from `models/base/Base.php`. This class implements the extremely feature-rich **Yii ActiveRecord** functionality. `models/base/Base<model name>.php` classes are responsible for communicating with the Mysql database. These files will be updated (overwritten) every time you change and generate the data model template. Therefore, *do not change* the code inside any `Base*.php` files. (Of course, mDIS developers who know PHP may still change them *temporarily*, for debugging purposes for instance).
- `models/<model name>.php` files. These **classes** inherit from `Base*.php` classes. All the specializations you want to implement should be written in here. The `models/<model name>.php` files will *not* be accidentally overwritten during any subsequent code generation steps. This also means, the `models/<model name>.php` files get generated *only once* (unless you check "overwrite" in a certain modal dialog box that pops up when working with the Templates manager GUI). Thus, keep them under version control (git), and manually roll back any unwanted changes.

- **Form template** generator outputs:

  - `forms/<form name>Form.php` files. These classes inherit from `models/<model name>.php` classes (the specialized one, not the base one). If there you need some customizations that belong only to this specific form, but not any other form that was derived from the same data model, they should be implemented in here.
  - `dis_templates/forms/*.json` (not shown in **figure above**): These files contain form definitions and form metadata. These files can be used to transfer basic form definitions from one mDIS instance to another.

Frontend

- **Form template** generator output, directories and files:
  - The component `DisSmartForm.vue` is responsible for rendering the form on the fly, depending on the information in the template.
  - Both SmartForm and customized forms use the `DisForm.vue` component internally.
  - `src/components/*.vue.generated` files (not shown in **figure above**). These files are just a template, ready for copy-and-paste-and-rename. They are the starting-point for software developers, to implement customizations that belong to the client-side of a form. As long as these files have an extension other than `.vue`, they will be ignored by the mDIS application and the Webpack build system (which mDIS relies on internally). In particular, `.vue.generated` files will not be used during the `npm run build` process, and will *not* be bundled with the mDIS application. In contrast, `.vue` files will be included in th build process.
  - In case a customization is needed, remove the `.generated` extension, change the new `.vue` file according to your needs, and build the mDIS frontend to use the customized form. Do so with `npm run build`. Keep the new `.vue` file under version control (git).

> **Historical Perspective**

> For a comparison of code generation within the Legacy DIS app, see also an older (internal) document: "**Code generation in the mdis app**"

Summary

If you used the **Templates-Manager** to generate a new database-backed form, `LithologyTestForm3`, associated with a tableset `Lithology`, the mDIS code generator would create these 9 files:

1. `backend/dis_templates/forms/lithology-test-form-3.json`
2. `backend/dis_templates/models/LithologyLithologyTestForm3.json`
3. `backend/forms/LithologyTestForm3Form.php`
4. `backend/forms/LithologyTestForm3FormSearch.php`
5. `backend/models/LithologyLithologyTestForm3.php`
6. `backend/models/LithologyLithologyTestForm3Search.php`
7. `backend/models/base/BaseLithologyLithologyTestForm3.php`
8. `backend/models/base/BaseLithologyLithologyTestForm3Search.php`
9. `src/forms/LithologyTestForm3Form.vue.generated`

Most of these files belong to the mDIS server backend. See **figure above**.

The `.json` files (1,2) contain form definitions and table definitions. File (3) implements the main panels of the actual form. Files (5,7) implement the linking of the form to the database table. The `*Search.php` files (4,6,8) implement the Filter Bar and the "Filter by Values" Dialog. Only one file, the `.vue.generated` file (9), belongs to the Frontend, and customizing this file is optional.

> **Frontend**
>
> See **figure above** and **DisForm Tutorial** on how to customize `.vue` files. This is rarely needed, though.

For a detailed explanation of the structure of the PHP code generated and the class hierarchy, read this **background document**.

The Yii console runners

This topic is related to code generation. Yii2 comes with a built-in tool that can generate SQL code.

yii and yii.bat

In the main directory of the mDIS application, there is a small command-line program called `yii`. There is also a second program called `yii.bat`, which can be used on Windows.

These small console programs do not much more than loading a configuration file, and then delegate to the main program which is located in `backend/vendor/bin/yii`.

Open a shell and change into the directory at the root of the code base. Just run `./yii` to get a glimpse of what the program can do. It will output the command line switches and arguments.

Creating users and setting credentials with `./yii` is one important task of many.

Also take a look at the configfile `backend/config/console.php`. This is the main configuration file for `./yii`.

yii Migration Scripts

The main task of `./yii` is calling the built-in `migrate` command. For details, run `./yii help migrate`.

The `migrate` keyword is just a synonym for "custom script". Often these are system intialization scripts or scripts for maintenance tasks. The "migration" keyword was "inspired" by the web application framework **Ruby on Rails**.

When you run `yii run migrate` or its shorthand `yii migrate`, Yii checks all classes inside the `backend/migrations/` subdirectory, which are sorted naturally by the datetime stamps in their filename, and executes the method `safeUp()` in all of them, from first to last.

The method `up()` does the same. However, unlike method `up()`, method `safeUp()` executes each migration as a *database transaction. Thus calling `safeUp()` is a bit safer than `up()`, because it is easier to roll back in case something goes wrong.

There is a whole lot more to learn about migrations. See also the **About migrations** section in the Sysadmin page, and (optionally) the reference literature on Yii and Ruby on Rails to find more information.

For example:

- **The Definitive Guide to Yii 2.0**
- **RubyOnRails Documentation** (optional)

ICDP custom migrations

For mDIS we have created a few custom migrations.

These are the migrations we have in place. Most of them are only needed when a new mDIS needs to be set up and configured.

#	Filename	Task
1	m180907_120508_init_rbac.php	init Role based Access Control

#	Filename	Task
2	m180928_092529_add_api_token_field_to_user.php	add api token field to user table
3	m181015_162413_list_values_table.php	create list values table
4	m181015_162424_create_project_module_tables.php	create project module tables
5	m181015_162524_create_core_module_tables.php	create core module tables
6	m181015_162554_create_archive_file_table.php	create archive file table
7	m190218_144637_add_token_expire_field.php	add token expire field
8	m190709_135452_create_widgets_table.php	create widgets table
9	m190716_140314_create_post_table.php	create post table
10	m190717_122306_add_colors_to_widgets.php	add colors to widgets
11	m190719_133741_create_message_of_the_day_tables.php	create message of the day tables
12	m190809_124330_update_widget_to_be_clonable.php	update widget to be clonable
13	m200309_084456_create_some_sql_views.php	create some sql views

> (*Table is Output of* `ls -1 m* | perl -nli -E "\$f = \$_; s/^m\d+_\d+_//; s/\\.php//; s/_/ /g; say qq(\$f, \$_)" | csvlook --no-header -l`)

To see how they are loaded, check the `backend/config/console.php` configuration script (section `controllerMap => [...]`), and in particular see the `backend/commands/DisMigrateControllers` PHP file. See also the `migrations` table in the mySQL database of your mDIS application.

> **WARNING**
>
> Actually, when you run mDIS in production, the `*.json` files in directory `backend/dis_templates/` are more important than the old migration scripts. The json configuration files hold more information, e.g. the exact form designs, which are not available during system setup (because You have to design them).
>
> The mDIS code generator will only look in this directory, ignoring everything else (e.g. SQL `create table` statements), and it will give precedence to the information in the json templates.

mDIS MVC Architecture

Yii2 is an MVC framework. As such it is useful to explain what these terms mean with respect to mDIS.

The term "Controller" mentioned below and in the sidebar refers to the **Model–View–Controller** (MVC) idiom of software architecture. A controller class accepts input and converts it to commands for the models (here, database tables) or views (here, webpages).

mDIS Models

The code for most models gets generated with an interactive model designer. That is the point of the sections **"Templates manager"** and **"Code generation principles"** mentioned above. The PHP files for linked tables of science data reside in **directory** `backend/models` .

mDIS Views

There are very few distinct views rendered by mDIS, because most webpages get rendered by an "independent" frontend. See the **figure above**. The mDIS frontend is based on the VueJS framework, see **below** and **DisForm.vue**. The frontend make REST API calls to get their data.

- The View code for the login-page resides in `backend/views/site/index.php` .
- Headers for Reports, and simple User-Management Pages can also be found in `backend/views/site/` .

TBC

mDIS Controllers

The most important mDIS controllers are where the **REST API** implementation resides. That is in directories `backend/modules/api/common/controllers` , `backend/modules/api/modules/v1/controllers` , and `backend/modules/cg/controllers` . Furthermore there are controllers for basic tasks, e.g., rendering the login page, a controller to render the imported Archive-Files, and for rendering the Pull-Down Menu with the List of Exportable Reports. These controllers are in `backend/controllers` .

TBC

REST API

The mDIS web application uses a ***REST*** programming interface to communicate between frontend and backend. It can be used to create new types of reports, for instance. But the REST API is also available to external developers.

mDIS actually has two REST APIs: a regular REST API for science data querying and manipulation, and the code-generation API (informally named *cg*-API) for data definition operations such as "create table", "create model", etc. The cg-REST-API is for internal use and, therefore, less exposed and more awkward to use.

The REST API implementation resides in directory `backend/modules/api/modules/v1/controllers` , and the cg API in `backend/modules/cg/controllers` .

See **REST API** page for extensive documentation. For tutorials, see **Worked Examples**.

Worked Examples

Most calls to the mDIS REST API are password protected. To skip password requests, you need to login only once and get an authorization token called a **Bearer-Token**. This security token looks like a long password, e.g. `DssMim-2QERdh6DcxPV0Saj9qYuKo267` , and it works like a session-id or HTTP cookie, however it is not stored in a `Cookie:` -Header but in a different HTTP header field, and in Local Storage, optionally.

> **Unwanted Logouts**
>
> **Experimenting with the mDIS REST API might log you out of your "normal" mDIS session.**
>
> There can only be *one Bearer-Token active per user* at any given time. If a user logs in twice with a new token (and not the token already gathered), the web server will reject the older token from the previous login, and only accept the newer token. This also holds for "regular" browser windows with open mDIS sessions.
>
> Therefore, if possible,
>
> - do not use the same login from two different browsers on the same computer at the same time
> - do not use the same login from two different devices simultaneously (e.g. your Laptop and your Smartphone)
> - use a test account with its own credentials, rather than your personal account, for your scripts calling the mDIS REST API. Then, at the same time, you can use your personal account for your mDIS browser window without being logged out.

JavaScript

There is a set of **tutorials** for using the mDIS REST API with JavaScript. How to render your own HTML table, how to perform asynchronous access.

See also: Not a real tutorial, but the "Postman collection" is also based on JavaScript (and Node00) internally. See **testsuites**.

R

There is a brief **tutorial** for R. Fetch some data from the mDIS REST API, perform a few exploratory analyses.

bash

This example uses Unix command line tools to get some data from the mDIS REST API:

- `curl` to perform web requests, and

- `jq` to parse the JSON response from mDIS.

Powershell

Powershell 6 script: for quick and dirty command-line processing on Windows.

Excel, Libreoffice-Calc TBC

There are two scenarios:

- Import from within Excel, from within an empty spreadsheet by connecting to mDIS, via an Excel-Extension or VBA.
- Creating an Excel Sheet with an external program, filling it with science data, and opening it with Excel. See **Powershell** example above. Take a close look at the commented-out lines.

TBC

Google Sheets TBC

There are two scenarios:

- Import from within a Google Sheet by connecting to mDIS, via Google Apps Script
- Pushing/Exporting to a new Google Sheet from some other software, or from the command line

TBC

Perl, Python, Go, Java, ..., PHP

> Coming Soon
>
> really

DisForm.vue

The role of the `DisSmartForm` and `DisForm` components was shown in the figure **"Code Generation Principles/Frontend"** above. `DisForm` is a powerful component that establishes the basis for all data-input forms.

See **DisForm.vue** for more background and how-to-use instructions.

Calculated Fields

Calculated fields on the client side can be created with a feature in the Template Manager. A calculation formula should contain no more than a single line of Javascript code.

> **Reminder**
>
> See "For **Operators/Viewers**" what Calculated Fields are and how to perform short one-line calculation expressions.

Calculated Fields should *not* be set to "required" in the form designer. This can create problems when saving records.

There is also an **extensive tutorial** on how to create more complex calculated fields in input-forms. This is needed when the calculation procedure is longer than a one-liner.

Form Validation

Server-Side

On the server-side, form-data validation happens when a user attempts to save a record.

Validators are stored in `backend/models/base/Base*.php` files. Check the code of the `rules()` method in each class. The mDIS validators work declaratively by inheriting from the many built-in **Yii input validators** in the `backend/vendor/yiisoft/yii2/validators` directory.

Do not change `rules()` methods in any PHP-file of the `backend/models/base/` directory. You *can* change the PHP-files containing the *derived* classes in the `backend/models/` directory. Some files in that directory already provide their own `rules()` methods, for instance `ImageOfTheDay.php`, `ListValues.php`, `ListValuesSearch.php`, `MessageOfTheDay.php`, `Post.php`, `Widgets.php`. Take a look at them for inspiration.

For example, in `ListValues.php` the validation rules are declared like this:

```php
public function rules()
  {
      return [
          [['listname', 'display'], 'required'],
          [['sort'], 'integer'],
          [['listname'], 'string', 'max' =>  50],
          [['display'], 'string',  'max' => 100],
          [['remark'], 'string',   'max' => 255],
          [['listname', 'display'], 'unique', 'targetAttribute' => ['listname', 'display']],
      ];
  }
```

There is an array of arrays returned. The first array contains the field names that are required (here `'listname'` and `'display'`).

TBC

The `rules()` method might have some additional value, `'safe'`. No attribute is marked 'safe' above. But see `ListValuesSearch.php`:

```php
public function rules()
{
    return [
        [['listname', 'display', 'remark'], 'safe'],
        [['sort'], 'integer'],
    ];
}
```

This means this attribute is not constrained by some validation rules and still marked as 'safe'. Users can assign anything to this field. If it was not marked as safe, it would be ignored by the `setAttributes()` method of the `yii\base\Model` class. Attributes known to be Unsafe must be marked with '!', e.g. "!last_command". (We do not use this feature in 2019).

Calculated Fields (server-side)

The transformation of the formula entered in "Calculate" to JavaScript is performed by the PHP method `getJsCalculate()` in File `backend/components/templates/FormTemplateField.php` and Javascript in Vue Files `src/components/DisSmartForm.vue` and `src/components/DisForm.vue`.

For details, see **Form Validation - Calculated Fields examples**.

Yii Behaviors

Yii Behaviors allow to implement functionality outside of a data table. Yii Behaviors are implemented in PHP code. They are related to mixins or **PHP traits**.

In mDIS, Behaviors are used to create complex constraints and calculated fields. Similar to database triggers, Yii Behaviors can be bound to many different events on a data table. Usually they are bound to `insert` - and `update` - Events.

mDIS makes use of Yii Behaviors to ...

- concatenate column-values in a very specific way (as "combined keys", e.g. "5063_1_A_3_1")
- create **IGSNs**
- create versatile autoincremented values that can be customized (e.g. they can have gaps, can be nonunique, can be given a start value > 1)
- create numeric multi-row constraints on some groupings inside a data table (ChildrenLimitBehavior)
- and more.

The mDIS Template Manager comes with a Graphical User Interface (GUI) to set some of these behaviors and attach them to tables and models. See the table creation page of the **Template manager**.

TBC

(Unfortunately the documentation for the GUI for the template behaviors is still lacking at this time.)

TBC

However it is still very likely that you need to change the mDIS source code to create or adapt the behaviors *you* require, to manage the specifics of your geological materials and science data. Thus you probably need access to the generated source code for the model and for the derived class. The code is in `backend/models/Base*.php` files.

> **Intermediate PHP programming skills needed**
>
> Creating or adapting existing Yii Behaviors requires changing the PHP source code of mDIS. This entails opening `.php` files, changing some code, saving the files, reloading web page. (A compilation step is not required).

See **Customizing models and templates with Yii Behaviors** for details.

File Importers

Types of Importers

File Importers are available through the **"Uploading files"** feature.

For their implementation in PHP, see the directory `/backend/importers`. It contains files implementing this inheritance hierarchy:

```
Base.php                  # abstract class, contains 11 methods
  CsvImporter.php         # contains 13 methods
    ExampleCsvImporter.php  # contains specialized beforeRun(), readRow()
  ListValuesImporter.php
```

For the controller class, see `backend/controllers/ImporterController.php`.

The controller searches for Importers in directory `backend/importers`. They *must* end with `.*Importer.php`. Every importer inherits (via `backend/importers/Base.php`) from `yii\web\ViewAction` and (therefore) is run automatically by the Yii framework.

CSV Importer

There exists one "universal" CSV importer, `backend/importers/CsvImporter.php`.

- For each importer, specify different file extensions of the data files to be imported

- Ignoring columns in the import file, if necessary
- Renaming (and remapping) columns of input files during import
- Add extra columns that are used to calculcate other columns in the target table
- Set model/table attributes to fixed values
- Check and Correct the imported data values
- Determine the parent record (Foreign Key) in a custom way
- Target Table/ "data model" does not have to be selected, since it is fixed or determined by the import filename extension (?)

There is a short file, `backend/importers/ExampleCsvImporter.php` , that shows how to specialize the universal CSV importer. It sets some default values specifically for attributes of the `core_core` table. Details are not shown here.

Based on these two production code examples of importers it should not be too difficult to implement further importers for different formats.

> **Hints and Limitations**
>
> Adding new file-importers requires custom PHP programming.
>
> Specifying foreign keys must happen before a datafile import starts. Foreign Keys cannot be added easily to a nonempty table through the Templates-Manager GUI. Dependent tables must be emptied first.
>
> For Importer usage see also **"For Operators/Viewers"** (basic use) and **section 2.9.5** (optional).

ListValues Importer

The list values files must be simple text files, more specifically: semicolon-separated values.

TBC

Reports

> **Reminder**
>
> As mentioned in the Beginner documention, there are **single-item reports and multi-item reports**.

Report items

Storage Locations of graphical items in the Backend

Many reports contain graphical items such as logos, headers, footers, formatting directives, etc. Where are these items stored?

Answer: Internally, some reports use **Yii Asset Bundles**. `AssetBundle` represents a collection of asset files, such as CSS files, JS files, images. They are stored in directory `backend/assets`.

Other HTML-, CSS- and JS-code fragments are stored as **Here-Document Blocks** inside some methods of some PHP files such as `/backend/reports/SectionMarumQrCodeReport.php`. From there they are returned by methods such as `getJS()`.

Customizing Reports

Backend: Change the PHP files in directory `/backend/reports/`. (Continued below).

Frontend: Any client-side changes are unnecessary. `DisForm.vue` has code that dynamically filters and loads any reports available from that form. The files are loaded automatically from `/backend/reports/`. For illustration, this is the code snippet that loads all single-item reports of a form, when you click on the upper "Export" button, and then displays a little pulldown menu with available reports:

```
<v-list v-if="selectedItem && selectedItem.id">
  <v-list-tile
     v-for="(item, index) in reports.single"
     :key="index"
     :href="`/report/${item.name}?model=${dataModel}&id=${selectedItem.id}`"
     target="_blank"
     >
    <v-list-tile-title>{{ item.title }}</v-list-tile-title>
  </v-list-tile>
</v-list>
```

The code for displaying the pulldown menu of multi-item reports looks very similar.

Backend (implementation details)

File `backend/controllers/ReportController.php` is in charge of processing the `:href=/report/` URL-path-fragment in the code sample above. The controller is configured in `/backend/config/web.php`. It looks for PHP files in directory `/backend/reports` following this naming convention: `$reportsPath . $reportName . "Report.php"`, so reports *must* end with `Report.php` (similar to imorters mentioned above). PHP classes must be named/namespaced according to this rule: `"\app\reports\" . $reportName . "Report";`. Inside any such class, it must have a constant property `MODEL` which specifies on which form's pull-down-menu the linked report will appear, and a property `SINGLE_RECORD` specifying if it is a **single-item report or a multi-item report**:

```php
class CoreQrCodeReport extends Base
{
    const TITLE = 'Print Core QR code';

```

```
    /**
     * This report can only be applied to Core forms.
     */
    const MODEL = 'CoreCore';

    /**
     * This report prints labels for all filtered records of the form.
     */
    const SINGLE_RECORD = false;

    // more code to generate CSS/JS/HTML fragments, not shown
    function getTemplate(){}
    function getLabelTemplate(){}
    function getJS(){}

    function generate(){} // return the whole report
}
```

The PHP class above contains a few methods which will generate and return the report's content. For brevity only empty "function stubs" of these methods are displayed here (the empty `{}` pairs).

continued in : **For Developers - ICDP internal**

Developer page

DisForm.vue

Purpose of DisForm

`DisForm` is a feature-rich Vue component that is used to filter, list and update the data records of a data model (table). Using the Template-Manager, mDIS Admins **design** new tables/models and new data input forms that are typically based on `DisForm`. The newly generated "artifacts" - a set of `.vue files`, `.json` files and others (PHP files) - can then be customized. In Legacy DIS, a similar process (of customizing data-models) is also known as *specialization*.

> WARNING
>
> Files `src/components/DisForm.vue` and its container `src/components/DisSmartForm.vue` should not be changed by any user.

The `<template>` section of `DisForm` typically contains two other **Form** components (which should not be changed either):

- `DisFilterForm.vue`: Implements the **Filter Bar** visible at the top of each form. `DisFilterForm` is used to render the filter components and filter-by-value form. This component updates the current URL query parameters corresponding to the selected filter component's value. It also offers a modal *Filter-by-Values* dialog-box that processes the new filter expressions entered by the user. The filter settings then "trickle through" the other components embedded in the form, and the filter is applied on the record-listing, at the bottom of the page.
  After receiving the new value of a filter expression, all affected components update themselves. This process is triggered by events emitted by changes in `DisFilterForm`.
- `DisDataTable.vue` which lists and paginates the **linked table records**. The listing depends on the current filter settings, visible as URL parameters in the browser's address bar on the current page. Alternatively, the listing may also depend on the current filter-by-value settings obtained from the `DisFilterForm.vue` component (in case this dialog box was used).

The `<script>` section of `DisForm` contains a rich `data` array and a lot of `props` and `methods`. There are too many to discuss them here. Most of them are either Click-handlers for the many buttons inside the navigation bar and filter bar, or other types of Event Listeners.

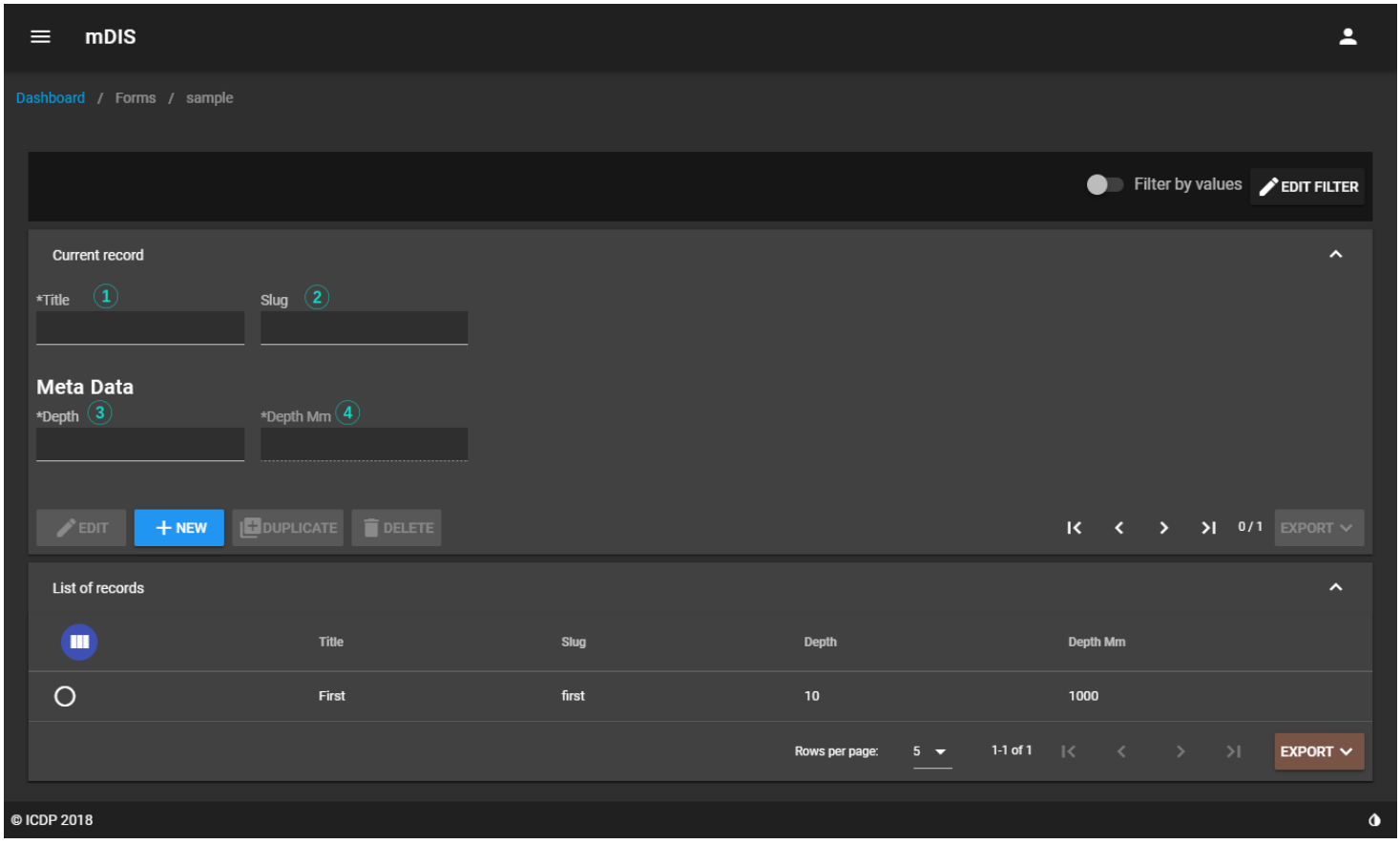
Tutorial: customizing a `.vue` file

Form Specialization: Data Fields

Forms generated by mDIS admins are typically based on `DisForm`. The new forms can be customized, or specialized. The remainder of this document is a tutorial how to perform various customizations on a file `SampleForm.vue`. (You might use a different name for your form, and that's okay.)

> **Info**
>
> - This tutorial demonstrates how to customize the forms with a text editor, *by editing the generated source code*.
> - Therefore, the Templates-Manager GUI plays only a minor role in this tutorial, mainly during initial form generation.
> - If you follow along in your IDE, copy the sections in **green font** from the code blocks below. After pasting them into your `SampleForm.vue` file (or equivalent), *remove the plus signs* (**+**) from the beginning of each line!
> - If you work in development mode (`npm run serve`), after saving any `.vue` file, the changes should be immediately visible in the browser. In production mode, you need to run `npm run build`.

To make form specialization easier to understand, we will start from this example:



A simple form with four fields. Here are the model templates and the form templates, both in JSON format:

Model Template

Generate this model with the Template-Manager. Alternatively, save this code into a `.json` file, then upload it with the "Upload a model template (*.json)" feature of the Template-Manager. "Save & Generate" the model.

On line 2, 3, and 4 of the following code, you can change the "module", "name", and "table" entries to something you prefer. ("module" means "TableSet", "name" is the Model Name as it appears on the web page, and "table" is the name of the database table. Avoid blanks .):

```json
{
    "module": "Sa",
    "name": "Sample",
    "table": "sa_sample",
    "importTable": null,
    "parentModel": "",
    "columns": {
        "id": {
            "name": "id",
            "importSource": "",
            "type": "integer",
            "size": 11,
            "required": false,
            "primaryKey": true,
            "autoInc": true,
            "label": "ID",
            "description": "",
            "validator": "",
            "validatorMessage": "",
            "unit": "",
            "selectListName": "",
            "calculate": "",
            "defaultValue": ""
        },
        "title": {
            "name": "title",
            "importSource": "",
            "type": "string",
            "size": 50,
            "required": true,
            "primaryKey": false,
            "autoInc": false,
            "label": "Title",
            "description": "The title of this record",
            "validator": "",
            "validatorMessage": "",
            "unit": "",
            "selectListName": "",
            "calculate": "",
```

```
            "defaultValue": ""
        },
        "slug": {
            "name": "slug",
            "importSource": "",
            "type": "string",
            "size": 100,
            "required": false,
            "primaryKey": false,
            "autoInc": false,
            "label": "Slug",
            "description": "A unique kebab-case title",
            "validator": "",
            "validatorMessage": "",
            "unit": "",
            "selectListName": "",
            "calculate": "",
            "defaultValue": ""
        },
        "depth": {
            "name": "depth",
            "importSource": "",
            "type": "double",
            "size": null,
            "required": true,
            "primaryKey": false,
            "autoInc": false,
            "label": "Depth",
            "description": "Depth in meters",
            "validator": "",
            "validatorMessage": "",
            "unit": "",
            "selectListName": "",
            "calculate": "",
            "defaultValue": ""
        },
        "depth_mm": {
            "name": "depth_mm",
            "importSource": "",
            "type": "double",
            "size": null,
            "required": true,
            "primaryKey": false,
            "autoInc": false,
            "label": "Depth Mm",
            "description": "Depth in millimeteres",
            "validator": "",
            "validatorMessage": "",
```

```
            "unit": "",
            "selectListName": "",
            "calculate": "",
            "defaultValue": ""
        }
    },
    "indices": {
        "pk_id": {
            "name": "pk_id",
            "type": "PRIMARY",
            "columns": [
                "id"
            ]
        },
        "unique_slug": {
            "name": "unique_slug",
            "type": "UNIQUE",
            "columns": [
                "slug"
            ]
        }
    },
    "foreignkeys": [],
    "createdAt": 1556631283,
    "modifiedAt": 1556631295,
    "generatedAt": 1556631298,
    "fullName": "SaSample"
}
```

Form Template

Generate this form with the Forms-Manager, or upload it with the "Upload a form template (*.json)" feature of the Template-manager. "Save & Generate" the form. Note the file names that are being generated.

json

```
{
    "name": "sample",
    "dataModel": "SaSample",
    "fields": [
        {
            "name": "title",
            "label": "Title",
            "description": "The title of this record",
            "validators": [
                {
                    "type": "required"
                },
                {
```

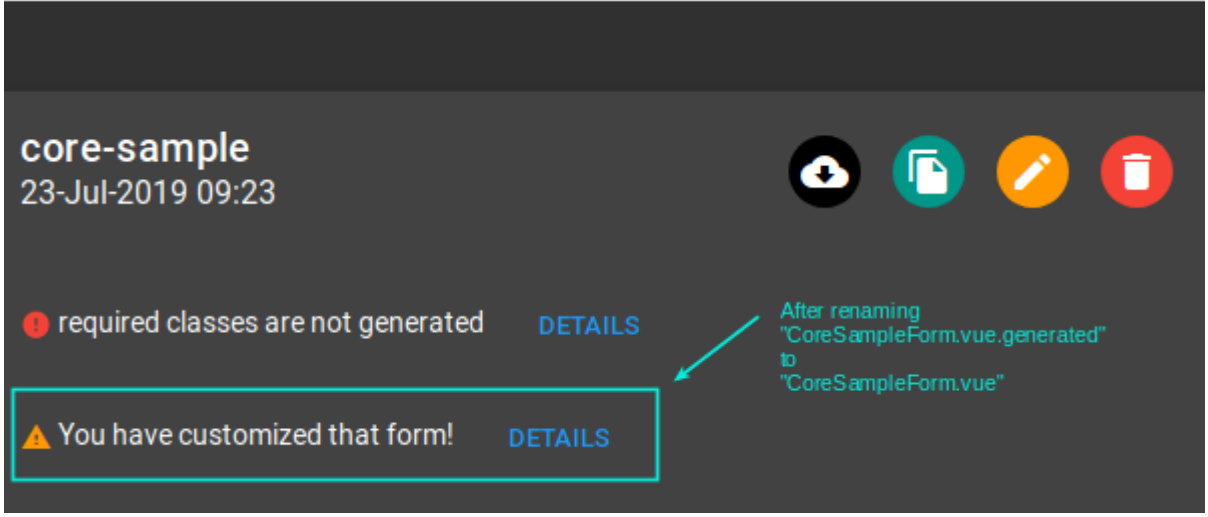
```
                "type": "string"
            }
        ],
        "formInput": {
            "type": "text",
            "disabled": false,
            "calculate": "",
            "jsCalculate": ""
        },
        "group": "-group1",
        "order": 0
    },
    {
        "name": "slug",
        "label": "Slug",
        "description": "A unique kebab-case title",
        "validators": [
            {
                "type": "string"
            }
        ],
        "formInput": {
            "type": "text",
            "disabled": false,
            "calculate": "",
            "jsCalculate": ""
        },
        "group": "-group1",
        "order": 1
    },
    {
        "name": "depth",
        "label": "Depth",
        "description": "Depth in meters",
        "validators": [
            {
                "type": "required"
            },
            {
                "type": "number"
            }
        ],
        "formInput": {
            "type": "text",
            "disabled": false,
            "calculate": "",
            "jsCalculate": ""
        },
```

```
            "group": "Meta Data",
            "order": 0
        },
        {
            "name": "depth_mm",
            "label": "Depth Mm",
            "description": "Depth in millimeteres",
            "validators": [
                {
                    "type": "required"
                },
                {
                    "type": "number",
                    "min": null,
                    "max": null
                }
            ],
            "formInput": {
                "type": "text",
                "disabled": false,
                "calculate": "[depth] * 100",
                "jsCalculate": "this.formModel['depth'] * 100"
            },
            "group": "Meta Data",
            "order": 1
        }
    ],
    "filterDataModels": [],
    "requiredFilters": [],
    "subForms": [],
    "supForms": [],
    "createdAt": 1556631418,
    "modifiedAt": 1556631472,
    "generatedAt": 1556631490
}
```

With this example we will demonstrate a typical form specialization case.

The first step is to remove the `.generated` extension from the generated form file. In directory `src/forms`, rename the file `SampleForm.vue.generated` to `SampleForm.vue`.

In the TemplatesManager/FormsManager GUI, a warning appears:



Be careful during later runs of the form-generator for that model, you might overwrite your own changes. Put the files you customized under version control (git). For this tutorial, this is not really necessary.

Toggle editable/uneditable

In `SampleForm.vue` find the textfield you want to edit, and simply change the value of the `disabled` property from `false` to `true` :

```
<DisTextInput
    :class="{'c-dis-form__input': true, 'c-dis-form__input--modified': formScenario === 'edit'
-    :disabled="false"
+    :disabled="true"
    :validators="validators['title']"
    name="title"
    label="Title"
    :serverValidationErrors="serverValidationErrors"
    hint="The title of this record"
    v-model="formModel['title']"
    :readonly="formScenario === 'view'"/>
```

> This edit can also be done via Templates Manager. But for demonstration purposes we *change the file* `src/forms/SampleForm.vue` .

Hide/unhide

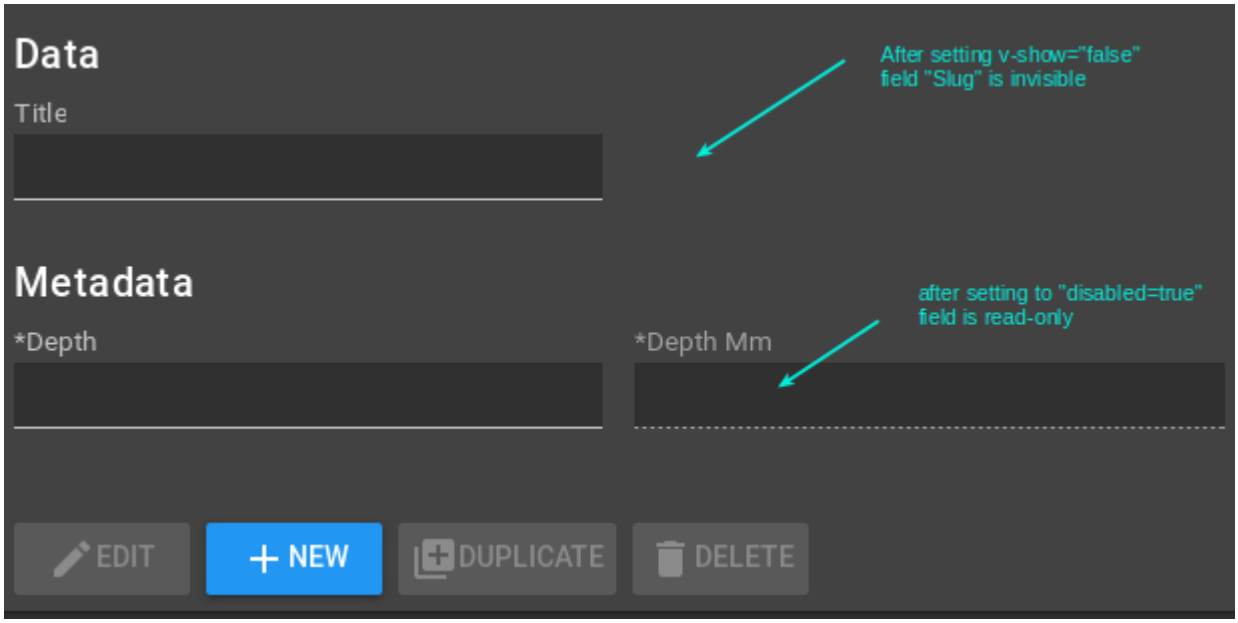
This edit is similar to the previous edit above. But this time add attribute **v-show** to control the field visibility:

```
<DisTextInput
    :class="{'c-dis-form__input': true, 'c-dis-form__input--modified': formScenario === 'edit'
    :disabled="false"
+    v-show="false"
    :validators="validators['title']"
```

```
    name="title"
    label="Title"
    :serverValidationErrors="serverValidationErrors"
    hint="The title of this record"
    v-model="formModel['title']"
    :readonly="formScenario === 'view'"/>
```

If the line with `v-show` is not there, you must add it.

Now the form should look like this (details can differ):



Change field widths

The fields are arranged using a Vuetify layout scheme called **"Grid System"**. You are free to modify the grid, and thus to re-arrange the input fields, as you wish. As an example, we will change the layout from a 2x2 grid to a 4x1 grid, thus render every field on its own full row. We accomplish this by removing the four `"md3 sm6"` snippets from the `SampleForm.vue` file.

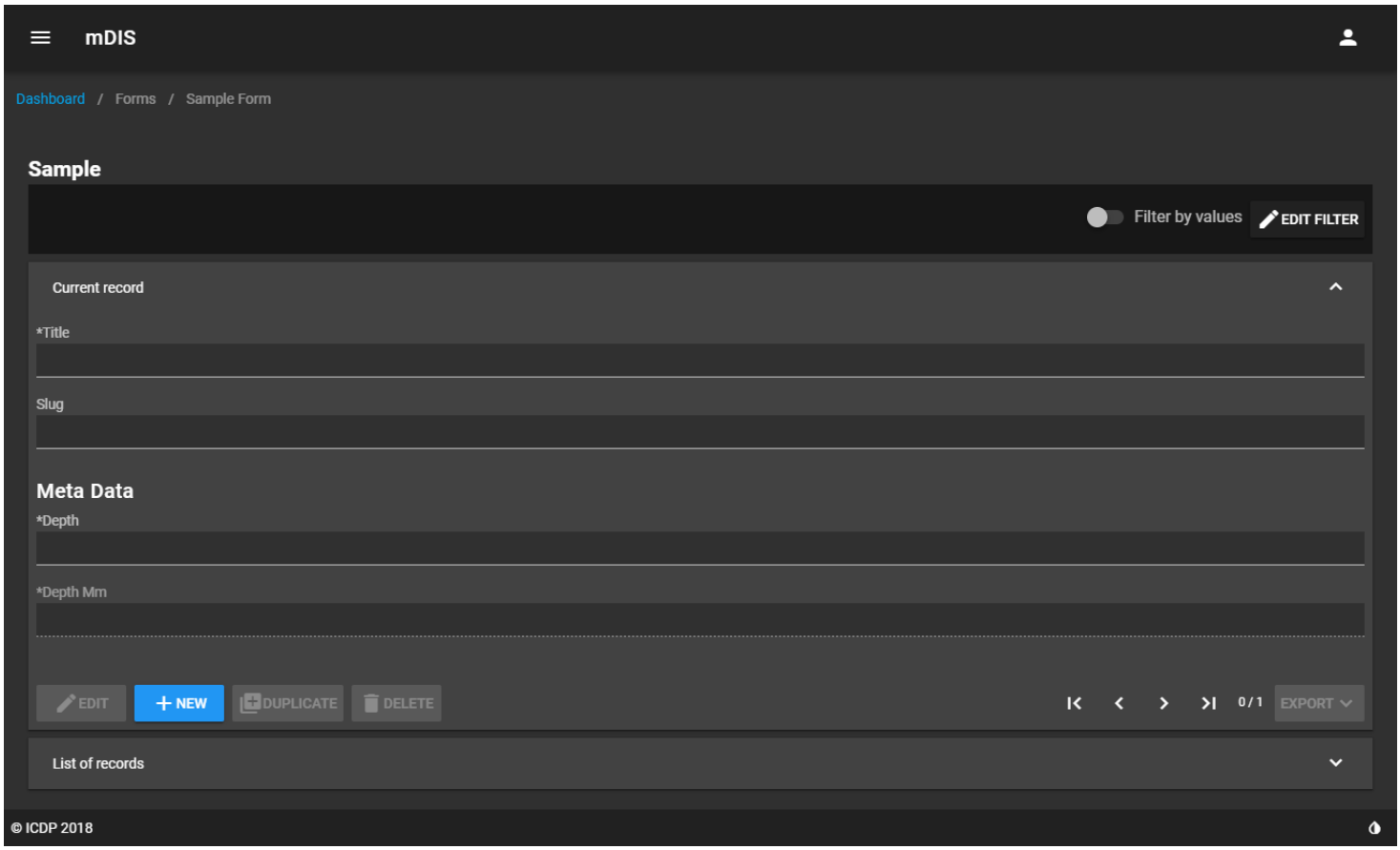
```
<v-layout wrap mb-3>
    <v-flex xs12 pl-2 pt-2>
        -group1
    </v-flex>
-    <v-flex lg2 md3 sm6 xs12 pr-2 pl-2>
+    <v-flex lg2         xs12 pr-2 pl-2>
        <DisTextInput name="title"/>
    </v-flex>
-    <v-flex lg2 md3 sm6 xs12 pr-2 pl-2>
+    <v-flex lg2         xs12 pr-2 pl-2>
        <DisTextInput name="slug"/>
    </v-flex>
</v-layout>
<v-layout wrap mb-3>
```

```
15      <v-flex xs12 pl-2 pt-2>
16          Meta Data
17      </v-flex>
18  -    <v-flex lg2 md3 sm6 xs12 pr-2 pl-2>
19  +    <v-flex lg2         xs12 pr-2 pl-2>
20          <DisTextInput name="depth"/>
21      </v-flex>
22  -    <v-flex lg2 md3 sm6 xs12 pr-2 pl-2>
23  +    <v-flex lg2         xs12 pr-2 pl-2>
24          <DisTextInput name="depth_mm"/>
25      </v-flex>
26  </v-layout>
```

> These changes effectively remove the table rendering rules for medium (md) screens and small (sm) screens, treating all screens as extra small. This forces a full-width rendering on most screens, regardless of size. (The terms `md3`, `sm6` etc are CSS classes from the `Vuetify` component library which itself is inspired by *Bootstrap*.)

Result:

Note the larger widths of the input fields after changing the 2x2 grid to a 4x1 grid.



Change field order

Changing the order of the fields is also possible. but it's not shown here. Simply switch positions of any `DisTextInput` elements inside the `Sample.vue` file.

> This feature is available via Templates Manager, too. Drag-and-drop the field positions.

Set label/hint

This change can be demonstrated with another couple of very simple one-liners:

```
<DisTextInput
    :class="{'c-dis-form__input': true, 'c-dis-form__input--modified': formScenario === 'edit'
    :disabled="false"
    :validators="validators['depth']"
    name="depth"
-    label="Depth"
+    label="Driller Depth"
    :serverValidationErrors="serverValidationErrors"
-    hint="Depth in meters"
+    hint="Driller Depth in meters"
    v-model.number="formModel['depth']"
    :readonly="formScenario === 'view'"/>
```

(Result not shown)

> Available via Templates Manager, too.

Calculated fields

Let's try an advanced example: Add a new field `error_estimate` that is calculated on the fly.

> In principle, available via Templates/Forms Manager GUI, too. Here we accomplish this in code, with more fine-grained control.

In the `<script>` element of the `Sample.vue` file, *after* the `this.simpleFields = ...` line insert the following Javascript code.

This JS snippet adds a textfield definition `error_estimate` that does not exist in the model. Nevertheless it will be displayed on the form:

```
this.simpleFields = JSON.parse('[{"name":"title","label":"Title","description":"The title of t
+    this.simpleFields.push({
+      name: 'error_estimate',
+      label: 'Error Estimate',
+      description: 'Displays an error estimate of the Driller Depth',
+      group: 'Meta Data',
+      order: 2,
+      inputType: 'text'
+    })
```

```
        this.requiredFilters = JSON.parse('[]')
        this.subForms = JSON.parse('[]')
```

Previously, in Section **"Hide/unhide"**, you have changed a `<DisTextInput>` element by setting `v-show="false"` . In the `<template>` element of the `Sample.vue` file, find that `<DisTextInput>` element. Immediately before the opening `<DisTextInput>` , add the following HTML/Vue code (without `+` s). The green snippet defines an input field for the new field `error_estimate` :

```
        <v-flex xs12 pr-2 pl-2>
            <DisTextInput
                :class="{'c-dis-form__input': true, 'c-dis-form__input--modified': formScenario ==
                :disabled="true"
                :validators="validators['depth_mm']"
                name="depth_mm"
                label="Depth Mm"
                :serverValidationErrors="serverValidationErrors"
                hint="Depth in millimeteres"
                v-model.number="formModel['depth_mm']"
                :readonly="formScenario === 'view'"/>
        </v-flex>
+       <v-flex xs12 pr-2 pl-2>
+           <DisTextInput
+               :class="{'c-dis-form__input': true, 'c-dis-form__input--modified': formScenario =
+               :disabled="true"
+               :validators="validators['error_estimate']"
+               name="error_estimate"
+               label="Error Estimate"
+               :serverValidationErrors="serverValidationErrors"
+               hint="Displays an error estimate of the Driller Depth"
+               v-model.number="formModel['error_estimate']"
+               :readonly="formScenario === 'view'"/>
+       </v-flex>
    </v-layout>
```

> Developers: It is important to make the field reactive. We do this by binding `formModel['error_estimate']` object to `v-model` , see third green line from below.

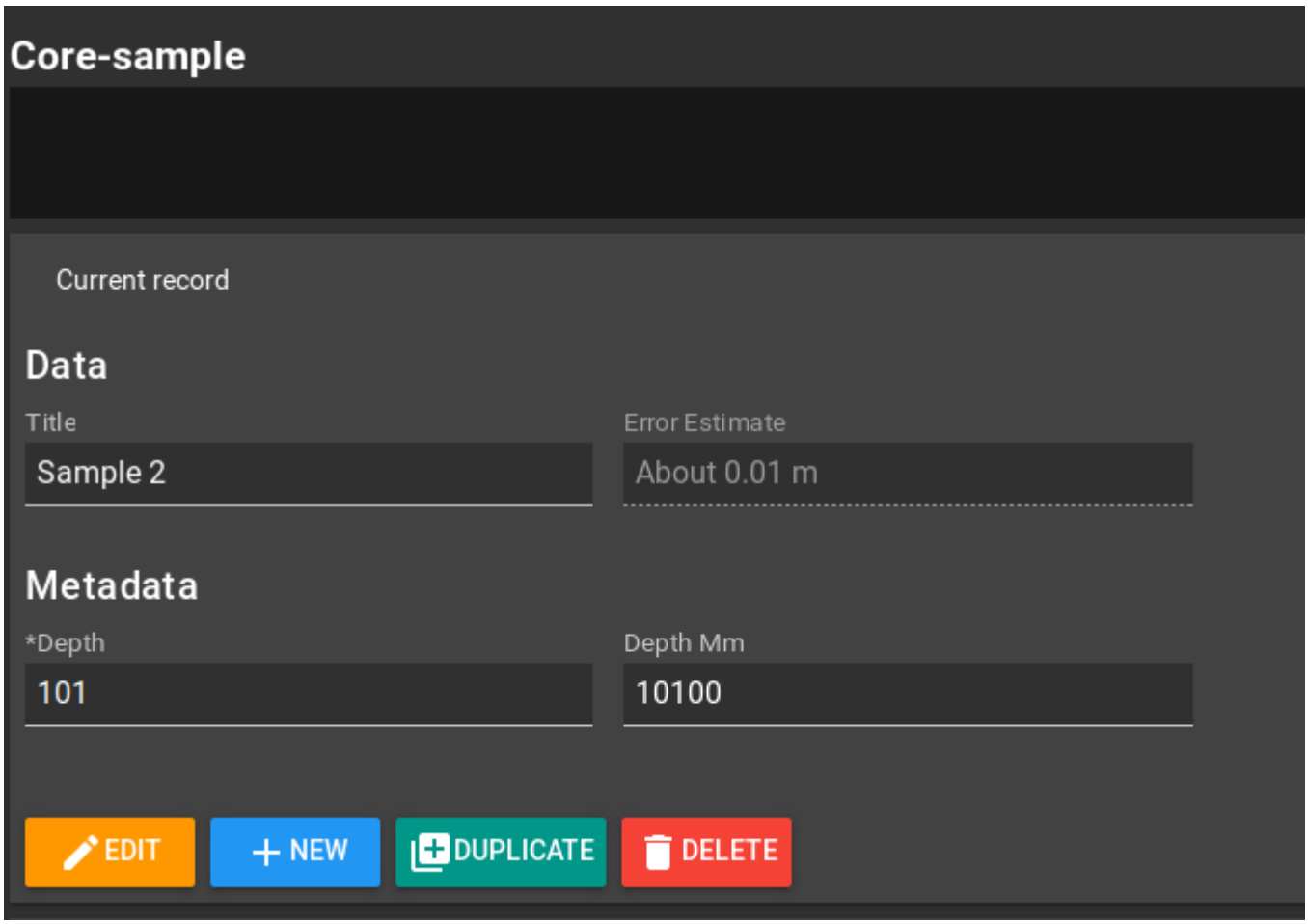
At this time, we have a new text field `error_estimate` , but the field is empty. This is because the form does not do any estimations yet.

In the `<script>` element of the `Sample.vue` file, add this Javascript code. This code defines a custom anonymous `function (){ }` with a primitive "error estimation algorithm".

Add the `this.calculatedFields['error_estimate']` object property:

```
this.calculatedFields = {}
    this.calculatedFields['depth_mm'] = function () {
      return this.formModel['depth'] * 100
    }
+    this.calculatedFields['error_estimate'] = function () {
+      if (this.formModel['depth'] > 1000) {
+        return 'About 1 m'
+      }
+      if (this.formModel['depth'] > 500) {
+        return 'About 0.5 m'
+      }
+      if (this.formModel['depth'] > 250) {
+        return 'About 0.10 m'
+      }
+      if (this.formModel['depth'] > 100) {
+        return 'About 0.01 m'
+      }
+      return 'Not Enough Data!'
+    }

    this.simpleFields = JSON.parse('[{"name":"title","label":"Title","description":"The title
```

Result (Note the new "Error Estimate" text field):



Exercises

1. Test the form reactivity: Update any value of any record. Are the calculated fields updated immediately? Are they displayed correctly?
2. Add a similar calculation rule for the "Depth mm" field. It should take the value of Depth (m) and multiply it with 100.

Vue Slots

> **At a glance**
>
> Slots could be used to inject static content, e.g. text blocks such as "How-To" descriptions, into forms with complicated data entry requirements, or to add further instrutions for mDIS operators.
>
> This tutorial here is only very basic, and does only scratch the surface of what is possible with Vue Slots.

Vue Slot Code in DisForm.vue

The `<template>` element of the `DisForm` and `DisSmartForm` components each contain two named `<slot>` elements: slot `form-fields` and slot `extra-form-actions`. These are Vue **scoped named slots** that could be used for form specialization. (*Scoped* slots have access to data only available in the child component. *Named* slots are different from the *default* slot of which there may exist only one per `<template>` block. Named slots thus are more flexible than default slots.)

There are many more slots defined in the mDIS `.vue` files but let's discuss the `form-fields` and `extra-form-actions` slots.

> **Slots vs. Props**
>
> **Vue Slots** can be used to pass HTML fragments (and even Template code) between Vue components. Slots can do this better than **Vue Props**, which are more suitable for JS code snippets.

Now you should take a look at source code of file `src/components/DisForm.vue`, where the slots are defined. (The line numbers are lines ~27 and ~132. Do not change the file).

Slot " `form-fields` "

As the name states, the content of this slot will be rendered in the position where the form fields reside. This slot provides access to the following variables in `DisForm.vue`:

formScenario

The current form scenario. `create` , `edit` or `view`

selectedItem

The current selected item (the selected item in the list)

formModel

An object that holds the current record that is loaded on the form

serverValidationErrors

The validation errors that are returned form the server will be assigned to this variable.

Slot " `extra-form-actions` "

The content of this slot will be rendered at the bottom of the form. By default, the child / parent forms are rendered in this slot. This slot provides access to the following:

selectedItem

The current selected item (the selected item in the list)

onSubFormClick

event handler for the child forms (see implementation in `DisForm.vue`)

onSupFormClick

event handler for the parent forms (see implementation in `DisForm.vue`)

Customizing .vue File (cont'd)

Continue customizing `SampleForm.vue` . See next long code snippet below.

Insert extra component

See this example. On lines ~ 84-100 of `SampleForm.vue` , we add new code via the `extra-form-actions` slot, to embed a small clickable picture on the form.

You could add/pass *any* component (icon, static image, link, button, ...etc) in the two available slots.

We insert an additional Vue-template fragment `<v-flex><h3>Extra Elements</h3><v-layout>...</v-layout></v-flex>` into the form's `template v-slot:...` element (on line 6).

If `<template v-slot:form-fields=...>` did *not* contain a `v-slot` attribute ("slot-prop"), any unkown components provided between its opening and closing tag would be *discarded* by Webpack.

Instead, because it is part of a `v-slot`, this *will* be converted to HTML, and then be displayed on the form. But the injected HTML is *not* part of the *model*, hence it is not stored inside the database.

```html
<!-- SampleForm.vue -->
<template>
    <v-container fluid>
        <v-flex align-start>
            <h2>Sample</h2>
            <dis-form formName="sample" dataModel="SaSample" :fields="simpleFields" :requiredF
                <template v-slot:form-fields="{fields, hasNumberValidator, getInputComponent,
                    <v-layout wrap lg-2>
                        <v-flex md12 lg6 pr-2 pl-2>
                            <v-layout wrap mb-3>
                                <v-flex xs12 pl-2 pt-2 class="title">
                                    Basic Info
                                </v-flex>
                                <v-flex xs12 pr-2 pl-2>
                                    <DisTextInput
                                            :class="{'c-dis-form__input': true, 'c-dis-form__i
                                            :disabled="false"
                                            :validators="validators['title']"
                                            name="title"
                                            label="Title"
                                            :serverValidationErrors="serverValidationErrors"
                                            hint="The title of this record"
                                            v-model="formModel['title']"
                                            :readonly="formScenario === 'view'"/>
                                </v-flex>
                                <v-flex xs12 pr-2 pl-2>
                                    <DisTextInput
                                            :class="{'c-dis-form__input': true, 'c-dis-form__i
                                            :disabled="false"
                                            :validators="validators['slug']"
                                            name="slug"
                                            label="Slug"
                                            :serverValidationErrors="serverValidationErrors"
                                            hint="A unique kebab-case title"
                                            v-model="formModel['slug']"
                                            :readonly="formScenario === 'view'"/>
                                </v-flex>
                            </v-layout>
                        </v-flex>
                        <v-flex md12 lg6 pr-2 pl-2>
                            <v-layout wrap mb-3>
                                <v-flex xs12 pl-2 pt-2 class="title">
                                    Meta Data
                                </v-flex>
```

```
                    <v-flex xs12 pr-2 pl-2>
                        <DisTextInput
                                :class="{'c-dis-form__input': true, 'c-dis-form__i
                                :disabled="false"
                                :validators="validators['depth']"
                                name="depth"
                                label="Depth"
                                :serverValidationErrors="serverValidationErrors"
                                hint="Depth in meters"
                                v-model.number="formModel['depth']"
                                :readonly="formScenario === 'view'"/>
                    </v-flex>
                    <v-flex xs12 pr-2 pl-2>
                        <DisTextInput
                                :class="{'c-dis-form__input': true, 'c-dis-form__i
                                :disabled="true"
                                :validators="validators['depth_mm']"
                                name="depth_mm"
                                label="Depth Mm"
                                :serverValidationErrors="serverValidationErrors"
                                hint="Depth in millimeteres"
                                v-model.number="formModel['depth_mm']"
                                :readonly="formScenario === 'view'"/>
                    </v-flex>
                    <v-flex xs12 pr-2 pl-2>
                        <DisTextInput
                                :class="{'c-dis-form__input': true, 'c-dis-form__i
                                :disabled="true"
                                :validators="validators['estimation']"
                                name="estimation"
                                label="Estimation"
                                :serverValidationErrors="serverValidationErrors"
                                hint="Gives an opinion about the depth"
                                v-model="formModel['estimation']"
                                :readonly="formScenario === 'view'"/>
                    </v-flex>
                </v-layout>
            </v-flex>
            <v-flex xs12 pr-2 pl-2>
                <h3>Extra Elements</h3>
                <v-layout>
                    <v-flex xs12 sm4 md3 lg2>
                        <v-card color="pink" shrink>
                        <!-- src="img/other/odai-gravatar.jpeg">-->
                            <v-img aspect-ratio="1" as src="https://s.gravatar.com
                            </v-img>
                            <v-card-title primary-title>
                                Hey how are you today?
```

```
                                    </v-card-title>
                                    <v-card-actions>
                                        <v-btn flat @click="onGoToCoresClick">Go to Cores
                                    </v-card-actions>
                                </v-card>
                            </v-flex>
                        </v-layout>
                    </v-flex>
                </v-layout>
            </template>
        </dis-form>
    </v-flex>
</v-container>
</template>
```

We can even add JavaScript code that refers to the above-mentioned HTML/Vue code. For example, we can register a click-handler on the dynamically-injected HTML snippet. Add this change to the `<script>` element. (The added lines are shown in green font):

```
    this.simpleFields.push({
      name: 'estimation',
      label: 'Estimation',
      description: 'Gives an opinion about the depth',
      group: 'Meta Data',
      order: 2,
      inputType: 'text'
    })
    this.requiredFilters = JSON.parse('[]')
    this.subForms = JSON.parse('[]')
    this.supForms = JSON.parse('[]')
    this.filterDataModels = JSON.parse('{}')
-  }
+  },
+  methods: {
+    onGoToCoresClick () {
+      this.$router.push('/forms/cores-form')
+    }
+  }
}
</script>
```

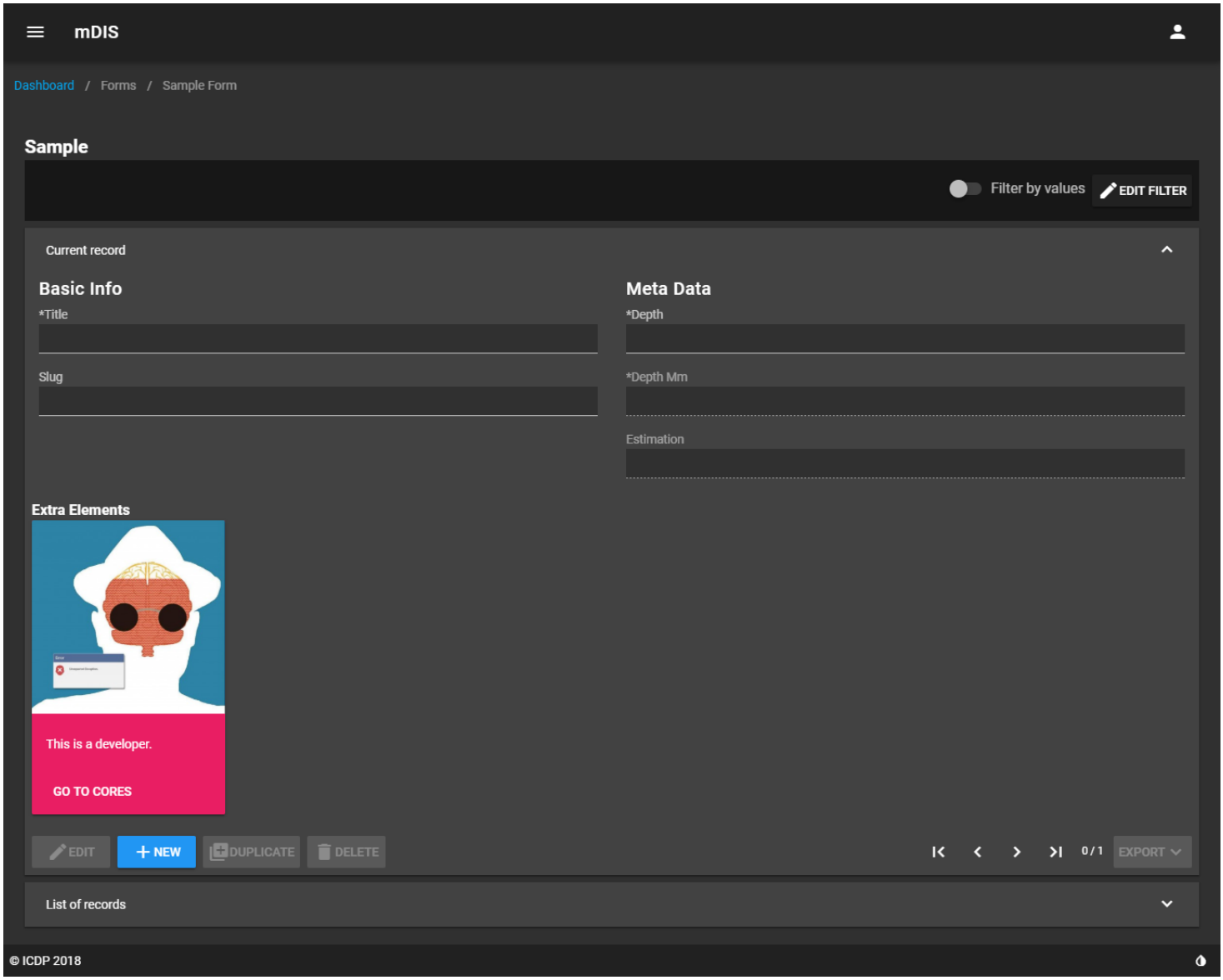
What just happened?

- In the first part of the tutorial, the fields layout was changed (From 4 to 5 text fields. Field ErrorEstimate is new.)

- An Extra element was added using Vuetify **cards** (the blue/pink picture)
- A button with a customized click handler was added to the card (A click on "GO TO CORES" text below will open the Cores Form)

Result

The "GO TO CORES" string in the picture is actually a clickable link.

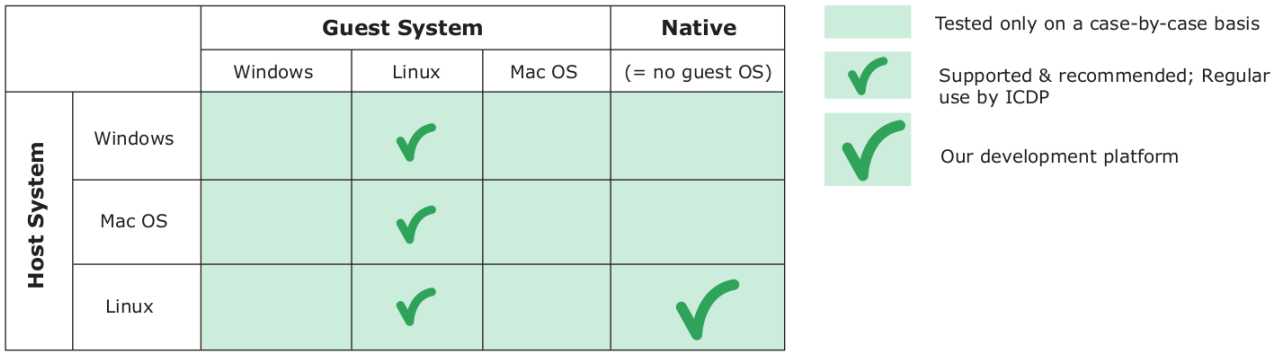


Now read the **official Slot documentation**.

Installation with VirtualBox

Supported Host/Guest Relationships

Regarding the VM hosts and VM guests, the supported operating systems are shown in the figure below.



Green background means all versions should work, and we have tried them at least once, briefly. But we cannot test all combinations thoroughly. The best supported combinations are shown with a checkmark.

For details, see also the **Virtualbox Virtual Machines** section of the general **"Technical Requirements"** page.

Hardware Requirements / BIOS settings

The VirtualBox software will not run, or will be extremely slow, if hardware support for Virtualization is unavailable.

Therefore, the CPU of the host system that will run VirtualBox must be fairly recent. That means, the PC/laptop must be manufactured after, say, the year 2010. It must be an Intel-CPU or AMD CPU. Both CPU manufacturers are well-supported by VirtualBox.

It is currently not possible to install mDIS on tablets (e.g. Apple iPad), because VirtualBox is not available for these systems.

> **CPU Version and BIOS Settings**
>
> The PC/Laptop must be manufactured after 2010, and Virtualization Support must be *enabled* in the **BIOS settings**.

Still, even if you own such a modern CPU, the support for Virtualization may be *disabled* in the BIOS settings of the PC or Laptop. *Make sure the Virtualization is set to "enabled" or "on".* The exact name of the setting is different for each Laptop/PC manufacturer, and may vary even between BIOS versions.

It may be set to *disabled* to optimize the battery performance. Setting Virtualization Support to *enabled* will increase power demand only by a little bit.

You can reach the screen showing the BIOS settings by hitting the F2 key during the first 5 seconds after switching on the PC. (The key can also be F10, or some other key - again, this depends on the laptop-/mainboard-manufacturer).

Other Hypervisors

Virtualbox 6 is recommended. Virtualbox 6.0. runs mDIS well, as does Virtualbox 6.1. However, some versions of Virtualbox 6.1 have some issues with clipboard support so working with VirtualBox versions 6.1.1 to 6.1.4 is a bit less convenient.

There are a lot of other hypervisors available (Microsoft Hyper-V, VMWare ESX, Xen, Qemu, KVM, ...). We have not tested mDIS thoroughly with any of them at this time (autumn 2019).

> **Do not use Microsoft's Hyper-V on Windows hosts**
>
> On Windows Hosts, the Microsoft **Hypervisor** built-in feature called "Hyper-V" *must not* be enabled. It cannot co-exist with the Virtualbox management software. Virtualbox guests will not boot. To disable **Hyper-V**, go to "Control Panel"/"Programs and Features"/Left Sidebar/"Turn Windows Features On and Off". A long selection list of optional Windows Features will appear. Check off "Hyper-V". Check the list for more entries containing "Virtualization". Turn them off as well. Reboot the Windows Host.

Required VirtualBox Versions

As mentioned **above** and **here**, Machine Images are usually *created* with **Virtualbox (Major Version 6)** on a Linux-, Windows- or MacOS Host.

Therefore, any prospective hosts intended for running mDIS instances as Virtualbox Guests should also have Virtualbox version 6 installed on the Host machine. Virtualbox Versions with lower major version numbers (v5) might work, but the import process of the Virtualbox `.ova` file (see below) may be less smooth. Interactive, manual adaptation of guest-machine configuration parameters might be necessary by the system administrator during the import of the `.ova` file. The import takes about 3-5 Minutes for a 5 GB `.ova` file.

How to generate a new mDIS by duplicating an existing preconfigured mDIS

Assumptions

- You have a preconfigured mDIS available inside a runnable Virtual Machine
- The mDIS is "empty", it contains no science data, except for some demo datasets, for example. (If mDIS *does* contain science data, you should keep the data as they are, clone the VM, and delete the data manually in the new copy of the mDIS. Do this deletion according to your needs.)
- You want to access the cloned mDIS using NAT networking (NAT means Guest and Host share the same IP address).

- It will be the *only* Virtualbox with that name and setting on that host computer. In other words, you do not want to "clone" an existing Virtualbox already present on your PC. You want to import an external one. ("Cloning" VMs on the same host machine has its own set of issues. In particular, some duplicated virtual network settings can create conflicts. For example, MAC Addresses, IP addresses and port forwarding rules should be tweaked slightly on the cloned/imported VM, because they should exist *only once* on the host).

The procedure entails generating a Virtualbox `.ova` image and importing this as a virtual guest OS into a host system. An `.ova` file (Open Virtualization Appliance) is a container file which contains all data and configuration needed to start up a new mDIS that is ready to run. However, it should only contain very few science datasets from previous unrelated drilling projects.

Exporting the virtual machine as *.ova* file

If you do not have an `.ova` file that you can import: Do this to create an `.ova` file that you can then import on a different host running VirtualBox.

- Inside the VM, reset any changed network settings to DHCP (don't use static IP addresses.
- Concerning the VirtualBox settings menu of the VM, reset VM network settings to "NAT" ("Bridged" is also possible but this page describes the "Network Address Translation" configuration, "NAT").
- Check how mySQL (inside the guest) listens for network traffic. Check the settings in the `/etc/mysql/mysql.cnf` file. Make sure that mySQL is not bound to a specific IP Address. Do so by commenting out the `bind_adress=... line` to allow access to Mysql from anywhere.
- From VirtualBox "File" menu select "Export appliance"
- Select the VM from the list
- Enter descriptive information like product, producer, version, ...
- Set an appropriate output path and descriptive filename
- Keep all other settings at the default values
- Click "Export"

See also (at end of page):

- section **Shrinking a virtual partition** (optional)
- section Installing the VirtualBox Guest Extensions (optional)
- section How to create an mDIS VirtualBox instance from a native instance

How to install a new mDIS inside a VirtualBox on a Host System

Key terms

- Host: Computer on which Virtualbox is installed (Host system)
- VM = Virtual Machine = Guest: Virtual computer emulated by the Virtualbox Service/Process running on the host computer. VM appears to run "inside" the host.

- .ova-image: A single file containing exported VM configuration, Guest Operating System, mDIS components, and data such as predefined mDIS users (initially, no Science Data)

1. Download and install current version of the free VirtualBox software (as of 2019: v6)

We mentioned before, under "Assumptions", that installing VirtualBox was really not needed, but was already taken care of, by you system administrator for example. For completeness this is mentioned here briefly anyway.

You need superuser/Administrator access on the host machine, becuse installing VirtualBox substantially alters the hosts' operating system. On a Linux host, for instance, VirtualBox installs 4 kernel modules (`vboxdrv, vboxpci,vboxnetadp,vboxnetflt`). On Windows, there will appear several Dialog boxes, containing both Warnings and Confirmation Dialogs. Aside from that, running the Graphical Installer for the VirtualBox Program is pretty straightforward.

2. Import Guest VM as *.ova* appliance

- The VM requires around 10GB of hard disk space on the host. If you work with the mDIS application and upload files, the required hard disk space will increase.
- The hard disk of the VM is configured to increase to a maximum of 32GB.
- In the settings, assign at least about 2 Gb of RAM for the virtual machine guest. 512 MB might also work for testing purposes.
- In the Virtualbox Menu select File -> Import appliance
- Choose "*.ova" file (Ignore "Oracle Cloud" stuff if it appeared)
- On Mac, the option "Import hard disk as VDI" should be enabled (otherwise the guest VM immediately requires the 32GB hard disk space).
- Finish the installer
- Wait for 10 Minutes. Done.

Troubleshooting

- If you get errors "Invalid Args", you can try to import the file via the command line tool "*VBoxManage*". See Virtualbox documentation.
- If that still does not work, you can create a new virtual machine and use an existing vdi disk file from a working VirtualBox installation, but this is a more advanced task.

3. Adjust VM settings

- System:
  - Adjust RAM, number of CPUs, etc. Usually no change is necessary here.
- Network:
  - decide and setup network as described below (see Chapter 4)
  - The network settings in the VM have to be adjusted, since the specifics (IPv4 Address, MAC Address...), of the virtual network device will probably change on a different host.
- Shared folder:

- Set the path for the shared folder to a convenient location on the host system. Recommended: Use the point-and-click GUI interface to the file selection dialog in order to avoid cutting-and-pasting invalid pathnames, prepended with witespace characters for instance.
- In the Virtualbox dialog box, keep the device name "Upload": in file '/etc/fstab' this is mapped to /var/www/dis/backend/data/upload (important!)
- If shared folder "Upload" does not exist, the guest VM will boot, but the shared folder mechanism will not work properly. Many kinds of error messages are possible. See Section "Troubleshooting" below.
- On Linux guest systems: there is a cronjob (in the crontab of unixuser *root*) that will mount the shared folder after restart. This means: (1) You do not need to set a mountpoint in the configuration form of the shared folder. (2) After the cronjob ran once, then the mount point and mount options for the shared folder are stored inside '/etc/fstab'.

4. Decide on network settings

There are two alternatives: NAT or Bridged network

- About NAT Networking
  - NAT means: Guest and Host share the same IP address.
  - NAT = "Network Address Translation" - from a Private IP Address Range to a Public IP Address.
  - NAT works standalone and enables small client-server setups. It is optional have the host connect to the public internet.
  - Inside-Out: Guest VM *can* connect to the public internet, if the host is connected to the public internet. The guest can access printers (and other computers) in the subnets accessible to the host.
  - Outside-In: from within the host and other computers in the host subnet, the guest VM is only accessible via *port forwarding* configured on the host (see below); i.e. the port 80 (normal Webserver port) on the host will be forwarded to some port of the guest VM
  - Other computers in the network of the host have to access the host via the forwarded port to connect to the Guest VM.
  - On the host, settings must be adjusted to allow the access to the forwarded ports
    - On *Mac* you may not forward ports less than port number 1024 (e.g. 80); you can use a *different* port (> 1024, e.g. 8080), so in the browser you would have to enter `http://ip-address-of-mac:8080/` to access the guest VM. Alternatively, use bridged networking (see below)
    - On *Linux*, it should be possible to allow VirtualBox to use privileged ports below 1024: **See post**
    - On *Windows* Hosts, Port 80 *can* be forwarded to the guest VM, but the Windows Firewall may complain on first access and proposes to immediately create a new firewall rule for port 80. Allow this. - Otherwise, use a nonprovileged port such as 8080, just like for Mac computers.
- About Bridged Network
  - Slightly more complicated than NAT Networking setup, because you need to make more decisions and do more research up front.
  - Host has to be connected to some network (by cable or WIFI)

- Guest VM: gets assigned a separate, own IP address on that network. (You must research up front which IP addresses are available, e.g. by asking your sysadmin, and you are in charge of that assignment). Guest and host are then separated "more strongly" than with NAT Networking.
- Host and other computers on the network access the guest VM via its IP address
- If the network provides DHCP, no network configuration has to be done on the guest VM, but the IP address could change between reboots. Even on a DHCP network, you could set the guest VM to a static IP address. (On Ubuntu Linux in file /etc/dhcpd/dhcp.conf).
- TBC

4.a) Setup network mode "NAT"

Create a custom NAT network configuration

Configure a preset of port-forwarding rules, that later gets assigned to your virtual machine.

- In Virtualbox Manager Window, open File/Preferences...
  - Choose tab "Network"
  - Add a new NAT network
  - Edit its configuration:
    - Name: mDIS NatNetwork - this name is arbitrary, choose a convenient name
    - Network CIDR: 10.0.2.0/24 - first three bytes (here 10.0.2) is network portion)
    - Network Options: Support for DHCP: Enabled
    - Network Options: Support for IPv6: Disabled
    - Port forwarding:
      - Add two Forwarding Ports and enter the following values.
        - HTTP to mDisBox: Protocol: TCP; Host IP: ; Host port: <e.g. 80 if Windows host, try 8080 if MacOS host)>; Guest IP:10.0.2.15; Guest port: 80
        - SSH auf mDisBox: Protocol: TCP; Host IP: ; Host port: <e.g. 22/win, 2222/MacOS>; Guest IP:10.0.2.15; Guest port: 22
        - add other well-known ports as you might see fit (e.g. 3306 for mysql direct access), but then make sure that the Windows Firewall is not blocking this

Use/assign the custom NAT network configuration

- Edit the Virtualbox configuration (click on Settings button) of the VM
  - Choose "Network" and choose "NAT Network" selection box. This activates the next selection box below (which was greyed out before). Click and select the previously assigned configuration name (here "mDIS NATNetwork") for the network adapter 1.
- Start the guest VM

- After starting the guest VM, to check the IP address the guest VM has received from the DHCP server, do this:
    - Login as User: Administrator; Password: 19DisBox
    - Open a terminal window, and type "hostname -I" (big i) to get the IP address(es) displayed or, alternatively:
    - Click on the network symbol on the top right of the guest VM's screen (next to the speaker icon)
        - Open "Wired Connected" and click on "Wired Settings"
        - Click on gear icon in section "Wired"
        - In tab "Details", check the IPv4 Address
    - If that IP address is different from the one you entered as the Guest IP in the port-forwarding dialog-box, go back to the port forwarding settings (see above), change that IP to the IP adress visible in the Details tab, and reboot the guest VM.
- Check if you can access the webserver running on the guest VM using the browser on the *guest* enter `http://localhost:80/` .
- Check if you can access the guest VM from the host using the forwarded port; in the browser's address bar *on a MacOS host*, enter `http://localhost:8080/` (or the port you've entered in "Host port" in the port-forwarding dialog-box mentioned above). On Windows hosts, `http://localhost:80` might also work.
- Troubleshooting:
    - If Windows Hosts or Mac Hosts show an alert box during startup:
      Adjust the settings of the software firewall on the host to allow external access to the redirect ports;
      Check if you can access the guest VM from a different computer (but on the same subnet) by entering in browser: `http://ip-address-of-host:forwarded-port/`
    - If Guest VM does not boot properly and the shared folder is not available:
      In file /etc/fstab, comment out line "Upload ..." by prepending it with a hashmark ("#Upload..."). Now the Shared Folders will not be available after booting up the guest, but the system is more likely to boot up properly. After booting up you can remove the hashmark from "#Upload" and type "mount -a" (without rebooting). Do this as 'root' user. This re-enables the shared folder. Optional: Before the next reboot, comment out the upload line again."
    - Post-Bootup Startup Scripts and Cronjobs: On Linux guests, see ... TBC
    - Linux guest should be rebooted after host network settings have been changed substantially, e.g. after logging into a different WLAN. Linux guest might show weird behaviour (e.g., ping still works, but DNS doesn't)

4.b) Setup network mode "Bridged"

This is an alternative to the NAT Network setup described above. Use either Bridged *or* NAT but not both.

- Edit the Virtualbox configuration of the VM
  - Choose tab "Network" and select "Bridged Network" for the network adapter 1
  - Choose the physical network device with which the host is connected to the network
    - On many hosts, it can be difficult to determine the correct network adapter to use. Things can become complicated in modern computer networks, e.g when you have Docker installed, or when you have VPN connections active. We cannot cover everything here. However:
    - On Microsoft Windows 10:
      - Open the settings and click on "Network and internet"
      - In tab "status" you see the current network connection
      - Click on the link below "Change network settings"
      - You can now see the details of the connection including IP address and then name of the network device
    - On Mac OS X:
      - Open the system settings and choose "Network"
      - Click on the connected device (hightlighted) and remember the IP address
      - Unfortunately here you do not see the device name listed in VirtualBox
      - Open a terminal window, enter ifconfig and look for the devices with that IP address
      - Is there a better way on Mac to determine the network device? ❓
    - On Linux:
      - In a terminal window enter "ifconfig" and see, which network device has an IP address in your network
- Start the guest VM
  - Login as User: Administrator; Password: 19DisBox
  - Click on the network symbol on the top right of the guest VM's screen (next to the speaker icon)
    - Open "Wired Connection" and click on "Wired Settings"
    - Click on gear icon in section "Wired"
    - In "Details" tab, note the IP address
    - The IP address should be somehow similar to that of the host. Hosts are the same subnet, so that means that the guest vm's number behind rightmost dot ("."") is different)
- Enter the IP address in the browser of the host: `http://ip-address/`
- If you prefer a "fixed" IP address for the guest VM:
  - Open the "Wired connection" settings (see above)
    - Write down the current IP address, default route, and DNS ip
    - Choose tab "IPv4"
      - Set "IPv4 Method" to "manual"

    - Enter the same values as written down before
    - Netmask is 255.255.255.0 in most small networks
        - Which IP address you can use, depends on your network and DHCP server
    - Try with a high number smaller than 250 for the last number in the IP address, i.e. 192.168.3.245
    - Klick button "Apply" on top right of dialog
    - Switch off the connection an switch on again
    - Check the ip settings again

- Test if you can still access the guest VM from the host's browser

- supported host/guest combinations

Optional, recommended

If you are using Virtualbox-based MDIS on a Linux host with limited amounts of memory, Virtualbox can slow down at times. This might also happen if you do memory-intensive tasks at the same time, e.g. when you edit a large document with lots of embedded graphics.

To make Virtualbox faster, it is recommended to create a **swap file or a swap partition**.

To see if you have a swap file enabled, type `free -m` which shows types of memory available, and the amount of free memory, in Megabytes (`-m`).

To add a swap file to your host machine, you can use these shell commands:

```sh
#!/bin/sh
sudo swapon --show
free -h
sudo fallocate -l 1G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
sudo swapon --show
```

This creates a one gigabyte swapfile on the root partition `/` of the VM host.

If you have large amounts of RAM, you can **configure the kernel parameter** `swappiness` from default `60` (percent) to `15` :

```sh
cat /proc/sys/vm/swappiness
sudo bash -c "echo 'vm.swappiness = 15' >> /etc/sysctl.conf"
sudo sysctl -p
cat /proc/sys/vm/swappiness
```

More technical info on swappiness

The guest VM is not configured to have a swap file.

Optional, advanced: Automating VirtualBox installations

You can use the free **"Vagrant"** tool to automate Virtualbox installations. For Virtualbox 6, at least version 2.2 of Vagrant is needed.

For details, see the **github repository** of the Vagrant based mDIS installer. Details are displayed in the **README file** and the source code there.

Given that Vagrant is installed properly, There is only one `vagrant` command to know `vagrant up` :

```sh
# create the Virtual machine, according to 'Vagrantfile'
vagrant up

# optional: login into the new Virtualbox (if it is Linux based and has no graphical desktop)
# username: vagrant, password: vagrant
vagrant ssh
```

Added value provided by Vagrant

This is really an advanced topic which requires some experience with Vagrant. However it's the *automation capabilities* that make learning Vagrant worthwhile.

(For example, it is a single command to install the VirtualBox Guest Extensions inside the newly created VirtualBox Guest. Try that with shell scripting!)

TBC

Optional, advanced: Shrinking a virtual hard-drive partition

When your Virtualbox host runs out of disk space too quickly

If the size of the virtual disk file (*.vdi) is much bigger than the disk space actually used by the virtual machine, this can become wasteful. Inside the box, check remaining diskspace with shell command `df -h` . Check for low values, e.g. "10%" in the `Use%` column of the `df -h` output.

If this value of percent-used is too small (that's your personal, subjective decision), you can reduce the size of the partition. This is a two-step process:

1. Prepare the machine for shrinking. This happens inside of the VM guest.
2. Shrink the partition of the Guest. Perform this on the host machine,outside of the guest.

Details:

1. Prepare for shrinking inside the VM

- (optional?) Boot from a virtual live CD, with the root file system of the linux guest instance unmounted (*TODO: verify this requirement*)
- Run application ***BleachBit***, select "Wipe free space" and run it on the root file system

2. Shrink the partition of the guest

- Remove CD from VM, change boot order of the guest VM back to Hard Drive (hd)
- On the host, search the paths for the program VBoxManage and your vdi disk file and shrink the partition of the hard drive by entering:

```
/path/to/program/VBoxManage modifymedium /path/to/mDisBox2-001.vdi --compact
```

Afterwards the .vdi file should be smaller.

Installing the VirtualBox Guest Extensions

Installing the VirtualBox Guest Extensions is not required for basic use. However, for using the "Shared Folders" feature it *is* required. On Shared Folders you can typically save backups and dumpfiles, or access and import datafiles from the host.

On the preconfigured mDIS Virtualbox images, the VirtualBox Guest Extensions are already installed. (You should check the version)

Inside a Linux guest, check if they are really installed, with the shell command: `lsmod | grep vboxguest`. There should appear the output `vboxguest`.

Alternatively, from the host, use the key combination `Host-N` which works for any guest OS i.e. type `Right CTRL-N` (if you use the default Host key configured by VirtualBox). A dialog box will appear with two tabs with metadata about the session. Check tab "Runtime Information", look for item "Guest extensions".

There is also an "extension to the extensions" called "Oracle VirtualBox Extension Pack". It is not required for mDIS. Actually it is recommended to not install the "Oracle VirtualBox Extension Pack" because it is distributed with a commercial End-User License Agreement (EULA) issued by Oracle. Agreeing to this EULA has some undesirable implications that we cannot mention here.

Optional, advanced: How to create an mDIS VirtualBox instance from a native instance

Suppose you have an mDIS running natively (without VirtualBox) on a Laptop, and you want to distribute the mDIS instance, but you do not want to give the laptop away. So you decide to create a copy of the mDIS instance inside a Virtual Box Guest. The VVirtualBox Guest can be distributed easily as an `.ova` image, as mentioned above.

This migration can be achieved in many ways. It depends on your goals (what you need to distribute) and on the software you have available. This means, importing from Linux host to Linux Guest is different than

from Windows Host to Windows Guest.

This is one way how to do this for Linux to Linux:

> **Work In Progress**
>
> - You need to have substantial Linux Experience to do the following.
> - This will probably not work unmodified. Use common sense.

- Design the native instance on a small disk partition (shrink the partition if possible, e.g. with `gparted`)
- Create a disk image of that partition with `dd`, or an appropriate disk partitioning tool.
- Create a fresh operating system instance as a VirtualBox Instance. Use an installation DVD, or DVD image, from inside VirtualBox, run the installer. Or use a preconfigured image you grabbed from elsewhere.
- Add the disk image of the partition as an addtional partition to the new VirtualBox instance. Use the VirtualBox Manager GUI to do this.
- Mount the disk image under an appropriate mount point, e.g. /mnt/mDIS, /data/mDIS, or /data
- Mount a live cd/dvd, or a third partition, such that the new Linux Guest will boot from that partition next time
- Shut down the new guest system
- Boot up the VirtualBox instance from a Linux live CD such that the (root) partition created previously is not mounted
- Use `dd` again to overwrite the root partition inside the Linux guest
- Remove the partition that you've imported previously, remove the live CD
- Boot up the Linux guest
- **Export the Linux Guest as a "VirtualBox Appliance"** using the VBox Manager GUI. This creates an ".ova" file, a container that includes the cloned native System, which is now "virtualized".
- You can distribute that .ova file.

> **Tips**
>
> - If you only want to add/import some directories, you achieve the same thing much more easily with the "shared folder" mechanism.
> - You can also create the directory, which contains mDIS, on its own disk partition. unmount the partition in the host, mount it inside the guest.

Now continue with setting up the Operating System of the Virtual Box Guest.

Tooling

(okay, not really a "Tooling" Article but saved here for now)

Changing the GUI Design using Stylesheets

Changing the design of the mDIS Graphical User Interface (GUI) might be required during customization and "branding" of a project-specific mDIS instance.

There are several ways to change the design of items on the web page. Some ways are easy but not recommended; some are appropriate but not so easy, because a build step is required, e.g. by running the `npm run build` command/script in a terminal window, or by running the equivalent "task" in the Vue-Cli UI **dashboard**

This article describes how to change the GUI design using Stylesheets. This is a simple, safe editing operation that does **not necessarily** require a build step; assuming customizing CSS is familiar to many technical users.

Stylus - "Expressive, dynamic, robust CSS built for nodejs"

mDIS uses the Vuetify GUI widget library and component collection. Specifically it uses the one of its free pre-made layouts, the "Dark" Layout theme.

The Vuetify layout depends on the *Stylus* CSS preprocessor.

In general, the Stylus processor transforms styles code to css code. Given input is a file `stylus/my.styl`, Stylus transforms it and writes output file `css/my.css`. Our mDIS does *not* do exactly that - instead, it compresses, "sourcemaps", and transforms the CSS code to **inline styles**.

For consistency, we also use *.styl* files for our design modifications.

You can put CSS fragments into .styl files without changing the CSS code.

The *.styl* syntax is compatible with CSS syntax. This means you can rename file `my.css` to `my.styl`, and the stylus preprocessor will still process it, (but will not change anything, because here input = output).

In our project the stylus files have been stored here:

```sh

./src/style/main.styl
./src/style/icdp-extra.styl
./src/style/components/_dis-form.styl
```

```
./src/style/components/_dis-data-table.styl
```

To change the design of certain features, modify file `/src/style/icdp-extra.styl` , e.g. by adding CSS classes to it.

This file `icdp-extra.styl` gets loaded after `src/style/main.styl` , overriding any existing rules in there, also adding new rules. The syntax used in the `icdp-extra.styl` file can be CSS syntax, or Stylus syntax.

File `icdp-extra.styl` is available only on host wb45; in git on development branch "knb".

After editing .styl file on wb45, the development build (which has hot module replacement enabled) should reload immediately, and show the design modifications.

Production build needs a recompilation step (run `npm run build` script, and restart the webserver).

Possible enhancements

- Supporting "pure CSS" alternatives in mDIS
- Use css files instead of inline styles in the production build

See **https://cli.vuejs.org/guide/css.html**. (Not much used/implemented/applied/used in mDIS) TBC

Using .css files directly (e.g. as assets/css/icdp-extra.css) is probably supported but Webpack, which starts the transformation process, needs the 'css-loader' activated. (Currently it is not activated.)

Appendix

Vuetifyjs Stylus files:

```sh

./node_modules/vuetify/src/stylus/components/_alerts.styl
./node_modules/vuetify/src/stylus/components/_app.styl
./node_modules/vuetify/src/stylus/components/_autocompletes.styl
./node_modules/vuetify/src/stylus/components/_avatars.styl
./node_modules/vuetify/src/stylus/components/_badges.styl
./node_modules/vuetify/src/stylus/components/_bottom-navs.styl
./node_modules/vuetify/src/stylus/components/_bottom-sheets.styl
./node_modules/vuetify/src/stylus/components/_breadcrumbs.styl
./node_modules/vuetify/src/stylus/components/_buttons.styl
./node_modules/vuetify/src/stylus/components/_button-toggle.styl
./node_modules/vuetify/src/stylus/components/_cards.styl
./node_modules/vuetify/src/stylus/components/_carousel.styl
./node_modules/vuetify/src/stylus/components/_chips.styl
./node_modules/vuetify/src/stylus/components/_content.styl
./node_modules/vuetify/src/stylus/components/_counters.styl
```

```
./node_modules/vuetify/src/stylus/components/_data-iterator.styl
./node_modules/vuetify/src/stylus/components/_data-table.styl
./node_modules/vuetify/src/stylus/components/_date-picker-header.styl
./node_modules/vuetify/src/stylus/components/_date-picker-table.styl
./node_modules/vuetify/src/stylus/components/_date-picker-title.styl
./node_modules/vuetify/src/stylus/components/_date-picker-years.styl
./node_modules/vuetify/src/stylus/components/_dialogs.styl
./node_modules/vuetify/src/stylus/components/_dividers.styl
./node_modules/vuetify/src/stylus/components/_expansion-panel.styl
./node_modules/vuetify/src/stylus/components/_footer.styl
./node_modules/vuetify/src/stylus/components/_forms.styl
./node_modules/vuetify/src/stylus/components/_grid.styl
./node_modules/vuetify/src/stylus/components/_icons.styl
./node_modules/vuetify/src/stylus/components/_images.styl
./node_modules/vuetify/src/stylus/components/_inputs.styl
./node_modules/vuetify/src/stylus/components/_item-group.styl
./node_modules/vuetify/src/stylus/components/_jumbotrons.styl
./node_modules/vuetify/src/stylus/components/_labels.styl
./node_modules/vuetify/src/stylus/components/_lists.styl
./node_modules/vuetify/src/stylus/components/_menus.styl
./node_modules/vuetify/src/stylus/components/_messages.styl
./node_modules/vuetify/src/stylus/components/_navigation-drawer.styl
./node_modules/vuetify/src/stylus/components/_overflow-buttons.styl
./node_modules/vuetify/src/stylus/components/_overlay.styl
./node_modules/vuetify/src/stylus/components/_pagination.styl
./node_modules/vuetify/src/stylus/components/_parallax.styl
./node_modules/vuetify/src/stylus/components/_pickers.styl
./node_modules/vuetify/src/stylus/components/_progress-circular.styl
./node_modules/vuetify/src/stylus/components/_progress-linear.styl
./node_modules/vuetify/src/stylus/components/_radio-group.styl
./node_modules/vuetify/src/stylus/components/_radios.styl
./node_modules/vuetify/src/stylus/components/_range-sliders.styl
./node_modules/vuetify/src/stylus/components/_rating.styl
./node_modules/vuetify/src/stylus/components/_responsive.styl
./node_modules/vuetify/src/stylus/components/_ripples.styl
./node_modules/vuetify/src/stylus/components/_selection-controls.styl
./node_modules/vuetify/src/stylus/components/_select.styl
./node_modules/vuetify/src/stylus/components/_sliders.styl
./node_modules/vuetify/src/stylus/components/_small-dialog.styl
./node_modules/vuetify/src/stylus/components/_snackbars.styl
./node_modules/vuetify/src/stylus/components/_speed-dial.styl
./node_modules/vuetify/src/stylus/components/_steppers.styl
./node_modules/vuetify/src/stylus/components/_subheaders.styl
./node_modules/vuetify/src/stylus/components/_switch.styl
./node_modules/vuetify/src/stylus/components/_system-bars.styl
./node_modules/vuetify/src/stylus/components/_tables.styl
./node_modules/vuetify/src/stylus/components/_tabs.styl
./node_modules/vuetify/src/stylus/components/_textarea.styl
```

```
./node_modules/vuetify/src/stylus/components/_text-fields.styl
./node_modules/vuetify/src/stylus/components/_timeline.styl
./node_modules/vuetify/src/stylus/components/_time-picker-clock.styl
./node_modules/vuetify/src/stylus/components/_time-picker-title.styl
./node_modules/vuetify/src/stylus/components/_toolbar.styl
./node_modules/vuetify/src/stylus/components/_tooltips.styl
./node_modules/vuetify/src/stylus/components/_treeview.styl
./node_modules/vuetify/src/stylus/components/_windows.styl
./node_modules/vuetify/src/stylus/elements/_blockquote.styl
./node_modules/vuetify/src/stylus/elements/_code.styl
./node_modules/vuetify/src/stylus/elements/_global.styl
./node_modules/vuetify/src/stylus/elements/_headings.styl
./node_modules/vuetify/src/stylus/elements/_lists.styl
./node_modules/vuetify/src/stylus/elements/_typography.styl
./node_modules/vuetify/src/stylus/generic/_animations.styl
./node_modules/vuetify/src/stylus/generic/_colors.styl
./node_modules/vuetify/src/stylus/generic/_elevations.styl
./node_modules/vuetify/src/stylus/generic/_reset.styl
./node_modules/vuetify/src/stylus/generic/_transitions.styl
./node_modules/vuetify/src/stylus/main.styl
./node_modules/vuetify/src/stylus/settings/_colors.styl
./node_modules/vuetify/src/stylus/settings/_elevations.styl
./node_modules/vuetify/src/stylus/settings/_rtl.styl
./node_modules/vuetify/src/stylus/settings/_theme.styl
./node_modules/vuetify/src/stylus/settings/_variables.styl
./node_modules/vuetify/src/stylus/theme.styl
./node_modules/vuetify/src/stylus/tools/_bootable.styl
./node_modules/vuetify/src/stylus/trumps/_display.styl
./node_modules/vuetify/src/stylus/trumps/_helpers.styl
./node_modules/vuetify/src/stylus/trumps/_spacing.styl
./node_modules/vuetify/src/stylus/trumps/_text.styl
./node_modules/vuetify/src/stylus/trumps/_transition.styl

## ? came with stylus
./node_modules/stylus/lib/functions/index.styl
```

mDIS online

mDIS Websites

Web-based versions of mDIS can be accessed at the locations shown in the table below

The mDIS software is still under active development. The websites mentioned below demonstrate the latest code. They show mDIS with its newest features added.

These websites are not production systems. Instead they serve as a reference, enabling stakeholders to have a quick look.

Sign-in is required.

Purpose	Instance Name	Content	Address	Version	Who has access
ICDP internal	internal Curation DIS	testing data	**internal.rundis.com** ⬛ 🔑 🔓	1.3.4-86 (2020-05-18), master + FB-reports-merge + FB-knb-202005wk3)	ICDP-OSG team
ICDP **GRIND Project**	grind Expedition DIS	Real World Data	**grind.rundis.com** ⬛ 🔑 🔓 🔌	1.3.4-86 (2020-05-18), master + FB-reports-merge + FB-knb-202005wk3)	ICDP, Tina, Melanie, Ana
ICDP **JET Project**	jet Expedition DIS	testing data	**jet.rundis.com** ⬛ 🔑 🔓	1.3.4-86 (2020-05-18), master + FB-reports-merge + FB-knb-202005wk3)	ICDP-OSG team Tom
ICDP **TADP Project**	tadp Expedition DIS	testing data	**tadp.rundis.com** ⬛ 🔑 🔓 🔌	1.3.4-86 (2020-05-18), master + FB-reports-merge + FB-knb-202005wk3)	ICDP-OSG team

Click on a symbol to open:

⬛ - mDIS Dashboard. 🔑 - Database Administration page. 🔓 - insecure site. 🔌 - preemptible.
Abbreviations: FB - Feature Branch. - .ova - VirtualBox image (**see below**)

> **mDIS Online**
>
> Online Versions will be **erased and rebuilt** once or twice per week, so do not put data in there that you expect to last!
>
> Online Instances can get **"preempted"** (🔌 = put offline without notice) quite frequently. This can happen *anytime*, and often happens in rapid succession!

But this "pre-empted" pricing model is the most affordable, offered by the server-hosting company (Google).

Hence, although pre-emption can be annoying, we tolerate pre-emption because these online versions are *training instances*, designed to try new features, to jointly discover new requirements, and to get accustomed to working with the data enty system, as an end user or administrator.

VirtualBox Instructions

Before the fieldwork starts, properly configured VirtualBox-based versions of mDIS will be created and shipped to project participants. These finalized VirtualBox-based mDIS instances will run on computers that are typically offline, in remote areas, or in research labs and in core repositories.

Virtualbox - Current Status

> **Lots of post-installation tasks required**
>
> **VM Instances are not finalized.**

> Download the VirtualBox instances only if you want to take a closer look at the code, or if you want to remote-debug the source code in a running instance. Besides, using a VirtualBox instance, you can avoid having your changes overwritten; and advanced visualization tools are only available in the VirtualBox-Versions of mDIS, not in any Online version.

- At this time (June 2020) VirtualBox instances are usable, although they are not completely setup and configured. VM Instances still require quite a few post-installation tasks that must be carried out manually.
- This is so because recently we have added a lot of software features. These need to be "activated", or administrators need to "opt-in" to use them in a meaningful, productive way. (We are working on completing the automated setup).
- Upgrade your VirtualBox Software to v6.1.6+, otherwise the Shared Clipboard might not work properly.
- The advanced Visualization-Tools mentioned above are **"PSICAT"** for creating lithological profiles, and **"Corelyzer"** for plotting full-resolution core-scans. Type `psicat` or `corelyzer` on a command line to start them.

Logging in

Use your mDIS credentials when asked for passwords. They are the same for both initiating the download, and for logging into the mDIS dashboard.

Import the .ova file with your VirtualBox-Manager App. Start up the newly imported VirtualBox instance.

Optional: For logging into the VirtualBox guest operating system (the Login Screen of the Graphical Desktop of Ubuntu Linux 18.04 or 20.04), use username "vagrant", password "vagrant". SSH Login also works, on port 2222.

VirtualBox-based mDIS in your browser

To log into the mDIS web application, you do *not* need to log into Ubuntu's graphical desktop. It is sufficient to just start up the VirtualBox instance. Then on the *host* computer, start a webbrowser, and open this web page: **http://localhost:8888** . (*Inside* the Virtualbox, just use **http://localhost** .)

Optional, advanced: To get rid of the "8888" you need to change a setting: the "Port Forwarding" rule in the Ubuntu Guest: choose Settings / Network / Tab "Adapter 1" (in the VirtualBox Manager Application). Your operating system might croak a bit, though. Restart the Guest VM.

Check your Shared Folder setting

We build the Ubuntu Linux virtual machines on Linux. Therefore, the default shared folder mapping is:

> `/var/tmp/upload` on the Host is shared with `/var/tmp/upload` on the Guest.

Both are Unix/Linux paths.

By default, mysql-backups go into subfolder `/var/tmp/upload/backup/mysql` .

If you run VirtualBox on Microsoft Windows or on a Mac, you need to change the Shared Folder Setting on the Host to a different folder. VirtualBox will warn you about this. (We cannot anticipate how you prefer these paths to be configured, by default, on *your* Windows PCs or *your* Mac.)

Notes

.ova-Files: Images of Virtual Machines are distributed as VirtualBox `.ova` files. This is a container format similar to `.zip` or `.tar` files.

The contents of the Virtual Machines and the respective web site (mentioned in the table above) are similar, but not exactly identical. The .ova files lag behind the releases of new online mDIS instances, typically for a few days.

> **Recommended mode of mDIS operation**

> Currently mDIS is under active software development. Use the online mDIS versions to experiment freely.

Download and import a VirtualBox image (the .ova file) for working with your own datasets that you would like to persist longer, for any reason. Using VirtualBox, you can also have a closer look at the source code of mDIS. It is also *easier to debug* PHP- and Vue-Files when you work with a Virtualbox running on *localhost*. However, for productive work, we recommend to use the above-mentioned online instances. - We know, pre-emption can be very annoying.

> **Contribute to mDIS development**
>
> If you have developed some feature (tables, forms, uploaded files, customizations) that you want to keep permanently in mDIS, please notify us. Then we will make it part of the default setup.

Mysql Administration

MySQL Root Password

For Linux

To restore mysql database dumps on Debian/Ubuntu based Linux distributions, you need credentials for a highly priviledged mysql user which is different from the mDIS user. By default, on mysql, this user is called `root`. This user is *also* different from the `root` superuser of the operating system.

> **Optional: Reset mysql root**
>
> Any mysql user who can restore databases from backups must have permission to drop and re-create database objects such as tables or indexes.
>
> Try `sudo --user=root mysql # use OS-superuser to access mysql`
>
> In case you have lost the credentials for such an mysql user, or cannot get to know them, check file `/etc/mysql/debian.cnf`. This file might contain a username and a password which you can use.
>
> Otherwise you need to "reset" the mysql system:
>
> The predefined Mysql user `root` can do anything inside mysql. Therefore you might need to know how to *reset* the mysql user root password:
>
> ```
> use mysql;
> update user set authentication_string=PASSWORD("?????") where User='root@"localhost';
> flush privileges;
> quit
> ```
>
> This was only a first step. Extra work is necessary, before and after issuing these SQL statements!
>
> Read "**mysqld_safe Directory '/var/run/mysqld' for UNIX socket file don't exists**" for details.
>
> Now you should have mysql- `root` credentials. You can continue restoring the mysql database(s) from backups, or you can create a special-purpose account for restoring backups.

More Credentials

> **TIP**

> You need valid credentials to login to mysql to see any mDIS data in tabular format; and (ideally) you need a root password (see above) to make full backups of the mysql server.

Maybe store these credentials elsewhere!

- mysql User Account used by the Webserver:
  - Username `dis` , Password (wb33: similar to pw of the `mdis` or `administrator` ssh account)
- mysql Root Account on localhost:
  - Username `root` , Password `??????`

perhaps in file `/home/mdis/.my.dis_backup.cnf` or equivalent.

```
[client]
user="dis"
password=
host=localhost
```

This config file is used in the mysqldump script below.

Dump script

This is a minimal example. It writes compressed dump files with a certain naming convention, and stores them in dir `/var/www/dis/backend/data/upload/backup/mysql/` . Adapt to your needs.

```sh
#!/bin/sh
# create a mysql dump , put that .sql file into a zipfile
# also save json templates
# knb 202004
#
host=$(hostname -I | sed -r "s/ .+$//")
host="$(uname -n)--${host}"
if [  $# -eq 1 ]
then
  host=$1
fi
host="$(echo "${host}" | tr -d '[:space:]')"


date=$(date -I)
db=dis

mkdir -p /var/tmp/upload/backup/mysql

outd=/var/tmp/upload/backup/mysql
```

```
outd_base=/var/www/dis/
outd_json=backend/dis_templates/
myd=/usr/bin
outf=mysqldb_backup_vbox_${host}--${date}--${db}.sql
outzip=${outf}.zip

if [ -d /tmp ]
then
  cd /tmp
  ${myd}/mysqldump --defaults-extra-file=$HOME/.my.dis_backup.cnf \
  --add-drop-table  --add-locks --no-create-db --extended-insert \
  --quick --lock-tables --databases $db > "${outf}"

  # quiet, move sql-dump into zip, line separator is cr-lf
  zip -q -m -l  "${outd}/${outzip}" "${outf}"

  chmod g+w ${outd}/${outzip}
fi

# also save the all-important *.json files
# (containing forms- and models-metadata) into the same zip file
cd ${outd_base}
zip -q -r  "${outd}/${outzip}" "${outd_json}"

```

Dumping creates compressed outfiles named like this: `mysqldb_backup_vbox_mdis-jet--10.0.2.15--2020-04-10--dis.sql.zip` .

Uncompress with `zip` , `7z` , or use the built-in "Extract" feature of the file managers of your Windows-PC or Mac laptop.

Here is a description what the zipfile contents could look like: **Zipfile Listing**

Cronjob

It is easy to *automatically* make backups with the above-mentioned scripts.

Check if a there is a Job running periodically. Run `crontab -l | grep -v "#"` , it should show output similar to the following:

```sh
57 14 * * * $HOME/bin/dump_dis_mysql.sh
```

If this is missing, add such an entry with `crontab -e` running as unixuser `administrator` (which is a *regular* user, not `root`).

> **TIP**
>
> Making mysql dumps can be automated, but *loading* a database is a manual task and cannot be automated. Therefore there is no cronjob for restoring databases.

Load script

It loads files from dir `/var/www/dis/backend/data/upload/backup/mysql/` . Adapt to your needs.

Script content:

```sh
#!/bin/sh
#load databases dumped at a foreign server into localhost
#
# knb 2019

host=`uname -n`
pw=
myd=/usr/bin
db=dis

if [ $# -ne 1 ]
then
echo $0: load mysql databases dumped at a foreign server into host ${host}
echo "$0: expects input file extension .sql or .bz2"
echo ""
echo Invalid format: $# argv parameters provided, 1 required
echo Required format: $0 '<DUMPFILENAME> '
echo If the filename extension is .bz2, this script will bunzip2 the dumpfile first.

exit 1
fi



fullfile="$1"
shift

filename=$(basename -- "$fullfile")
extension="${filename##*.}"
infile_no_ext="${filename%.*}"

dumpdir=/var/www/dis/backend/data/upload/backup/mysql/
infile="${dumpdir}/${filename}"
echo "dir: '$dumpdir'"
echo "looking for infile '$infile' "
```

```

if [ -z "$1" -a  "$extension" = "bz2" ]
then
  ext=".$2"
  cmd="bunzip2 --keep --force $infile"
  $($cmd)
  cmd="echo '${cmd}'"
  infile="$dumpdir/$infile_no_ext"
echo ""

elif [ -z "$1" -a "$extension" = "sql" ]
then
  echo "Trying to load dumpfile '$filename' into mysql"
  echo "$infile"

else
  echo "Nothing to import. Filename extension must bei either .bz2 or .sql "
  echo ""
fi


echo ""
ls -l "$infile"
echo ""
echo ""
echo "head -25 "${dumpdir}/$infile_no_ext" | grep -C 3 $db"
head -25 "${dumpdir}/$infile_no_ext" | grep -C 3 $db

echo "Execute this:"
echo ""
echo " ${myd}/mysql -hmyhost.gfz-potsdam.de -udis  -p$pw --database=$db -f  < $infile"


```

Example Call on commandline:

```sh
./load_some_mysqldbs.sh /var/www/dis/backend/data/upload/backup/mysql/mysqldb_backup--dis.2.sq
```

Output

```sh

dir: '/var/www/dis/backend/data/upload/backup/mysql/'
looking for infile '/var/www/dis/backend/data/upload/backup/mysql//mysqldb_backup--dis.2.sql.b


-rwxrwxrwx 1 administrator administrator 1039342 Jul 25 17:10 /var/www/dis/backend/data/upload
```

```
head -25 /var/www/dis/backend/data/upload/backup/mysql//mysqldb_backup--dis.2.sql | grep -C 3
-- MySQL dump 10.13  Distrib 5.7.27, for Linux (x86_64)
--
-- Host: localhost    Database: dis
-- ------------------------------------------------------
-- Server version 5.7.27-0ubuntu0.18.04.1

--
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Current Database: `dis`
--

USE `dis`;

--
-- Table structure for table `archive_file`
Execute this:

 /usr/bin/mysql -hwb33.gfz-potsdam.de -udis  -pXXXXX --database=dis -f  < /var/www/dis/backend

```

> **WARNING**
>
> The last line of the output above is the actual load command which you can copy and paste, adapt to your needs, and the reload the `dis` database.

More about restore

Sys-Admin page - higher-level documentation of backup/restore.

Templates Manager - understanding the interplay of json-templates and SQL database is essential. Read this if you prefer to restore *individual* forms or mDIS tables. You can do this from within the mDIS webpage 'Templates-Manager' itself. You do not need to use the command line.

More about mysql

Mysql commands are case-insensitive, and both double quotes and single quotes are allowed for quoting strings. TBC

mySQL Versions and `sql_mode` used in 2020

The actual SQL commands are commented out with `--` prefixes below:

```
    select @@version
--    5.7.30-0ubuntu0.18.04.1
-- when mDIS runs mySQL version 5.7

-- SELECT @@GLOBAL.sql_mode

ONLY_FULL_GROUP_BY,
STRICT_TRANS_TABLES,
NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
NO_AUTO_CREATE_USER,
NO_ENGINE_SUBSTITUTION

-- when mDIS runs mySQL version 8
 8.0.20-0ubuntu0.20.04.1 |
SELECT @@GLOBAL.sql_mode;

ONLY_FULL_GROUP_BY,
STRICT_TRANS_TABLES,
NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
NO_ENGINE_SUBSTITUTION
```

References

Comparison of relational database management systems

Native Installation on Linux

Detailed Instructions

- Example: Ubuntu 18.04 LTS server
- Several of the following steps are based on the script of Henning Lorenz.
- The installation has to be conducted by a user with "sudo" privileges. The instructions assume the Linux username "administrator". If your username is different, adapt the commands accordingly. (`adduser administrator administrator; usermod -aG sudo administrator`)

Assumptions

- Admin has some Linux experience
- **Suitable** Linux operating system is already installed and configured
- Network is configured, Linux server is accessible over the network
- Instructions below are given for Debian-based Linux Distributions, and their respective package management systems (`apt` , `dpkg` , etc.).

> **"Native" installation in Virtualbox guest**
>
> - If you'd like to practice, perform a "dry-run" of the installation inside a Virtualbox Linux guest, before "going native".
> - If running as a guest in Virtualbox, you should install the Virtualbox guest extensions.
> - Use the Virtualbox menu "Device" to add (virtual) CD with guest extensions
> - Mount CD ROM drive
>
> ```sh
> sudo mkdir /media/cdrom
> sudo mount /dev/cdrom /media/cdrom
> ```
>
> - Run the Guest Extensions Installer
>
> ```sh
> sudo /media/cdrom/VBoxLinuxAdditions.run
> ```

Installing the prerequisites

```sh
sudo apt install dkms build-essential linux-headers-generic linux-headers-$(uname -r)
```

Update package resources

```sh
sudo apt update
```

Install software packages

- MySQL-Server (or MariaDb-Server)
- Apache Web Server
- PHP with extensions:
  - gd
  - xml
  - mysql
  - mbstring
  - zip
  - intl
- git
- composer
- npm
- imagemagick
- ssh (server)

Install some important binaries

```sh
sudo apt install mysql-server apache2 imagemagick ssh git
```

PHP should be php7. In 2019 the current version is PHP 7.3 but your Linux might come with a different version. It should be at least PHP 7.1.

```sh
sudo apt install php php-gd php-xml php-mysql php-mbstring php-zip php-intl composer
```

Install **NodeJS**, the Javascript Runtime.

To do so, you can use `nvm`, the bash script to manage multiple active node.js versions.

```sh
# The script clones the nvm repository to ~/.nvm
# and adds the source line to your profile
# (~/.bash_profile, ~/.zshrc, ~/.profile, or ~/.bashrc).
#
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
```

Open a new shell to have nvm in your PATH.

```sh
# this installs nodejs
nvm ls-remote --lts   #   list only long term support versions
nvm install   --lts 10.16.3    #   currently the most recent lts version
nvm use 10.16.3
```

Alternatively you can use the package manager of your Linux Distribution.

```
apt install nodejs npm
```

Create dis directory and set user as owner

Create a directory for the dis website. This should usually be in /var/www.

```sh
sudo mkdir -p /var/www/dis
# substitute appropriately
sudo chown youruser:youruser /var/www/dis

# only for VM guests
#sudo chown administrator:administrator /var/www/dis

```

> **WARNING**
>
> Running the web server process under an account that has `sudo` privileges **is not secure** and can **attract malicious hackers**. Running this setup might be practical only in your intranet where you know what's going on. Do not connect an mDIS configured like this to the wider Internet!

Install mDIS from the repository server (Credentials required)

Clone the repository from the Gitlab server to the dis website directory.

```sh
# Optional Use the su command to switch to the administrator account.
# su - administrator
# or do sudo -u administrator
git clone https://gitlab.informationsgesellschaft.com/dis/dis.git /var/www/dis
```

Install PHP dependencies and Node modules

Use command-line-tool `composer` to install the **required PHP dependencies** for setting up the backend.

```sh
cd /var/www/dis
composer update composer
```

```
composer update --prefer-dist
composer install
mkdir runtime
composer run-script post-create-project-cmd
```

Use command-line-tool `npm` to install the required Node modules, required to build the frontend Compile the web application.

```sh
npm install
# npm audit fix # optional if know what you are doing
npm run build
```

Create `mysql` database and user

Install mysql as required by the installer. Create mysql user `root`, or **lookup its credentials**.

Create a database named "dis" with character set `utf8mb4` and collation `utf8mb4_unicode_ci` (UTF-8, multibyte character set for full unicode, case insensitive sorting order). Create a user (perhaps named `disuser`, and assign a random password. Allow `disuser` to connect from localhost. You may need to give this user additional permissions if you want to allow database access from *other* hosts (e.g. a backup server) from within your network.

```sh
sudo mysql -u root -e "CREATE DATABASE dis CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;"
PASSWORD=$(echo "$(openssl rand -base64 50)" | tr -d [:space:])
sudo mysql -u root -e "GRANT ALL ON dis.* TO 'disuser'@'localhost' IDENTIFIED BY '${PASSWORD}'
sudo mysql -u root -e "FLUSH PRIVILEGES;"
```

Create yii configuration file to access the database

Create a file `".env"` in the `dis` directory. You can use the file `".env-dev"` as a template. Enter the randomly created password from above:

```sh
# first make sure you are not overwriting your own .env file
# then:
bash -c 'cat > .env' << EOF
MYSQL_ROOT_PASSWORD=
MYSQL_DATABASE=dis
MYSQL_USER=disuser
```

```
MYSQL_PASSWORD=$PASSWORD
MYSQL_HOST=localhost
YII_DEBUG=true
YII_ENV=dev
EXTERNAL_ALLOWED_IP=$(hostname -I) # e.g. 172.21.0.1
EOF
unset PASSWORD
```

Details of these settings can change slightly depending on your computing environment. Use your developer knowledge to set appropriate values.

Initialize database content

Run the yii scripts to create the required tables, install some users and test data.

```
# this command keeps all tables, but might throw some errors
./yii migrate --interactive=0
#
# this variant of the command drops all existing tables.
# (run this if tables have been redesigned substantially):
# ./yii migrate/fresh --interactive=0

# additional tasks
./yii seed/users
./yii seed/example-dump
./yii seed/form-permissions # backend, in *.php files
```

The last 3 `seed/` tasks are defined in `backend/commands/`, e.g. in file `SeedController.php` and in file `backend/modules/api/common/controllers/FormController.php`, method `updateAccessRights()`.

Configure web server

- Setup the configuration files for the dis website.
- Disable the default website
- Enable the apache rewrite module
- Restart the web server

sh

```
sudo bash -c 'cat > /etc/apache2/sites-available/dis.conf' << EOF
<VirtualHost  *:80>

  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/dis/web

  <Directory /var/www/dis/web>
    RewriteEngine on
```

```
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule . index.php
  </Directory>


  ErrorLog \${APACHE_LOG_DIR}/error.log
  CustomLog \${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
EOF

sudo a2dissite 000-default.conf
sudo a2ensite dis.conf
sudo a2enmod rewrite
sudo systemctl reload apache2
```

Run web server under account of `administrator`

Working with the mDIS Templates manager will create new PHP files in directory `backend/`. By default, the Web Server Process will write to this directory, but some other unixusers might as well.

To avoid access right problems after changes in the subdirectories of `backend/`, configure the web server such that it runs under the account of the user "administrator".

On production servers accessible from the internet you should use Access Control Lists (ACLs) instead. ACLs installed on a Router or on a Firewall provide more fine-grained security.

sh

```
sudo sed -i 's/www-data/administrator/g' /etc/apache2/envvars
sudo systemctl reload apache2
```

Developer page

REST API

mDIS has two REST APIs

1. a "regular" REST API for querying and manipulation of science-data, the focus of this article
2. the code-generation API (informally named *cg*-API) for data definition operations (create tables/models, forms, PHP-code, etc). See bottom of page.

Almost all endpoints take query string parameters (or POST data) as arguments. For a listing of the acceptable arguments, click on the respective links in the Controller Documentation **below**.

Implementation Notes

Versioning

The mDIS REST APIs support versioning. The current version is v1.

Security

Authentication is handled via usernames and passwords.

After successful authentication an OAuth2 **Bearer Token** is granted. It is valid for 1 day, but the duration of validity can be refreshed and extended.

API calls to REST API endpoints must submit that Bearer token in the HTTP hader of each request.

HTTPS is not supported though.

Directories

The REST API implementations reside in directories `backend/modules/api/modules/v1/controllers` and `backend/modules/cg/controllers`. Both call classes from the `yii\rest` namespace and from **Gii**, the automatic code generator that comes with the Yii framework. Gii is an optional **extension** of the Yii framework. Gii's role in MDIS is explained **here**. The `api` and `cg` Modules and the `Gii` extension are being cleverly loaded (**"bootstrapped"**) by Yii. See file `backend/config/web.php`.

Happy Paths only

In the following API examples, only successful REST-API calls are documented. Any HTTP Server Error responses will *not* be documented since they are edge cases. The error texts as well as a proper description are provided as a notification. In mDIS forms, this notification appears as a yellow popup box in the upper right corner of the screen. In API calls, notifications and error information can be seen in the browser's Developer Console.

Controllers

API calls are controller method calls. Available mDIS controllers are explained **below**.

The term "Controller" mentioned below and in the sidebar refers to the **Model–View–Controller** idiom of software architecture. A controller class accepts input and converts it to commands for the models (here, database tables) or views (here, Webpages or JSON-data-chunks).

URL Routing

To understand how the REST API works, it is helpful but not required to learn which URL paths the controllers understand, that is, how URL routing mechanisms work inside Yii2. A quick **look** into file `backend/config/web.php`, section `'urlManager' => [...]` helps a lot. Also look into `backend/modules/api/Module.php` and `backend/modules/cg/Module.php` which contain many important URLManager rules.

TBC

Postman collection

There exists an ICDP-internal "testsuite" for REST API calls, a demonstrating selected mDIS REST-API calls mentioned below. The collection is available as a **Postman** collection. It can automate create, inserts, updates, deletes, gets, logins. It also contains many tests and a test runner. This collection is available to all ICDP team members as a Postman Shared Workspace.

Open Endpoints

Open endpoints do not require Authentication.

Login

- **Login** `POST /api/v1/auth/login`

> **Reminder**
>
> At this time, the Login Screen is the *only* endpoint that does not require any authentication.

However its sole purpose is to accept login requests with username and password, and to give out security tokens, enabling the user to proceed to closed endpoints.

Closed Endpoints

Closed Endpoints are endpoints that *do* require authentication

Closed endpoints require a valid Security Token to be included in the header of the request. A Token can be acquired from the Login response above. The token acts like a temporary password. The benefit of the token is that the "real" username and password are submitted *only once*, during login. Later, instead, the token is submitted with each HTTP request.

All endpoints that require authentication get an error response of `Unauthorized`, **HTTP status 401**:

```json
{
  "name": "Unauthorized",
  "message": "Your request was made with invalid credentials.",
  "code": 0,
  "status": 401,
  "type": "yii\\web\\UnauthorizedHttpException"
}
```

However, HTTP status **422 ("Unprocessable Entity")** is also returned often.

A *successful* authentication response looks like this.

```json
{
  "id": 3,
  "username": "knb",
  "email": "knb@gfz-potsdam.de",
  "token": "f5uKee5XVMB6X666eaaugFr_Aw-YCMz4", //temporary password
  "roles": ["sa"],
  "permissions": [
    "form-archive-file:edit",
    "form-archive-file:view",
    "form-cores:edit",
    "form-cores:view",
    "form-init-gas:edit",
    "form-init-gas:view",
    "form-section:edit",
    "form-section:view"
  ]
}
```

Note that an authentication token was provided, which is valid until it expires, or until the user logs out. Expiration date can be extended by "refreshing" any mDIS webpage.

Auth Controller

- **Login** `POST /api/v1/auth/login` - see above
- **Logout** `POST /api/v1/auth/logout` - returns `1` if successful (= Security Token was still valid)
- **Refresh** `GET /api/v1/auth/refresh`

App Controller

This lists the forms created by mDIS admins. The result of this call is used to construct the listing in the sidebar of the mDIS dashboard.

- **Forms** `GET /api/v1/app/forms`

There are no other API endpoints to change the forms-listing. However, another useful URI to know is `/cg/api/summary`, which returns a complex object with keys `modules, models, forms` which returns read-only metadata about tablesets (internally named *modules*), models, and forms.

Global Controller (CRUD controller)

The endpoints under `global` are used to perform CRUD operations (Create, Retrieve, Update, Delete, Defaults, ..etc) for data models under directory `/backend/models`. Mainly to manage `ArchiveFiles` and `ListValues` data models, and also to get data from all columns of the datatables. This is in contrast to the forms controller which only returns the subsets that end users users can actually see with that form.

- **Index** `GET /api/v1/global`
- **Create** `POST /api/v1/global`
- **Update** `PUT /api/v1/global`
- **Delete** `DELETE /api/v1/global`
- **Duplicate** `POST /api/v1/global/duplicate`
- **Defaults** `POST /api/v1/global/defaults`
- **Filter Lists** `GET /api/v1/global/filter-lists`
- **Reports** `GET /api/v1/global/reports`

Form Controller

Form is a specialization of `global`. The endpoints under `form` are used to manage (create, update, delete, defaults, ...etc) data models under directory `/backend/forms` which inherit from the data models in directory `/backend/models`. They usually only return a subset of the data.

- **Index** `GET /api/v1/form`
- **Create** `POST /api/v1/form`

- **Update** `PUT /api/v1/form`
- **Delete** `DELETE /api/v1/form`
- **Duplicate** `POST /api/v1/form/duplicate`
- **Defaults** `POST /api/v1/form/defaults`
- **Filter Lists** `GET /api/v1/form/filter-lists`
- **Reports** `GET /api/v1/form/reports`

List Values Controller

List Values is also a specialization of `global` i.e. it contains all the actions in `global` (except `filter-list` & `reports`) and the following extra ones:

- **List Names** `GET /api/v1/list-values/list-names` - Names of Select-Lists
- **List Values** `GET /api/v1/list-values` - Values for each Select-List

File Controller

Used to manage uploaded files, and to connect or attach them to existing datasets.

- **Index** `GET /api/v1/file`
- **Assign** `POST /api/v1/file/assign`
- **Delete** `POST /api/v1/file/delete`
- **Upload** `POST /api/v1/file/upload`
- **MetaData** `GET /api/v1/file/meta-data/<name>`
- **Unassign** `PUT /api/v1/file/unassign/<fileId>`

Code Examples for using these REST API calls with *curl* can be found in some **mDIS-installer scripts** for customizing the mDIS dashboard.

TBC Widgets Controller

Dashboard widgets controller, manages visibility, alignment, cloneability.

The code examples mentioned in the previous paragraph demonstrate some calls.

TBC

TBC Importer Controller

CSV file controller, ValueList controller.

TBC

TBC Posts Controller

TBC - Dashboard Posts controller

Missing REST API Endpoints

These items do not have a REST API endpoint, or documentation is missing:

- Reports. There is no ReportsController. Reports can be listed via the **Form Controller** or the **Global Controller** though. However new reports cannot be *created* via a REST API.
- Everything covered by the cg-API (see **below**)

TBC

Physical Implementation

Filesystem Locations

The most important PHP file governing API functionality is `backend/modules/api/Module.php`. The `bootstrap()` Method in this file decides which REST endpoints are available, and which standard HTTP methods (GET/PUT/DELETE,...) are supported.

The PHP bases classes actually implementing the endpoints and methods supported by the API are stored in `backend/modules/api/common/controllers`, whereas the classes in `backend/modules/api/modules/v1/controllers` are simple wrapper-classes that only `extend` the base classes (without overriding anything).

PHP Third-Party Libraries

Authentication and all security matters are handled by the Yii framework.

See `vendor/yiisoft/yii2/filters/auth/HttpBearerAuth` and `vendor/yiisoft/yii2/filters/VerbFilter` for details. The Sysadmin page contains a big-picture **list** of third-party libraries. (Look for "dektrium").

TBC

Tutorials

The following tutorials show how the mDIS can be accessed without a Webbrowser, without using the preconfigured data input forms. Following along these tutorials is interesting for developers.

All tutorials perform a simple task: logging in and fetching data. No tutorial performs a more complex action (e.g. smart-copying an existing dataset) at this time.

Javascript

There is a **tutorial** for modern browsers. It's actually a set of tutorials.

R

R tutorial: data access and some simple analysis.

bash

bash script: for quick and dirty command-line processing on Linux.

Powershell

Powershell 6 script: for quick and dirty command-line processing on Windows. Here, csv file export.

More languages coming soon

More tutorials for Python, Perl, Go, PHP...

cg API

cg = Code-Generation

The *cg*-API is designed for internal use during interactive work with the Templates Manager. Endpoints exposed by *cg* require privileged access with the *developer role* or the *sa role*. Therefore they are not accessible to external users, and they are more complicated to use. They are not explained here. Still, check out the following paragraph.

TBC

Useful URLs / endpoints

- `/cg/api/summary` , which returns a complex object with keys `modules, models, forms` which returns read-only metadata about tablesets (internally named *modules*), models, and forms
- `/cg/api/verify-uploaded-template` - imports a table-definition `.json` file. Prerequisite to create a new model.
- `/cg/api/delete-model?name=...` - deletes an empty table from the database, and a corresponding .json file from `backend/dis_templates/models/`

Knowledge about **Gii**, the Yii2 extension for code generation, is recommended.

TBC

mDIS Developer pages

Save this code block as Save-CsvAsExcel.ps1 .

```
# source: https://github.com/gangstanthony/PowerShell/blob/master/Save-CSVasExcel.ps1
# knb 20190807
function Save-CSVasExcel {
    param (
        [string]$CSVFile = $(Throw 'No file provided.')
    )

    BEGIN {
        function Resolve-FullPath ([string]$Path) {
            if ( -not ([System.IO.Path]::IsPathRooted($Path)) ) {
                # $Path = Join-Path (Get-Location) $Path
                $Path = "$PWD\$Path"
            }
            [IO.Path]::GetFullPath($Path)
        }

        function Release-Ref ($ref) {
            ([System.Runtime.InteropServices.Marshal]::ReleaseComObject([System.__ComObject]$r
            [System.GC]::Collect()
            [System.GC]::WaitForPendingFinalizers()
        }

        $CSVFile = Resolve-FullPath $CSVFile
        $xl = New-Object -ComObject Excel.Application
    }

    PROCESS {
        $wb = $xl.workbooks.open($CSVFile)
        $xlOut = $CSVFile -replace '\.csv$', '.xlsx'

        # can comment out this part if you don't care to have the columns autosized
        $ws = $wb.Worksheets.Item(1)
        $range = $ws.UsedRange
        [void]$range.EntireColumn.Autofit()

        $num = 1
        $dir = Split-Path $xlOut
        $base = $(Split-Path $xlOut -Leaf) -replace '\.xlsx$'
        $nextname = $xlOut
        while (Test-Path $nextname) {
            $nextname = Join-Path $dir $($base + "-$num" + '.xlsx')
            $num++
```

```
        }

        $wb.SaveAs($nextname, 51)
    }

    END {
        $xl.Quit()

        $null = $ws, $wb, $xl | % {Release-Ref $_}

        # del $CSVFile
    }
}
```

Forms

Get infos about available forms. The infos are generated based on the existing vue files in /src/forms/. Only forms visible to the current user are returned.

URL : `/api/v1/app/forms/`

URL Parameters : None

Method : `GET`

Auth required : YES

Permissions required : None

Data constraints: None

Data example : None

Success Response

Code : `200 OK`

Content example

```json
{
    "Core": [
        {
            "key": "cores",
            "label": "Cores",
            "Component": "CoresForm.vue",
            "dataModel": "CoresForm",
            "module": "Core"
        },
        {
            "key": "init-gas",
            "label": "Init-gas",
            "Component": "InitGasForm.vue",
            "dataModel": "InitGasForm",
            "module": "Core"
        },
        {
            "key": "section",
            "label": "Section",
```

```
            "Component": "SectionForm.vue",
            "dataModel": "SectionForm",
            "module": "Core"
        }
    ]
}
```

Login

Used to collect a Token for a registered User.

URL : `/api/v1/auth/login`

Method : `POST`

Auth required : NO

Data constraints

```json
{
    "username": "[valid email address or username]",
    "password": "[password in plain text]"
}
```

Data example

```json
{
    "username": "administrator",
    "password": "abcd1234"
}
```

Success Response

Code : `200 OK`

Content example

```json
{
  "id": 1,
  "username": "administrator",
  "email": "alali@domain.com",
  "token": "Mq9WH5pgvLOpdwZi5BXCpkms_n5-M742",
  "roles": [
    "sa"
  ],
  "permissions": [
    "form-archive-file:edit",
    "form-archive-file:view",
```

```
    "form-core-samples:edit",
    "form-core-samples:view",
    "form-cores:edit",
    "form-cores:view",
    "form-hole:edit",
    "form-hole:view",
    "form-init-gas:edit",
    "form-init-gas:view",
    "form-init-temp:edit",
    "form-init-temp:view",
    "form-project-site-1:edit",
    "form-project-site-1:view",
    "form-section:edit",
    "form-section:view"
  ]
}
```

Error Response

Condition : If 'username' and 'password' combination is wrong.

Code : `422 Unprocessable entity`

Content :

```json
[
  {
    "field": "password",
    "message": "Login data is invalid"
  }
]
```

Logout

Used to collect a Token for a registered User.

URL : `/api/v1/auth/logout`

Method : `POST`

Auth required : YES

Permissions required : None

Data constraints : None

Data example : None

Success Response

Code : `200 OK`

Content example None

Refresh

Get user object information to mainly get updated permissions

URL : `/api/v1/auth/refresh`

URL Parameters:

Method : `GET`

Auth required : YES

Permissions required : None

Data constraints : None

Data example: None

Success Response

Code : `200 OK`

Content example

Same as Login response

```json
{
    "id": 1,
    "username": "administrator",
    "email": "alali@domain.com",
    "token": "NSdZMa2joGm8BdepRFsiPJUYU6klExqZ",
    "roles": [
        "sa"
    ],
    "permissions": [
        "form-cores:edit",
        "form-cores:view",
        "form-init-gas:edit",
        "form-init-gas:view",
        "form-section:edit",
        "form-section:view"
    ]
}
```

Assign

Used to assign one or more file to an expedition, site, hole, core or a section

URL : `/api/v1/file/assign`

URL Parameters : None

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`

Data constraints

see `rules()` in class `/backend/module/apu/common/models/FilesUploadedFormModel.php`

- `actionSave` must be set to true

Data example

```json
{
  "expeditionID": 1,
  "siteID": 41,
  "holeID": 205,
  "coreID": null,
  "sectionID": null,
  "fileType": "CT",
  "number": null,
  "remarks": "Some notes about the file assignment",
  "selectedFilenames": [
    "I29.BW_5063_1_A_1.jpg"
  ],
  "actionSave": true
}
```

Success Response

Code : `200 OK`

Content example

The number of assigned files.

```json
1
```

Delete

Used to delete one or more file in the upload directory.

URL : `/api/v1/file/delete`

URL Parameters : None

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`

Data constraints

- `selectedFilenames` must be an array of a valid file names in the upload directory.
- `actionDelete` must be set to true

Data example

```json
{
  "selectedFilenames": [
    "I31.BW_5063_1_A_1.sql",
    "I30.BW_5063_1_A_1.txt"
  ],
  "actionDelete": true
}
```

Success Response

Code : `204 No Content`

Content example : None

Index

Used to get the list of the files in the upload directory.

URL : `/api/v1/file`

URL Parameters :

- per-page=[integer] : number of records in page
- page=[integer] : page number
- sort=[string] : the name of the field used to sort result (prefix with `-` for desc sort)

Method : `GET`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator` , `viewer`

Data constraints : None

Data example : None

Success Response

Code : `200 OK`

Content example

> GET /api/v1/file?per-page=5&page=1&sort=id

```json
{
    "items": [
        {
            "name": "I29.BW_5063_1_A_1.jpg",
            "size": 25331,
            "modified": "2017-09-18 15:55:10",
            "mime": "image/jpeg"
        },
        {
            "name": "I30.BW_5063_1_A_1.txt",
            "size": 597,
            "modified": "2017-11-01 09:41:58",
```

```
            "mime": "text/plain"
        },
        {
            "name": "I31.BW_5063_1_A_1.sql",
            "size": 200825,
            "modified": "2018-09-19 13:11:23",
            "mime": "text/plain"
        },
        {
            "name": "ImportCores.csv",
            "size": 36217,
            "modified": "2019-04-17 14:42:54",
            "mime": "text/plain"
        },
        {
            "name": "UN_5063_1_A_2017-03-23_1.jpg",
            "size": 83853,
            "modified": "2019-03-21 14:27:44",
            "mime": "image/jpeg"
        }
    ],
    "_links": {
        "self": {
            "href": "http://localhost:8000/api/v1/file?page=1"
        },
        "next": {
            "href": "http://localhost:8000/api/v1/file?page=2"
        },
        "last": {
            "href": "http://localhost:8000/api/v1/file?page=3"
        }
    },
    "_meta": {
        "totalCount": 11,
        "pageCount": 3,
        "currentPage": 1,
        "perPage": 5
    }
}
```

Meta Data

Used to get the meta data of a file in the upload directory

URL : `/api/v1/file/meta-data/<name>`

URL Parameters :

- name=[string] : a valid file name that exists in the upload directory.

Method : `GET`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator` , `viewer`

Data constraints : None

Data example : None

Success Response

Code : `200 OK`

Content example

> GET /api/v1/file/meta-data/UN_5063_1_A_2017-03-23_1.jpg

```json
{
    "FILE": {
        "FileName": "UN_5063_1_A_2017-03-23_1.jpg","FileSize":83853,"FileDateTime":"2019-03-21
    },
    "thumbnail":"......AASUVORK5CYII="
}
```

Error Response

Condition : If 'core_type' is empty.

Code : `422 Unprocessable entity`

Content :

json

Unassign

Used to unassign a file from an entity.

URL : `/api/v1/file/unassign/<fileId>`

URL Parameters :

- fileId=[integer] : a valid file id

Method : `PUT`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`

Data constraints: None

Data example : None

Success Response

Code : `200 OK`

Content example

> PUT /api/v1/file/unassign/7

```json
{
  "id": 7,
  "parent_combined_id": "5063_1_A_1",
  "type": null,
  "number": null,
  "filename": "CT_5063_1_A_1.F7.jpg",
  "original_filename": "UN_5063_1_A_2017-04-19_1.jpg",
  "filesize": 56300,
  "mime_type": "image/jpeg",
  "checksum": "e1abd5eb9bf664c1e2e9c98dce29e220",
  "upload_date": "2019-03-21 02:27:45",
  "expedition_id": 1,
  "site_id": 41,
  "hole_id": 205,
```

```
  "core_id": 38,
  "section_id": null,
  "remarks": "Some notes about the file assignment",
  "metadata": "FILE:\n- FileName. UN_5063_1_A_2017-04-19_1.jpg\n- FileSize. 56300\n- FileDateT
}
```

Upload

Used to upload a file to the upload directory

URL : `/api/v1/file/upload`

URL Parameters : None

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`

Data constraints

- `FilesUploadNewFormModel[files][0]` : (binary)
- `FilesUploadNewFormModel` : true

Success Response

Code : `200 OK`

Content example The uploaded file model is returned

```json
{
    "name":"authors_2.txt",
    "size":597,
    "modified":"2019-04-30 12:32:44",
    "mime":"text/plain"
}
```

Create

Used to create a new record of a data model.

URL : `/api/v1/form/`

URL Parameters :

- name=[string] : a valid form data model name (the same name which was defined in the template)

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Data constraints Depends on the data model. These constraints are defined in the data model class usually under `rules()` . Check **Yii2 - Validating Input**

Data example

Depends on the data model structure. An object that represents a table row (Model Instance) should be sent as pairs of the field name and corresponding values.

> CoreCore item create example `POST http://localhost:8080/api/v1/form?name=cores`

```json
{
  "rqd_abundance": 0,
  "rqd_intensity": 0,
  "hole_id": 205,
  "core": 278,
  "site_id": 41,
  "expedition_id": 1,
  "program_id": 1,
  "analyst": "NB",
  "core_catcher": 1,
  "core_type": "H",
  "drilled_length": 12,
  "last_section": 2,
  "top_depth": 12,
  "bottom_depth": 24,
```

```
  "core_ondeck": "2019-04-24 10:37"
}
```

Success Response

Code : `201 Created`

Content example

```json
{
  "rqd_abundance": 0,
  "rqd_intensity": 0,
  "hole_id": 205,
  "core": 278,
  "analyst": "NB",
  "core_catcher": 1,
  "core_type": "H",
  "drilled_length": 12,
  "last_section": 2,
  "top_depth": 12,
  "bottom_depth": 24,
  "core_ondeck": "2019-04-24 10:37:00",
  "combined_id": "5063_1_A_278",
  "core_recovery_pc": 0,
  "id": 335,
  "igsn": "ICDP5063ECB9001",
  "site_id": 41,
  "expedition_id": 1,
  "program_id": 1,
  "archive_files": {
    "filter": {
      "expedition": 1,
      "site": 41,
      "hole": 205,
      "core": 335
    },
    "files": []
  }
}
```

Error Response

Condition : If 'core_type' is empty.

Code : `422 Unprocessable entity`

Content :

```json
[
    {
        "field": "core_type",
        "message": "core_type cannot be blank."
    }
]
```

Defaults

Used to get the defaults values of a form data model.

URL : `/api/v1/form/defaults/`

URL Parameters :

- name=[string] : a valid form data model name

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Data example

```json
{
    "hole_id":205
}
```

Success Response

Code : `200 OK`

```json
{
  "rqd_abundance": 0,
  "rqd_intensity": 0,
  "hole_id": 205,
  "core": 278,
  "site_id": 41,
  "expedition_id": 1,
  "program_id": 1,
  "archive_files": {
    "filter": {
      "expedition": 1,
      "site": 41,
      "hole": 205,
      "core": null
```

```
    },
    "files": []
  }
}
```

Error Response

Condition : If a required parent id was not set

Code : `400 Bad Request`

Content :

```json
{
    "name": "Bad Request",
    "message": "required parameter hole_id is not sent.",
    "code": 0,
    "status": 400,
    "type": "yii\\web\\BadRequestHttpException"
}
```

Delete

Used to delete a record of a data model

URL : `/api/v1/form/<id:integer>`

URL Parameters :

- name=[string] : a valid form data model name

Method : `DELETE`

Auth required : YES

Permissions required :

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Data constraints : None

Data example : None

Success Response

Code : `204 No Content`

Content example

```
```

Error Response

Condition :

Code : ``

Content :

```
1
```

Duplicate

Used to duplicate a record taking into consideration the values that should not be copied like `igsn` , auto increments and `combined_id`

URL : `/api/v1/form/duplicate/`

URL Parameters :

- name=[string] : a valid data model name
- id=[integer] : a valid id of the record you want to duplicate

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Success Response

> POST /api/v1/form/duplicate?name=cores&id=335

Code : `200 OK`

json


```
{
  "rqd_abundance": 0,
  "rqd_intensity": 0,
  "hole_id": 205,
  "core": 279,
  "analyst": "NB",
  "bottom_depth": 24,
  "continuity": null,
  "core_catcher": 1,
  "core_diameter": null,
  "core_loss_reason": null,
  "core_ondeck": "2019-04-24 10:37:00",
  "core_recovery": null,
  "core_recovery_pc": 0,
  "core_type": "H",
  "drilled_length": 12,
```

```
  "last_section": 2,
  "mcd_offset": null,
  "oriented": null,
  "remarks": null,
  "temperature": null,
  "top_depth": 12,
  "site_id": 41,
  "expedition_id": 1,
  "program_id": 1,
  "archive_files": {
    "filter": {
      "expedition": 1,
      "site": 41,
      "hole": 205,
      "core": null
    },
    "files": []
  }
}
```

Filter Lists

Used to get the main filter components' lists

URL : `/api/v1/form/filter-lists/`

URL Parameters :

- name=[string] : a valid form data model name

Method : `GET`

Auth required : YES

Permissions required : None

Data constraints : None

Data example : None

Success Response

Code : `200 OK`

Content example

Return a list for every filter component of the current data model. each filter component is one of the parent data models that belongs to the main hierarchy

```json
{
  "expedition": [
    {
      "value": 1,
      "text": "DSEIS"
    }
  ],
  "site": [
    {
      "value": 41,
      "text": 1,
      "expedition_id": 1
    },
    {
      "value": 56,
```

```
      "text": 7,
      "expedition_id": 1
    }
  ],
  "hole": [
    {
      "value": 205,
      "text": "A",
      "site_id": 41
    },
    {
      "value": 220,
      "text": "B",
      "site_id": 41
    },
    {
      "value": 221,
      "text": "A",
      "site_id": 56
    },
    {
      "value": 228,
      "text": "C",
      "site_id": 41
    },
    {
      "value": 229,
      "text": "D",
      "site_id": 41
    }
  ]
}
```

Index (List records)

Used to get a filtered list of a form data model

URL : `/api/v1/form`

URL Parameters :

- name=[string] : a valid form data model name (the same name which was defined in the template)
- per-page=[integer] : number of records in page
- page=[integer] : page number
- sort=[string] : the name of the field used to sort result (prefix with `-` for desc sort)
- filter=[array] : fields values to used for filter e.g. `filter[expedition_id]=1&filter[site_id]=4`

Method : `GET`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator` , `viewer`
- Permissions: `form-<data model name>:edit` , `form-<data model name>:view`

Data constraints : None

Data example : None

Success Response

> GET http://localhost:8080/api/v1/form?name=cores&per-page=3

Code : `200 OK`

Content example

```json
{
    "items": [
        {
            "rqd_abundance": 32,
            "rqd_intensity": 1,
            "id": 38,
            "hole_id": 205,
            "core": 1,
```

```
            "combined_id": "5063_1_A_1",
            "analyst": "NB",
            "bottom_depth": 3.1,
            "continuity": null,
            "core_catcher": 1,
            "core_diameter": 67,
            "core_loss_reason": "DRILL",
            "core_ondeck": "2017-06-14 09:00:00",
            "core_recovery": 2.9,
            "core_recovery_pc": 93.548387096774,
            "core_type": "R",
            "drilled_length": 3.1,
            "igsn": "ICDP5063EC81001",
            "last_section": 2,
            "mcd_offset": 0,
            "oriented": 0,
            "remarks": "Simple tube barrel used to drill the casing. Fractures of excavation d
            "temperature": null,
            "top_depth": 0,
            "site_id": 41,
            "expedition_id": 1,
            "program_id": 1,
            "archive_files": {
                "filter": {
                    "expedition": 1,
                    "site": 41,
                    "hole": 205,
                    "core": 38
                },
                "files": [
                    {
                        "id": 1,
                        "parent_combined_id": "5063_1_A_1",
                        "type": "BA",
                        "number": null,
                        "filename": "BA_5063_1_A_1.F1.jpg",
                        "original_filename": "BW_5063_1_A_2_BW_2.jpg",
                        "filesize": 25386,
                        "mime_type": "image/jpeg",
                        "checksum": "0e0599831ef80b92a88d25a179c220ba",
                        "upload_date": "2017-09-18 03:55:12",
                        "expedition_id": 1,
                        "site_id": 41,
                        "hole_id": 205,
                        "core_id": 38,
                        "section_id": null,
                        "remarks": null,
                        "metadata": "FILE:\n- FileName. BW_5063_1_A_2_BW_2.jpg\n- FileSize. 25
```

```
                }
            ]
        }
    },
    {
        "rqd_abundance": 25,
        "rqd_intensity": 1,
        "id": 39,
        "hole_id": 205,
        "core": 2,
        "combined_id": "5063_1_A_2",
        "analyst": "NB",
        "bottom_depth": 6,
        "continuity": null,
        "core_catcher": 1,
        "core_diameter": 67,
        "core_loss_reason": "BROKEN",
        "core_ondeck": "2019-03-31 22:00:00",
        "core_recovery": 2.7,
        "core_recovery_pc": 90,
        "core_type": "R",
        "drilled_length": 3,
        "igsn": "ICDP5063EC91001",
        "last_section": 2,
        "mcd_offset": 0,
        "oriented": 0,
        "remarks": "Simple tube barrel used to drill the casing. Fractures of excavation d
        "temperature": null,
        "top_depth": 3,
        "site_id": 41,
        "expedition_id": 1,
        "program_id": 1,
        "archive_files": {
            "filter": {
                "expedition": 1,
                "site": 41,
                "hole": 205,
                "core": 39
            },
            "files": []
        }
    },
    {
        "rqd_abundance": 95,
        "rqd_intensity": 4,
        "id": 40,
        "hole_id": 205,
        "core": 3,
```

```
            "combined_id": "5063_1_A_3",
            "analyst": "NB",
            "bottom_depth": 9,
            "continuity": null,
            "core_catcher": 1,
            "core_diameter": 49,
            "core_loss_reason": null,
            "core_ondeck": "2017-06-15 08:00:00",
            "core_recovery": 3,
            "core_recovery_pc": 100,
            "core_type": "R",
            "drilled_length": 3,
            "igsn": "ICDP5063ECC1001",
            "last_section": 1,
            "mcd_offset": 0,
            "oriented": 1,
            "remarks": "Core drilled with double core barrel",
            "temperature": null,
            "top_depth": 6,
            "site_id": 41,
            "expedition_id": 1,
            "program_id": 1,
            "archive_files": {
                "filter": {
                    "expedition": 1,
                    "site": 41,
                    "hole": 205,
                    "core": 40
                },
                "files": []
            }
        }
    ],
    "_links": {
        "self": {
            "href": "http://localhost:8000/api/v1/form?name=cores&per-page=3&page=1"
        },
        "next": {
            "href": "http://localhost:8000/api/v1/form?name=cores&per-page=3&page=2"
        },
        "last": {
            "href": "http://localhost:8000/api/v1/form?name=cores&per-page=3&page=98"
        }
    },
    "_meta": {
        "totalCount": 292,
        "pageCount": 98,
        "currentPage": 1,
```

```
153             "perPage": 3
154         }
155     }
```

Reports

Used to get the available reports of a form.

URL : `/api/v1/form/reports/`

URL Parameters :

- name=[string] : a valid form data model name

Method : `GET`

Auth required : YES

Permissions required :

- Roles: `sa` , `developer` , `operator` , `viewer`
- Permissions: `form-<data model name>:edit` , `form-<data model name>:view`

Data constraints : None

Data example : None

Success Response

> GET /api/v1/form/reports?name=cores

```json
{
    "single": [
        {
            "name": "Details",
            "title": "Details of record",
            "model": ".*",
            "showInMenu": true,
            "singleRecord": true
        }
    ],
    "multiple": [
        {
            "name": "ListAllCsv",
            "title": "Export full records as CSV file",
            "model": ".*",
            "showInMenu": true,
            "singleRecord": false
```

```
        },
        {
            "name": "ListCsv",
            "title": "Export records as CSV file",
            "model": ".*",
            "showInMenu": true,
            "singleRecord": false
        },
        {
            "name": "List",
            "title": "List of records",
            "model": ".*",
            "showInMenu": true,
            "singleRecord": false
        }
    ]
}
```

Update

Used to update an existing record of a data model

URL : `/api/v1/form/<id:integer>`

URL Parameters :

- name=[string] : data model name
- id=[integer] : id of an existed data model record

Method : `PUT`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Data constraints Depends on the data model. These constraints are defined in the data model class usually under `rules()` . Check **Yii2 - Validating Input**

Data example

Depends on the data model structure. An object that represents a table row (Model Instance) should be sent as pairs of the field name and corresponding values.

> ListValue item update example

```json
{
  "core_catcher": 1
}
```

Success Response

> PUT /api/v1/form/314?name=cores

Code : `200 OK`

Content example The updated record

```json
{
  "rqd_abundance": 0,
  "rqd_intensity": 0,
  "id": 314,
  "hole_id": 205,
  "core": 35,
  "combined_id": "5063_1_A_35",
  "analyst": "MR",
  "bottom_depth": 102,
  "continuity": null,
  "core_catcher": 1,
  "core_diameter": 47.5,
  "core_loss_reason": null,
  "core_ondeck": "2017-10-11 07:00:00",
  "core_recovery": 1.65,
  "core_recovery_pc": 110,
  "core_type": "R",
  "drilled_length": 1.5,
  "igsn": "ICDP5063ECQ8001",
  "last_section": 2,
  "mcd_offset": 0.8,
  "oriented": 0,
  "remarks": null,
  "temperature": null,
  "top_depth": 100.5,
  "site_id": 41,
  "expedition_id": 1,
  "program_id": 1,
  "archive_files": {
    "filter": {
      "expedition": 1,
      "site": 41,
      "hole": 205,
      "core": 314
    },
    "files": []
  }
}
```

Error Response

Condition : If 'core_catcher' is a string.

```json
{
  "core_catcher": "break it"
```

```
}
```

Code: `422 Unprocessable entity`

Content:

```json
[
    {
        "field": "core_catcher",
        "message": "CC must be an integer."
    }
]
```

Create

Used to create a new record of a data model.

URL : `/api/v1/global/`

URL Parameters :

- name=[data model name]

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Data constraints Depends on the data model. These constraints are defined in the data model class usually under `rules()` . Check **Yii2 - Validating Input**

Data example

Depends on the data model structure. An object that represents a table row (Model Instance) should be sent as pairs of the field name and corresponding values.

> ListValue item create example

```json
{
  "listname": "ANALYST",
  "sort": 10,
  "display": "UG",
  "remark": "Underground"
}
```

Success Response

Code : `201 Created`

Content example

```json
{
  "id": 21735,
  "listname": "ANALYST",
  "sort": 10,
  "display": "UG",
  "remark": "Underground"
}
```

Error Response

Condition : If 'display' is empty.

Code : `422 Unprocessable entity`

Content :

```json
[
    {
        "field": "display",
        "message": "Display cannot be blank."
    }
]
```

Defaults

Used to get the defaults values of a data model.

URL : `/api/v1/global/defaults/`

URL Parameters :

- name=[string] : a valid data model name

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Notes

- This endpoint is used only for forms where such functionality is needed.
- check **defaults** under form controller for more real examples

Delete

Used to delete a record of a data model

URL : `/api/v1/global/<id:integer>`

URL Parameters :

- name=[string] : a valid data model name

Method : `DELETE`

Auth required : YES

Permissions required :

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Data constraints : None

Data example : None

Success Response

Code : `204 No Content`

Content example

```
```

Error Response

Condition :

Code : ``

Content :

```
1
```

Notes

- some notes

Duplicate

Used to duplicate a record taking into consideration the values that should not be copied like `igsn` , auto increments and `combined_id`

URL : `/api/v1/global/duplicate/`

URL Parameters :

- name=[string] : a valid data model name
- id=[integer] : a valid id of the record you want to duplicate

Method : `POST`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Notes

- This endpoint is used only for forms where such functionality is needed.
- check **duplicate** under form controller for more real examples

Filter Lists

Used to get the main filter components' lists

URL : `/api/v1/global/filter-lists/`

URL Parameters :

- name=[string] : a valid data model name

Method : `GET`

Auth required : YES

Permissions required : None

Data constraints : None

Data example : None

Success Response

Code : `200 OK`

Content example

Return a list for every filter component of the current data model. each filter component is one of the parent data models that belongs to the main hierarchy

```json
{
  "expedition": [
    {
      "value": 1,
      "text": "DSEIS"
    }
  ],
  "site": [
    {
      "value": 41,
      "text": 1,
      "expedition_id": 1
    },
    {
      "value": 56,
```

```
      "text": 7,
      "expedition_id": 1
    }
  ],
  "hole": [
    {
      "value": 205,
      "text": "A",
      "site_id": 41
    },
    {
      "value": 220,
      "text": "B",
      "site_id": 41
    },
    {
      "value": 221,
      "text": "A",
      "site_id": 56
    },
    {
      "value": 228,
      "text": "C",
      "site_id": 41
    },
    {
      "value": 229,
      "text": "D",
      "site_id": 41
    }
  ]
}
```

Index (List records)

Used to get a filtered list of a data model

URL : `/api/v1/global`

URL Parameters :

- name=[string] : a valid form name
- per-page=[integer] : number of records in page
- page=[integer] : page number
- sort=[string] : the name of the field used to sort result (prefix with `-` for desc sort)
- filter=[array] : fields values to used for filter e.g. `filter[expedition_id]=1&filter[site_id]=4`

Method : `GET`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator` , `viewer`
- Permissions: `form-<data model name>:edit` , `form-<data model name>:view`

Data constraints : None

Data example : None

Success Response

Code : `200 OK`

Content example

```json
{
  "items": [
    {
      "id": 34,
      "parent_combined_id": null,
      "type": "BA",
      "mime_type": "image/jpeg",
      "filename": "I34.BW_5063_1_A_3_BW_1.jpg",
      "original_filename": "BW_5063_1_A_3_BW_1.jpg",
      "filesize": 26574,
```

```
      "checksum": "7f2977066aec5193811e585c47e579eb",
      "upload_date": "2017-11-28 02:56:40",
      "number": null,
      "expedition_id": 1,
      "site_id": 41,
      "hole_id": 205,
      "core_id": 38,
      "section_id": null,
      "remarks": null,
      "metadata": null
    },
    {
      "id": 36,
      "parent_combined_id": null,
      "type": "UN",
      "mime_type": "text/plain",
      "filename": "UN_5063_1_A_1_internalmetadata.F36.csv",
      "original_filename": "internalmetadata.csv",
      "filesize": 149,
      "checksum": "391cccd22287080cdd8b4eee85e0fe65",
      "upload_date": "2019-03-25 11:11:48",
      "number": null,
      "expedition_id": 1,
      "site_id": 41,
      "hole_id": 205,
      "core_id": 38,
      "section_id": null,
      "remarks": null,
      "metadata": null
    },
    {
      "id": 38,
      "parent_combined_id": null,
      "type": "TS",
      "mime_type": "image/tiff",
      "filename": "TS_5063_1_A_1_2.F38.tif",
      "original_filename": "5054-1-A-121Z-3WR.tif",
      "filesize": 53692312,
      "checksum": "9422346f864e5acbf7c0ad4de63248fa",
      "upload_date": "2019-04-04 02:22:44",
      "number": 2,
      "expedition_id": 1,
      "site_id": 41,
      "hole_id": 205,
      "core_id": 38,
      "section_id": null,
      "remarks": null,
      "metadata": null
```

```
    }
  ],
  "_links": {
    "self": {
      "href": "http://dis-app-2018/api/v1/global?name=ArchiveFile&per-page=5&page=1&sort=id&fi
    }
  },
  "_meta": {
    "totalCount": 3,
    "pageCount": 1,
    "currentPage": 1,
    "perPage": 5
  }
}
```

Reports

Used to get the available reports of a form.

URL : `/api/v1/global/reports/`

URL Parameters :

- name=[string] : a valid data model name

Method : `GET`

Auth required : YES

Permissions required :

- Roles: `sa` , `developer` , `operator` , `viewer`
- Permissions: `form-<data model name>:edit` , `form-<data model name>:view`

Data constraints : None

Data example : None

Notes

- This endpoint is used only for forms where such functionality is needed.
- check **reports** under `form-controller` for more real examples

Update

Used to update an existing record of a data model

URL : `/api/v1/global/<id:integer>`

URL Parameters :

- name=[string] : data model name
- id=[integer] : id of an existed data model record

Method : `PUT`

Auth required : YES

Permissions required

- Roles: `sa` , `developer` , `operator`
- Permissions: `form-<data model name>:edit`

Data constraints Depends on the data model. These constraints are defined in the data model class usually under `rules()` . Check **Yii2 - Validating Input**

Data example

Depends on the data model structure. An object that represents a table row (Model Instance) should be sent as pairs of the field name and corresponding values.

> ListValue item update example

```json
{
    "display": "MZZ"
}
```

Success Response

Code : `200 OK`

Content example The updated record

```json
{
    "id": 100,
```

```
    "listname": "ANALYST",
    "display": "MZZ",
    "remark": "Martin Ziegler",
    "sort": 81
}
```

Error Response

Condition : If 'display' is empty.

Code : `422 Unprocessable entity`

Content :

```json
[
    {
        "field": "display",
        "message": "Display cannot be blank."
    }
]
```

Index

The index action (same as in global) could be used to get a list's items

URL : `/api/v1/list-values/`

URL Parameters :

- per-page=[integer] : number of records in page (set to -1 to get all records)
- sort=[string] : the name of the field used to sort result (set to `sort`)
- filter=[array] : fields values to used for filter e.g. `filter[expedition_id]=1&filter[site_id]=4` (set to `filter[listname]=<list name>`)

Method : `GET`

Auth required : YES

Permissions required : None

Data constraints : None

Data example : None

Success Response

> GET /api/v1/list-values?per-page=-1&filter[listname]=ANALYST

Code : `200 OK`

Content example

```json
{
    "items": [
        {
            "id": 92,
            "listname": "ANALYST",
            "display": "BL",
            "remark": "Bennie Liebenberg",
            "sort": 40
        },
        {
            "id": 94,
            "listname": "ANALYST",
```

```
            "display": "GH",
            "remark": "Gerhard Hofmann",
            "sort": 23
        },
        {
            "id": 95,
            "listname": "ANALYST",
            "display": "GVA",
            "remark": "Gerrie van Aswegen",
            "sort": 72
        },
        {
            "id": 96,
            "listname": "ANALYST",
            "display": "HO",
            "remark": "Hiroshi Ogasawara",
            "sort": 59
        },
        {
            "id": 97,
            "listname": "ANALYST",
            "display": "JH",
            "remark": "John Hunt",
            "sort": 25
        },
        {
            "id": 99,
            "listname": "ANALYST",
            "display": "MM",
            "remark": "Musa Manzi",
            "sort": 43
        },
        {
            "id": 93,
            "listname": "ANALYST",
            "display": "NB",
            "remark": "Nicolas Berset",
            "sort": 2
        },
        {
            "id": 101,
            "listname": "ANALYST",
            "display": "NW",
            "remark": "Neta Wechsler",
            "sort": 75
        },
        {
            "id": 106,
```

```
            "listname": "ANALYST",
            "display": "RD",
            "remark": "Raymond Durrheim",
            "sort": 16
        },
        {
            "id": 98,
            "listname": "ANALYST",
            "display": "SB",
            "remark": "Sifiso Bucibo",
            "sort": 7
        },
        {
            "id": 102,
            "listname": "ANALYST",
            "display": "SM",
            "remark": "Sylvester Morema",
            "sort": 48
        },
        {
            "id": 103,
            "listname": "ANALYST",
            "display": "TO",
            "remark": "Tullis Onstott",
            "sort": 62
        },
        {
            "id": 104,
            "listname": "ANALYST",
            "display": "TW",
            "remark": "Tony Ward",
            "sort": 74
        },
        {
            "id": 105,
            "listname": "ANALYST",
            "display": "YY",
            "remark": "Yasuo Yabe",
            "sort": 77
        }
    ],
    "_links": {
        "self": {
            "href": "http://localhost:8000/api/v1/list-values?per-page=-1&filter%5Blistname%5D
        }
    },
    "_meta": {
        "totalCount": 14,
```

```
        "pageCount": 1,
        "currentPage": 1,
        "perPage": -1
    }
}
```

List Names

Used to get the available list names.

URL : `/api/v1/list-values/list-names/`

URL Parameters : None

Method : `GET`

Auth required : YES

Permissions required : None

Data constraints : None

Data example : None

Success Response

Code : `200 OK`

Content example

```json
[
    "ABUNDANCE",
    "ACCESSORIES",
    "ACTIVITY_TYPE",
    "AFFILIATION",
    "ALTERATION",
    "ALTERATION_NAME",
    "AMOUNT",
    "ANALYST",
    "ANNO_CLASS",
    "ANNO_TYPE_COMPONENTS",
    "ANNO_TYPE_SAMPLING",
    "ANNO_TYPE_STRUCTURES",
    "BEDDING",
    "BHA_TYPE",
    "BHM_COMPANY",
    "BIOEROSION_INT",
    "BIOGENETICS",
    "BIOTURBATION",
    "BIOTURBATION_INT",
```

```
    "BIO_REPL",
    "BOUNDARY",
    "BOX_POSITION",
    "BOX_TYPE",
    "CASING_TYPE",
    "CHECKED_BY",
    "CHIP_TYPE",
    "CHL_REPL",
    "CLASS_SYSTEM",
    "CLAST",
    "CLAST_ROUNDING",
    "CLAST_SORTING",
    "CLAST_SPHERICITY",
    "CLAST_TEXTURE",
    "CL_ROUNDING",
    "COLOUR",
    "COMPANY",
    "COMPONENT",
    "COMPONENT_OLD",
    "CONTACTS",
    "CONTACT_BOTTOM",
    "CONTACT_TOP",
    "CORE_CONDITION",
    "CORE_CONTINUITY",
    "CORE_CONTINUITY_NAT_EXTEND",
    "CORE_CONTINUITY_NAT_OTH",
    "CORE_CONTINUITY_NAT_SHEAR",
    "CORE_DEPTH_CCSF",
    "CORE_DEPTH_CSF",
    "CORE_LOSS_REASON",
    "CORE_REMARKS",
    "CORE_TYPE",
    "DEFINITION_STRUCTURE_OODP",
    "DESTRUCTIVE_USE",
    "DISCONTINUITIES",
    "DISTURBANCE",
    "DRILLING_DEPTH_DSF",
    "DRILLING_METHOD",
    "FILE_TYPE",
    "FOLIATION",
    "FOLIATION_TYPE",
    "FOSSILS",
    "FOSSILS_OLD",
    "FOSSIL_OLD",
    "FOSSIL_OLD_FULL",
    "FRACTURES",
    "FRACTURE_FILLING",
    "FRACTURE_WAVINESS",
```

```
    "FRACTURE_WEATHERING",
    "FRACTURING",
    "FRAMEWORK_ASSEMBLAGE",
    "GRAINSIZE",
    "GRAINSIZE1",
    "GRAINSIZE2",
    "GRAIN_ASSEMBLAGE",
    "GROUNDMASS",
    "GSIZE_QFM",
    "HOLE_TYPE",
    "ICHNOFOSSILS",
    "IMAGE_TYPE",
    "LAT_DIR",
    "LINEATION_SLIP",
    "LINEATION_TYPE",
    "LINER_TYPE",
    "LITHOLOGY",
    "LITHOLOGY1",
    "LITH_ACCESSORIES",
    "LOCATION",
    "LOGGED_BY",
    "LON_DIR",
    "MAIN_ACTIVITY",
    "MEASUREMENT",
    "MEASUREMENT_MONTPEILLIER",
    "MEDIA",
    "METHOD",
    "MINERAL",
    "MINERAL_OLD",
    "MUD_ADDITIVE",
    "MUD_FROM",
    "MUD_GAS_TYPE",
    "MUD_SIZE_UNIT",
    "MUD_TYPE",
    "PARAMETER",
    "PARAMETERS",
    "PHENOCRYST",
    "PH_ABUNDANCE",
    "PH_DISTRIBUTION",
    "PH_SIZE",
    "PLATFORM",
    "PROGRAM",
    "PUMP_TYPE",
    "PURPOSE",
    "QUICK_SEARCH",
    "REDEPOSITED",
    "REPOSITORY",
    "REQUEST",
```

```
    "REQUEST_TYPE",
    "ROCK_CLASS",
    "ROCK_COLOR",
    "ROCK_NAME",
    "ROCK_NAME1",
    "ROCK_SYMBOL",
    "ROCK_SYMBOL1",
    "ROCK_TYPE",
    "ROCK_TYPE1",
    "RQD_ABUNDANCE",
    "RQD_INTENSITY",
    "RUN_TYPE",
    "SAMPLED_BY",
    "SAMPLE_SERIES_TYPE",
    "SAMPLE_SERIES_TYPE_OLD",
    "SAMPLE_TYPE",
    "SAMPLE_VOLUME",
    "SECTION_HALF",
    "SECTION_REMARKS",
    "SEQUENCE_BOUNDARY",
    "SHIFTS",
    "STATE",
    "STATUS",
    "STRAT_LEVEL",
    "STRAT_LEVEL1",
    "STRAT_LEVEL2",
    "STRAT_SERIE",
    "STRAT_SERIES",
    "STRAT_SYSTEM",
    "STRENGTH",
    "STRUCTURE",
    "STRUCTURE_MEASUREMENT_OODP",
    "STRUCTURE_OLD",
    "SUBSAMPLE_PRESERVATIVE",
    "SUBSAMPLE_PRES_AMOUNT",
    "SUBSAMPLE_SERIES_TYPE",
    "SUBSAMPLE_STORAGE",
    "SUBSAMPLE_TYPE",
    "SUBSAMPLE_UNIT",
    "SUBSAMPLE_VIAL",
    "SUBSAMPLE_VIAL_MAT",
    "SUBSAMPLE_VOLUME",
    "SUBSECTION_TYPE",
    "SUB_UNIT_TYPE",
    "TECTO_ELEMENT",
    "TEXTURES",
    "THINSECTION",
    "TOOL_TYPE",
```

```
    "UNITS",
    "UNIT_CLASS",
    "UNIT_CLASS_NEW",
    "UNIT_CLASS_ORIG",
    "UNIT_TYPE",
    "UNIT_TYPE_DFDP",
    "UNIT_TYPE_ORG",
    "UPLOAD_FILE_TYPE",
    "VEIN_FILLINGS",
    "VEIN_INTENSITY",
    "VEIN_SHAPE",
    "VESICLES_ABUNDANCE",
    "WD_METHOD",
    "WETNESS",
    "WHAT",
    "WIRELINE_DEPTH_WSF",
    "YES_NO",
    "ZOOM"
]
```

mDIS REST API access with R

Developer page

Build your own reports and data analyses. Using the mDIS REST API, you can get mDIS data independently from any predefined forms and reports. All you need is a current login to mDIS.

This tutorial demonstrates how to get data from mDIS with R, and how to analyze some data from the ICDP DSEIS Project. (To load this dataset, you need to run the `seed/example-dump` Yii migration. You can also work with your own mDIS data, but then some URL parameters, axis labels and plot titles must be changed accordingly).

This tutorial does *not* demonstrate how to *edit* mDIS datasets (create, delete, duplicate, etc). Such API calls are **available**, but this tutorial focuses on the simple case of getting data in a read-only mode.

What the R code does

- login to mDIS
- fetch some data
- convert it to a table (an R data frame)
- simple exploratory analysis

> **TIP**
>
> Download the **RMarkdown File** directly, for usage in RStudio for example.

Necessary R packages

```
library(tidyverse)    # data processing + plotting helpers
library(jsonlite)     # read/write JSON
library(httr)         # give R web browsing capabilities
library(kableExtra)   # nicer tables
theme_set(theme_bw()) # ggplot layout theme
```

Login to mDIS

Get credentials, and URL info, where to get the data from. Assign it to the `logindata` list.

(We read it in as a JSON structure, because we'll deal with such JSON structures later. If you struggle with the following code block, later blocks will be difficult to deal with, too.)

```
# copy JSON structure from other tutorial
logindata <- jsonlite::fromJSON(txt = '{
  "username": "knb",
  "password": "knb",
  "urlhost": "http://34.76.214.94",
  "urlpaths": [
    "/api/v1/auth/login",
    "/api/v1/form?name=cores&per-page=5000&page=0&sort=id&filter[expedition_id]=1&filter[site_
  ]
}')

#logindata$password = Sys.getenv("MDIS_PW")
#jsonlite::toJSON(logindata)
```

You can insert your own password on line 5, or you can set it on line 12 (which is commented out at this time).

To actually connect and fetch data with the `logindata`, we need to create some helper functions first.

Create Functions

Create a function `mdis_api()` - a single function for login and data fetching. This is suboptimal, but will suffice for now.

If the user is not logged in , the `logindata` list will *not* contain a `token` element. Thus we need to **log in** first, with data provided in `logindata`. We will receive a Security token ("Bearer Token"), which we will use as 32-character long "temporary password" from now on.

If the user is logged in, `logindata$token` will have a valid value, e.g. `9nxFYoXoBA7Eb6n1hc-FbwcTqeU5vf3a`. Then we can perform HTTP GET requests, sending the token as an extra HTTP header.

```
# function according to the httr vignette
# returns simple s3 object for easier debugging
# used for login and HTTP-GETting data
mdis_api <- function(logindata, path) {
  url <- modify_url(logindata$urlhost, path = path)
  if(is.null(logindata$token)){
    resp <- POST(url, ua = logindata$ua,
            body = list(username = logindata$username,
                        password = logindata$password))
  } else {
    resp <- GET(url,
                ua = logindata$ua,
                add_headers(Authorization = sprintf("Bearer %s", logindata$token)))
  }
```

```
  if (http_type(resp) != "application/json") {
    stop("API did not return json", call. = FALSE)
  }

  parsed <- jsonlite::fromJSON(content(resp, "text"), simplifyVector = FALSE)

  if (http_error(resp)) {
    stop(
      sprintf(
        "mDIS API request failed [%s]\n%s\n<%s>",
        status_code(resp),
        parsed[[1]],
        resp$url
      ),
      call. = FALSE
    )
  }

  # return simple S3 object, see comment below
  structure(
    list(
      content = parsed,
      path = path,
      response = resp
    ),
    class = "mdis_api"
  )
}

# Rather than simply returning the response as a list,
# I think it's a good practice to make a simple S3 object.
# That way you can return the response and parsed object,
# and provide a nice print method.
# This will make debugging later on much much much more pleasant.

print.mdis_api <- function(x, ...) {
  cat("<mDIS ", x$path, ">\n", sep = "")
  str(x$content)
  invisible(x)
}
```

Actually log in. Get token and add it to `logindata` list.

```
# good practice: to identify yourself as user agent, ua
logindata$ua = user_agent("http://github.com/knbknb")

```

```
loggedin <- mdis_api(logindata, logindata$urlpaths[[1]])

logindata$token <- loggedin$content$token
```

Fetch data

Make **API request** to `/api/v1/form?name=cores&per-page=5000&page=0&sort=id&filter[expedition_id]=1&filter[site_id]=41&filter[hole_id]=205` to get all cores of a certain DSEIS drillhole (site 1, hole A)

```
parsed <- mdis_api(logindata, logindata$urlpaths[[2]])
```

JSON often contains `NULL` values. However in R, we prefer `NA` instead. `NA` values are easier to work with than NULL. Replace NULLs with NAs in list parsed from JSON, and put items in a data frame, `cores_df` .

```
cores <- as.list(parsed)$content$items %>%
  map(function(x) map(x, function(y) ifelse(is.null(y), NA, y)))

cores_df <- map_df(cores, as_tibble, .name_repair = "minimal")

cores_df <- cores_df %>%
  mutate(core_ondeck = as.POSIXct(core_ondeck))
```

The data

What does the table/data frame look like?

We got a 276x28 table from the mDIS REST API.

Column names, data types, some example values:

```
#skimr::skim(cores_df)
glimpse(cores_df)
```

```
## Observations: 276
## Variables: 28
## $ rqd_abundance    <int> 32, 25, 95, 88, 100, 89, 80, 94, 72, 94, 0, 72,…
## $ rqd_intensity    <int> 1, 1, 4, 3, 4, 3, 3, 4, 2, 4, 0, 2, 4, 4, 4, 4,…
## $ id               <int> 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,…
## $ hole_id          <int> 205, 205, 205, 205, 205, 205, 205, 205, 205, 20…
## $ core             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, …
```

```
## $ combined_id      <chr> "5063_1_A_1", "5063_1_A_2", "5063_1_A_3", "5063…
## $ analyst          <chr> "NB", "NB", "NB", "NB", "NB", "NB", "NB", "NB",…
## $ bottom_depth     <dbl> 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39…
## $ continuity       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ core_catcher     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
## $ core_diameter    <dbl> 67, 67, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49,…
## $ core_loss_reason <chr> "DRILL", "BROKEN", NA, NA, NA, NA, NA, NA, NA, …
## $ core_ondeck      <dttm> 2017-06-15 08:00:00, 2017-06-15 08:00:00, 2017…
## $ core_recovery    <dbl> 2.9, 2.7, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.…
## $ core_recovery_pc <dbl> 96.67, 90.00, 100.00, 100.00, 100.00, 100.00, 1…
## $ core_type        <chr> "R", "R", "R", "R", "R", "R", "R", "R", "R", "R…
## $ drilled_length   <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,…
## $ igsn             <chr> "ICDP5063EC81001", "ICDP5063EC91001", "ICDP5063…
## $ last_section     <int> 3, 2, 1, 1, 3, 5, 1, 3, 5, 1, 3, 3, 1, 1, 1, 3,…
## $ mcd_offset       <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, -0.71, -0.71, -0.…
## $ oriented         <int> 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
## $ remarks          <chr> "Simple tube barrel used to drill the casing. F…
## $ temperature      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ top_depth        <dbl> 0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36,…
## $ site_id          <int> 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41,…
## $ expedition_id    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
## $ program_id       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
## $ archive_files    <list> [[1, 41, 205, 38], [1, 41, 205, 39], [1, 41, 2…
```

A table view would be very, very wide so we list only the column definitions above, with some sample values.

Analysis

Do some descriptive statistics with the drillcore data.

Core Diameter Counts

We are studying drillhole 1, site A (ICDP DSEIS project). How "deep" did they drill? (Actually they directionally drilled *upwards* from inside a mine gallery at ~3000 m depth.)

```
cores_df %>%
  group_by(core_diameter) %>%
  summarize(`max_depth (m)`= max(bottom_depth), n_Cores = n())  %>%
  arrange(desc(core_diameter)) %>%
  knitr::kable(caption = "DSEIS Core diameters (mm), Hole A, site 1", col.names = names(.)) %>
  kableExtra::kable_styling(bootstrap_options = c("striped"))
```

DSEIS Core diameters (mm), Hole A, site 1

core_diameter	max_depth (m)	n_Cores

core_diameter	max_depth (m)	n_Cores
67.0	6.0	2
49.0	773.6	256
47.5	824.6	18

Who are the most active analysts/geologists?

```
cores_df %>%
  count(analyst, sort =TRUE) %>%
  knitr::kable(caption = "Initials of active curators in the DSEIS project", col.names = names
  kableExtra::kable_styling(bootstrap_options = c("striped"))
```

Initials of active curators in the DSEIS project

analyst	n
NB	254
MR	20
HO	1
MZ	1

Actually one geologist, NB, did almost all the work.

Core loss

Tabulating core loss

```
table(as.Date(cores_df$core_ondeck), cores_df$core_loss_reason) %>%
  kable(caption = "When were DSEIS cores lost? How many, and why?") %>%
  kableExtra::kable_styling(bootstrap_options = c("striped"))
```

When were DSEIS cores lost? How many, and why?

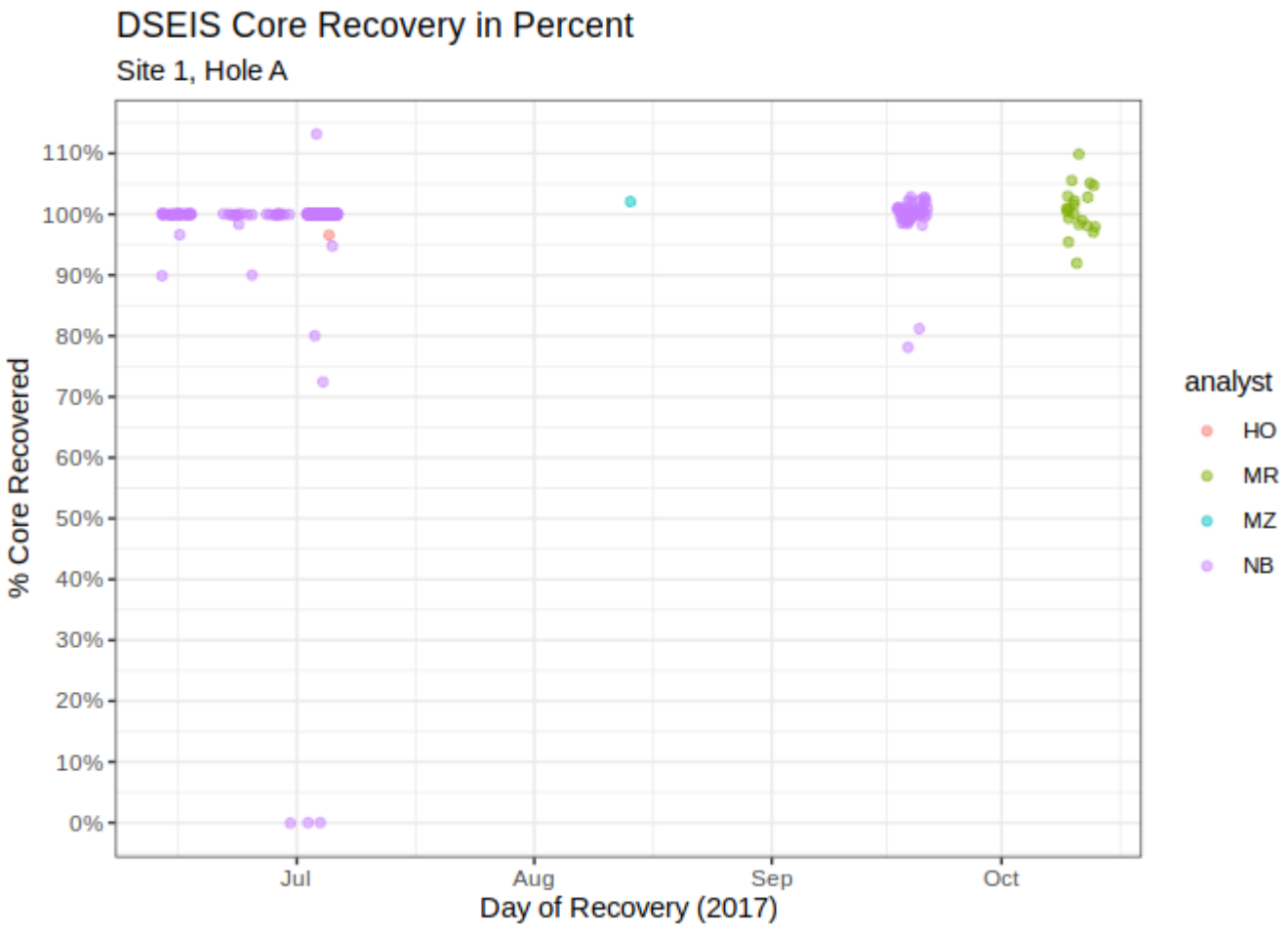
	BROKEN	CHIPS	CONTAM	DISC	DRILL
2017-06-15	1	0	0	0	1
2017-06-23	0	0	0	1	0
2017-06-28	0	0	0	0	0
2017-07-04	1	1	0	0	0

	BROKEN	CHIPS	CONTAM	DISC	DRILL
2017-08-15	0	0	1	0	0
2017-09-19	0	0	0	0	0
2017-10-11	0	0	0	0	0

Core recovery in percent

See figure.

```
cores_df %>% ggplot(aes(core_ondeck, core_recovery_pc/100, color=analyst)) +
  geom_jitter(alpha = 0.5) +

  labs(title = "DSEIS Core Recovery in Percent",
       subtitle = "Site 1, Hole A",
       y = "% Core Recovered",
       x = "Day of Recovery (2017)") +
  scale_y_continuous(labels = scales::percent_format(accuracy = NULL),  breaks = seq(0, 1.2, 0
```



Values larger than 100% are possibly due to solid core material broken off a bit below the core barrel. Sometimes cores are retrieved with a bit of extra rock sticking out of the pipe. Alternatively it might be elastic material (mud) decompressing and expanding in the liner.

Many more analyses are now possible. These were just illustrations.

REST API

Documentation

See **REST API** page for extensive documentation.

Shell Scripting

Developer page

bash Example

Accessing the mDIS REST API with bash

This example uses Unix command line tools:

- `curl` to perform web requests, and
- `jq` to parse the JSON response from mDIS.

This example uses an mDIS loaded with data from the GRIND project.

Instructions

Put the code block below into a shell script, make it executable.

sh

```sh
#!/bin/bash
## get data from mDIS
## knb Aug 2019, May 2020
if [[ -z "$user1password" ]]; then
echo "\$user1password env var is empty!"
echo "change the script code, or type this:"
echo "export user1password=user1password"
return
fi

host="http://34.76.214.94" # or other
url="$host/api/v1/auth/login"
payload="{\"username\":\"user1\",\"password\":\"$user1password\"}"

## perform curl request to get bearer token
json=$(curl -s -H "Content-Type:application/json" \
    --data $payload $url)
token=$(echo $json | jq -r .token)
#echo -n $token

# perform curl request to fetch JSON data.
# here we use some hardcoded IDs for simplicity.
url2="$host/api/v1/form?name=core&per-page=5000&page=0&sort=id&filter[expedition_id]=2&filter[
json2=$(curl --globoff -s \
```

```
-H 'Accept: application/json, text/plain, */*'\
-H "Content-Type:application/json" -H "Authorization: Bearer $token" $url2)

# ## process JSON data with jq
echo "$json2" | jq '.items[0]'
```

To set a password, run this command in a shell, before running this script above:

`export user1password="..."` . Of course you can put in a different username/password, or change the source code of the script.

The run the shell script.

Expected output with valid credentials:

```json
{
  "items": [],
  "_links": {
    "self": {
      "href": "http://34.76.214.94/api/v1/form?name=cores&per-page=5000&page=1&sort=id&filter%
    }
  },
  "_meta": {
    "totalCount": 0,
    "pageCount": 0,
    "currentPage": 1,
    "perPage": 5000
  }
}
```

Here no data were found for the selected filter in `$url2` , to keep this codeblock short.

Getting names of all forms and models

Here we try the less exposed `cg` API, not the rest API.

- Call: `"$host/cg/api/summary`
- jq filter: `jq -c '[.forms[].name],[.models[].name] '`

Expected Output

```js
["cores","daily-message","expedition","hole","pieces","program","sample-request","sample-scien
["File","Core","InitGas","InitTemp","Pieces","Section","DailyMessage","Expedition","Explog","H
```

These are names of Tables and Input forms, they can come back in any order.

Expected output with WRONG username/password:

```json
{
  "name": "Unauthorized",
  "message": "Your request was made with invalid credentials.",
  "code": 0,
  "status": 401,
  "type": "yii\\web\\UnauthorizedHttpException"
}
```

Getting all expeditions

> **WARNING**
>
> To get this example to work, there must exist a form named `expedition` in your mDIS.

Modify the above-mentioned script. Try the `"$host/api/v1/form?name=expedition"` call

and process it with

```
echo $json3 | jq '.items[] | {(.acr):(.chief_scientist)}
```

Expected Result:

```json
{"Hibo":"F. Anselmetti, A. Schwalb"}
{"GRIND":"Tony Prave"}
```

Developer page

See **developer page**

REST API

See **REST API** page for extensive documentation.

More advanced examples

The **mdis-installer** git repository has more advanced use cases. There are **shellscripts** demonstrating how the mDIS REST API can be used for file upload, for example.

JavaScript

Developer page

Build your own reports and data analyses. Using the mDIS REST API, you can get mDIS data independently from any predefined forms and reports.

All you need is a current login to mDIS. You need at least web-access to mDIS.

Actually, it is helpful if you had physical access to the file system of the mDIS backend, and some programming skills.

- If you *do not* have such access, do the **"In a Browser Console"** tutorial described below.
- If you *do* have shell-access, e.g. inside your own mDIS VirtualBox, then you can do the "full" tutorial.

You also need to know the exact Primary Key IDs for a valid current `hole` and `site` dataset in your mDIS.

> **Does not work**
>
> 2020: The mDIS instance that this tutorial refers to is no longer online. Update coming soon.

mDIS REST API access with JavaScript

This tutorial demonstrates 3 methods to get data from mDIS with JavaScript.

1. As a Web page
2. In a Browser console
3. As a Node script (coming soon)

This tutorial does *not* demonstrate how to edit data (create, delete, duplicate, etc). Such API calls are **available**, but this tutorial focuses on the simple case of getting data in a read-only mode.

The tutorials use data from the ICDP DSEIS Project. To load this dataset, you need to run the `seed/example-dump` Yii migration. You can also work with your own mDIS data, but then some URL parameters must be changed accordingly.

Tutorial 1: As a Web page

Step-by step introductions are displayed further below.

What the JavaScript code does

- login to mDIS
- fetch some data
- display data as a table

Your task

It is your task to build a web page that displays a data table. Use copy-and-paste.

1. create an empty HTML file
2. Paste some HTML code into this file
3. Load the (incomplete) web page
4. Paste some JavaScript code into the HTML file.
5. Copy two JavaScript modules (files) to the web server
6. Load the (now complete) web page

HTML File

Inside your mDIS VirtualBox instance, open a file editor, navigate to folder `web/` . In there, create an empty page `apitest.html` .

Copy and paste in this code:

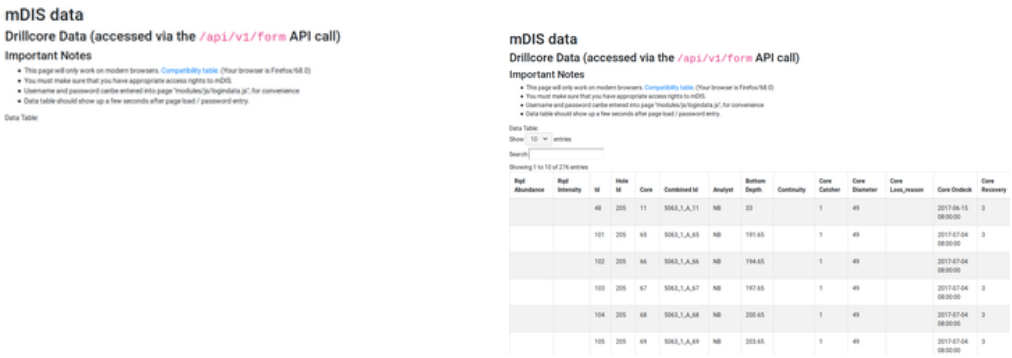
HTML Code

```
<!DOCTYPE html>
<html lang='en'>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>mDIS data</title>
    <!-- Bootstrap 4 and Jquery 3 and DataTables CSS and JS-->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootst
        integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" cr
    <link rel="stylesheet' href='//cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
        integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
        integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
        crossorigin="anonymous"></script>
    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>
```

```
    <!-- our custom modules -->
    <script type="module">

    </script>
</head>

<body>

    <div class="container mx-2">
        <div class="row">
            <h1>
                mDIS data
            </h1>
        </div>
        <div class="row">
            <h2>Drillcore Data (accessed via the <code>/api/v1/form</code> API call)</h2>
        </div>
        <div class="row">
            <h3>Important Notes</h3>
        </div>
        <div class="row">
            <ul>
                <li>This page will only work on modern browsers.
                    <a
                        href="https://developer.mozilla.org/en-US/docs/Web/JavaScript/Referenc
                        table</a>.
                    (Your browser is
                    <script>document.write(navigator.userAgent.replace(/^.+ /, ""))</script>)
                </li>
                <li>You must make sure that you have appropriate access rights to mDIS.</li>
                <li>Username and password can be stored into page "modules/js/logindata.js", f
                <li>Data table should show up a few seconds after page load / password entry.<

            </ul>
        </div>
        <div id="hint" class="row">Data Table:</div>
        <div class="row">
            <table id="mdisdata" class="table table-striped table-bordered"></table>
        </div>
    </div>
</body>

</html>
```

Open this file as a webpage in a modern web browser. On my computer, the URL is **http://34.76.214.94/apitest.html**. On your computer, it might be different.

An almost blank HTML page is shown. There is no data table. This is what you should see, and a preview of the result (when you are done with this tutorial):



In order to display the missing table, you must add some JavaScript code.

> **Note**
>
> There is this text block with an empty `script` element on line 24:
>
> ```
> 1  <script type="module">
> 2
> 3  </script>
> ```
>
> Line 2 here corresponds to line 24 in the long codesample before. Put some JavaScript code on that empty line element. (Which code? - read on)

JavaScript Code

The subtasks "login, fetch, display" mentioned above are implemented with 3 functions: `loginToDis()`, `getFromDis()`, `renderTable()`.

These functions are already provided to you, in the next 3 code blocks below. They should run unmodified. Cut and paste.

1. **login to mDIS**. The first function, `loginToDis()`, uses the `/api/v1/auth/login` API method. It will return a JSON object with ~ 6 text entries, for details see **auth-login documentation**. We need the `token` item, e.g. `{"token":"DssMim-2QERdh6DcxPV0Saj9qYuKo267"}`, for future logins. We'll send this token (instead of a password) along as an HTTP header field in all future requests:

```
Authorization: Bearer DssMim-2QERdh6DcxPV0Saj9qYuKo267
```

2. **fetch some data**. The second function, `getFromDis()`, gets some corebox data from mDIS via the `/api/v1/form` API method. (A `console.log()` statement inside the function body prints a string such as `Success! Found 276 records` to the Browser console.)

3. **display data as a table**. The third function, `renderTable()` will put a wide, striped data table on the web page.

This is the code that does the work mentioned in the "What the JavaScript code does" paragraph above. Put the following snippet into the empty `<script> </script>` element on line 24 of the `apitest.html` HTML page that you've just created.

```js
import {
  loginToDis,
  getFromDis,
  titleCase,
  renderTable
} from "./js/modules/apitest.js"
import { logindata } from "./js/modules/logindata.js"

var token = ""
var apicalldata = {}
var mdisdata = {}
if (!logindata.password || !logindata.username) {
  ;[logindata.username, logindata.password] = prompt(
    "Enter username and password separated by blank",
    "user2 user2 password"
  ).split(" ")
}

logindata.url = `${logindata.urlhost}/${logindata.urlpaths[0]}`
loginToDis(logindata)
  .then(function(response) {
    token = JSON.parse(response).token
  })
  .then(
    _ => {
      console.log("Login successful! got token " + token)
      apicalldata = {
        url: `${logindata.urlhost}${logindata.urlpaths[1]}`,
        authheader: `Bearer ${token}`
      }

      getFromDis(apicalldata)
        .then(function(response) {
          mdisdata = JSON.parse(response)
        })
        .then(_ => {
          renderTable(mdisdata, $)
        })
    },
    _ => {
```

```
      console.log("Login unsuccessful!")
      $("#hint")
        .text("Login unsuccessful!")
        .css("color", "red")
    }
  )
```

After entering this, the rendering of the datatable still does not happen. We have not yet provided actual implementations of the functions `loginToDis` , `getFromDis` , and `renderTable` . We will do this now.

Again, using only copy-and-paste, you must place the code that contains the functions `loginToDis()` , `getFromDis` , `renderTable()` into 2 appropriate files.

The code snippets are provided for you here. Copy them one after the other.

Create a new directory `web/js/modules/` , and in there create two empty files, `web/js/modules/apitest.js` , and `web/js/modules/logindata.js` .

Contents of file `web/js/modules/logindata.js` :

```
export let logindata = {
  username: "",
  password: "",
  urlhost: `${location.protocol}//${location.host}`, // "http://34.76.214.94",
  urlpaths: [
    // should always be first item in array
    "/api/v1/auth/login",
    // get all cores for exp. 1, site 41, hole 205 from the mDIS
    "/api/v1/form?name=cores&per-page=5000&page=0&sort=id&filter[expedition_id]=1&filter[site_
  ]
};
```

Editing this file is optional. You can put your username and password in there. Then, you will no longer be prompted to enter username and password. You can also edit the `hole_id` and other query string parameters.

Contents of file `web/js/modules/apitest.js` :

The JavaScript code of the function `loginToDis()` is highly verbose with lots of comments and error-handling. Adapted from examples found on the Mozilla Developer Network, **mdn**. Do not modify it.

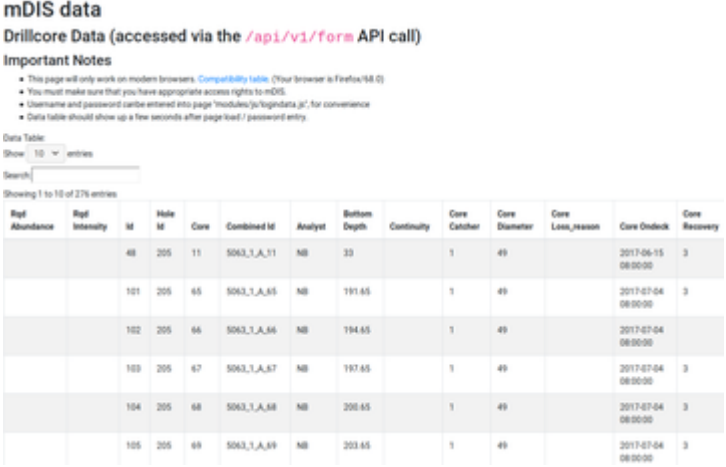
```
/**
   * Login to mDIS asynchronously and return a Promise to give us an Authorization token
   */
export function loginToDis(logindata) {
```

```
  // Create new promise with the Promise() constructor;
  // This has as its argument a function
  // with two parameters, resolve and reject
  return new Promise(function (resolve, reject) {
    var formData = new FormData(); // Currently empty
    formData.append("username", logindata.username);
    formData.append("password", logindata.password);
    // Standard function to load something from the web
    var request = new XMLHttpRequest();
    request.open('POST', logindata.url);
    request.responseType = 'text';
    // When the request loads, check whether it was successful
    request.onload = function () {
      if (request.status === 200) {
        // If successful, resolve the promise by passing back the request response
        resolve(request.response);
      } else {
        // If it fails, reject the promise with a error message
        reject(Error('Cannot login to mDIS; error code:' + request.statusText));
      }
    };
    request.onerror = function () {
      // Also deal with the case when the entire request fails to begin with
      // This is probably a network error, so reject the promise with an appropriate message.
      reject(Error('Cannot send data. There was a network error.'));
    };
    // Send the request
    request.send(formData);
  });
}

/**
 * Make an single authorized API call to mDIS, and return a Promise with
 * a response to be processed in a then() function.
 * apicalldata is a JSobject with 2 keys: url and authheader
 */
export function getFromDis(apicalldata) {
  return new Promise(function (resolve, reject) {
    var request = new XMLHttpRequest();
    request.open('GET', apicalldata.url);
    request.setRequestHeader("Authorization", apicalldata.authheader);
    request.responseType = 'text';
    request.onload = function () {
      if (request.status === 200) {
        resolve(request.response);
      } else {
        reject(Error('Cannot get data from mDIS; error code:' + request.statusText));
      }
```

```
    };
    request.onerror = function () {
      reject(Error('There was a general network error.'));
    };
    request.send();
  });
}

export function titleCase(str) {
  str = str.toLowerCase().split(' ');
  let final = [];
  for (let word of str) {
    final.push(word.charAt(0).toUpperCase() + word.slice(1));
  }
  return final.join(' ')
}

export function renderTable(mdisdata, $) {
  // console.log(mdisdata.items[0]);
  let i = 0;
  let columns = Object.keys(mdisdata.items[0]).map(k => (
    {
      targets: [i++],
      render: function (data, type, full, meta) {
        // handle NAs
        return full[k] ? full[k] : '';
      },
      defaultContent: '.',
      title: titleCase(k.replace(/_+/, " "))
    }
  ));
  // console.log(columns)
  $('#mdisdata').DataTable({
    dom: 'lfirtip',
    select: true,
    data: mdisdata.items,
    columns: [],
    columnDefs: columns
  });
}

```

The code uses native JavaScript Promises because it is an idiom widely used currently. Moreover, the chained `do_this().then()` style allows for a "separation of concerns".

Again, this is what you should see at the end of this tutorial:



Tutorial 2: In a Browser console

This tutorial is very similar to the JavaScript/Browser example shown above. However, physical access to the file system is not necessary. There is no need to create or copy files. Run everything in a browser console, by copy-and-paste. Unlike the tutorial above (which creates a pretty table), the end result of this tutorial is "only" some mDIS data displayed in the Browser console.

Open the console with `CTRL+SHIFT+K` .

Use a modern web browser to execute the following. (**List of browsers** supporting the codesample below.)

Important: Open a new Browser tab. Navigate to an instance of the mDIS, but do not log in. I have access to `http://34.76.214.94/` , but the URL to your mDIS is likely different.

> **Advanced**
>
> Opening the mDIS page in its own browser tab is necessary to avoid security problems (JS-sandboxing issues and cross-domain access restrictions).
>
> Do the following all in the browser tab that you have just opened.

Type F12, or Shift-Ctrl-k. Execute the JavaScript code below in the browser console *that belongs to the mDIS browser tab*.

Optional: Empty the Browser Console to get rid of older, irrelevant warnings and error messages. To remove them, click on the trashbin icon in the top left corner of the console.

Your task

1. Copy function definitions into the console
2. Set credentials
3. Enter JS code that fetches the data from mDIS.
4. See JSON data object appear in the console.

Detailed instructions:

1. **Copy function definitions.** Copy this JavaScript code into the console, hit the "Enter" key. (Alternatively, if you are on Firefox, you can open the JavaScript Scratchpad with `Shift + F4`, paste code into that window, click "Run").

Definitions of `loginToDis()` and `getFromDis()`, for copy and paste:

```
function loginToDis(logindata) {
  return new Promise(function (resolve, reject) {
    var formData = new FormData(); // Currently empty
    formData.append("username", logindata.username);
    formData.append("password", logindata.password);
    var request = new XMLHttpRequest();
    request.open('POST', logindata.url);
    request.responseType = 'text';
    request.onload = function () {
      if (request.status === 200) {
        resolve(request.response);
      } else {
        reject(Error('Cannot login to mDIS; error code:' + request.statusText));
      }
    };
    request.onerror = function () {
      reject(Error('Cannot send data. There was a network error.'));
    };
    request.send(formData);
  });
}

function getFromDis(apicalldata) {
  return new Promise(function (resolve, reject) {
    var request = new XMLHttpRequest();
    request.open('GET', apicalldata.url);
    request.setRequestHeader("Authorization", apicalldata.authheader);
    request.responseType = 'text';
    request.onload = function () {
      if (request.status === 200) {
        resolve(request.response);
      } else {
        reject(Error('Cannot get data from mDIS; error code:' + request.statusText));
      }
    };
    request.onerror = function () {
      reject(Error('There was a general network error.'));
    };
    request.send();
```

```
  });
}

```

2. To **set mDIS credentials**, and specify API call details, enter this code into the console:

```
let logindata = {
  "username": "user2",
  "password": "user2password",
  "urlhost": `${location.protocol}//${location.host}`,
  "urlpaths": [
    "/api/v1/auth/login",
    "/api/v1/form?name=cores&per-page=5000&page=0&sort=id&filter[expedition_id]=1&filter[site_
  ]
};
```

If you want to execute this code a second time, and the browser console responds with a error `SyntaxError: redeclaration of let logindata` , you must remove the `let` from the JavaScript expression.

3. **Fetch data** Fetch the mDIS core data and display them in the console:

```


var token = "";
var apicalldata = {};
var mdisdata = {};
if (!logindata.password || !logindata.username) {
    [logindata.username, logindata.password] = prompt("Enter username and password separated b
}

logindata.url = `${logindata.urlhost}/${logindata.urlpaths[0]}`;
loginToDis(logindata).then(function (response) {
    token = JSON.parse(response).token;
}).then(_ => {
    console.log("Login successful! got token " + token);
    apicalldata = {
        url: `${logindata.urlhost}${logindata.urlpaths[1]}`,
        authheader: `Bearer ${token}`
    };

    getFromDis(apicalldata).then(function (response) {
        mdisdata = JSON.parse(response);
    }).then(_ => {
        console.log(mdisdata);

    });
```

```
}, _ => {
    console.log("Login unsuccessful!");
});
```

This is what you should see:

The `> Object` item in the console can be inspected interactively. It contains a property `items`, an array of 276 objects representing the 276 core datasets from the mDIS database.

You can copy this object to the clipboard, and then paste it into your favorite editor, for further processing.

Tutorial 3: As a Node Script

This tutorial is also very similar to the JavaScript/Browser example shown above. However, everything is run on the command line. There is no need to open a browser, and there are no cross-domain issues. The data can be saved to disk which is difficult to achieve with the browser console. However, you need to have the **NodeJS** JavaScript runtime installed, and need to know what it is and how it can be used. Run everything from your command line, or from inside your IDE.

Task

> **TBC**
>
> coming soon

REST API

Documentation

See **REST API** page for extensive documentation.

This dir is here only for illustration purposes, must reside in the main mDIS dir tree.

Powershell

Developer page

Quick and dirty command-line processing on Windows.

Setup Powershell

Any user can run Powershell interactively by executing commands in the window. However, by default, running Powershell *scripts* (`*.ps1` files) is not permitted by Windows. Microsoft decided to apply this restrictive measure, to prevent that inexperienced users run malicious scripts.

But you can change **that setting**:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned -Force
```

> **Optional**
>
> To change this restrictive setting, once and for all, *for all users*, you need Administrator permissions. Inside a second Powershell Window (started with "*Run as Administator*"), you must execute this command at least once:
>
> ```
> Set-ExecutionPolicy RemoteSigned
> ```
>
> Now open a new Powershell window, as a regular user. You should be able to run `*.ps1` scripts now.

Tutorial

Download the **powershell-example-mdis.ps1** file for easier usage.

Alternatively, save this code block as `powershell-mdis-example.ps1`. You can also enter the 7 commands into a Powershell window, with cut and paste. of course you have to change the value of the `$server` variable (line 7).

```
1  #!/usr/bin/pwsh
2  #!/snap/bin/powershell
3  # Get a simple JSON dataset from the REST API of the ICDP mDIS
4  # The dataset is "All Expeditions" (2 rows at this time)
5  # tested on Powershell Core 6.2.2, 6.1.0
6  # knb 20190807
```

```
$server = "http://34.76.214.94"

# set username and password here
$postParams = @{username = 'user1'; password = 'user1password'}

$response = Invoke-WebRequest -Uri "$server/api/v1/auth/login" -Method POST -Body $postParams
#$json = ConvertFrom-Json $response.Content -AsHashtable # $json['token']
$token = ConvertFrom-Json $response.Content  | select 'token' |  %{$_.PSObject.Properties.Valu
$headers = @{'Authorization' = "Bearer $token" }

$uri = "$server/api/v1/form?name=site"
$response2 = Invoke-RestMethod -Method Get -AllowUnencryptedAuthentication  -Headers $headers
# Print it as a table. Expected output: see end of this file
$items = $response2 | select items |   %{$_.PSObject.Properties.Value}
$items | Format-Table
#$items |Out-GridView    # First: Install-Module Microsoft.PowerShell.GraphicalTools
# UNCOMMENT THIS to Write data to a CSV file
#$items  | Export-csv expeditions.csv -Append -NoTypeInformation

# OPTIONAL
#
# convert CSV File to Excel:
# download this file https://github.com/gangstanthony/PowerShell/blob/master/Save-CSVasExcel.p
# import it, then convert it.
# IF YOU HAVE PROBLEMS: YOU MUST DO THIS ONCE: Set-ExecutionPolicy RemoteSigned
#
#Import-Module .\Save-CsvAsExcel.ps1
# see https://stackoverflow.com/a/42597053/202553 Powershell+Export-Excel module
#$items | Save-CSVasExcel expeditions.csv

#
## expected result
#id program_id expedition acr   area  chief_scientist   comment             end_date
#-- ---------- ---------- ---   ----  ---------------   -------          # 1          1
# 2          2       5064 GRIND Southern Namibia Witputs Basin Tony Prave               in pr

```

Execute this by typing `powershell -noLogo -f powershell-mdis-example.ps1` on the command line. You can also uncomment line 21 (create CSV file), or try the instructions starting in line 25.

Optional

To convert mDIS results to Excel, you must download this Powershell Helper Script to your computer: **Save-CsvAsExcel.ps1**.

You can also install the `GraphicalTools` Powershell Module with `Install-Module Microsoft.PowerShell.GraphicalTools` and then you can use `Out-GridView` instead of `Format-Table` .

mDIS Backups

(continued from **System Administration**) related: (**mysql Administration**)

Backup/Restore

Basics

Daily cron job

By default, there exists a backup job that is scheduled to automatically run daily. The job puts database dumps and form designs into a single zip file. Usually these files are stored in directory `backend/data/upload/backup/mysql` . A typical filename of such a zipfile would be `mysqldb_backup_mdis-empty--10.132.0.3--2020-04-03--dis.sql.zip` . For more info about this zipfile, see **below**.

Alternative ways

There exists alternative methods to make backups of science-data files and of the whole mDIS installation itself. This section also describes how to restore a corrupt mDIS from a Backup.

Full and Incremental

Backups can be "full" or "incremental".

Full backups are copies of the entire system as-is.

Incremental Backups are small and contain a day's work (or similar).

Simple Full Backups

Single multi-gigabytye *.ova* file

A simple way to make a "full" backup is simply to create an *.ova* image of the virtual machine. An .ova image is single large multi-gigabyte file that contains everything. The image creation process was described in "**Installation with VirtualBox**".

You can *restore* a backup using the same mechanism as installing a virtual server using the *.ova image (see above). However, this method can consume a lot of disk space in long-running projects, and the act of backing up and/or restoring is quite slow (~10 -30 minutes).

Command line script

A simple command line script for making backup from the command line.

Code sample / Shellcscript

```sh
# get list of VirtualBox images Installed on machine

vboxmanage list vms

# pick one VBox from  the list,
# then export it to .ova file

vboxmanage export mdis-jet_20200329 -o  /data/virtualbox/shared_boxes/mdis-jet_20200329.ova
```

On Windows the command would be very similar, except the pathnames would look different of course.

VirtualBox Snapshots

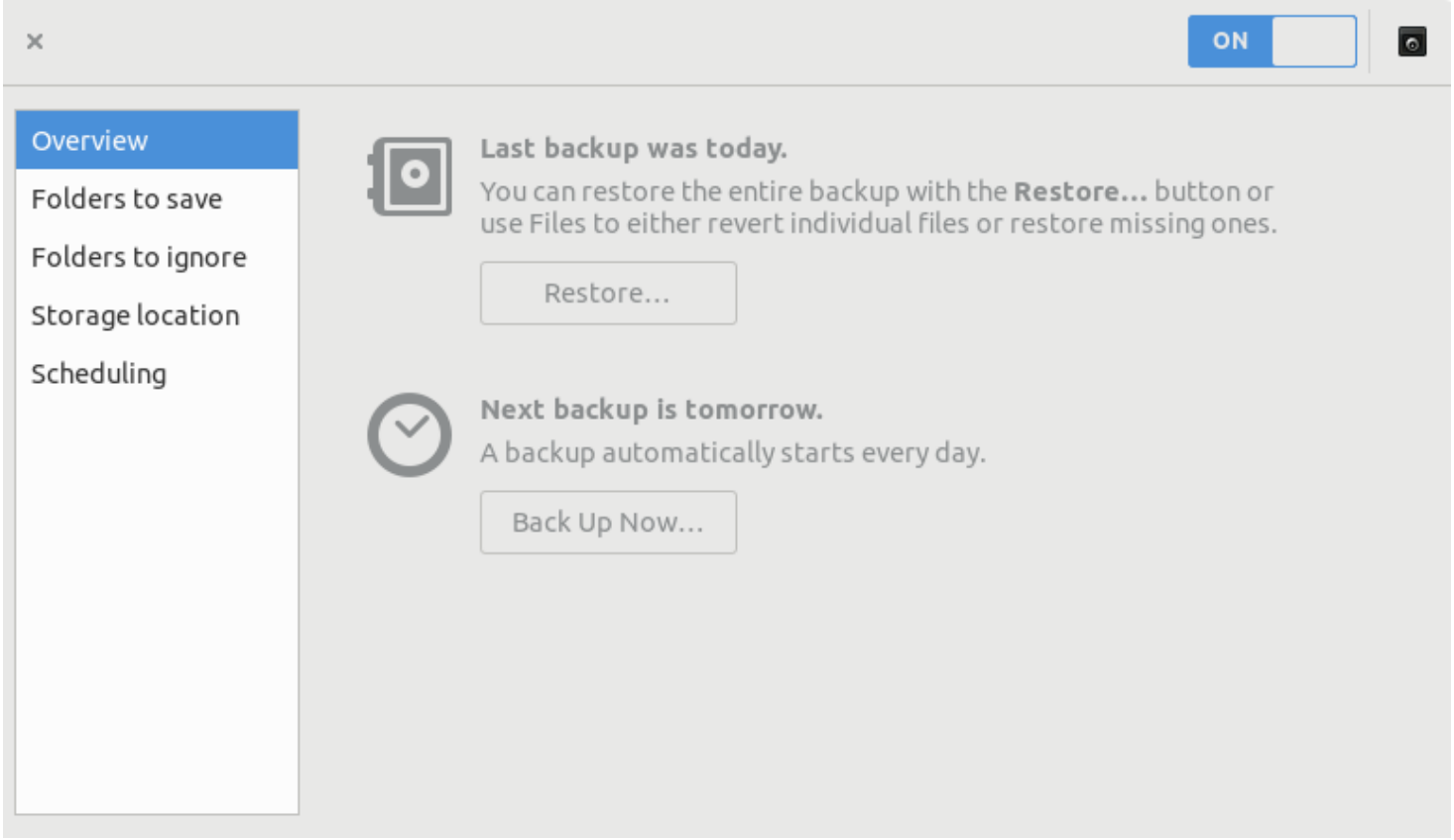
- On the host system, create "Snapshots" with the VirtualBox Software. This creates much smaller files than full backups, but are they are a bit harder to maintain and restore.
- In the list of snapshots, right-click on any snapshot you have taken and select *Restore*. By restoring a snapshot, you go back or forward in time. The current state of the machine is lost, and the machine is restored to the exact state it was in when the snapshot was taken.
- More information on snapshots can be found in the **Virtualbox documentation**

Additional techniques

The techniques described here both apply to Virtual machines *and* in particular to mDIS instances running natively on a Server.

Running a "native" mDIS means running an mDIS instance without VirtualBox. Backing up such instances cannot be accomplished with *.ova* files or Snapshots. Instead:

- Use backup software to create backups at the operating system level. This process works by finding all the changed files and saving only these files to backup storage.
  - For Ubuntu this could be **Deja Dup**, a software to automatically create incremental backups, and full backups. See the **documentation for that program**. Install it with `sudo apt install deja-dup duplicity python-gi` . Commandline-Program `deja-dup` also has a GUI frontend which is simply called "Backups". Start it from the command line with `deja-dup --name=backups` . The configuration via GUI is easy and pretty self-explanatory:



- Additionally, you should have the `/var/www/dis` directory under version control (git). This protects frontend code and backend code even better, if you follow established practices (commit often, push it somewhere).
- To save only the database content, you should create a daily dump of the MySQL database (using *mysqldump*). That will create 1 single backup file per day, which can be saved away with the chosen backup software (or stored directly in appropriate location).
- See also section **Mysql Administration** and **mysql-backup-and restore** page. Backing up the Mysql Server independently via DB dumps has additional benefits. For example, it enables you to restore individual tables if necessary.

> **Saving backups**
>
> You must also save the backups to offsite storage. That means: other network locations, or external media (e.g. USB sticks which are removed and stored elsewhere).
>
> Use `git push` to save your version-controlled directories to a remote git repo. This protects against scenarios such as failed hardware, stolen laptops, etc.

Restore Operations

For each of the above-mentioned backup strategies there are several ways to do teh inverse process, to restore a damaged system.

TBC

Explaining this in great detail would take a lot of space even for a single backup method.

Some hints:

- You should have the `/var/www/dis` directory under version control. If you have several branches in your code tree, you can restore a damaged file by copying it from a different branch to the current branch. Do this with `git checkout mybranch ./myfile`. If you have messed things up in a big way, you can then try a `git reset --hard <COMMIT-ID>` to go back in time to a previous state.
- **Git commands** work for frontend and backend code, but not for the database tier and the mysql data. Thus, for a shellscript that can import mysql backups/dumps, see also **mysql-backup-and restore** page.
- Restoring Virtualbox snapshots: see above
- **TBC**

Restoring Forms and Tables

Try to find a recent backup file that the daily backup cron job creates every day.

Perhaps it resides in `/var/tmp/upload/backup/mysql/` and could be named similar to `mysqldb_backup_vbox_mdis-jet--10.0.2.15--2020-04-10--dis.sql.zip`.

Your system administrator might have configured a different output directory and filename convention though.

The contents of the file should look like this:

```
  Length      Date    Time    Name
---------  ---------- -----   ----
   233326  2020-04-10 10:10   mysqldb_backup_vbox_mdis-jet--10.0.2.15--2020-04-10--dis.sql
     2132  2020-04-09 21:27   backend/dis_templates/forms/program.json
     6283  2020-04-09 21:27   backend/dis_templates/forms/expedition.json
     5976  2020-04-09 21:27   backend/dis_templates/forms/site.json
    13192  2020-04-09 21:27   backend/dis_templates/forms/hole.json
    18377  2020-04-09 21:27   backend/dis_templates/forms/core.json
     9719  2020-04-09 21:27   backend/dis_templates/forms/section.json
    11350  2020-04-09 21:27   backend/dis_templates/models/ArchiveFile.json
     7665  2020-04-09 21:27   backend/dis_templates/models/ProjectSite.json
    14298  2020-04-09 21:27   backend/dis_templates/models/ProjectHole.json
    18732  2020-04-09 21:27   backend/dis_templates/models/CoreCore.json
     7880  2020-04-09 21:27   backend/dis_templates/models/CoreSection.json
     7939  2020-04-09 21:27   backend/dis_templates/models/ProjectExpedition.json
     2580  2020-04-09 21:27   backend/dis_templates/models/ProjectProgram.json
---------                     -------
   360345                     19 files

```

> Output of `unzip -l /var/tmp/upload/backup/mysql/mysqldb_*--2020-04-10*.zip | grep -v "txt\| 0"`

In order to restore forms and tables, you must do this:

For forms, put the respective `*.json` file into `backend/dis_template/forms` .

For models/tables put the respective `*.json` file into `backend/dis_template/templates` .

The Code Generator of the **Templates Manager** will pick up the uploaded json file, and generate a new form or model for you.

If the table has not been created yet, start with the `.json` for the table, then upload the `*.json` for the form.

> **Replacing Forms is easy, but Replacing Tables is hard**
>
> It is easy to restore a data-entry-form by uploading JSON files. The form will be created instantly, because a form is a standalone object in mDIS.
>
> It is harder to restore tables. This has to be done in a certain sequential order, and even if the restoring operation succeeds, the table will be *empty*. Loading the newly restored table with science data is another matter and can be a quite tedious task.
>
> Filling empty mysql tables must be done in a certain sequential order, and requires working with the mDIS File-Importer which is part of the **"File-Upload"** GUI feature, but see also the **developer docs**.
>
> For low-level access try the **Adminer** mySQL Administration tool (if it is installed with mDIS).

For details describing the process of restoring the mDIS mySQL database, also read this **extra document**.

Further System Administration

(continued from **System Administration**)

Configuration management

HTTP and HTTPS connectivity

HTTPS enables secure login and secure data transfer.

HTTPS connectivity for mDIS can be achieved with a set of free tools and services, as explained below.

However, at this time, only insecure HTTP is supported out-of-the-box.

> **mDIS and HTTPS**
>
> When installing and configuring HTTPS support for mDIS, the mDIS instance must be connected to the public internet *at least once*, to automatically sign the server certificate.

Why is HTTPS not enabled by default? Because HTTPS support requires a unique *server certificate* for each mDIS instance. The non-profit organisation **Letsencrypt.org** issues these certificates for free.

However, we cannot re-use one of our older LetsEncrypt certificates. These are not meant to be redistributed freely. Each newly configured mDIS intance needs to get a certificate separately. Hence we *canno simply put an older certificate inside the mDIS installation package.

When mDIS is installed, mDIS needs to be reachable from the public internet at least once, to register the certificate with LetsEncrypt.org. Moreover, the certificate needs to be renewed every 3 months. This can be automated, though.

Use these free services to register a domain name for mDIS:

freedns: Paid & No-Charge Domain Hosting & DNS Hosting Services. (Many others such services exist).

LetsEncrypt: A nonprofit Certificate Authority providing TLS certificates to 200 million websites.

CertBot: Certbot is a free, open source software tool for automatically using Let's Encrypt certificates on manually-administrated websites to enable HTTPS. *Install and run this software on your mDIS.*

SSL Server Test: An optional service to list all properties of LetsEncrypt certificates, to perform a quality check.

(TBC)

PWAs -Progressive Web App

HTTPS is also needed for **Service Workers**, a key component of PWAs, Progressive Web Apps. PWAs deliver a nice user experience on smartphones, by changing the look of the webbrowser in a specific way. The mDIS already has PWA features built-in, but they are not enabled when HTTPS is not available.

Development and staging

A mDIS admin should maintain a development server where to verify the upcoming software updates, and to try out new things.

This could be an own development machine, but ideally the mDIS SA also maintains a *staging* virtual machine. That is a setup that is more or less identical to the Virtual machine that is used in production, i.e. during an ongoing drilling project.

OS Updates

Updates of the guest OS inside the virtual machine

- Most Linux distributions provide security updates and install these automatically. For the LTS (long time support) version of Ubuntu (like 18.04 LTS), security updates are provided for 5 years. After that you have to upgrade to a newer Ubuntu version. The Ubuntu developers provide a migration tool to do this.
- On Ubuntu automatic security updates are configured by default (on server and desktop version). You just have to apply them:
- If a Linux is used as a guest OS, use the self-upgrading software used by the operating system. You need to be Unixuser *root* or belong to the "sudo" group to do this (find out with `sudo -l -U YOURUSERNAME`).
  - For Ubuntu Linux, run `apt list --upgradable` to see packages that can be updated. Run `apt update && apt upgrade`, or use the GUI tool "Software Updater" for managing updates.
    - If you do not want *any* updates, open "Software & Updates" App and disable update-checks (See **Detailed easy tutorial**).
  - If you want to prevent updates for *specific packages* only, mark them as untouchable; in Ubuntu/Debian use **"apt-mark hold <package>" or similar**
  - Make sure that `update-grub` script can do its work properly: There must be one partition marked as *bootable*. [TBC], otherwise -sooner or later- the `apt upgrade` command will create a weird confirmation dialog asking "Into which partition should update-grub write the boot record?"

Updates of the Host operating system

For a permanent installation, the server administrator has to decide, if or when to update software versions. You *should* apply the security updates for the *host* operating system.

- Windows. Updates for the host operating system have to be applied for security reasons, especially if it is Microsoft Windows.

- Linux. Minor security updates are applied automatically, or you get reminded to manually update, if the Linux OS has an internet connection.

> **Recommendation: No major upgrades of the host OS**
>
> For a project installation (on a laptop), we recommend to *not* manually update or upgrade the system to *a new major release*, e.g. Windows 8 to Windows 10. If you have to upgrade, make a full backup before such a step.

For the guest, security updates are optional.

For any upgrade holds: it should not affect the VirtualBox installation. If Windows or VirtualBox does not work anymore after an update, but you still have a recent *.ova file, or the hard disk file ("*.vdi"), you can continue working with the mDIS installation on the same or a different computer without losing any data.

Updates of VirtualBox itself

There is normally no need to upgrade the Virtualbox application. Turn off the update-checks in the VirtualBox GUI, or set notification intervals to the maximum value.

> **Recommendation: Do not update the VirtualBox Application**
>
> For a project installation (on a laptop), we recommend: *do not* update the VirtualBox application.

Verification of software upgrades

It is important to verify that software upgrades did work properly.

General verification strategy:

- check output emitted during update process
- run the client-side test suites (npm run test:unit, npm run test:e2e) TBC
- run mDIS interactively and check the Browser Devtools Console (CTRL+SHIFT+K)
- check logfiles during mDIS startup.
  - mDIS logiles are in `backend/runtime/logs/` .
  - On Linux Guests: Inside Linux guests the kernel startup messages of the last boot can be seen with `dmesg -T` . Various operating system logs are inside directory `/var/log/` , but often you need to be *root* to see them. If the guest OS runs *systemd*, most recent errors can be seen with `journalctl -xe -p err --since today` . For a GUI tool, call `gnome-logs` , installed in dir `/snap/gnome-logs` .

File-Upload

File-Upload as VirtualBox Shared Folder

Accessing the Upload directories directly

There can be additional ways to access the upload directory `/backend/data/upload` as a folder of the hard disk on the host: In the VirtualBox appliance containing the Backend, this folder is mapped to a directory on the host computer, so you can easily bulk-copy files into this directory. The mDIS file importer will detect directory changes and try to import files automatically.

Remote upload folders: On permanent server installations there could be a remote folder mapped to the shared folder. For instance, on a Windows host system, a Windows Shared Folder could represent the upload directory. (On Linux, the Shared Folder could be mapped to a Samba share, to an FTP server directory mounted via FUSE filesystem, to an NFS mount, etc. *We never tried this.*).

Sending email from mDIS

Setting up the PHP mailer (TBC)

Emails should be sent to new users after registration, or when a user tries to reset the password, or uses the password reminder (*not implemented yet*). Therefore, configuring the mDIS for sending mail is sometimes required.

Configure Mailer

Inside configuration file *backend/config/web.php*, section "components" there is a configuration section "mailer" determining how mails are sent by the webserver. Default setting for development is `useFileTransport => true` . This implies that mails are not actually sent, but saved in a temporary directory. They are stored inside "*backend/runtime/mail*" as eml-files.

To enable sending of email, set this in `backend/config/web.php` , section `components` :

```php
// Production setting
'mailer' => [
    'class' => 'yii\swiftmailer\Mailer',
    'messageConfig' => [
        'charset' => 'UTF-8',
        'from' => 'knb@gfz-potsdam.de'
    ],
    'transport' => [
        'class' => 'Swift_SmtpTransport',
        'host' => 'smtp.gfz-potsdam.de',  // e.g. smtp.gmail.com
        'username' => 'icdpdc',
        'password' => '',  // use master password
        'port' => '465',
        'encryption' => 'ssl',
    ],
],
```

```
// Development setting - write to .eml files:

// 'mailer' => [
//     'class' => 'yii\swiftmailer\Mailer',
    // send all mails to a file by default. You have to set
    // 'useFileTransport' to false and configure a transport
    // for the mailer to send real emails.
//     'useFileTransport' => true,
// ],
```

To set the logging level and enable emails to be sent to you on Error, set this in `backend/config/web.php`, section `components`:

```php
'log' => [
    'traceLevel' => YII_DEBUG ? 3 : 0,
    'targets' => [
        'all_messages' => [
            'class' => 'yii\log\FileTarget',
            'levels' => ['info', 'trace', 'warning', 'error']
        ],
        'problems' => [
            'class' => \yii\log\EmailTarget::className(),
            'levels' => \yii\log\Logger::LEVEL_ERROR,
            'message' => [
                'to' => 'knb@gfz-potsdam.de'
            ]
        ]
    ],
],
```

See **Yii-Swiftmailer Documentation** and **Logging Documentation**

Concerning the mailer setup inside a VirtualBox instance: Sending mail should work. In a future release there should be an mDIS setting, configurable by the DIS admin in the "settings" page.

Image/Picture conversion

mDIS has a file-uploading feature. The most important use case is uploading pictures of core scans. See **Viewer-Operator, section "File Upload"**.

To use this with all features (e.g. preview-image generation), the free software **Imagemagick** must be installed on the webserver. The Imagemagick command line tools (`convert` , `identify`) must be available to the webserver process such that PHP's `exec()` function can call these executables. (PHP module `imagick` does *not* have to be installed.)

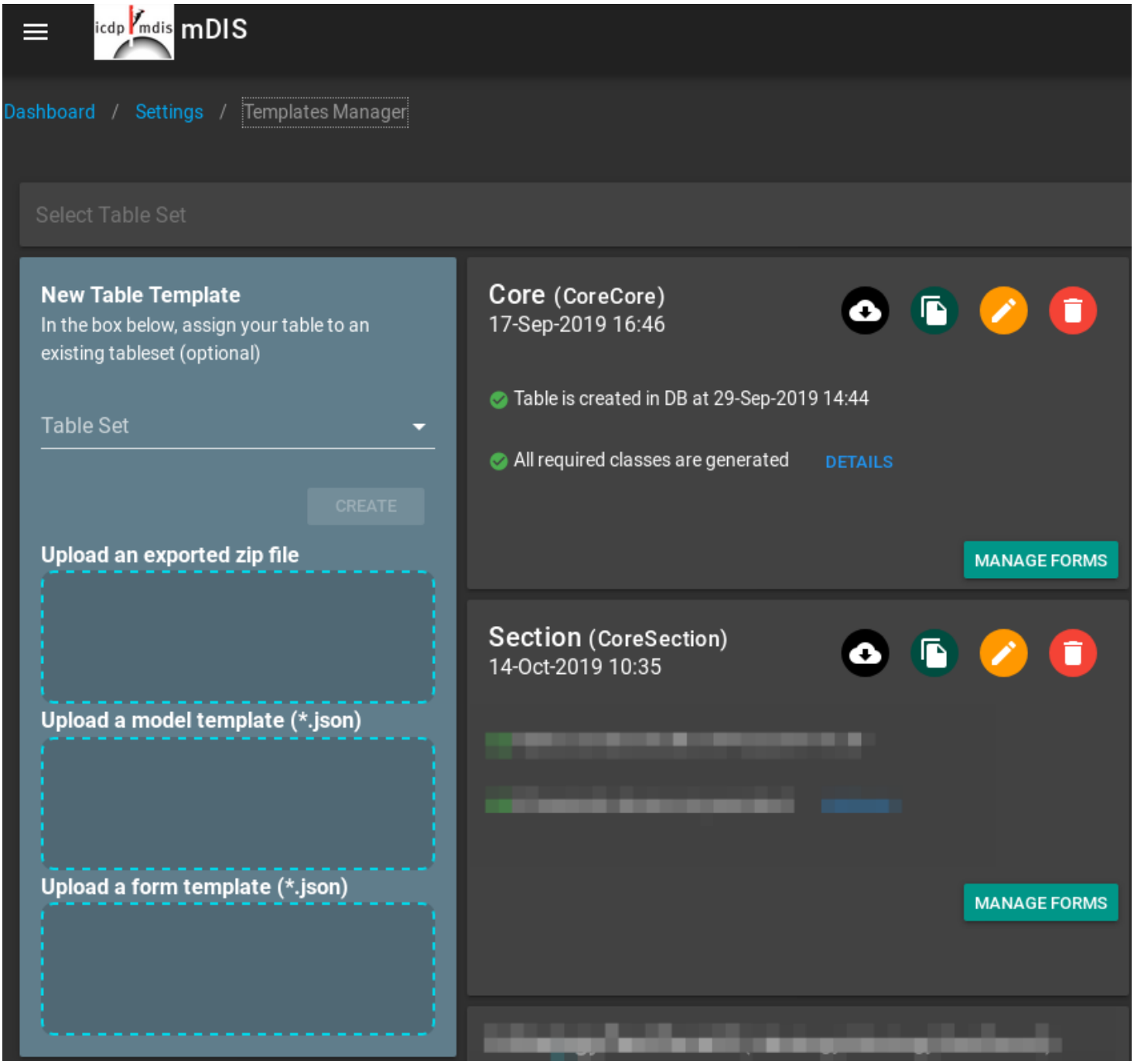
An mDIS operator uses these tools to create screen-sized JPEG Files and JPG thumbnails from oversized, high-resolution corescan pictures.

Importing CSV files

A CSV importer is located in `backend/importers/CsvImporter.php` . Usage: See **Viewer-Operator** documentation, and section **3.9 File-Importer**.

Transferring items between mDIS installations

Sometimes an mDIS admin will need to transfer some item, such as a preconfigured input form, from mDIS installation *A* to mDIS installation *B*. This can be done with the Templates-Manager.



It has two file-upload textfields where the user can drag-and-drop **.json* files into. These are now "registered" on the new mDIS (system *B*), but the changes must still be applied. After import, you still need to create the new data-tables and forms respectively, by clicking the "Save and Generate" Button.

The form can also import `.zip` File Archives, which have been exported from another mDIS. The zip-Files would contain both backend code (`.php` , `.json` files) and frontend code (`.vue` file).

The backend/config directory

In the `backend/config` directory you can find important information about many fine points on how the Yii2 framework was customized for mDIS.

See in particular the `config/web.php` configuration file.

The design of the server backend, i.e. the Yii2 app, closely follows the "Yii 2 Basic Project Template", with package `kartik-v/yii2-mpdf` added as an extra dependency.

The Yii2 main framework is installed, see directory `/backend/vendor/yiisoft` , and the Yii **extensions** installed separately by Composer are currently (Oct 2019): `yii2-authclient` , `yii2-bootstrap` (Twitter Bootstrap 3.4.x), `yii2-composer` , `yii2-debug` , `yii2-faker` , `yii2-gii` , `yii2-httpclient` , `yii2-swiftmailer` .

Better check yourself. Have a look at configfile `backend/vendor/yiisoft/extensions.php` and check which Yii extensions are actually being loaded.

PHP Dependencies in 2019

HTML 5 APIs

Local Storage

- "Local Storage" inside the Browser
- TBC: what's in there? How to change it

more HTML5 APIs

TBC

developer page

More... TBC

> **Still To Be Documented:**
>
> - Effect of Software Firewalls, Antivirus-Software on OS - ??? (TBC) (likely to be important for troubleshooting)
> - Effect of Adblockers, Do-Not-Track settings, JS permissions, JS same-host-origin-policy of Browsers (all not very important for mDIS)

System Administration

(contined from **further System Administration**)

PHP code dependencies

These are the PHP code dependencies as of Oct 31, 2019. They can be determined with `composer show`, executed in the directory where file `composer.lock` is located. Many of these packages are related to testing, and not relevant in production.

Number	Name	Version	Description
1	**behat/gherkin**	v4.6.0	Gherkin DSL parser for PHP 5.3
2	**bower-asset/bootstrap**	v3.4.1	
3	**bower-asset/inputmask**	3.3.11	
4	**bower-asset/jquery**	3.4.1	
5	**bower-asset/punycode**	v1.3.2	
6	**bower-asset/typeahead** .js	v0.11.1	
7	**bower-asset/yii2-pjax**	2.0.7.1	
8	**cebe/markdown**	1.2.1	A super fast, highly extensible markdown parser for PHP
9	**codeception/base**	2.5.6	BDD-style testing framework
10	**codeception/phpunit-wrapper**	7.7.1	PHPUnit classes used by Codeception
11	**codeception/specify**	0.4.6	BDD code blocks for PHPUnit and Codeception
12	**codeception/stub**	2.1.0	Flexible Stub wrapper for PHPUnit's Mock Builder
13	**codeception/verify**	0.3.3	BDD assertion library for PHPUnit
14	**dektrium/yii2-rbac**	1.0.1	RBAC management module for Yii2
15	**dektrium/yii2-user**	0.9.14	Flexible user registration and authentication module for Yii2

Number	Name	Version	Description
16	doctrine/instantiator	1.2.0	A small, lightweight utility to instantiate objects in PHP without invoking their constructors
17	ezyang/htmlpurifier	v4.11.0	Standards compliant HTML filter written in PHP
18	fzaninotto/faker	v1.8.0	Faker is a PHP library that generates fake data for you.
19	guzzlehttp/psr7	1.6.1	PSR-7 message implementation that also provides common utility methods
20	kartik-v/yii2-krajee-base	v2.0.5	Base library and foundation components for all Yii2 Krajee extensions.
21	kartik-v/yii2-mpdf	dev-master e513d17	A Yii2 wrapper component for the mPDF library which generates PDF files from UTF-8 encoded HTML.
22	kartik-v/yii2-widget-select2	v2.1.4	Enhanced Yii2 wrapper for the Select2 jQuery plugin (sub repo split from yii2-widgets).
23	mpdf/mpdf	v8.0.2	PHP library generating PDF files from UTF-8 encoded HTML
24	myclabs/deep-copy	1.9.3	Create deep copies (clones) of your objects
25	paragonie/random_compat	v9.99.99	PHP 5.x polyfill for random_bytes() and random_int() from PHP 7
26	phar-io/manifest	1.0.3	Component for reading phar.io manifest information from a PHP Archive (PHAR)
27	phar-io/version	2.0.1	Library for handling version information and constraints
28	phpdocumentor/reflection-common	2.0.0	Common reflection classes used by phpdocumentor to reflect the code structure
29	phpdocumentor/reflection-docblock	4.3.2	With this component, a library can provide support for annotations via DocBlocks or otherwise retrieve information that is embedded in a DocBlock.
30	phpdocumentor/type-resolver	1.0.1	A PSR-5 based resolver of Class names, Types and Structural Element Names
31	phpspec/php-diff	v1.1.0	A comprehensive library for generating differences between two hashable objects (strings or arrays).

Number	Name	Version	Description
32	phpspec/prophecy	1.9.0	Highly opinionated mocking framework for PHP 5.3+
33	phpunit/php-code-coverage	6.1.4	Library that provides collection, processing, and rendering functionality for PHP code coverage information.
34	phpunit/php-file-iterator	2.0.2	FilterIterator implementation that filters files based on a list of suffixes.
35	phpunit/php-text-template	1.2.1	Simple template engine.
36	phpunit/php-timer	2.1.2	Utility class for timing
37	phpunit/php-token-stream	3.1.1	Wrapper around PHP's tokenizer extension.
38	phpunit/phpunit	7.5.16	The PHP Unit Testing framework.
39	psr/container	1.0.0	Common Container Interface (PHP FIG PSR-11)
40	psr/http-message	1.0.1	Common interface for HTTP messages
41	psr/log	1.1.0	Common interface for logging libraries
42	ralouphie/getallheaders	3.0.3	A polyfill for getallheaders.
43	sebastian/code-unit-reverse-lookup	1.0.1	Looks up which function or method a line of code belongs to
44	sebastian/comparator	3.0.2	Provides the functionality to compare PHP values for equality
45	sebastian/diff	3.0.2	Diff implementation
46	sebastian/environment	4.2.2	Provides functionality to handle HHVM/PHP environments
47	sebastian/exporter	3.1.2	Provides the functionality to export PHP variables for visualization
48	sebastian/global-state	2.0.0	Snapshotting of global state
49	sebastian/object-enumerator	3.0.3	Traverses array structures and object graphs to enumerate all referenced objects
50	sebastian/object-reflector	1.1.1	Allows reflection of object attributes, including inherited and non-public ones
51	sebastian/recursion-context	3.0.0	Provides functionality to recursively process PHP variables

Number	Name	Version	Description
52	sebastian/resource-operations	2.0.1	Provides a list of PHP built-in functions that operate on resources
53	sebastian/version	2.0.1	Library that helps with managing the version number of Git-hosted PHP projects
54	select2/select2	4.0.10	Select2 is a jQuery based replacement for select boxes.
55	setasign/fpdi	v2.2.0	FPDI is a collection of PHP classes facilitating developers to read pages from existing PDF documents and use them as templates in FPDF.
56	swiftmailer/swiftmailer	v5.4.12	Swiftmailer, free feature-rich PHP mailer
57	symfony/browser-kit	v4.3.5	Symfony BrowserKit Component
58	symfony/console	v4.3.5	Symfony Console Component
59	symfony/css-selector	v4.3.5	Symfony CssSelector Component
60	symfony/dom-crawler	v4.3.5	Symfony DomCrawler Component
61	symfony/event-dispatcher	v4.3.5	Symfony EventDispatcher Component
62	symfony/event-dispatcher-contracts	v1.1.7	Generic abstractions related to dispatching event
63	symfony/finder	v4.3.5	Symfony Finder Component
64	symfony/polyfill-ctype	v1.12.0	Symfony polyfill for ctype functions
65	symfony/polyfill-mbstring	v1.12.0	Symfony polyfill for the Mbstring extension
66	symfony/polyfill-php73	v1.12.0	Symfony polyfill backporting some PHP 7.3+ features to lower PHP versions
67	symfony/service-contracts	v1.1.7	Generic abstractions related to writing services
68	symfony/yaml	v4.3.5	Symfony Yaml Component
69	theseer/tokenizer	1.1.3	A small library for converting tokenized PHP source code into XML and potentially other formats
70	vlucas/phpdotenv	v2.6.1	Loads environment variables from `.env` to `getenv()`, `$_ENV` and `$_SERVER` automagically.
71	webmozart/assert	1.5.0	Assertions to validate method **input/output** with nice error messages.

Number	Name	Version	Description
72	**yiisoft/yii2**	2.0.28	Yii PHP Framework Version 2
73	**yiisoft/yii2-authclient**	2.2.4	External authentication via OAuth and OpenID for the Yii framework
74	**yiisoft/yii2-bootstrap**	2.0.10	The Twitter Bootstrap extension for the Yii framework
75	**yiisoft/yii2-composer**	2.0.8	The composer plugin for Yii extension installer
76	**yiisoft/yii2-debug**	2.0.14	The debugger extension for the Yii framework
77	**yiisoft/yii2-faker**	2.0.4	Fixture generator. The Faker integration for the Yii framework.
78	**yiisoft/yii2-gii**	2.0.8	The Gii extension for the Yii framework
79	**yiisoft/yii2-httpclient**	2.0.12	HTTP client extension for the Yii framework
80	**yiisoft/yii2-swiftmailer**	2.0.7	The SwiftMailer integration for the Yii framework
81	**yiithings/yii2-dotenv**	1.0.2	PHP DotEnv for Yii2 framework

2. mDIS For System Administrators (SAs)

This page describes tasks that are usually performed only once, during setup and configuration of the mDIS system.

> **Note**
>
> Many tasks and responsibilites overlap with the mDIS **developer role**. A clear separation is not always possible.

> **Extra Pages**
>
> Stuff relevant for System-Administrators:
>
> 1. **Installation with Virtual Box**
> 2. **Native Installation on Linux**, or with Docker
> 3. Using the Template Manager
>   - See **extra Template Manager** page
>   - See also **"For mDIS Developers"** page, and references in there

Client-Server Architecture

The mDIS frontend, what the user sees in the Web-Browser on the client-side, is based on **Vue.js**. On the server-side mDIS is a PHP7-based Application backed by the **Yii2** Framework, the Apache Webserver and a mySQL Database.

You may decide to use different Webserver Vendor and Database backends. Webserver choices are various and a matter of personal preference. For databases, see what **Yii2** supports. You may create an SQLite, MySQL, PostgreSQL, MSSQL or Oracle database, and possibly others via Yii extensions (Informix, IBM DB2, Firebird, MariaDB...).

We develop and test on mySQL 5. 7 (or higher) and Apache 2. 4 (or higher).

Webserver configuration

Apache 2.4

In production mode, there is no particular Apache configuration necessary , except for enabling `mod_php7` of course. There is no authentication, very little URL rewriting, no proxy-ing for Node, etc.

Apache *mod_rewrite* code

Yii2 requires very little URL rewriting.

The directives shown below are pretty basic and standard. Put them into `web/.htaccess` or into an appropriate Apache configuration file in `/etc/apache2` (if you are on Linux).

```sh
# redirect to index.html by default
DirectoryIndex index.html index.php

# if a directory or a file exists, use it directly
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# otherwise forward it to index.php
RewriteRule . index.php
AddDefaultCharset UTF-8
```

With rewriting set correctly, you'll be able to put your own static HTML pages or PHP scripts into the `web/` directory. Use URLs such as `http://mdis.on-my.server/user/admin/index` .

In development mode a bit more configuration needs to be done, especially if mDIS runs inside Docker containers. This is subject to the personal tastes and preferences of the developers. This is not further described here.

If you run the **Adminer** database management tool on the same server as mDIS, you should to create an Alias in one of the config files in `/etc/apache` , or call `adminer.php` directly.

PHP 7

For a complete PHP 7.2 configuration of a working mDIS installation, see **2019 phpinfo()** output.

From this document you should infer which PHP extensions to install, etc.

> **Misleading 404 error**
>
> If a 404 error (Page not found) occurs after clicking on the previous link, **simply reload the page**. It should work and display the familiar `phpinfo()` output that all PHP developers know and love. The output is a static snapshot from a development server (not from the data.icdp... server).

Yii UrlManager

For Routing inside the Yii framework, take a look at the `urlManager` section in `backend/config/web.php` which defines the main URL Paths that mDIS uses:

```php
'urlManager' => [
            'enablePrettyUrl' => true,
            'enableStrictParsing' => true,
            'showScriptName' => false,
            'rules' => [
                '/' => 'site/index',
                '/test' => 'site/test',
                '/site/error' => 'site/error',
                '<module:(rbac)>/<controller>/<action>' => 'rbac/<controller>/<action>',
                '<module:(user)>/<controller>/<action>' => 'user/<controller>/<action>',
                '<controller:(report)>/<reportName:[\w-]+>' => 'report/generate',
                '<controller:(files)>/<id:\d+>' => 'files/view',
                '<controller:(files)>/original/<id:\d+>' => 'files/view-original',
                '<controller:(files)>/<action>' => 'files/<action>',
                '<controller:(importer)>/<action>' => 'importer/<action>',
                '<controller:(terminal)>/<action>' => 'terminal/<action>',
            ],
        ],
```

This gives you an overview about the main routes that mDIS defines in the Web layer.

Also look into files `backend/modules/api/Module.php` and `backend/modules/cg/Module.php` which contain many important URLManager rules for mapping REST api calls to controller method calls.

> **Write Permissions**
>
> You must give the unixuser owning the webserver unix-process write permissions to certain directories: (`backend/dis_templates/forms/` , `backend/dis_templates/models/` , `backend/forms/` , `backend/models/` , `src/forms/`). See also **"for developers" documentation**.

Installation with Virtual Box

See page **Installation with Virtual Box**.

A *Vagrant*-based installer is also **provided** as a public repository. **Vagrant**_ is a command line utility for managing the lifecycle of virtual machines. e.g. for scripting the setup of VirtualBox VMs. A _Vagrantfile_ is a declarative installation script.

See **README page** for details.

Native Installation on Linux, or with Docker

See page **Native Installation on Linux**.

A Docker configuration will be documented in a later release.

(Docker is a relatively new virtualization software that is free to use but can be complex to configure. It is more suited for software developer workstations.)

mDIS User Management

Frontend

sa Roles and Permissions

The role `sa` is the most powerful role in mDIS. *sa* is the only role that can define new tables and columns.

With regard to user management

- *sa* can define new Frontend users and roles
- *sa* can assign access permissions and privileges to roles
- *sa* can assign users to roles
- *sa* can define file access permissions roles relevant to the upload area

Backend

There is no predefined 'sa' role. There are however the users `root` and `administrator`, which have a role equivalent to the sysadmin `sa` role in the frontend. They can do everything inside the host OS and the guest OS, in particular:

- installation tasks on the host system and within the VirtualBox guest
- seamless rebuilds of predefined forms and tables from within the VirtualBox (**'yii migration'** tasks)
- can upload anything and set OS-related file access permissions relevant to the upload directories

Creating a New User and Assignment of User Credentials

mDIS Administrators perform registration of a new mDIS user by means of a **special-purpose web form**, url-path `/user/admin/create`.

Unlike those webpages and forms that work with science data, this page does not use the REST API but uses the widget sets and the APIs of the Yii framework directly.

However, adding users is a task performed rarely. It must be implemented in a secure manner, so we decided to rely on built-in features of Yii's RBAC extension directly.

The user's password is initially assigned by the mDIS administator, or, if the new user is physically present with the mDIS Admin at registration time, entered by the new user personally.

> TBC

> At this time the mDIS user cannot change the password at a later time.
>
> Email sent at registration time does not inform user about privileges what he can do. (Role Assignment happens AFTER email is sent.)
>
> User administration should also happen via the REST API.

Login to the mDIS

If you have sufficient access rights, then you can access the user management using the menu entry "Settings" (visible inside the left frame, the "drawer"). With the user management page, you can edit users, groups and assign permissions.

mDIS Roles (and permissions)

A role represents the function a user has inside the mDIS.

In the tab "Roles" you can see the set of roles already present with their names and description. You can create as many more roles as you need using the menu button "Create". You can edit or delete a role using the icons in the right column of the table.

Next to the name and description, every role can have children. You can add other roles meaning the role inherits the permissions of these other roles. Additionally you can add explicit permissions (view or edit) for the forms.

Some permission assignments are updated automatically. The role `viewer` has view access to all forms; the role `operator` has edit access to all forms. Every time a form is created, updated or deleted these permissions are updated.

Some roles have some implicit permissions:

- Only members of role `sa` have access to the user management
- Members of role `developer` have access to the template manager

Users

Users can log in to the mDIS application using their user name and password. Preferably, every user should have his own account. In the tab "Users" you can see the list of users. As usual you can sort the table by clicking on the headers. You can see the user name, the email address and the time of the last access to the system. You can block a user, so he cannot log in any more. With a click on the pencil you can edit the properties of a user.

You can edit the email address, user name and password using the tab "Account details". In the tab assignments, you can see and edit his roles (see below) and permissions. When you click on an empty area in the input field "items" you get a list of all roles and permissions.

If you have several users with the same permissions, you should create a new role, assign the permissions to the role and assign only that one role to the users.

Template Manager

This is more an ongoing "developer" task, than a one-off setup- and configuration task.

There is overlap with the contents described in pages **"For Developers"** and, in particular **Templates-Manager**.

See also **Customizing models and templates with Yii Behaviors** for a more advanced PHP programming task.

Console Commands

This section will describe which special-purpose commands are necessary for maintenance of an existing, working mDIS installation. Examples for these commands are `npm run ...` or `yii migrate/fresh` and related.

`npm run build`

Command `npm run build` creates a production build of the mDIS frontend, which consists of several `*.js` - and `*.css` files, and images. Ultimately `npm run build` copies these files to a special output directory, `web/`. Before doing so it deletes older versions of these files from `web/`.

> **Directory `web`**
>
> Do not delete the contents of the `web/` directory manually. Some files put in there might be there for a variety of reasons. E.g. they are static files not served from mDIS, but still important.

Internally, the node package manager `npm` relies on the bundler **webpack** to do approximately the following subtasks:

All of this happens automatically and requires no input from the mDIS Admin.

Webpack comes built-in with Node Module **@vue/cli**. Webpack gets installed when Vue is installed.

> **For developers**
>
> Custom tasks are not included in the list above. Our custom tasks are:
>
> - injecting the git-tag and the build number into the mDIS dashboard for display.
> - creating sourcemaps during a dev build.
>
> The relevant config files are `vue.config.js` and `babel.config.js` .
>
> See also directories `node_modules/webpack*` and `node_modules/@vue` , `node_modules/vue*` , and (less so) `node_modules/@vue/cli-service/webpack.config.js` . The Babel preset is **@vue/app** which is a specialization of **@babel/preset-env**. **Babel** is also in directory `node_modules` .
>
> TBC

Yii Migrations

> **Read this first**
>
> The **console runners section** and the **migrations section** on the developer page.

About Migrations

`yii migrate` scripts perform initialization tasks and are often needed during mDIS development, especially when you want to setup, reconfigure, or rebuild an mDIS from scratch. This might be necessary during a running project, e.g. after substantial database schema updates and redesigns; during a new setup of an mDIS; or after all data have been purged, and the system must be rebuilt, i.e. prepopulated with new tables, users, forms, etc. The migration scripts are stored in `/backend/migrations` .

If there were any *new* columns in any important database tables, and the PHP controller classes can't find these new columns, because the Yii initialization scripts (migrations) were not applied, mDIS will throw lots of internal errors and the system will refuse to work.

Rebuilding the mDIS database

This is an example of a shell script applying Yii migration scripts, and 4 of our custom *seed* scripts (custom tasks, related to migrations, another topic in itself).

The following code empties all tables, and deletes all tables and users. Then it rebuilds everything in the correct order, and loads a database-dump with example data, and sets some default permissions.

sh

```
# Optional: Enter the Docker container, if your mdis runs on with Docker.
# docker exec -it disapp_php_1 bash            # enter container

# this command keeps all data but, but might throw some errors
# only new migrations are applied
# ./yii migrate --interactive=0
#
# this variant of the command drops all existing tables.
# (run this if tables have been redesigned substantially):
./yii migrate/fresh --interactive=0

# additional tasks
./yii seed/users
./yii seed/example-dump
./yii seed/list-values
./yii seed/widgets

./yii seed/form-permissions #

# optional: promote knb to sa-role - execute this inside Adminer
```

```
# update auth_assignment
# set item_name = 'sa'
# where user_id = select max(id) from user where username = 'superstar';
```

The last 3 `seed/` tasks are defined in `backend/commands/`, e.g. in file `SeedController.php` and in file `backend/modules/api/common/controllers/FormController.php`, method `updateAccessRights()`.

The `./yii seed/form-permissions` task tries to analyze existing PHP Model files and Forms found in the `backend` directory, created during previous specializations, and tries to assign appropriate edit permissions to the new users just created with `seed/users`.

The `yii` script runner is stored at the top level of the mDIS app. Change into this directory and call it as `./yii ...`. On Windows, use `./yii.bat ...`.

Updating third-party code

> **Background**
>
> Both client-side parts and the server-side components of mDIS make heavy use of third party code. From time to time these code dependencies must be updated. This needs to be done by an administrator.

The upgrade-commands itself are simple, but verifying that upgrades do not break anything can be quite time consuming.

Therefore, updates should be done cautiously: mDIS admins should try stuff in a development environment first, and then on a staging server (configured similar to a production server, but for internal use only), and *then* deploy on the production server.

In practice

- The update process requires mDIS having access to the internet. Upgrades without internet access can be done but are cumbersome.
- About ~70% of the mDIS codebase is open-source third-party code. Updates for these libraries are *very common*, both for Javascipt/Node Modules and the PHP modules that mDIS uses internally. It is the responsibility of the project managers and software developers to come up with a concrete plan and a set of best practices for upgrades. It will vary from project to project.

What needs to be updated can be seen with:

- on the client side: `npm outdated`
- on the server side: `composer outdated`
- Linux guest operating system: `apt list --upgradable`

- Virtual Box GUI: Help menu / Check For Updates

Upgrading Client Side (production environment)

In a production environment, all javascript dependencies are packaged together (by Webpack, by `npm run build`, started by the mDIS developer) into a single file, `web/assets/js/app.<cachebuster-id>.js`. This file might load several other *chunk*.js* files. Together, they occupy very little space, about 2 MB in the `web/assets/js` directory of the mDIS source-code tree.

For updating or restoring the front end, i.e. the code that runs inside the browser of the mDIS user, it is therefore necessary to copy the contents of the `web/` folder from a development machine (or from any storage location) into the `web/` folder of the production machine.

Upgrading Client Side (development environment)

The update process of a *development* instance of the mDIS is much more complicated, because the code-bundling system itself also gets updated. This is described in **"For developers"**.

> **See also**
>
> Developer Documentation: "Advanced System setup and configuration" - **Updating Third-party Code**

Upgrading Server Side

For the server side, **3rd-party PHP code-dependencies** are stored in directory *backend/vendor*. In March 2019 there were 31 top-level subdirectories in *backend/vendor*, occupying 270 MB of disk space.

- Basically, upgrade server-side PHP code with
  - `composer list`
  - `composer outdated`
  - `composer upgrade`

Command `composer list` lists composer commands.

Command `composer outdated` shows a list of installed packages that have updates available, including their latest version.

Command `composer upgrade` upgrades the dependencies to the latest version according to composer.json, and updates the composer.lock file (which contains detailed info about the packages actually installed, their repo locations, creation dates, checksums, ...)

Run command `composer help` for details.

- Verification:

- see below (2.9. Further System Administration)

TODO: describe which versions of 3rd party code are easy to upgrade and which aren't. Describe why some composer packages have their version number fixed in file "composer.json" (and in package.json)

Mysql Administration

The *mysql* console client

An mDIS admin might want to interact with the mysql/mariadb database server directly.

How?

- can be done with commandline tools `mysql` (for interactive SQL) and `mysqldump` (for backups)
- can also use web based admin tool **Adminer**, see also **below**. Alternatively, use **Phpmyadmin** or MYSQL Workbench which are more powerful than Adminer, but setting them up is out of the scope of this article.
- check mysql installation directory */var/lib/mysql* (as root) and */var/log/mysql* (for textbased messagelog/errorlog)

Why?

- can perform backups and restore, independently from mDIS webserver layer
- ideal for incremental backups of science data
- can export data in a range of different formats not supported by mDIS-exporter
- mysql direct access maybe needed some day, e.g. for performance tuning or for moving db data to a different partition or network location

Details: See **mysql-backup-and restore** page.

Credentials: For VirtualBox instances, see a dot-file in the `$HOME` directory of the `mdis` or `vagrant` unixusers. See also a dot-file in `/root` .

How to install *Adminer* inside the virtual machine

Adminer is a web based admin tool for Mysql. Adminer consists of a single big PHP file.

You can simply download Adminer- `latest.php` and can put this in the `web/` subdirectory:

```
curl -sSL http://www.adminer.org/latest.php -o web/adminer.php
```

Alternatively, run this shell script to install Adminer persistently in the `/usr` directory:

```sh
#!/bin/sh
sudo mkdir /usr/share/adminer/
# download 'adminer.php' 1-page standalone php script
```

```
sudo wget "http://www.adminer.org/latest.php" -O /usr/share/adminer/latest.php
sudo ln -s /usr/share/adminer/latest.php /usr/share/adminer/adminer.php
# download 'nette' design = 1 css file
sudo wget -nd https://raw.github.com/vrana/adminer/master/designs/nette/adminer.css -O /usr/sh
# create new apache config file
echo "Alias /adminer.php /usr/share/adminer/adminer.php" | sudo tee /etc/apache2/conf-availabl
sudo bash -c 'echo "Alias /adminer.css /usr/share/adminer/adminer.css" >> /etc/apache2/conf-av
# load config persistently and restart apache
sudo a2enconf adminer.conf
sudo systemctl reload apache2
```

Occasionally you might need to update the `adminer.php` file. You can do so by simply replacing the file in the respective directory:

```sh
# use appropriate, more recent version numbers
# (these are from 2018/2019)
mv adminer.php adminer.4.6.2.php
curl -LO https://github.com/vrana/adminer/releases/download/v4.7.3/adminer-4.7.3.php
cp adminer-4.7.3.php adminer.php
```

The *Adminer* web interface can do most administrative tasks well. However, it cannot rename MySQL foreign keys, for example. For that you would need a different tool, perhaps *phpMyAdmin*, or *HeidiSQL*.

Maybe one of the many free **Adminer-Plugins** can do the missing task.

We recommend to also install the following Adminer plugins:

AdminerDumpArray, AdminerDumpMarkdownDict,
dump-json, fk-disable, floatThead, json-column, stickyColumns, suggest-tablefields.

TBC

Troubleshooting

Troubleshooting measures for beginnners and advanced users.

Frontend problems

- Watch the texts in yellow notification tabs and try to understand their meaning. However, often the yellow balloon is to small to display longer error texts, and it disappears after 10 seconds. Therefore:
- Hit the F12 function key on your keyboard (or CTRL-Shift-I, or CMD-Shift-I on a Mac) to open the console of the Browser's developer-tools. Hit the "Console" tab in the developer-tools window. There the same error should be visible, as in the yellow notification box. Also check the "Network" tab to see if any files are missing, after mDIS unsuccessfully tried to load them in the background. Loading in the

background is expected behavior. mDIS does this with an internal component called "Axios". This "axios" name might also appear in the error texts. But errors or missing files should not appear.

- If you are an administrator with access to the backend, you might run `npm run build`. This creates/transpiles a new frontend. Doing so might delete older cruft. This is a last-resort measure.
- The NodeJS Javascript runtime is required to build the mDIS frontend. NodeJS compiles Vue code to compact Javascript code.

> **Which NodeJS version?**
>
> Q: For development purposes it might be needed to install a recent **NodeJS** version. However NodeJS is undergoing heavy and rapid development by a community of Javascript developers. So which Node version can you actually use?
>
> A: mDIS was developed and extensively used on Node v8.11 and v8.12. However in 2020 we have switched to Node 12.14.1 LTS.

Backend problems

If you have Shell/Console access to the backend, go to directory `backend/runtime/log` and check files there. Start with file `app.log`. Look at the last lines of the file, or any other log file in that directory (they might have different names). These lines contain the most recent errors. The log file is very verbose because it is designed for readability. Information in the log file is spread out over many lines. You might need to go back 200 lines; even if the error happened just moments ago.

To shorten the output, try this shell command: `grep error app.log`. The output of this command is often informative enough.

Further System Administration

See **extra document**

The document describes special-purpose tasks related to working inside mDIS.

References

See **for-developers page**.

(**Developer page**) (**The backend/models directory**)

Templates Manager

Template Principles

Using the GUI of the Templates Manager, the mDIS sysadmins and developers can create, update, delete Data Model Templates and Forms Templates. The source-code files of the forms and tables are stored in directory `backend/models/` (and `backend/templates/`).

In this context, the term "Model" means a Mysql database table, and its binding to an associated PHP code via the Yii PHP framework. The "Model" represents a relational datatable in terms of a PHP object.

We are talking about a "**Model-View-Controller**" software design idiom here, where "Model" describes the data storage/persistence layer. "Model" *does not mean* "entity relationship model" (or a related concept) here, and it does not mean something scientific such as "Age Model" or "Linear Regression Model".

> **Create Model first, Forms later**
>
> A new *Data Model Template* needs to be generated first.
>
> Then, *Forms Templates* are generated afterwards, based on the Data Model Templates.
>
> Tables and Forms Templates ultimately are created as `.json`, `.php` and `.vue` files. All three types are text files. Of course, the Template Manager also creates various objects inside MySQL, e.g. tables, indices, foreign keys, and so on.

The Templates Manager generates *output* in the form of `*.json` files, which are stored here:

- `/backend/dis_templates/models/*.json` - metadata for mysql data table definitions (data models)
- `/backend/dis_templates/forms/*.json` - metadata for input forms (Form definitions)

> **Why .json files?**
>
> Unlike legacy DIS, there are no database tables involved to store form-, table-, or tableset-metadata. All metadata is stored in `.json` files.
>
> These `*.json` files are (pretty-printed) plain text files that can be easily understood and edited by humans. These text files can be put under version control (git). They are also useful for exchanging form definitions and table definitions between existing mDIS installations.

Templates *.json File Structure

A closer look into `.json` files

Which template properties are available in a model's JSON definition file, or in a form's JSON definition file?

- For data **models** templates structure, see **here**
- For **forms** templates structure, see **here**

> **WARNING**
>
> Be careful when you make a copy of a form's `.json` file inside the `/backend/dis_templates/forms/` directory, e.g. for experimenting. mDIS will automatically detect the new file and instantiate the form, and then there will be *two* forms visible with the same name, in the `Templates Manager / XxxXxx's Forms` GUI. This can be confusing.
>
> Better store the copy of unneeded `.json` files outside of the `/backend/dis_templates/forms/` directory.

So if you have syntactically invalid JSON files inside `backend/dis_templates/forms/` or `/backend/dis_templates/models/`, move them into a different directory outside of `backend/dis_templates`. Maybe the Templates Manager windows will then show up again.

> **Templates Manager**
>
> The entire Templates Manager will not render, when there is the tiniest problem with any JSON file inside `backend/dis_templates/` . (e.g. a comma or a quote is missing). This can be extremely annoying.

How to determine if there is a problem, and which file is causing it? You have to check the syntax of every JSON file individually. Some commands (of many) to do so are:

- `json_xs -t none < sample.json  # perl`
- `python -m json.tool sample.json  # python`
- `jq . sample.json  # jq`

All these perform the same "JSON Lint" job - they try to parse the file `sample.json` as JSON - if it is invalid JSON, they will print an error message and exit with non-zero exit status. (Otherwise `perl` returns nothing, and `python` and `jq` return the file contents).

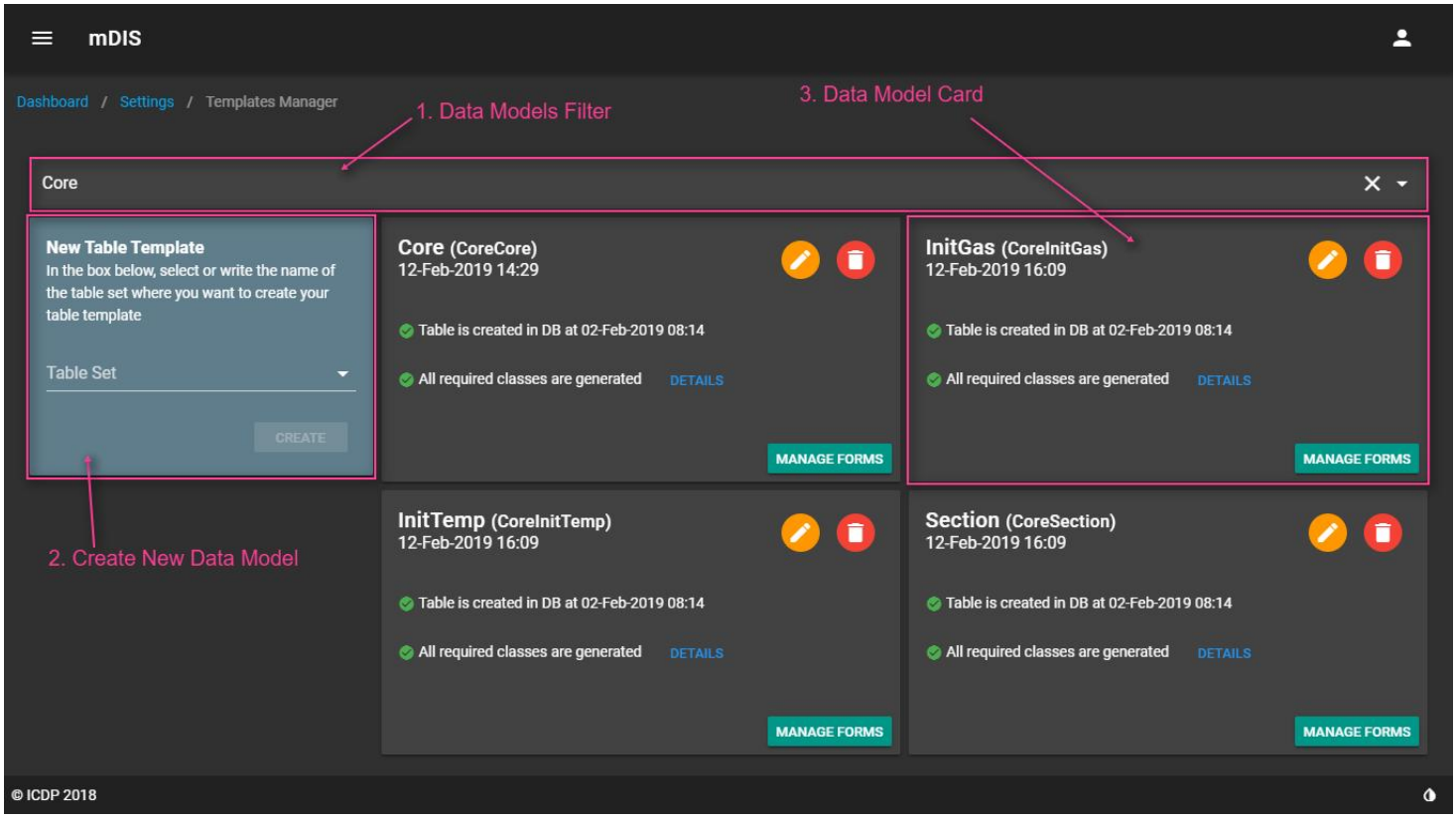
To check an entire folder full of JSON files for errors, use the PHP tool `jsonlint` like this (tested on Ubuntu Linux):

- `ls -1 backend/dis_templates/forms/*.json | xargs -i jsonlint-php "{}" | grep -q -v "Valid JSON" -B 1`

Templates Manager GUI

Main Page

Main Page of the Templates Manager.

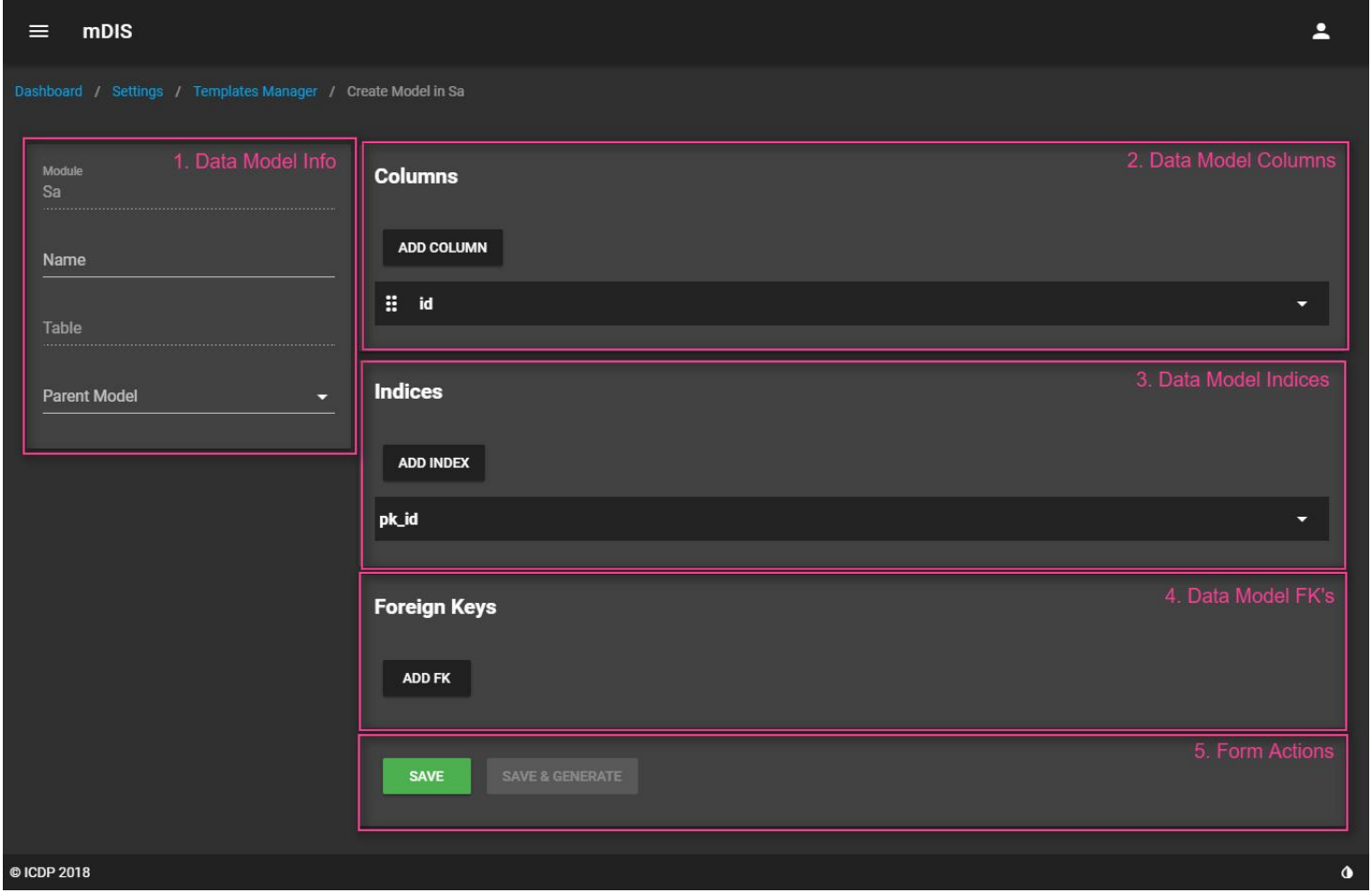


See picture above:

1. Data Models Filter: The value of this field is used to *filter* the data model cards (3). The value will be compared with the `fullName` property of the data model. The input is a dropdown list with the names of the table sets. Alternatively, any free text input could be entered.
2. Create New Data Model: Use this card to create a New Table Template (= "Linked Table", "Data Model"). Select the table set you want to assign the new data model to (or write a new table set name) then click `create` to be redirected to the data model creation GUI.
3. Data Model Card: From within a Data Model Card you can:

- update the model or delete it using the top right buttons.
- gather some useful information about the status of the generated files.
- manage the forms templates of a data model using the green button on the bottom right of any Card.

Create a Data Model Page

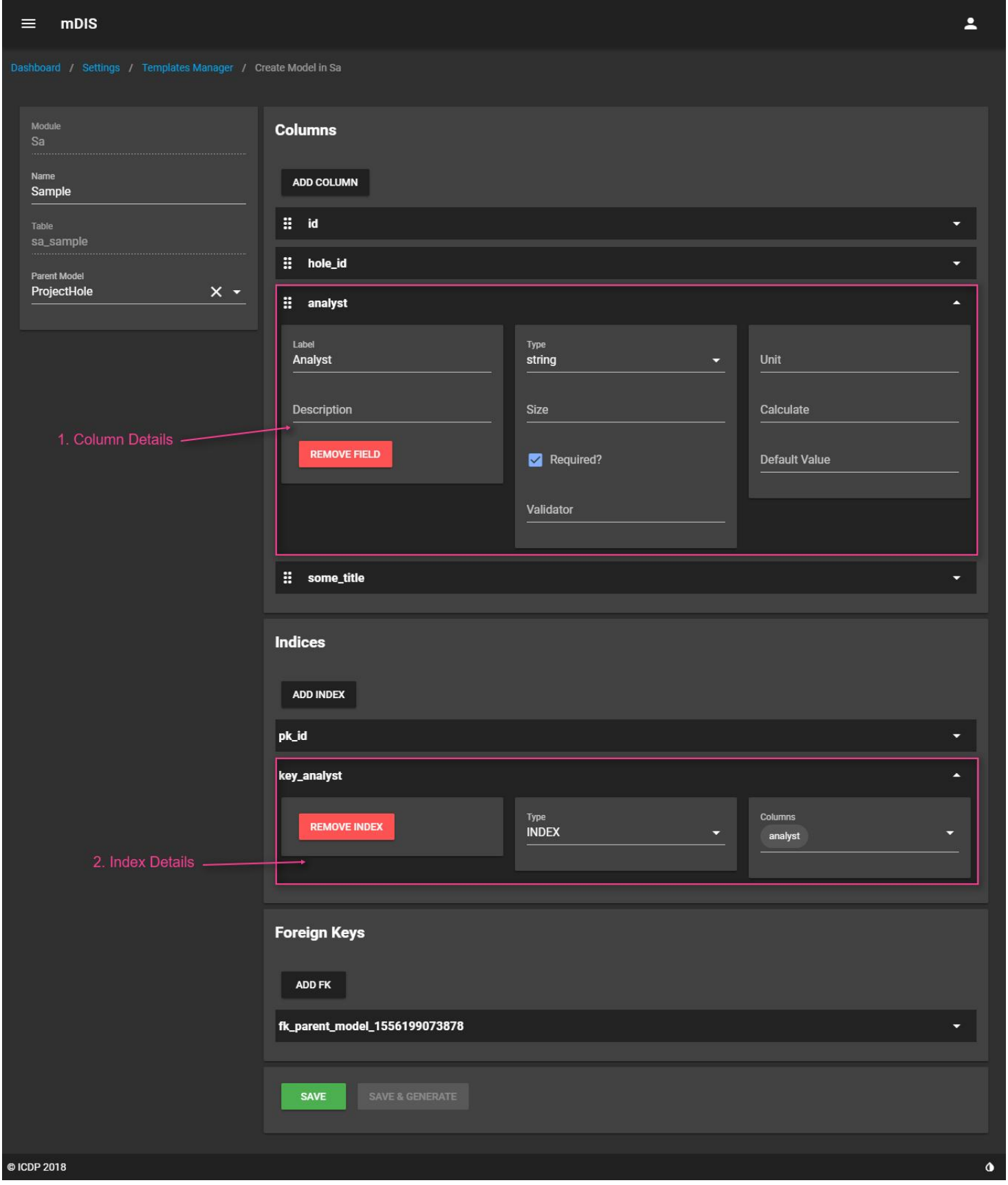


1. In the Data Model Info pane you can define `name` and `parentModel` values. See **data model templates structure** for more info. "Module" means "Tableset" here.

- Module field is disabled since it was selected in the main page using the data model create card.
- Table field is also disabled since it will be automatically generated from the Module name and the model name.
- Model name must be written in **"PascalCase"**, e.g. `CorePieces` .
- Parent Model could be selected from the list.

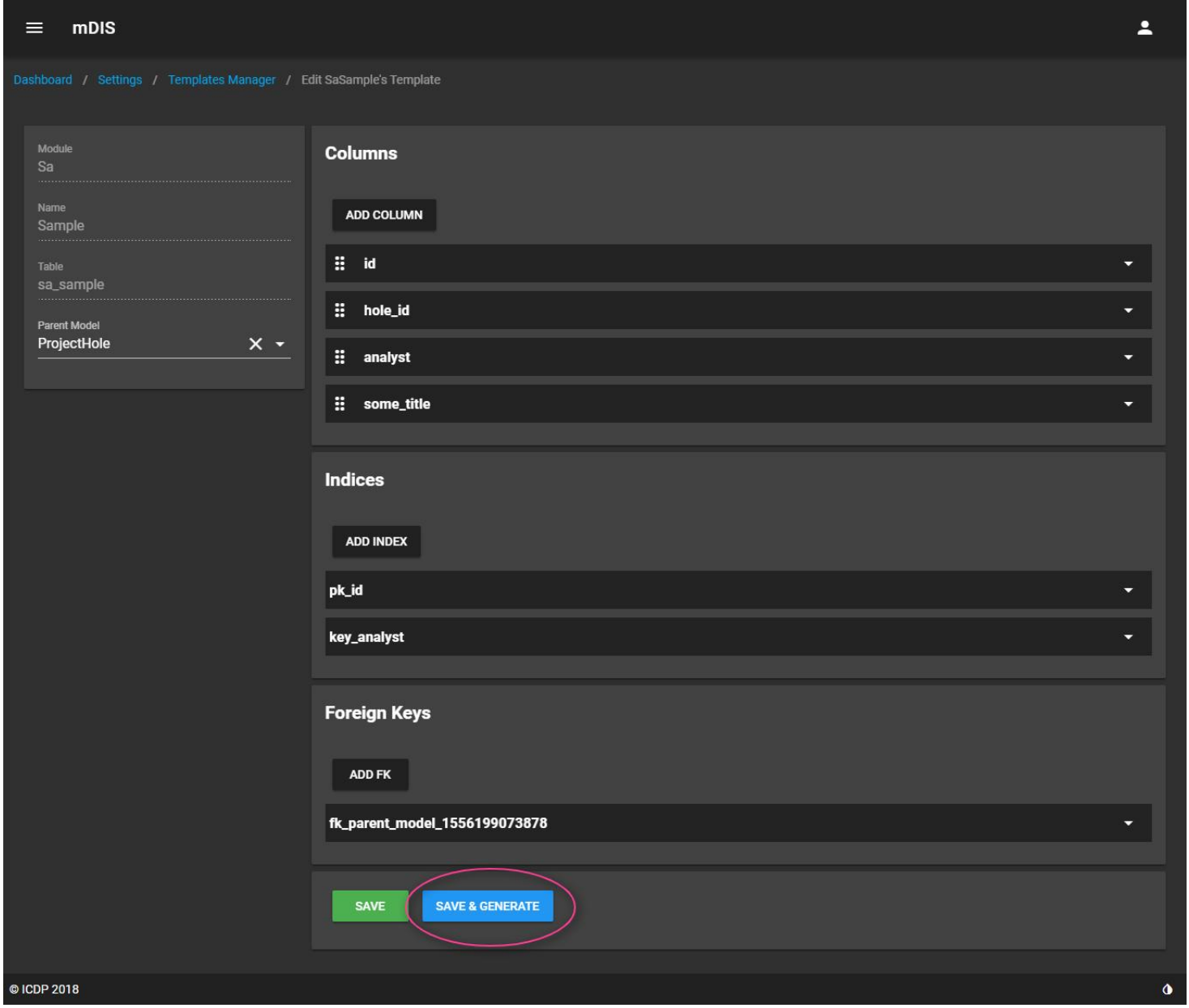
2. The Data Model Table pane, columns could be added here by clicking the top left button. Notice that `id` is already added for you since it is required in all tables as a primary key column.
3. The Data Model Table pane, indices could be added here by clicking the top left button. notice that `pk_id` is already added for you since it is required in all tables.
4. The Data Model Table pane, foreign keys could be added here by clicking the top left button. When choosing a parent model, a foreign key that represents this relation.
5. Click save to save the template and be redirected to the Data Model Update Template page. Notice that the generate button is disabled in this page since the template's `json` file should be saved first in order to be able to generate the `.php` and `.vue` files based on it.
6. (not shown in figure above) There is another pane for attaching **behaviors** to a form. This is a new feature and will be documented elsewhere.

Create a Data Model Example



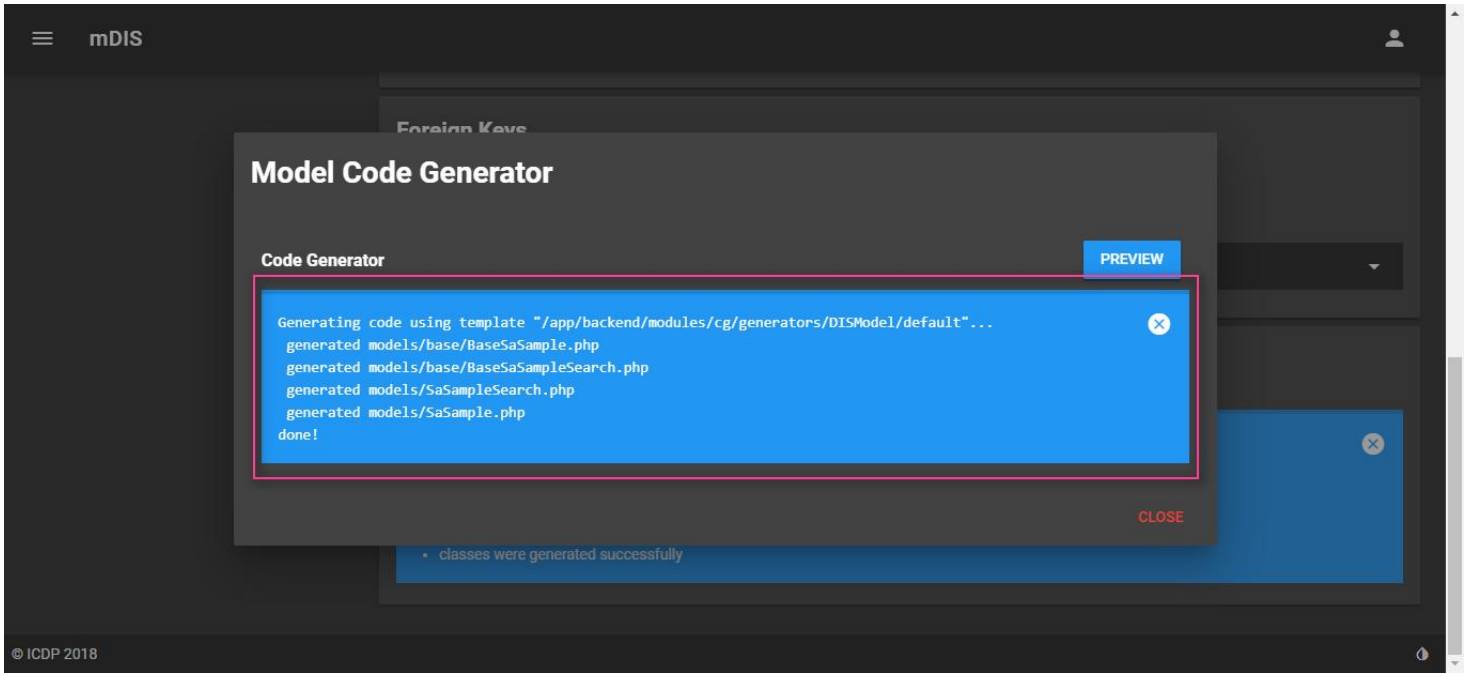
1. By expanding the column row you can edit the column details. see **here** to compare the field with the structure
2. The same for indices. see **here** to compare the field with the structure

After clicking the `Save` button, the app will be redirected to the data model template and the `SAVE & GENERATE` button will be enabled:



After clicking `SAVE & GENERATE` a dialog that will automatically save the template and generate four `php` classes which are responsible for managing the `sa_sample` table records.

> If you are updating a data model template, only **two** `php` classes will be generated



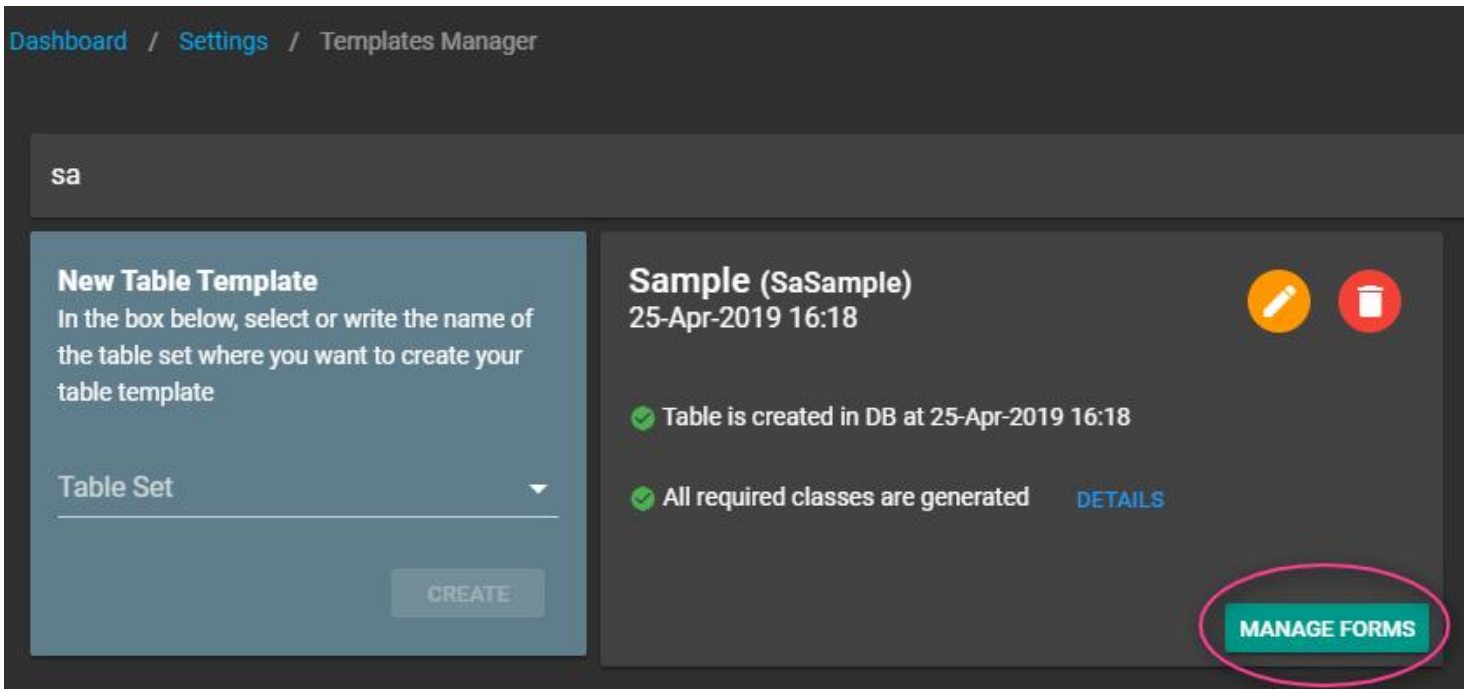
```php
Generating code using template "/app/backend/modules/cg/generators/DISModel/default"...
 generated models/base/BaseSaSample.php
 generated models/base/BaseSaSampleSearch.php
 generated models/SaSampleSearch.php
 generated models/SaSample.php
done!
```

1. `models/base/BaseSaSample.php` is the data model class. This class includes validation rules and provides the data manipulation methods (`save()` , `update()` , `delete()` , ..etc). This class will be overwritten every time you generate the data model template. Do not put important customization in here.
2. Generated `models/base/BaseSaSampleSearch.php` inherits form the previous class and has the required implementation needed to *filter* the data model records. This class is called by the Filter Bar on top of some forms, and by the **"Filter By Values"** pages. This class will also be updated every time you generate the data model template.
3. Generated `models/SaSample.php` is an empty class that inherits from (1). All the specializations you want to implement should be written in here. This file will *not* be updated or overwritten by the code-generator. It will be re-written in case it is missing, though.
4. `models/SaSampleSearch.php` an empty class that inherits from (2). All the specializations you want to implement should be written in here. This file will *not* be overwritten by the code-generator. It will be re-created in case it is missing, though.

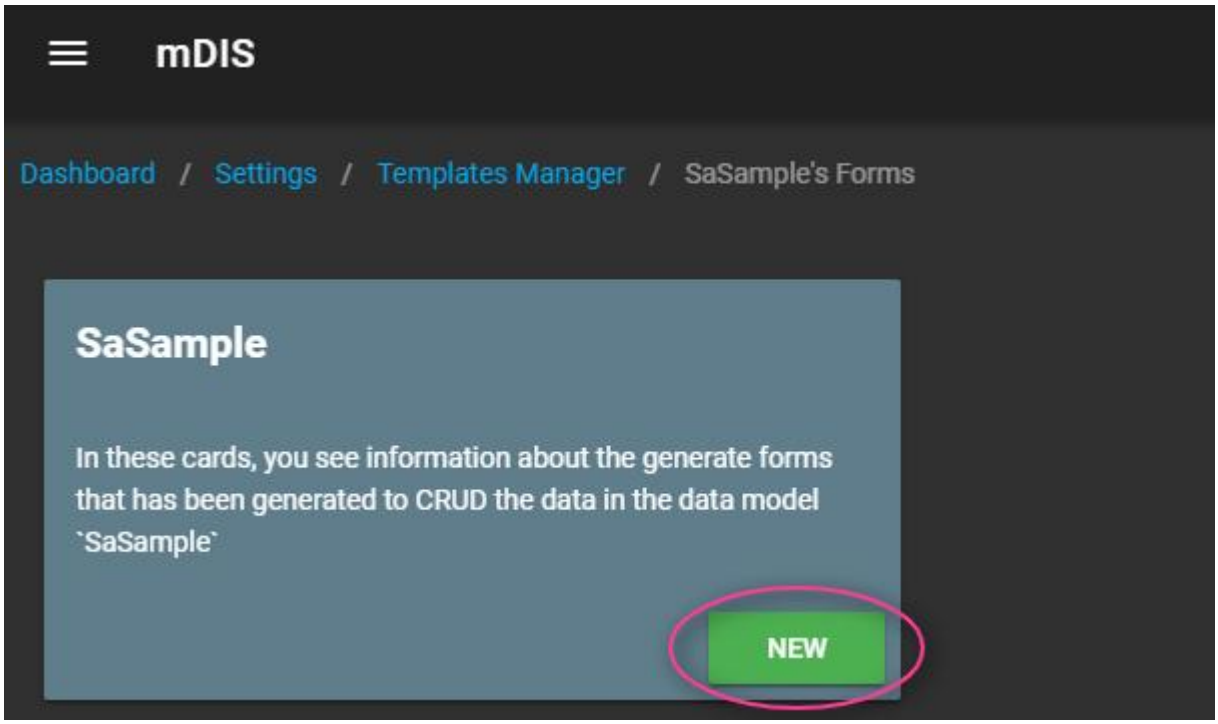
The class hierarchy is also explained in a **longer document** based on the "Cores" table.

Create a Form for a Data Model

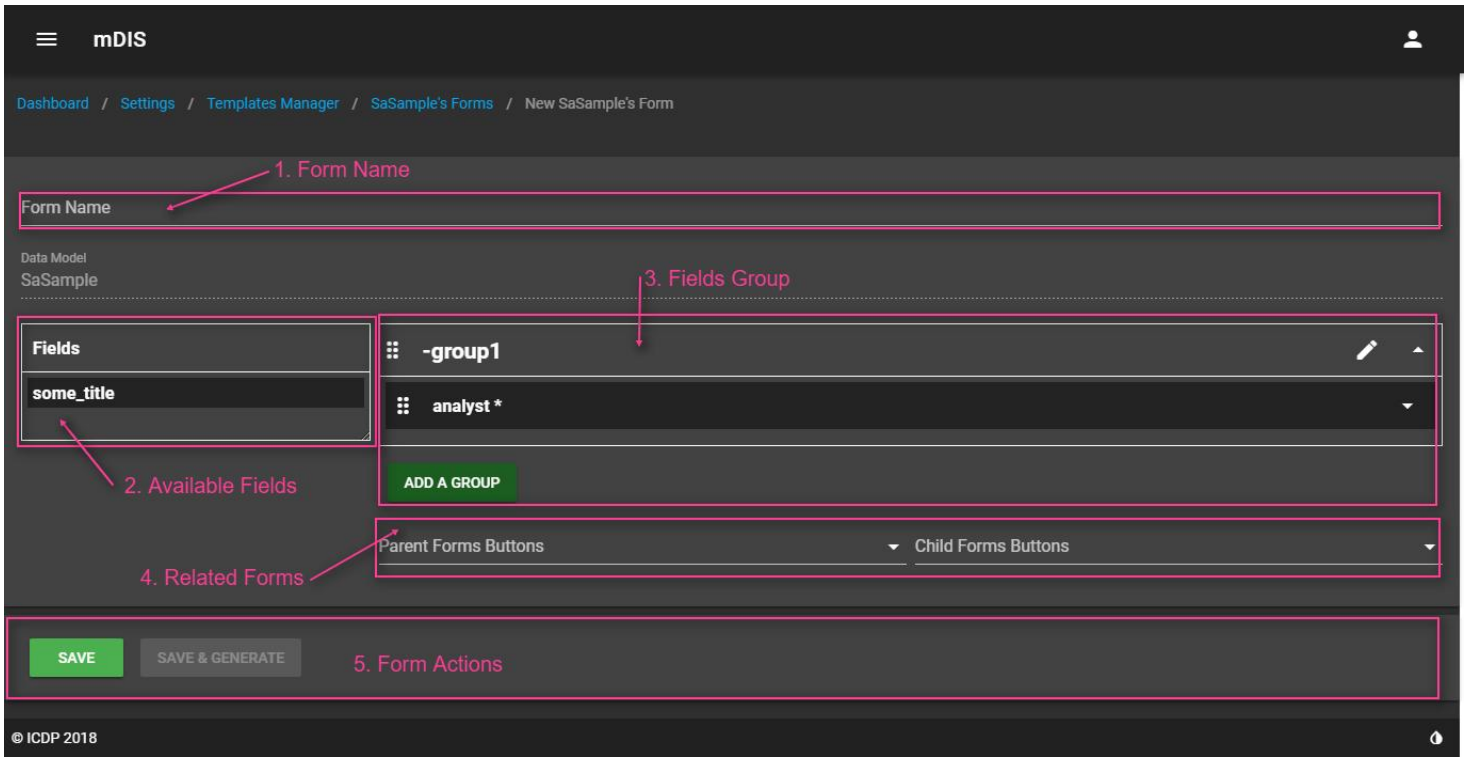
In order to create a form for the table `SaSample` , search for the model in the Templates Manager home page and click on `MANAGE FORMS` .



The "Manage Forms" page will open and you can click on `NEW` to create a new form.



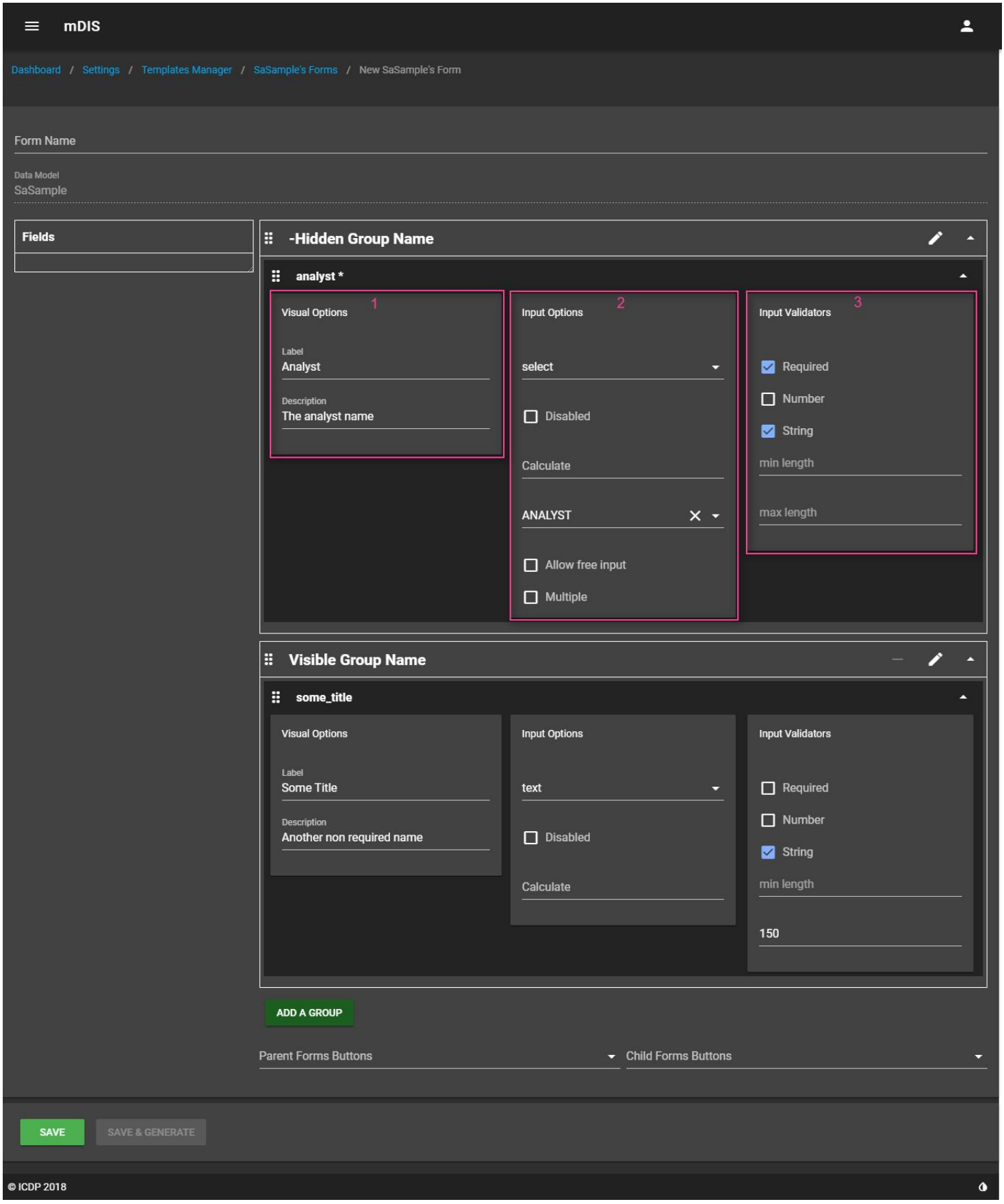
The form-template create-page will open and will generate a basic content based on the data model template. More information will follow



1. The form name must be unique and written in "**kebab-case**", e.g. "core-pieces".
2. On the left side, a list of the data-model columns is available. Each column can be dragged into a field group on the right side. Click on the draghandle formed by the 6 dots `:::` on the left side of any column name. Notice that all fields labeled as "required" are already added to the first field-group.
3. In this section the field-group can be managed.
   - the group name could be changed by clicking on the pencil icon (✏) on the right side of the group header.
   - a new group could be added by clicking on the `ADD A GROUP` button.

- in case more than one group is available, the order of the groups can be changed using the drag handle besides the group name (left). Notice that if the group name starts with `-` it will *not be displayed* in the generated form. This is sometimes useful to save screen space.

4. Depending on the data model relation (see **parentModel**), a list of the available child/parent forms will be available in these two lists. The user can then to choose one item from the list. Multiple forms can be selected and the corresponding form buttons will appear in the generated form.
5. As in data model template form, the `SAVE & GENERATE` button is disabled until the template's `.json` file is saved.



1. The values of these fields will be copied from the data model template and can be edited as well.
2. These options determines what kind of an input should be used in the form to edit the corresponding column
   - Notice that for `analyst` field, when choosing `select` as an input type, extra options are displayed that are specific for this kind of input
3. The validators are also checked automatically depending on the data model template and can be edited as well.
   - for `some_title` field a max string length is added to define the title maximum length.

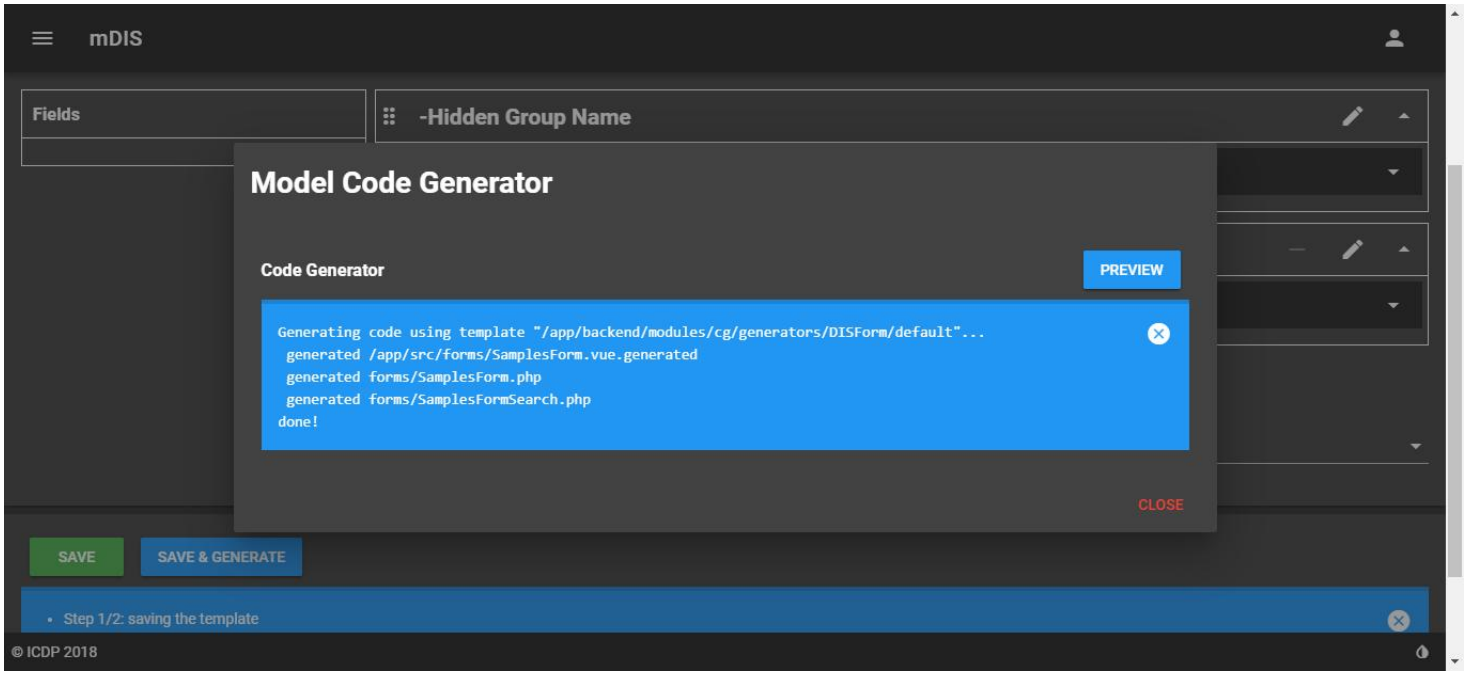
> See **Field Definition** for more information about these options.

> **File permissions**
>
> **Uploading `*.json` files with table- and form-definitions to the webserver**
>
> Make sure the web server process has write access to the following directories:

```sh
backend/dis_templates/forms     # code-generated .json files for input forms
backend/dis_templates/models    # code-generated .json files for models/tables
backend/forms                   # code-generated .php files for input forms
backend/models                  # code-generated .php files for tables
src/forms                       # contains *.vue.generated files
```

> An administrator could make these directories world writable. It is a security risk, though.

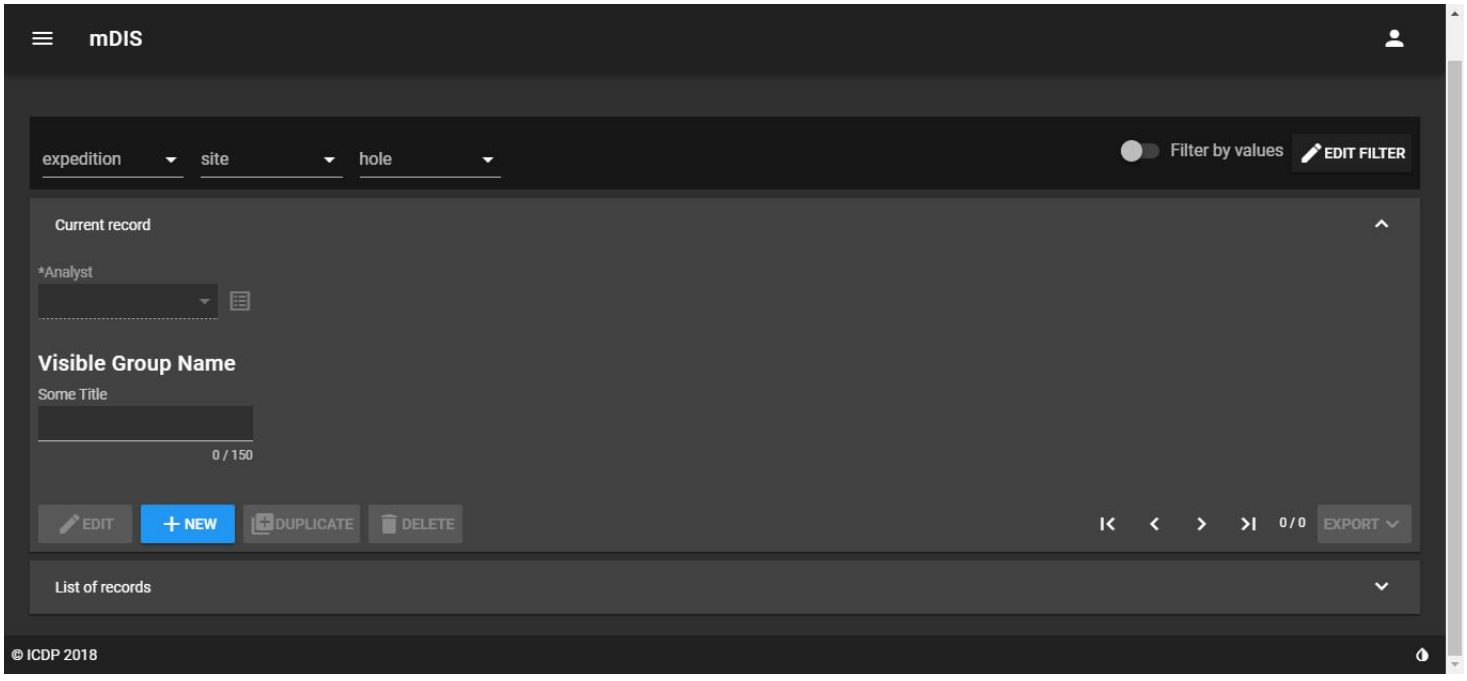
After saving the template, you will be redirected to the "Edit Template" page and the `SAVE & GENERATE` button will be enabled. Click on it to generate the required files.



```php
Generating code using template "/app/backend/modules/cg/generators/DISForm/default"...
 generated /app/src/forms/SamplesForm.vue.generated
 generated forms/SamplesForm.php
 generated forms/SamplesFormSearch.php
done!
```

1. `forms/SamplesForm.php` - an empty class that inherits from `generated models/SaSample.php`. If there were some specialization that belongs to this specific form and not the all forms that belongs to the same data model, it should be implemented here.
2. `forms/SamplesFormSearch.php` - an empty class that inherits from `generated models/SaSampleSearch.php`. same as (1), but this class contains specializations about how the form should be filtered.
3. `src/forms/SamplesForm.vue.generated` - this file is the template that should be used for form specialization that cannot be done through the templates manager e.g. adding a picture beside the field or giving the `remarks` field more width. the `.generated` extension should be removed and `npm run build` should be called on the terminal afterwards to build your form.

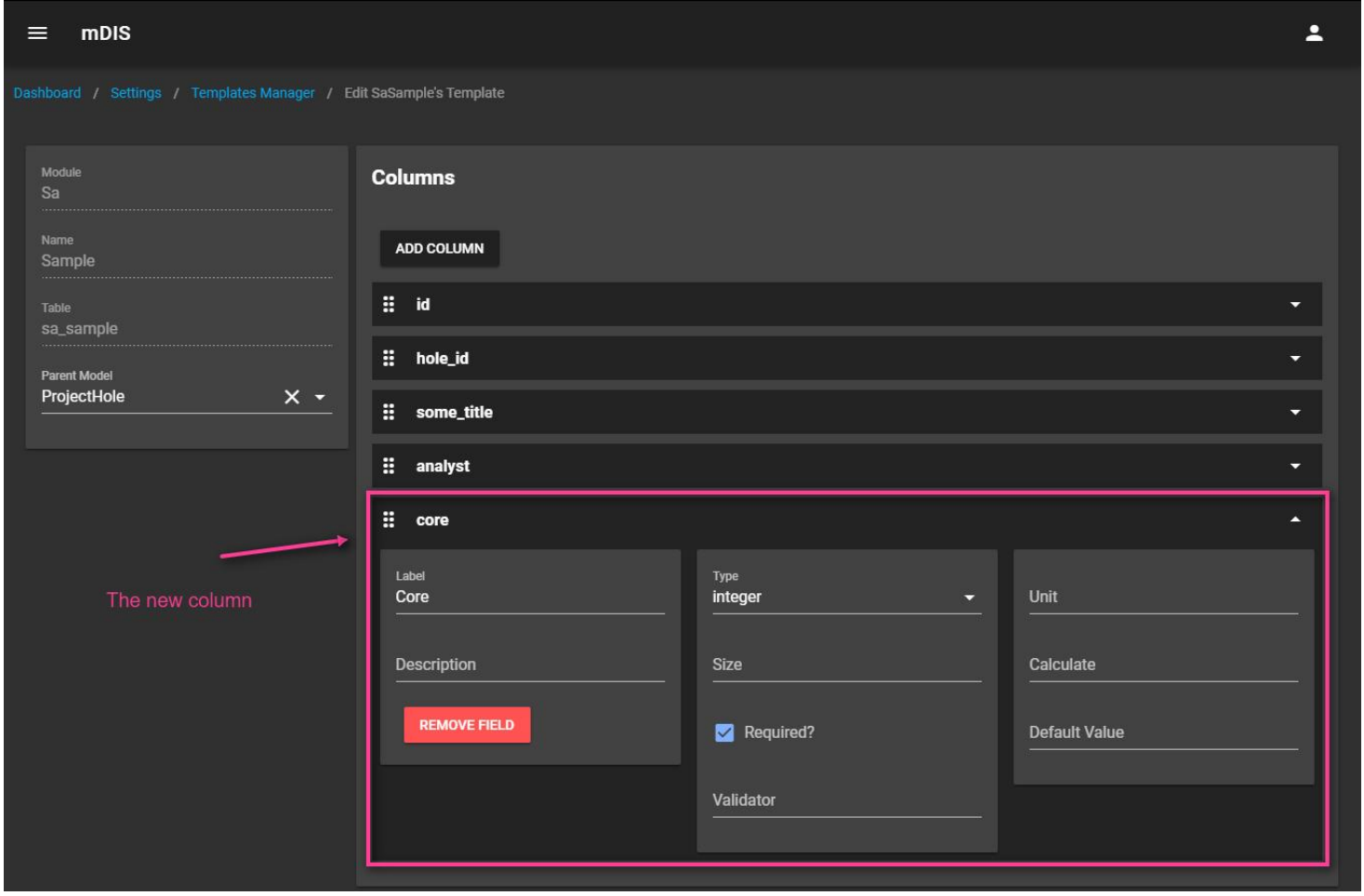
See **background document** to learn more about the PHP class hierarchy.



> How the generated form looks like. Notice the group names.

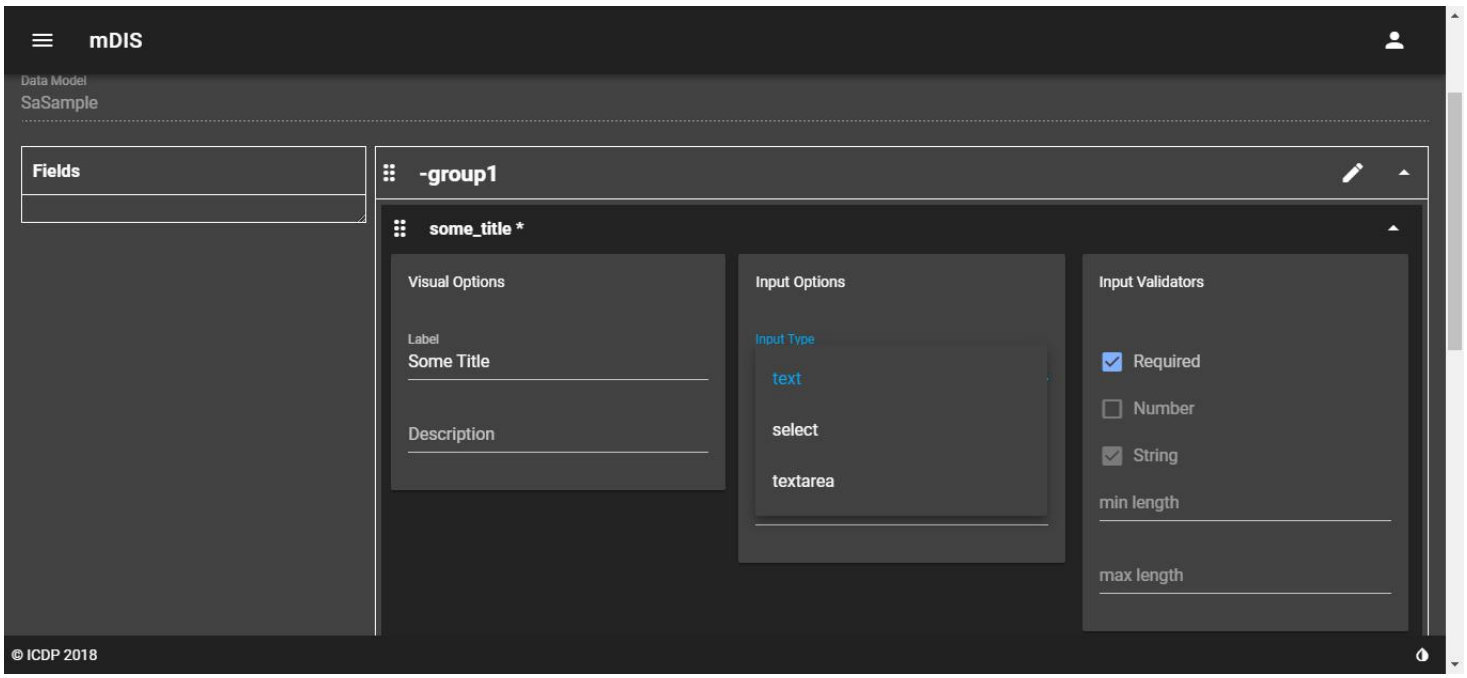
Update since v1.2.0-beta.0

Since this version some features and enhancements have been added to the form template manager. To present this features we will add a column to the previous example. the column called `core` and it is of type integer and expected to store a core id taken from the core table.



Filtered form input and auto validators

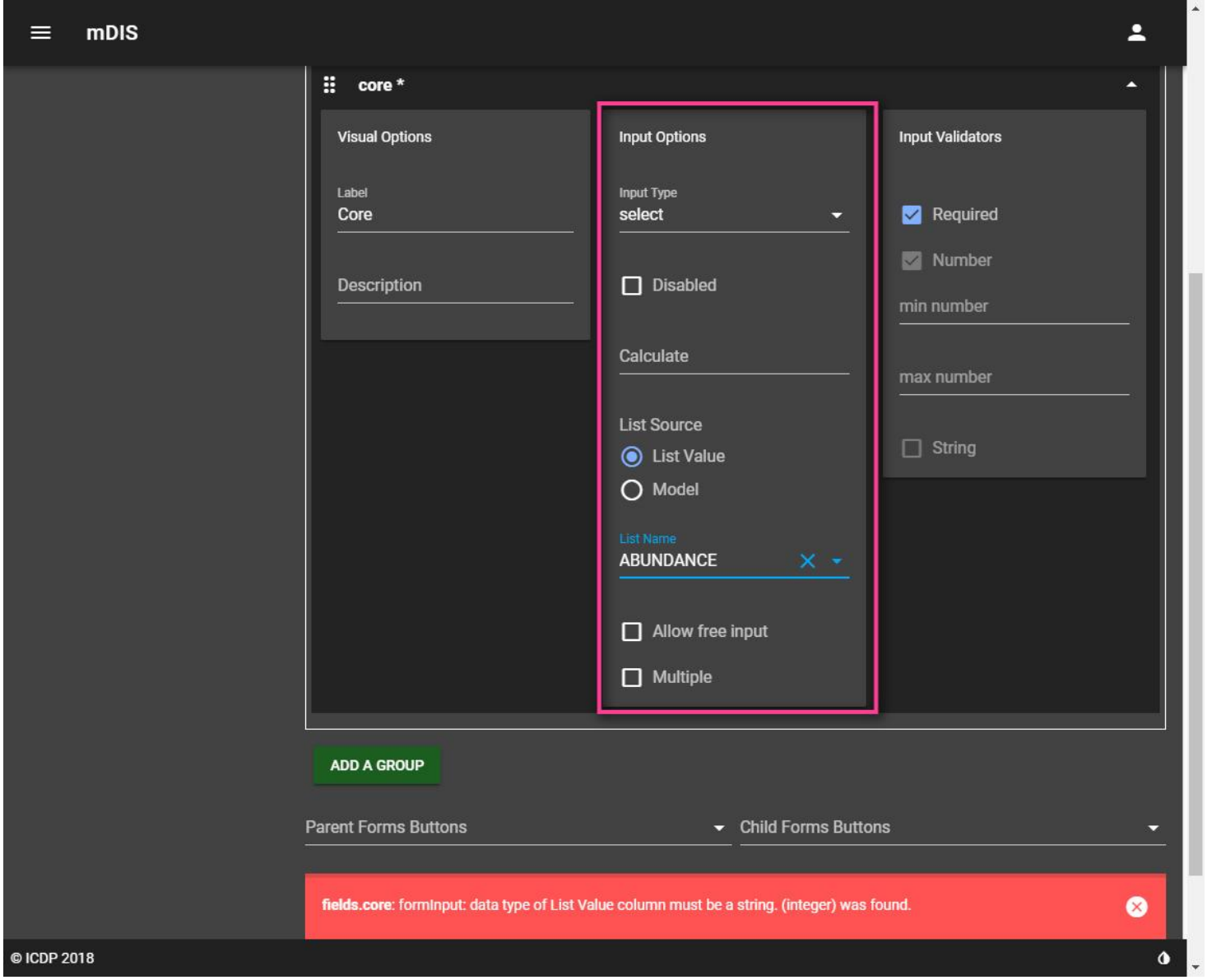
Now when you navigate to the corresponding form template page you will notice the following:

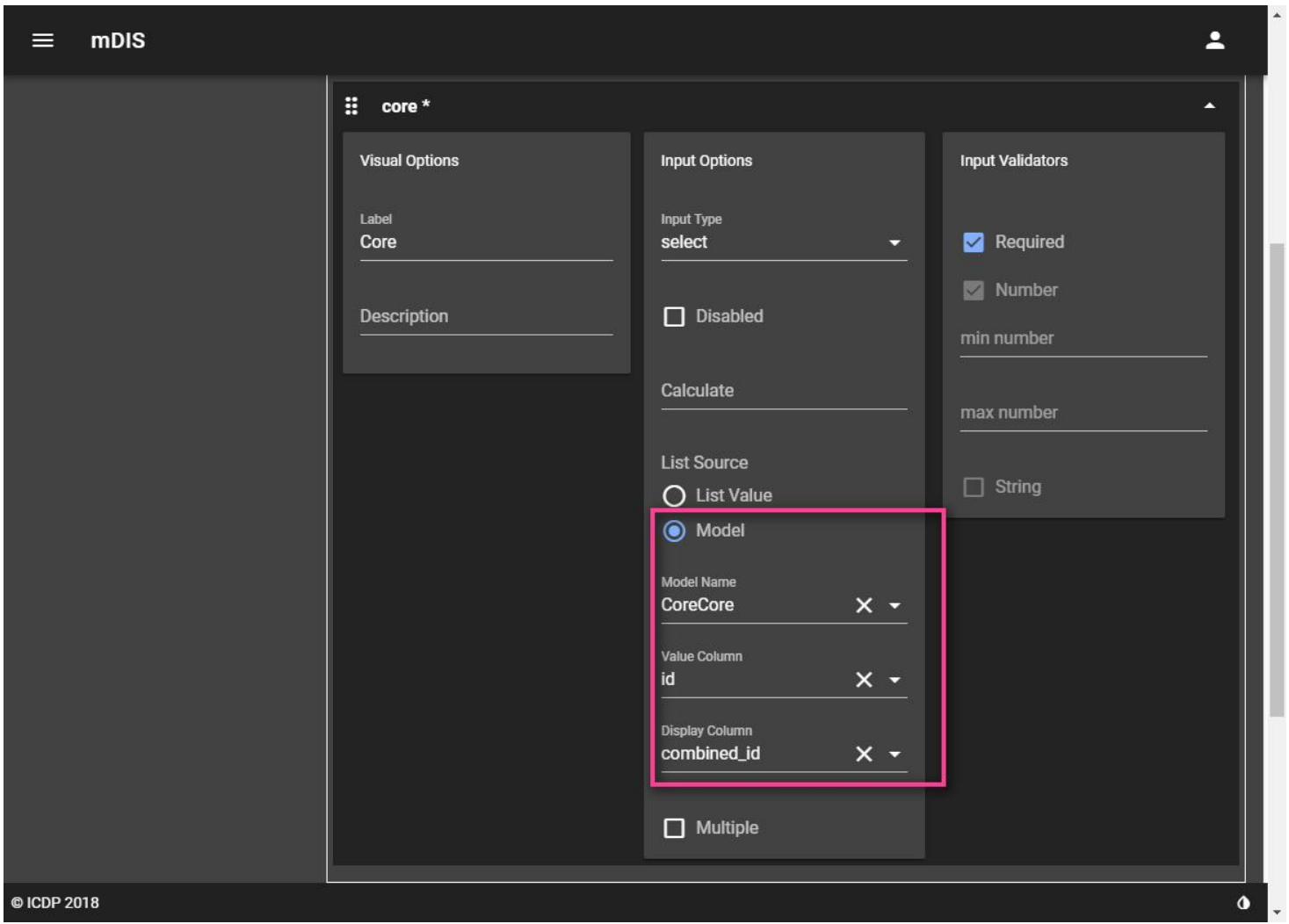


1. The list option of the input type are filtered and only valid input type for a column that holds a string value are shown.
2. The string and number validators are checked automatically based on the column data type. The min/max length are still available to change.

Select Source

For the field `core` , change the value of the Input Type to `select`

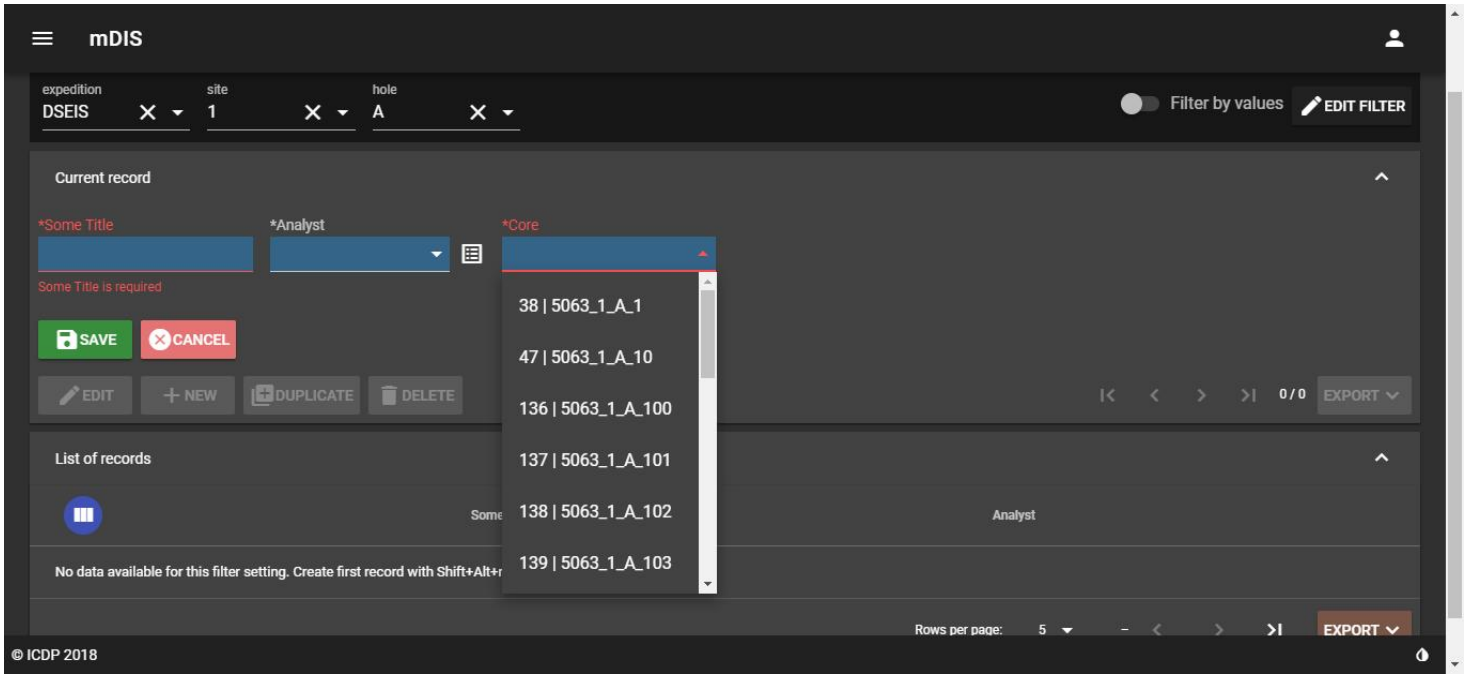

mDIS
core *
Visual Options
Label
Core
Description
Input Options
Input Type
select
Disabled
Calculate
List Source
List Value
Model
List Name
ABUNDANCE
Allow free input
Multiple
Input Validators
Required
Number
min number
max number
String
ADD A GROUP
Parent Forms Buttons
Child Forms Buttons
fields.core: formInput: data type of List Value column must be a string. (integer) was found.
© ICDP 2018


mDIS
core *
Visual Options
Label
Core
Description
Input Options
Input Type
select
Disabled
Calculate
List Source
List Value
Model
Model Name
CoreCore
Value Column
id
Display Column
combined_id
Multiple
Input Validators
Required
Number
min number
max number
String
© ICDP 2018

There is a new source for the list. **ValueList** and **Model**.

- The "ValueList" option will generate the options list from the `list_values` table based on the **List Name** value. The data type of the column must be a string since the `list_values` table's value are of String data type. Therefore, when trying to save the template you will get a type mismatch error.
- The "Model" option will generate the options list form the table of the selected model (CoreCore in the screenshot). Additional fields are available when Model is selected as the List Source. Make sure that the **Value Field** data type is matching the column data type, otherwise you will get a type mismatch error.

A select input will be generated in the form which gets the options list from the `core_core` table.



Templates Manager in PHP Code

Implementation

For the codebase of the Templates Manager GUI, see directory `backend/components/templates/` for both

- the tableset designer (4 .php-files)
- the forms designer (2 .php-files)
- base Classes (3 .php files)
- `BaseTemplate.php`
- `Component.php`
- `FormTemplateField.php`
- `FormTemplate.php`

- `ModelTemplateBehavior.php`
- `ModelTemplateColumn.php`
- `ModelTemplateForeignKey.php`
- `ModelTemplateIndex.php`
- `ModelTemplate.php`

For implementation details of the code generators, see directories `backend/modules/api/`, `backend/modules/cg/` and file `backend/config/web.php`. The Forms code-generator is class file `backend/modules/cg/generators/DISForm/Generator.php` and the Models code-generator is class file `backend/modules/cg/generators/DISModel/Generator.php`.

Extra Documents

For a detailed explanation of the structure of the PHP code generated and the 9-level class hierarchy involved, read this **background document**.

More about Gii, the Yii extension for code generation.

insolita/yii2-migration-generator: A third-party Gii extension that creates Migration Scripts for tables and models. Maintained until at least 2018.

Developer page

Form Template

Links

Developer page

Templates Manager page

Form Templates `*.json` File Structure

Which template properties are available in a model `.json` definition file, or in a form `.json` definition file?

- For data **models** templates structure, see **here**.
- For **forms** templates structure, see this file. For reference, also open this file **cores.json** to the side, in a new browser window.

In the table below, the values in column "Property" correspond to keys in a `*.json*` file, e.g. `backend/dis_templates/forms/cores.json`. They can also be found as `props` in `*.vue` files, e.g. `src/forms/CoresForm.vue.generated`.

The key ordering in the `.json` file is arbitrary, and can deviate from this order.

Property	Type	Required	Nullable
name	`string`	Yes	No
dataModel	`string`	Yes	No
fields	`array` of **Field**	Yes	No
filterDataModels	`object` see **details**.	Yes	No
requiredFilters	`array` of **RequiredFilter**	Yes	No
subForms	`object` see **details**.	Yes	No
supForms	`object` see **details**.	Yes	No
group	`string`	Yes	No
order	`integer`	Yes	No
createdAt	`number`	No	Yes
modifiedAt	`number`	No	Yes

Property	Type	Required	Nullable
generatedAt	`number`	No	Yes

Form name

The name of the module that this model belongs to or, in other words, the table groups that this table belongs to.

- is **required**
- type: `string`
- case: **kebab-case**

Form dataModel

The name of the data model that the form manages e.g. `CoreCore`

- is **required**
- type: string

Form fields

An array of fields definitions that the form has. Every object of this array is of **Field** type. The array must have at least fields for the required fields of the table.

- is **required**
- type: `array` of **Field**

Field Definition

Property	Type	Required	Nullable
name	`string`	Yes	No
label	`string`	Yes	No
group	`string`	Yes	No
order	`string`	Yes	No
description	`string`	No	Yes
validators	`array` of **Validator**	Yes	No
formInput	`object` of **FormInput**	Yes	No

Field name

The name of the field is the name of the column it edits. This name is copied automatically from the model template when using the templates manager.

- is **required**
- type: `string`

Field label

The label of the field is copied automatically from the model template when using the templates manager.

- is **required**
- type: `string`

Field group

The name of the group. if the name starts with `-`, the group name will not be shown on the generated form.

- is **required**
- type: `string`

Field order

The order of the field in the group it belongs to.

- is **required**
- type: `integer`

Field description

This value will be shown as a hint under the fields in the generated form. This value is also copied automatically from the model template when using the templates manager.

- is **required**
- type: `string`

Field validators

An array of validators for the field. Every object of this array is of **Validator** type. An empty array is also accepted.

These validators are independent from, and simpler than, data model validation. The real validation happens on the backend when submitting the form. On the frontend only three validators are available: `required`, `string` and `number`.

- is **required**
- type: `array` of **Validator**

Validator Definition

Property	Type	Required	Nullable
type	`string`	Yes	No
max	`number`	No	No
min	`number`	No	No

Validator type

The type must be one of the following values: `required` , `string` or `number` .

- is **required**
- type: `string`

Validator max

This value works with `string` and `number` validators only and represents that maximum string length in `string` validator and the largest number the field could have for `number` validator.

- is optional
- type: `number`

Validator min

This value works with `number` validator only and represents that smallest number the field could have for `number` validator.

- is optional
- type: `number`

Field formInput

Defines the type of the input that should be used to edit the matching field/column. The value of this field should be a **FormInput** type.

- is **required**
- type: `object`

FormInput Definition

Property	Type	Required	Nullable
type	`string`	Yes	No
disabled	`boolean`	No	No
readOnly	`boolean`	No	No

Property	Type	Required	Nullable
calculate	`string`	No	Yes
allowFreeInput	`boolean`	No	No
multiple	`boolean`	No	No
selectSource	`object`	No	No

FormInput type

Defines the input type for this field. The type can be one of the following values: `text` , `textarea` , `switch` , `select` , `date` , `time` or `datetime`

> depending on the type of the input, other options are possible in the FormInput Definition. `allowFreeInput` , `multiple` and `selectSource` works only with the type `select` otherwise they will be ignored.

- is **required**
- type: `string`

FormInput disabled

Determines if the input should be disabled or not.

- is optional
- type: `boolean`

FormInput readOnly

Determines if the input should be read only or not.

- is optional
- type: `boolean`

FormInput calculate

This value has the same value in the model template, **see here**, and it will be copied automatically when using the templates manager.

- is optional
- type: `string`

FormInput allowFreeInput

Determines whether the user is able to write a value that does not already exists in the list.

> Only for `select` form input type.

- is optional
- type: `boolean`

FormInput multiple

Determines whether the user is able choose more than one value from the list.

> Only for `select` form input type.

- is optional
- type: `boolean`

FormInput selectSource

Defines the source from which the list items should be loaded. For now only one source is supported which is the list from the ListValues data model.

> Only for `select` form input type.

- is **required** when input type is select
- type: `string`

Example

since version **1.2.0-beta.0** There are two types of select sources: **list** and **api**. If **list** is the type of the select source then the list of options will be generated from the `list_values` table based on the value of `listName`. In this case, the values of `textField` and `valueField` must be *remark* and *display* an in the following example.

```json
{
    "selectSource": {
        "type": "list",
        "listName": "ANALYST", <--- the name of the list in ListValues data model
        "textField": "remark",
        "valueField": "display"
    }
}
```

If **api** is the type of the select source them the list of options will be generated from the model defined in the corresponding property. In this case `textField` and `valueField` values must be a name of a column that belongs to the defined model's table. `textField` and `valueField` can be identical.

```json
{
    "selectSource": {
        "type": "api",
```

```
        "model": "ProjectHole",
        "textField": "hole",
        "valueField": "id"
    }
}
```

Form filterDataModels

This field defines the filter components that will be generated in the filter form at the top of the form. This value will be automatically generated when using the templates manager.

> the order is important

- is **required**
- type: `object` of (key, value) pairs
  - key: is a unique `string` for the filter component. This string will be used as the label of the filter input
  - value: is an `object` of **FilterDataModel** type

FilterDataModel Definition

Defines the filter components data source and relation to other components.

Property	Type	Required	Nullable
model	`string`	Yes	No
value	`string`	Yes	No
text	`string`	Yes	No
ref	`string`	Yes	No
require	`object`	Yes	No

FilterDataModel model

The name of the data model of which the list items will be loaded.

- is **required**
- type: `string`

FilterDataModel value

The name of the data model field which should be mapped as a list item value.

- is **required**

- type: `string`

FilterDataModel text

The name of the data model field which should be mapped as a list item text.

- is **required**
- type: `string`

FilterDataModel ref

The name of the field of the data model needed to be filtered. the value of the filter component will be compared with this field.

- is **required**
- type: `string`

FilterDataModel require

This field defines the dependency of the filter component. Dependency means the filter list items relations e.g. **hole** list items changes when the selected item in the **site** filter component changes.

- is optional
- type: `object`

This relation is defined as follows:

```json
{
    "require": {
        "value": "...",
        "as": "..."
    }
}
```

- **value**: defines which filter component value to use to filter the dependent list (one of the filterDataModels keys)
- **as**: defines against which field of the dependent list's data model should this list be filtered upon.

Form requiredFilters

This array defines the minimum fields that are required to be sent when submitting a form. Every item of the array should be of **RequiredForm** object.

- is **required**
- type: `array` of **RequiredForm**

RequiredForm Definition

```json
{
    "requiredFilters": [
        {
            "value": "hole",
            "as": "hole_id"
        }
    ]
}
```

- **value**: which filter component value is required (on of filterDataModels keys)
- **as**: the field name of this value to be sent when submitting the form.

Form subForms

Will be generated automatically in the templates manager depending on the selected child forms.

- is optional
- type: `object` of (key, value) pairs
  - key: is a unique key (preferably the name of the form)
  - value: is an `object` of **SubForm** type

SubForm Definition

Property	Type	Required	Nullable
buttonLabel	`string`	Yes	No
url	`string`	Yes	No
filter	`array`	Yes	No

SubForm buttonLabel

The label of the child form button in the form.

- is **required**
- type: `string`

SubForm url

The target form url

- is **required**
- type: `string`

SubForm filter

An array of objects that defines the values of the filter components to append to the url.

- is **required**
- type: `array` of object. Every object has two properties:
  - unit: `string` defines the filter component (one of filterDataModels of the target form template)
  - formField: `string` defines which value of the selected item which is loaded in the form should be set to this filter component

Form supForms

Will be generated automatically in the templates manager depending on the selected parent forms.

- is optional
- type: `object` of (key, value) pairs
  - key: is a unique key (preferably the name of the form)
  - value: is an `object` of **SupForm** type

SupForm Definition

Property	Type	Required	Nullable
buttonLabel	`string`	Yes	No
url	`string`	Yes	No
parentIdField	`string`	Yes	No
filter	`array`	Yes	No

SubForm buttonLabel

The label of the parent form button in the form.

- is **required**
- type: `string`

SubForm url

The target form url

- is **required**
- type: `string`

SubForm parentIdField

Which field value should be passed as the selected item id of the parent form

- is **required**
- type: `string`

SubForm filter

An array of objects that defines the values of the filter components to append to the url.

- is **required**
- type: `array` of object. Every object has two properties:
  - unit: `string` defines the filter component (one of filterDataModels of the target form template)
  - formField: `string` defines which value of the selected item which is loaded in the form should be set to this filter component

Form createdAt

A timestamp of when the template was created

- is optional
- type: `integer`

Form modifiedAt

A timestamp of when the template was modified

- is optional
- type: `integer`

Form generatedAt

A timestamp of when the template was generated

- is optional
- type: `integer`

Model Template

Links

Developer page

Templates Manager page

Model Templates `*.json` File Structure

Which template properties are available in a model `.json` definition file, or in a form `.json` definition file?

- For data **models** templates structure, see here, this file. For reference, also open this file **CoreCore.json** to the side, in a new browser window.
- For **forms** templates structure, see **here**.

In the table below, the values in column "Property" correspond to keys in a `*.json*` file, e.g. `backend/dis_templates/models/CoreCore.json` .

The key ordering in the `.json` file is arbitrary, and can deviate from this order:

Property	Type	Required	Nullable
module	`string`	Yes	No
name	`string`	Yes	No
fullName	`string`	Yes	No
table	`string`	Yes	No
importTable	`string`	Yes	Yes
parentModel	`string`	Yes	Yes
columns	`array` of **Column**	Yes	No
indices	`array` of **Index**	Yes	No
foreignkeys	`array` of **ForeignKey**	Yes	No
createdAt	`number`	No	Yes
modifiedAt	`number`	No	Yes
generatedAt	`number`	No	Yes

Model module

The name of the module that this model belongs to or, in other words, the table groups that this table belongs to.

- is **required**
- type: `string`
- case: **PascalCase**

Model name

The name of the model.

- is **required**
- type: `string`
- case: **PascalCase**

Model fullName

Is a combination of the `module` and `name` properties, simply concatenation of these properties.

- is **required**
- type: `strings`
- case: **PascalCase**

Model table

The table name in the DB where the records of the data model are stored. Naming of the table should follow a strict rule that is it must be the same `fullName` property but written in a **snake_case** format.

- is **required**
- type: `string`
- case: **snake_case**

Model importTable

used only when migrating from a legacy mDIS.

- is optional
- type: `string`

Model parentModel

The **fullName** of the parent model

- is optional
- type: `string`
- case: **PascalCase**

Model columns

An array of columns definitions that the table has. Every object of this array is of **Column** type. The array must have at least one columns **id** that used as the primary key of the table.

- is **required**
- type: `array` of **Column Type**

Column Definition

Property	Type	Required	Nullable
name	`string`	Yes	No
importSource	`string`	Yes	Yes
type	`string`	Yes	No
size	`integer`	Yes	No
required	`boolean`	Yes	No
primaryKey	`boolean`	No	Yes
autoInc	`boolean`	No	Yes
label	`string`	Yes	No
description	`string`	Yes	No
validator	`string`	Yes	No
validatorMessage	`string`	Yes	Yes
unit	`string`	Yes	Yes
selectListName	`string`	Yes	Yes
calculate	`string`	Yes	Yes
defaultValue	`any`	Yes	No

Column name

The name of the column.

- is **required**
- type: `string`
- case: **snake_case**

Column importSource

used only when migrating from a legacy mDIS.

- is **required**
- type: `string`

Column type

the type of the column.

- is **required**
- type: `string`
- can only have one of the following values `integer` , `double` , `string` , `boolean` , `dateTime` , `date` , `time`

Column size

The size of the table column. This value has an effect only on `integer` and `string` columns.

- is **required**
- type: `string`

Column required

If this column is required. This will be generated as a validation rule in data models and frontend.

- is **required**
- type: `boolean`

Column primaryKey

> Used only by import.

- is **required**
- type: `boolean`

Column autoInc

> Used only by import.

Define the column as an auto-incremented primary key. Only **ONE** column can has this value set to true which is the `id` column i.e. this field should always be `false` .

- is **required**
- type: `boolean`

Column label

The value of this field is used as a field label in forms.

- is **required**
- type: `string`

Column description

The value of this field is used as a hint label in forms.

- is optional
- type: `string`

Column validator

This field defines the validation rules for a column value. It accepts the multiple formats. If the written format does not match one of the supported ones, it will be ignored.

- is optional
- type: `string`

Supported validator formats

- compare operators: `>` , `<` , `>=` , `<=` , `<>` , `!=` appended with the value to compare with.
- between ... and ...: for a range validator
- in [..., ..., ...]: for a enum validator
- LIKE regex: string regex validator

Column validatorMessage

A custom validation message to be shown in case the value was not valid.

- is optional
- type: `string`

Column unit

> Used only by import.

- is optional
- type: `string`

Column selectListName

> Used only by import

The name of the list to select the values from. This value will be used when generating the form to show a suitable select input.

- is optional
- type: `string`

Column calculate

Define this if the column value should be calculated based on other columns. It supports limited operations `+`, `-`, `/`, `*` and parentheses `()` and functions `ABS()`. The columns are represented by there names inside brackets e.g. `[bottom_depth]`

- is optional
- type: `string`

Column defaultValue

The default value of the column

- is optional
- type: `string`

Model indices

An array of indices definitions that the table has. Every object of this array is of **Index** type. The array must have at least one index on **id** column that represent the primary key of the table.

- is **required**
- type: `array` of **Index Type**

Index Definition

Property	Type	Required	Nullable
name	`string`	Yes	No
type	`string`	Yes	No
columns	`array` of strings	Yes	No

Index Name

Name of the index

- is **required**

- type: `string`

Index Type

Type of the index can have one of the following values: `PRIMARY` , `KEY` , `UNIQUE`

- is **required**
- type: `string`

Index Columns

An array of columns names that forms the index value.

- is **required**
- type: `array` of strings

Model foreignkeys

An array of foreign keyes definitions that the table has. If the model has a parent model a foreign key will be automatically generated in the template manager that represents the relation. Every object of this array is a **ForeignKey** type.

- is **required**
- type: `array` of **ForeignKey Type**

ForeignKey Definition

Property	Type	Required	Nullable
name	`string`	Yes	No
foreignTable	`string`	Yes	No
localColumns	`array` of strings	Yes	No
foreignColumns	`array` of strings	Yes	No

ForeignKey name

Name of the key

- is **required**
- type: `string`

ForeignKey foreignTable

The name of the foreign table

- is **required**

- type: `string`

ForeignKey localColumns

An array of columns names that forms the key value.

- is **required**
- type: `array` of strings

ForeignKey foreignColumns

An array of columns names in the foreign table that forms the reference value.

- is **required**
- type: `array` of strings

Model createdAt

A timestamp of when the template was created

- is optional
- type: `integer`

Model modifiedAt

A timestamp of when the template was modified

- is optional
- type: `integer`

Model generatedAt

A timestamp of when the template was generated

- is optional
- type: `integer`

Development

Recommended Tools for Software Developers

In 2019, the best-tested IDE for mDIS develpment is Visual Studio Code.

Visual Studio Code

Vue Development

- Extension "Vetur"
- **How to use the VueJS Devtools Browser Extension**

Php Development

- PHP Zend Extension Xdebug
- **How to to stepwise remote debugging with XDebug**

Tutorials: The Vue.js Devtools Browser Extension

For developers and expert users (but Vue beginners)

Inspecting and debugging Vue components

The VueJS devtools extension runs inside the Browser's Development Console.

The main use case for the **Vue.js Devtools Browser Extension for Chrome** is to explore Vue's nested component hierarchy on any VueJS-backed web page of the mDIS.

> **Important Constraint**
>
> The Vue Devtools Extension works **on Vue Development builds only**. Start mDIS with `npm run serve` ,not `npm run build` .

Further Constraints

- Again, the Vue Devtools Extension works **on mDIS Development builds only** (`npm run serve`), e.g, on **wb45** (rarely online).
- Of course, any **changes** made (e.g. to the Vuex Store) inside the Console Panel of the Extension are **not permanent**. Vue Devtools are just a way to figure out how a feature on an mDIS web page is implemented, and what changes are possible in general. The extension enables users to explore Vue's support for **Reactive Programming**: *after input changes, or config changes, dependent objects on the web-pages self-update immediately*.
- **Paths shown in the pictures below might have been changed** during ongoing redesign of the mDIS application.
- This tutorial has been written for **Google Chrome** in 2019. There exists also a Vue.js Devtools Firefox Extension. It works similar to the Chrome extension, but the Chrome devtools are more powerful and easier to use.

Installation of the Extension

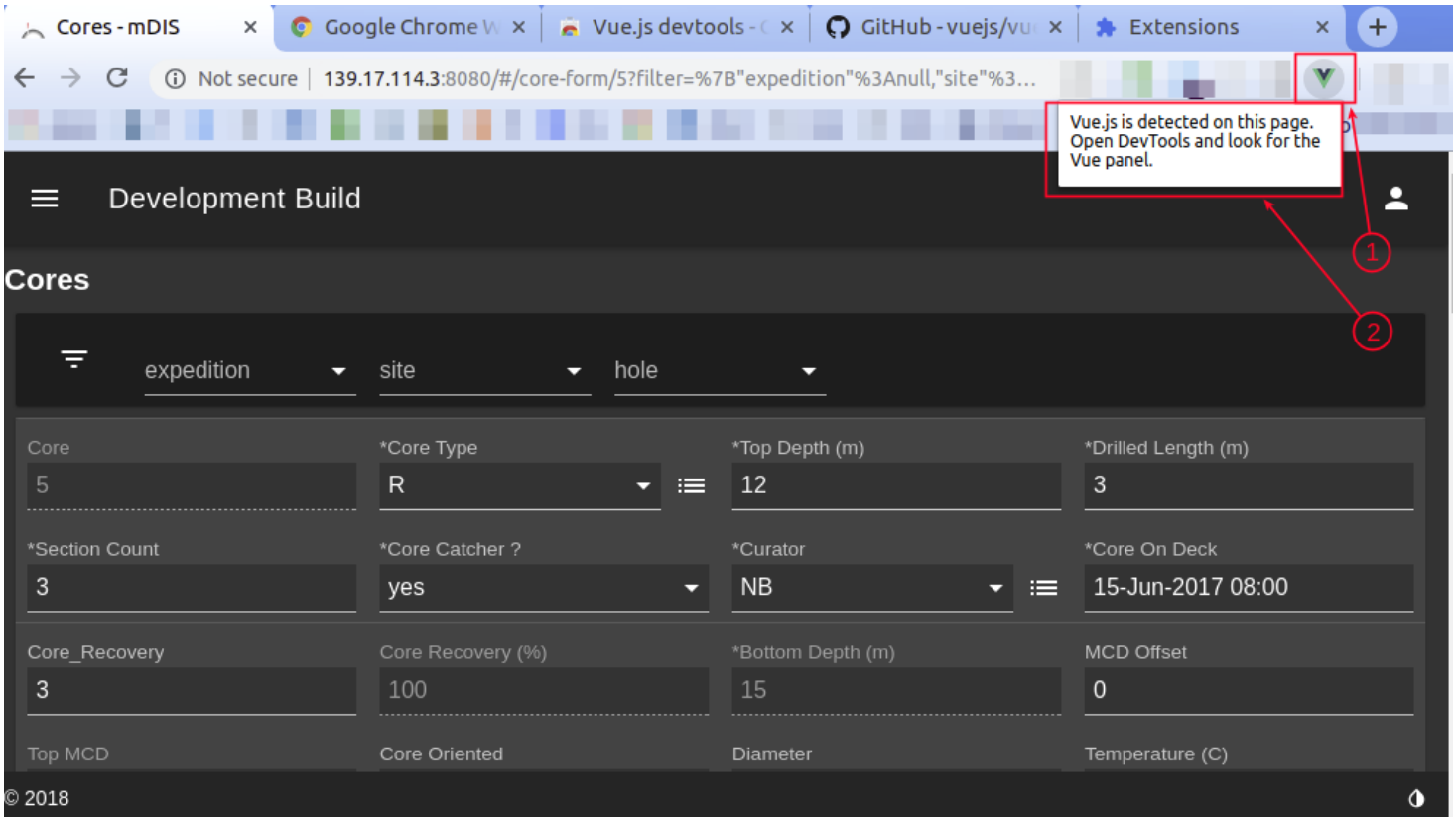
If you haven't done so, install the **Vue.js Devtools Browser Extension for Chrome**. This is a one-time task.

1. Open Google Chrome, go to the **Chrome Web Store**, search for Vuejs. The extension can be found **here** but this particular link might change
2. Click on the blue "Add to Chrome" button in the upper right corner.
3. (optional) Take note of the **Extension's Documentation** on Github.com. (There are some animated GIFs.)
4. (optional) Go to Chrome's **extension management panel**. On the Vue.js Devtools page, click on the white "Details" button. Click "Allow access to file URLs" for the Vue Devtools extension. You can temporarily

disable the extension here, too; or make sure that the extension is actually enabled.

Usage of the extension (general case)

1. Go to any mDIS Web page of the **Development Build** in the ICDP/GFZ intranet.
2. (see screenshot below) There should be a green "V" icon next to the Browser's Address bar (1). Make sure it is green and not grey. Reload page if necessary. Move the mouse over it and make sure that the tooltip (2) looks like as shown in the screenshot below.



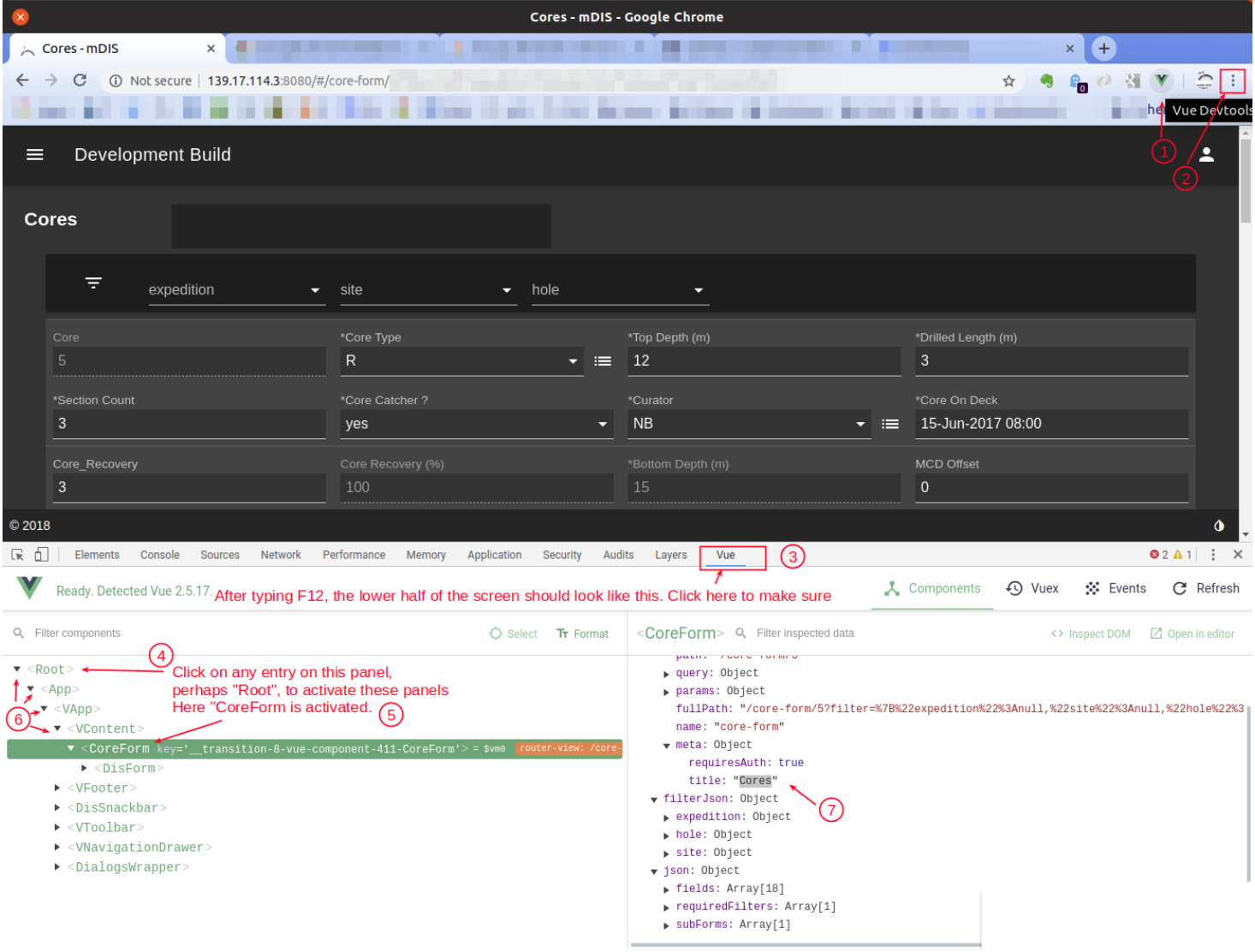
Finding the "Vue" panel in the Chrome Developer tools

(see screenshot below). Go to any mDIS data-entry page. (The Login Page is too basic and not recommended). Make sure the Vue.js Devtools Icon is green (1). Open the Chrome Developer Tools panel. To do so, hit the `F12` key (alternatively, `Ctrl+Shift+I`). Alternatively, with the Mouse, click the "vertical ..." Menu Icon of Chrome, choose the .../More Tools.../Developer Tools menu item (2).

Revealing the nested component structure of a Vue Application

This is the main use case for this extension.

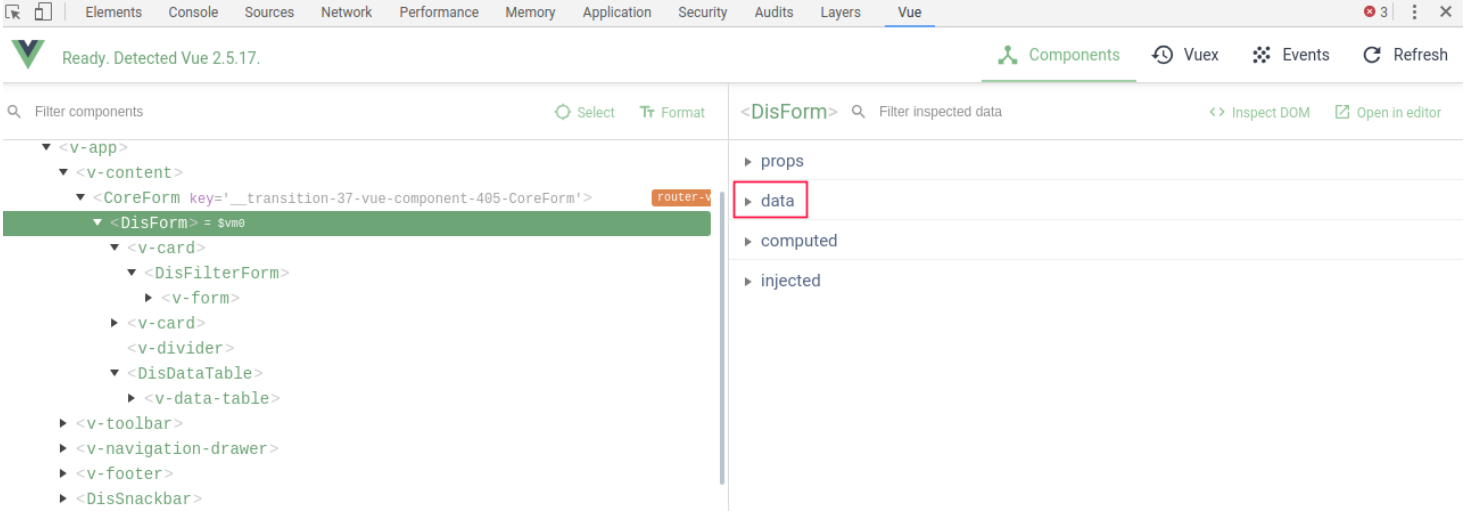
(see screenshot below). After the devtools panels opened, you should see something like in the screenshot below. click on "Vue" in the middle bar (3). Click on " `<Root>` " in the left half in the panel (4). **You can explore Vue's nested component hierarchy on that page by clicking on the black triangles (5) + (6).** The data, methods and properties associated with each component are visible on the right half of the Vue developer panel. A small subset of these items (7) is also *editable* here. See below.



Attention: The devtools extension might stop working, e.g. after some period of inactivity. Then nothing is selectable or editable any longer. If it is not editable, reload page; or close Chrome and navigate to your page again.

The 'properties panel' in the lower right

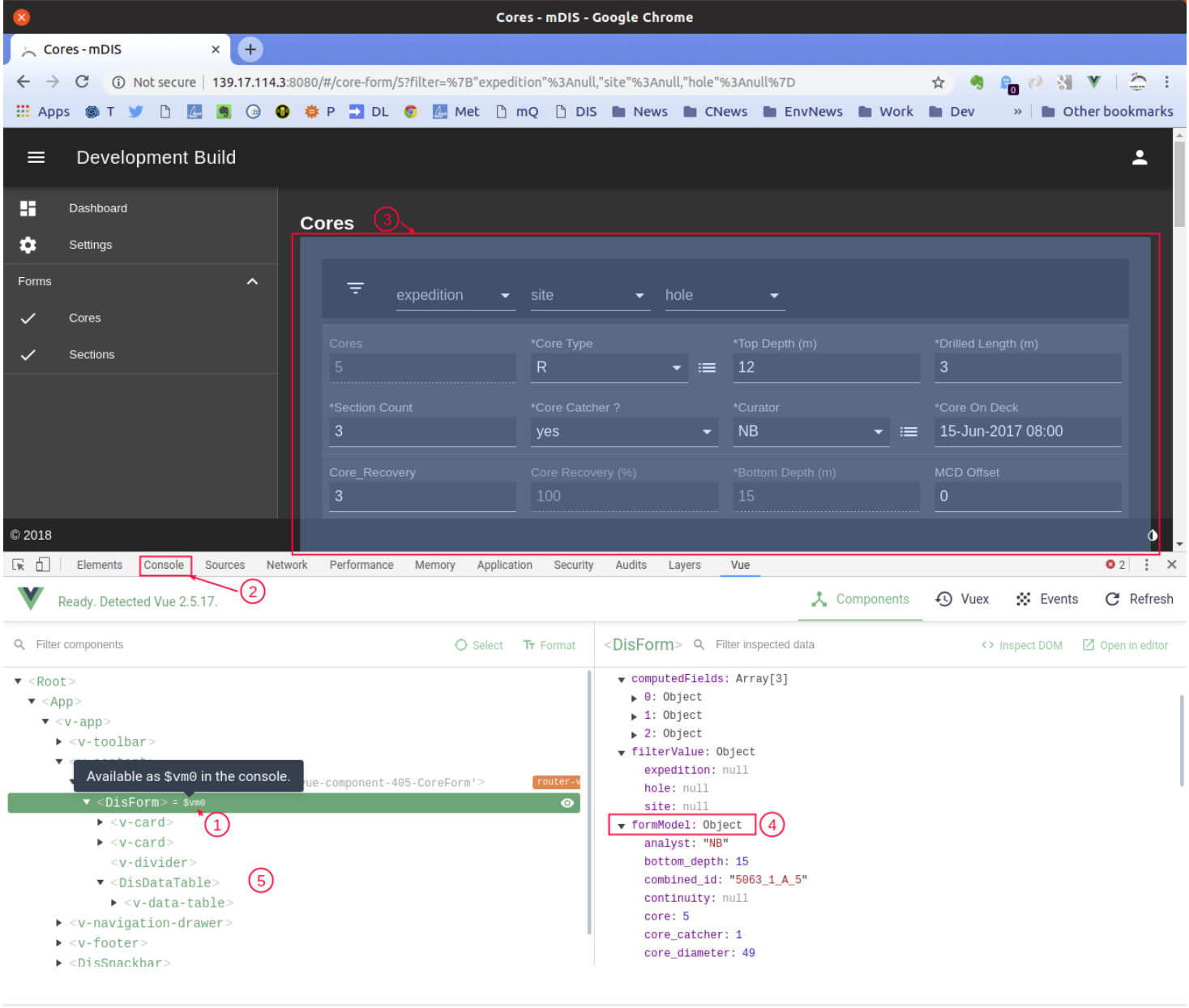
The panel in the lower right is similar to the properties page in the Visual Basic IDE (legacy DIS users). When the elements inside this panel are all collapsed, there are up to 4 main elements, `props`, `data`, `computed`, and `injected`. The most important is `data`, because its sub-elements are (sometimes) editable. Click on "data" to expand it.



Interesting: You can also click on "open in editor" icon in the top right corner of the right Vue panel.

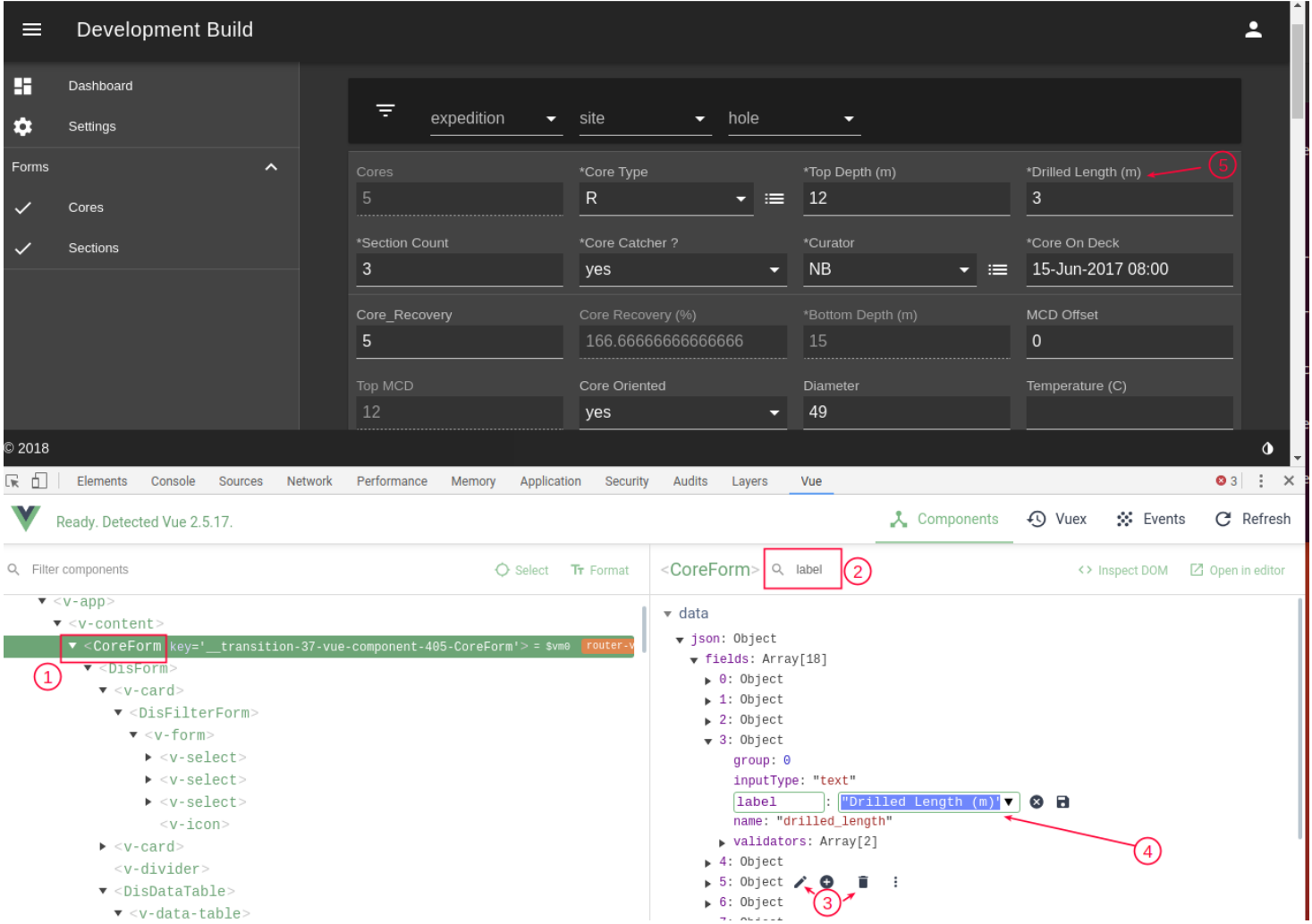
Using the Vue "Components" panel in the Developer Tools for Chrome

1. (see screenshot below) Use Black triangles to drill down to "DisForm" item, click on it to highlight it with a green background (1).
2. (optional) Hold mouse over `= $vm0` shown in white font inside the green bar (1). A black tooltip "Available as $vm in console" appears (1). This console would become visible after clicking tab "Console" (2). *Do not* click on "Console" at this time. This will be explained further in another section below, in section *Cool uses*.
3. After clicking "DisForm", the corresponding Vue component is highlighted inside the body of the black web page (3). Here it is the form that will posted to the web server, for processing with Yii/PHP.
4. The form data currently selected (and its field names as known to PHP are shown in the lower right panel. Click on "formModel" (4). The form field data is shown here. These fields are editable inside the lower right hand panel. The fields are also reactive, thus browser will self-update the current web page. Click formModel/core_recovery, change value, see it updated instantly in the body of the web page (upper half, black background). See also dependent values getting updated instantly, e.g. "core recovery "%").
5. (optional) The DisDataTable component (5) represents the Section child table. (It is not hightlighted in the screenshot.)



Use case example: How to edit the Labels of an mDIS Data Entry Form

1. In the lower left panel, click on "Core Form" (1)
2. In the lower right panel, enter "Label" in the search box. (2)
3. In the data/json/fields/ array, open any object (e.g. `3: Object`). Move the Mouse over the items, watch a pencil icon appear and a trashbin icon (3). Click on the *value* for the label (4). Change its value to something else. The (black) Web page should update instantly (5).



Of course this can also be done with Javascript code. See below.

Cool uses of the command line to interact with the DIS Vue App

Editing the panel as mentioned in the picture above requires a lot of clicks, and does not always work properly. It is easier to enter javascript commands in the 'Console' tab.

Prerequisite: Click on any `<...>` element in the left panel, such that it gets a dark green background, thus activating the $vm0 variable in the devtools.

Purpose	Left Panel, Click on Element	Javascript Commands in Console
See querystring parameters	`<ROOT>` , any	`$vm0.$route.query.filter`
Get last part of route	`<ROOT>` , any	`$vm0.$route.params`
open "drawer" (left sidebar)	`<App>`	`$vm0.$data.drawer = true`
close "drawer" (left sidebar)	`<App>`	`$vm0.$data.drawer = false`
change a label in group1	`<DisTextInput>` or `<v-text-field>`	`$vm0.$props.label = "Drilled Length (km)"`

Enter the commands from the rightmost column in the Javascript Console.

How to use the other GUI elements in the VueJS Devtools, e.g. "Vuex", "Events" tabs

Vuex tab - Vuex is a Vue library that is being used for state management. Vuex maintains data about the logged-in user and the session, for instance.

Events tab - this panel shows if there are custom events defined inside the Vue app. A component may emit custom events, and other components may listen for this event, and do something, e.g. update themselves. This `Event` tab mechanism facilitates communication among the components, especially from nested components ".

See also

Debug VueJS Pages with Chrome

This only works in `npm run serve` mode, not `npm run build` :

- How to use the Debugger for Chrome extension with VS Code to debug Vue.js applications generated by the Vue CLI. **By Microsoft / on Github**. Works for me. An alternative way to take a peek under the hood of a Vue Application, and see javascript being executed.

How to do stepwise remote debugging with Xdebug

Xdebug is an extension for PHP to assist with debugging and development.

From the XDebug Homepage

- It contains a **single step debugger** to use with IDEs. (**This topic is the most important for mDIS debugging**)
- it upgrades PHP's var_dump() function
- it adds stack traces for Notices, Warnings, Errors and Exceptions
- it features functionality for recording every function call and variable assignment to disk
- it contains a profiler
- it provides code coverage functionality for use with PHPUnit

Installation

Installation for Visual Studio Code

For remote debugging, there are two hosts who must communicate during a debugging session: (1) The mDIS Server, and (2) the developer's workstation, where the mDIS codebase is installed.

mDIS Server

On the mDIS Server, the **XDebug Installation Process** is straightforward. Do that first.

Instructions for Ubuntu Linux 18.04 and 20.04

```sh
sudo apt install php-xdebug
apt-file show php-xdebug # lists 4 files in package.
sudo phpenmod xdebug
sudo apachectl restart
# check contents of the .ini file, add a few lines as mentioned below
```

The steps for installations on other Linux distributions or platforms are not repeated here. Please read the xdebug documentation.

We have used xdebug v2.9.0 from late 2019.

Create a `phpinfo()` page.

On the server, a configuration file `xdebug.ini` should be created and stored in an appropriate location, e.g. where the other php-ini files are stored. See `phpinfo()` output. The absolute path should look similar to this one: `/etc/php/7.2/apache2/conf.d/20-xdebug.ini` . Do a `phpenmod xdebug` and restart the web server. Check the `phpinfo()` page for xdebug entries.

The contents of `xdebug.ini` should look like this:

```
; can also give an absolute path to .so file
zend_extension=xdebug.so
xdebug.remote_enable=1
xdebug.remote_autostart=1
xdebug.remote_connect_back=1
;;;ICDP-OSG Intranet
;xdebug.remote_host=139.17.81.xx
;;; avoid standard port (9000) conflicts by using 9005 port
xdebug.remote_port=9005
;;; my preferred editor
xdebug.idekey = VSCODE
;xdebug.remote_log = /var/tmp/xdebug.log
```

Alternatively, you can also set `xdebug.remote_connect_back=0` and remove the commented `;xdebug.remote_host=139.17.81.xx` .

Xdebug Troubleshooting

The xdebug logfile might be reachable at a directory similar to this one:

`/var/tmp/systemd-private-SOME_ID-apache2.service-Ify9fj/tmp/xdebug.log`

```
sudo tail -f /var/tmp/systemd-private-9b637db8485f46859a97389f791b08b6-apache2.service-Ify9fj/
[1034122] Log closed at 2020-06-09 13:48:32

[1034122] Log opened at 2020-06-09 13:48:32
[1034122] I: Checking remote connect back address.
[1034122] I: Checking header 'HTTP_X_FORWARDED_FOR'.
[1034122] I: Checking header 'REMOTE_ADDR'.
[1034122] I: Remote address found, connecting to 188.100.66.140:9005.
[1034122] E: Time-out connecting to client (Waited: 200 ms). :-(
[1034122] Log closed at 2020-06-09 13:48:32

```

If your server is protected by a firewall, you might need to open port 9005 or whatever port yoo prefer to listen on for xdebug. You might also need to disable the software-firewall on your own machine, or add a new rule.

mDIS Client

On the client, the PC workstation where the developers want to carry out the debugging session, some files need to be set up as well.

First, a copy of the codebase of mDIS that runs on the server must be present on the client. Both client and server should use the same codebase. To do so, simply clone the same git repository.

In VS Code, there should be should be a workspace created.

In the client directory, where the root folder of the mDIS codebase resides, there must be a directory `.vscode`. If it is not there, create it.

Inside `.vscode` ,create a file `launch.json` with this content:

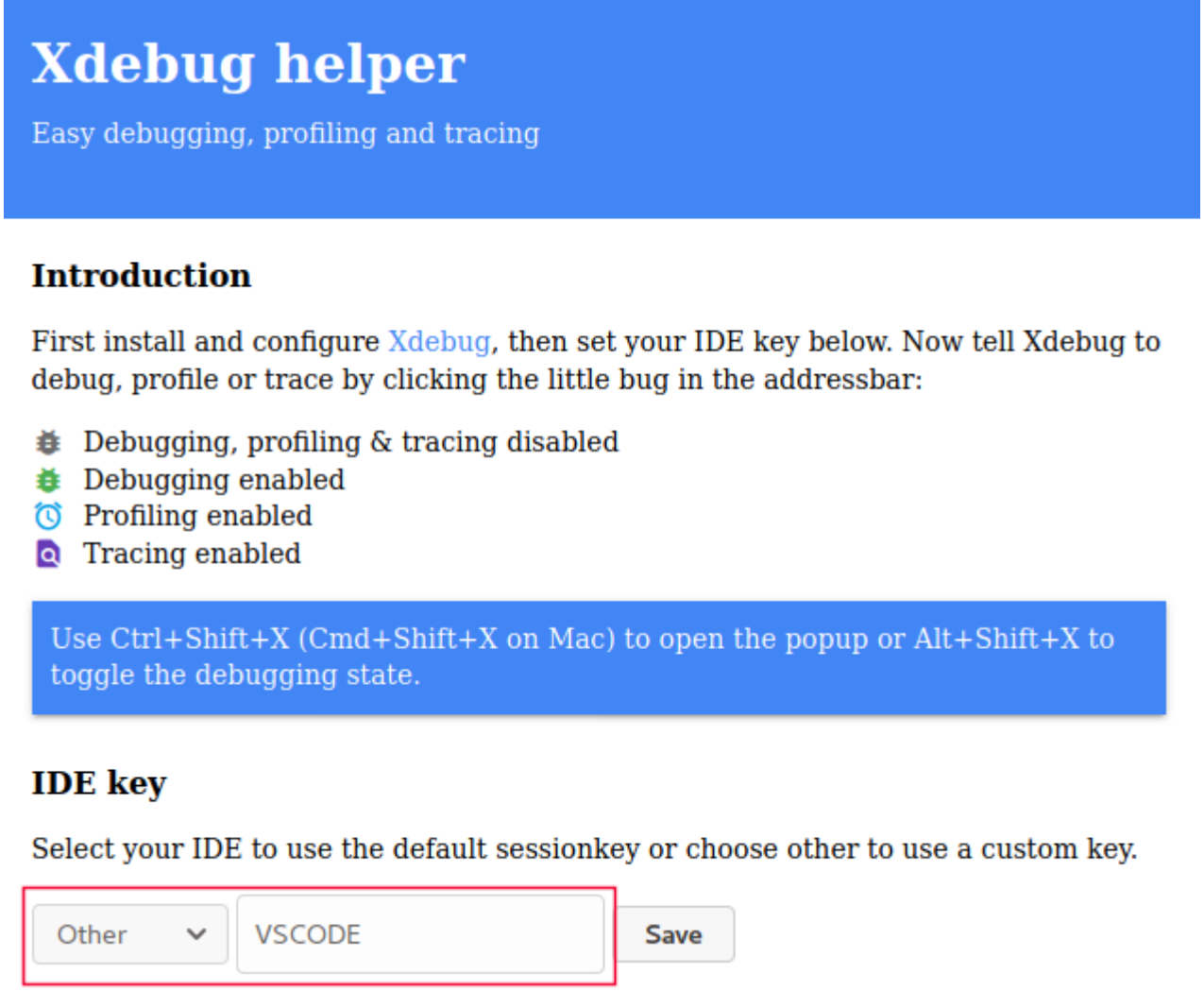
```json
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "php:xdebug",
            "type": "php",
            "request": "launch",
            "port": 9005,
            "stopOnEntry": false,
            "pathMappings": {
                "/var/www/dis": "${workspaceFolder}",
                "/app": "${workspaceFolder}/"
            },
            "ignore": [
                "**/Vendor/**/*.php"
            ],
            "xdebugSettings": {},
            "log": false,
            "hostname": "127.0.0.1"
        }
    ]
}
```

Adapt the `pathMappings` according to your needs. Here two settings are predefined, for two different mDIS Servers. On mDIS Server 1, the PHP code resides in `/var/www/dis` , on mDIS Server2 (a Docker container) the PHP code resides in Docker volume `/app` . They are mapped to a relatively simple VSCode Workspace, one without the "Add Folder to workspace" feature used. On the left side a directory that enables the debugger to be able to step into `web/index.php` .

Browser Extensions

Install Extension **"xdebug-helper" for Google Chrome** or the equivalent extension **"xdebug-helper for Firefox"**.

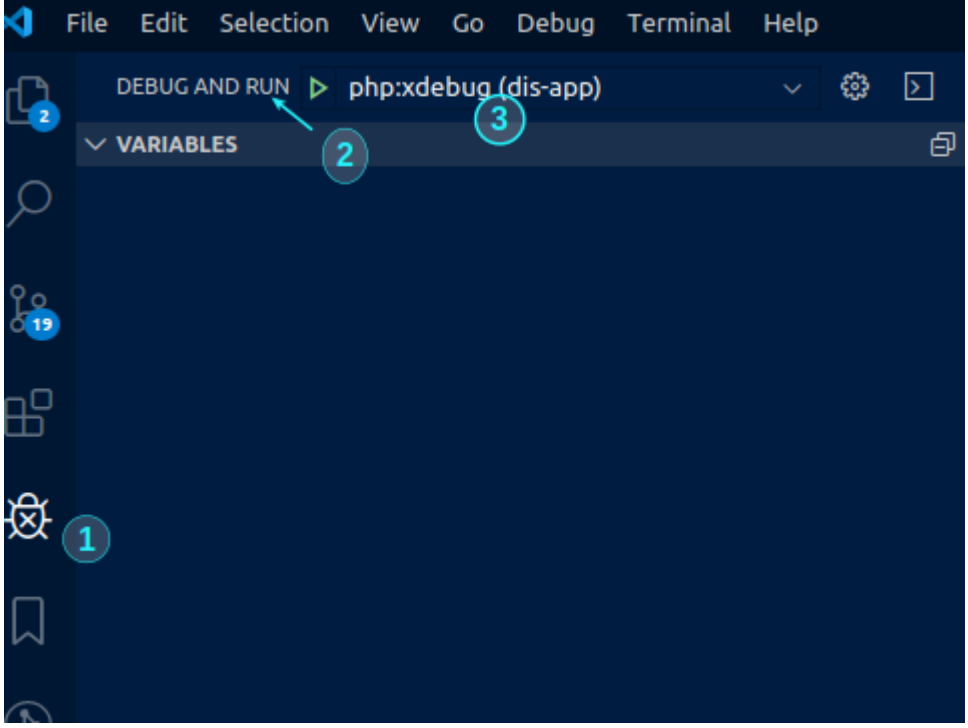
In the respective configuration page ("Preferences") of the Browser-Add-On, set the IDEKey Config Value to "VSCODE:



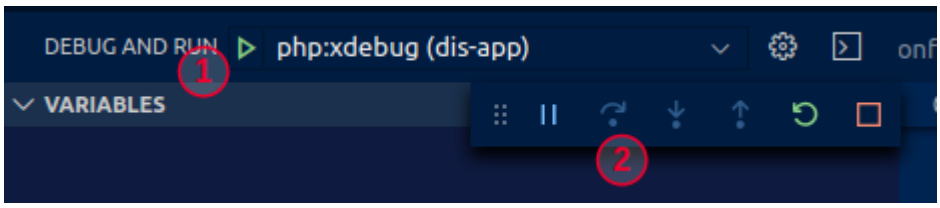
Look for the icon of this extension in the Browser Address Bar. Setting it to green is enabling it. The extension tells the browser to start or "switch on" a debugging session on the server and send some additional data with each HTTP request.

Debugging

Start your debugging session by clicking on the "Debug" Icon in the Sidebar of the VSCode IDE (1). Then click on `"Debug and run"` (2). The debug configuration selected (3) should be `php:xdebug` .



After clicking on "Debug and Run", see red (1) below, the DebugBar should appear.



To determine where Xdebug looks for on your development workstation, set keys `stopOnEntry` to `true` and `log` to `true` temporarily. Then click on "Continue" or "StepOver" icon in the Debugbar, see (2) in figure above. If your `pathMapping` is wrong, a dialog box with an error message will appear, and tell you where Xdebug looked for PHP files on your server, and why it didn't find them, most likely because of an incorrect path mapping.

Happy debugging!

(TBC)

git commands

Version control software, such as `git`, is an important tool for programming. Git allows you to keep track of what you did when, undo any changes you have decided you don't want, and collaborate at scale with other people.

For details, see wiki document **"Version Control with git"**, and the official **Git documentation**, the "git book".

The most important `git` commands are listed here, but they are not necessarily always applied in this order.

Most common *git* commands

Getting the current state of the repository

- `git status`
- `git log -5` - show last 5 commits (date, id and message)
- `git log <path> -5` - show last 5 commits of a single file/path (with hashes)
- `git diff <filename>` - compare changed file with last committed version in repo
- `git diff hash1..hash2 <filename>` - display changes between older versions

Managing local changes

Adding things:

- `git add .` - add everything what's new or changed to the "index" of the repository
- `git commit -m "an informative message"` - really make contents of "index" persistent and traceable

Undoing changes:

A single file:

- `git checkout <filename>` - restore latest committed version from repository, overwriting current file

Keeping two versions in parallel:

- `git branch` - create a branch (locally)
- `git checkout <branchname>` - change current "working directory" to `branchname`
- `git merge <frombranch> <tobranch>` - keeps perfect traceability
- `git pull --rebase` - get most recent changes and "flatten" the graph
- `git push --force` - send the rebased, simplified graph to remove repo. Must force

Less important

Less important or common commands, also for managing local changes

- `git init`
- `git version`
- `git ignore *.pdf` - (in pkg "git-extras") do not put PDF files under version control.

Working with a remote repository

- `git clone` - download and make first copy of remote branch
- `git fetch` - refresh metadata info. check what's new remotely (but don't apply changes)
- `git pull origin master` - get changes from remote master branch and apply them to local branch
- `git push`

Sys-Admin page

Version Control with `git`

What is version control?

A version control system is a tool that manages changes made to the files and directories in a project. Many version control systems exist; this webpage focuses on one called Git, which is the most popular version control system these days. Its strengths are:

- Nothing that is saved to Git is ever lost, so you can always go back to see which results were generated by which versions of your programs.
- Git automatically notifies you when your work conflicts with someone else's, so it's harder (but not impossible) to accidentally overwrite work.
- Git can synchronize work done by different people on different machines, so it scales as your team does.

Version control isn't just for software: books, papers, parameter sets, and anything that changes over time or needs to be shared can and should be stored and shared using something like Git.

Where does Git store information?

Each of your Git projects has two parts: the files and directories that you create and edit directly, and the extra information that Git records about the project's history. The combination of these two things is called a repository.

Git stores all of its extra information in a directory called `.git` located in the root directory of the repository. Git expects this information to be laid out in a very precise way, so you should never edit or delete anything in `.git`.

What is a hash?

Every commit to a repository has a unique identifier called a *hash* (since it is generated by running the changes through a pseudo-random number generator called a hash function). This hash is normally written as a 40-character hexadecimal string like `7c35a3ce607a14953f070f0f83b5d74c2296ef93`, but most of the time, you only have to give Git the first 6 or 8 characters in order to identify the commit you mean.

Hashes are what enable Git to share data efficiently between repositories. If two files are the same, their hashes are guaranteed to be the same. Similarly, if two commits contain the same files and have the same

ancestors, their hashes will be the same as well. Git can therefore tell what information needs to be saved where by comparing hashes rather than comparing entire files.

How can I view a specific commit?

To view the details of a specific commit, you use the command git show with the first few characters of the commit's hash. For example, the command `git show b37ee1679e44e8` produces this:

```
commit b37ee1679e44e8efc0b8cb6213fe017dc50284f0 (origin/master, origin/HEAD)
Author: Odai Alali <alali@informationsgesellschaft.com>
Date:   Mon Aug 12 10:02:23 2019 +0200

    fix(Project Information Widget): add website link settings

diff --git a/src/components/widgets/ProjectInformationWidget.vue b/src/components/widgets/Proj
index 5a59754..76c25d9 100644
--- a/src/components/widgets/ProjectInformationWidget.vue
+++ b/src/components/widgets/ProjectInformationWidget.vue
@@ -1,5 +1,5 @@
 <template>
-    <base-widget ref="baseWidget" :widget="widget" :editMode="editMode" :extraSettingsProps="
+    <base-widget ref="baseWidget" :widget="widget" :editMode="editMode" :extraSettingsProps="
         <v-layout :row="widget.extraSettings.imageTextStack === 'row'" :column="widget.extraS
             <v-flex xs6 :style="{ 'text-align': widget.extraSettings.imageAlignment || 'cente
                 <v-img
@@ -11,6 +11,7 @@
                 <div class="headline" v-if="widget.extraSettings.projectName">
                     {{widget.extraSettings.projectName}}
                 </div>
+                <a target="_blank" v-if="widget.extraSettings.websiteLink" :href="widget.extr
                 <div class="body-2" v-if="widget.extraSettings.projectDescription" style="whi
                 <div class="body-2" v-if="widget.extraSettings.projectContact" style="white-s
             </v-flex>
@@ -31,6 +32,7 @@
             <v-layout wrap>
                 <v-flex xs12 md6>
                     <v-text-field label="Project Name" v-model="extraSettingsFormModel.projec
+                    <v-text-field label="Project Website Link" v-model="extraSettingsFormMode
                     <v-textarea label="Project Description" v-model="extraSettingsFormModel.p
                     <v-textarea label="Project Contacts" v-model="extraSettingsFormModel.proj
                 </v-flex>
```

The first part is the same as the log entry shown by `git log`. The second part shows the changes; as with `git diff`, lines that the change removed are prefixed with `-`, while lines that it added are prefixed with

+ .

Reference

Introtext inspired by **Introduction to Git for Data Science**

Sys-Admin

Sys-Admin page

Tutorials and Screencasts

> **This page is incomplete**
>
> Links to videos work properly only inside the IGBremen's **gitlab wiki** for this documentation.
>
> Video links need to be updated to work on icdp-online.org servers.

Tutorials

Developer Tutorials

- See **REST API Documentation**

Screencasts

There were some short video tutorials. They are now deprected. They showed scenarios that execute without any trouble and produce an expected output. In mDIS, they represented what a trained mDIS user regularly performs, and when the software is properly configured and everything runs smoothly.

In this regard, here we provide brief video tutorials to demonstrate how the user interface of a properly configured mDIS is meant to be used.

Log in

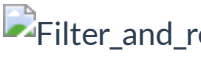
- The user goes to a webpage that is available on a private network. He enters username and password and logs in.
- Password Change: Not implemented

Data Entry

What you find on the dashboard

View Data

How to work with input forms

- Use the filter function and view datasets 

- Export some datasets, e.g. for printing. 

Add/Edit Record, Duplicate

- Only Operators or Admins can do this.
  - How to edit a record 
  - Create a new record
- Add a core - ...to be continued
- Add a section - ...to be continued
- Switch back and forth between core and section
- Duplicate a record - 
- Add a value to a value list - 

Create Parent-Child relationships

- e.g. add core section to a selected core run.

Delete items

- Delete Records
- Delete Forms
- Delete Forms and related tables
- Cascading Deletes (of Forms and Tables)

TODO Happy Paths

- refine documents, update links
- create more screencasts / videos

Sad Path Scenarios

Scenarios showing some trouble and error conditions.

Exact diagnostics might be a power-user task. (*Fixing* is DIS dev/admin task)

- Virtualization Problems (BIOS Settings, enable CPU Virtualization Support)
- Guest VM bootup problems
- Internet network problems
- Host/Guest interaction problems (networking, shared folders, webserver, Host firewall)
- insufficient permissions (User Rights, e.g Viewer tries to edit data, make API requests)

- Database-asserted Constraint Violations (Primary-Foreign key, Cascading Deletes, Column constraints/defaults/unique constraints/NOT NULLS, row duplications),
- Rename Operations (of columns, of tables)
- Client-Side-asserted Input validators (numeric/string datatypes, datetime issues, depth semantics...)
- Form/Tableset generation (= code generation issues. DB/File access permissions)
- Code editing issues (custom formfield validators).

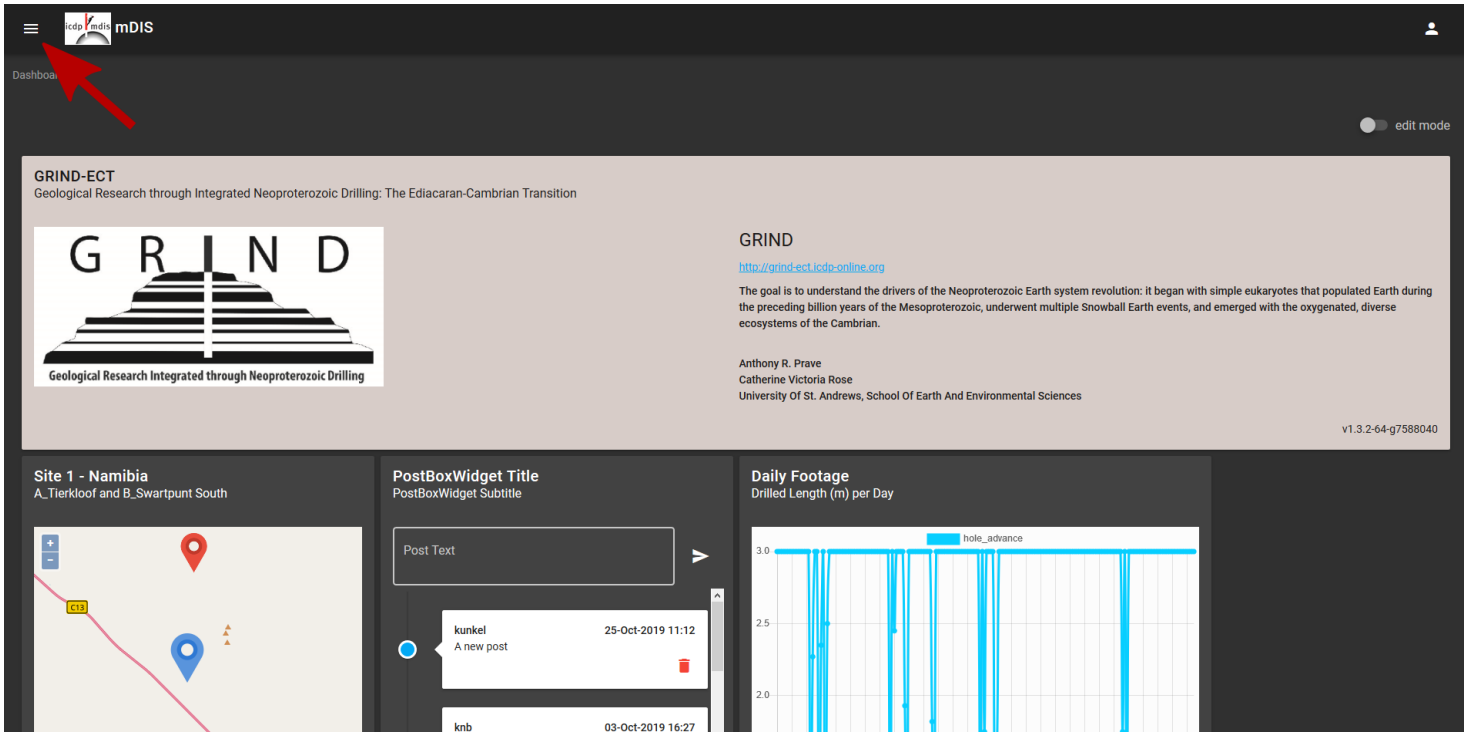
1. mDIS for Data Entry and Data Management

> **mDIS - the mobile Drilling Information System**
>
> This is the end user documentation for mDIS- the mobile Drilling Information System.
>
> These pages contain an instruction manual for normal end users: Earth Scientists or Engineers.
>
> Let's dive right in!

mDIS Basics

Dashboard

By logging into mDIS the project Dashboard is opened:



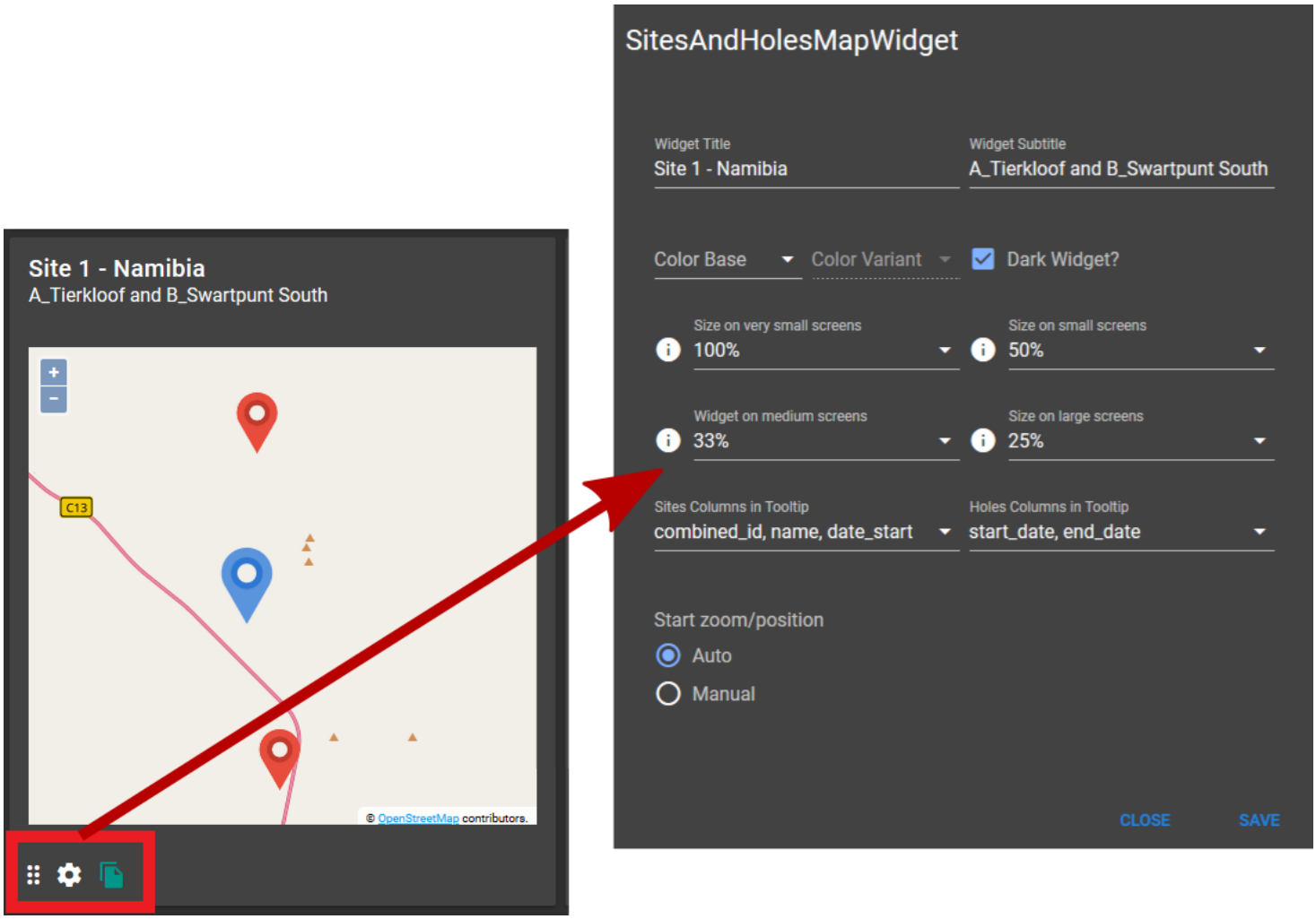
It is composed of different widgets:

- giving an overview of the general project information,
- the location of the sites and/or holes,
- the daily hole advance,
- the message of the day,
- a postbox for short messages or
- an instructions widget for tutorials, summaries, etc..

It can be modified personally, if you turn on the edit mode in the upper right corner of the dashboard.

Now each widget gets at least two icons in the bottom left (Fig. 2).

- A double column of three dots (drag points) and a gear.
  - If you move the mouse to the drag points and hold the left mouse button down, you can drag the widget to another position.
  - The gear icon opens the settings of a widget. Here you can modify the widget information and its appearance.
- Some widgets can also be used multiple types (green sheets symbol). For example one widget can show an overview of different sites and another can show the holes of a site.



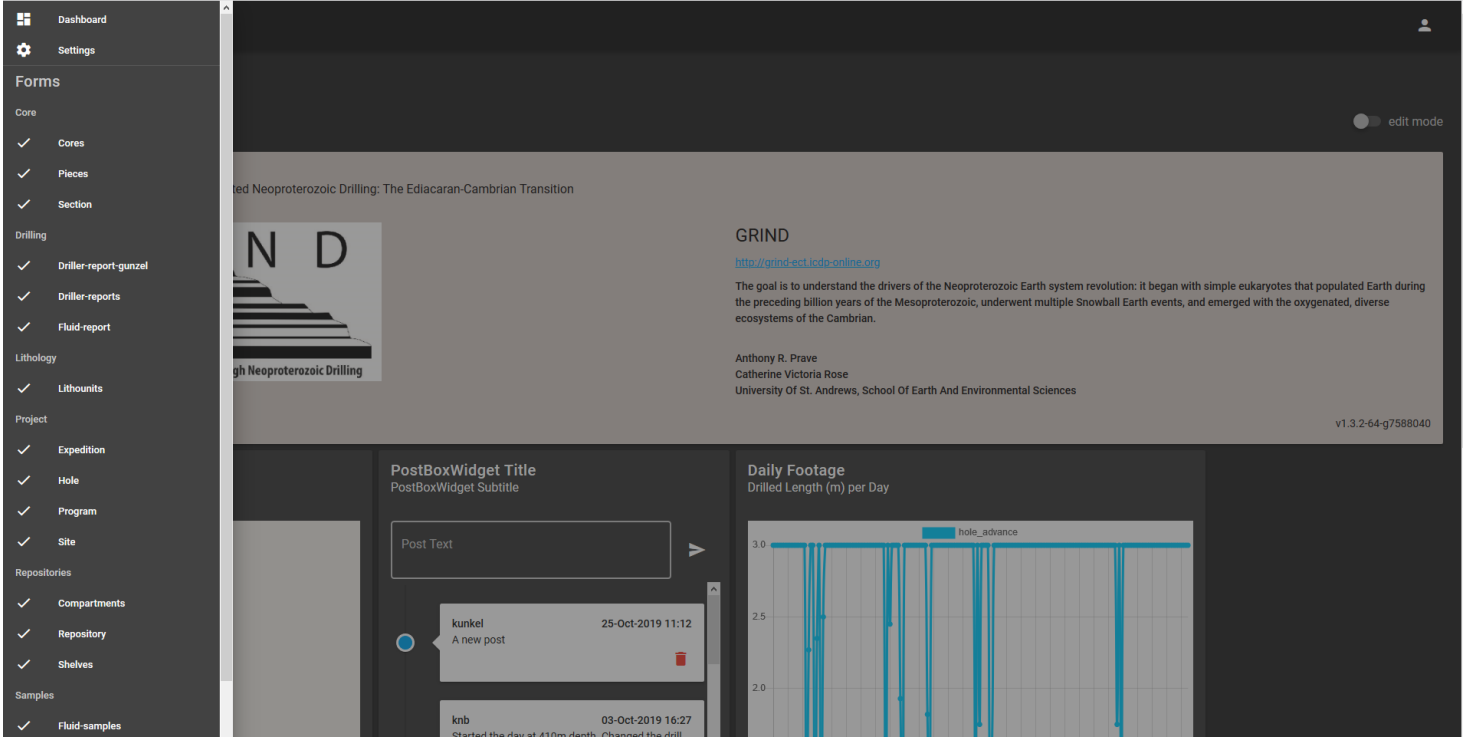
If a widget is not used, it can be found at the bottom of the dashboard under 'inactive widgets'. To use it, just drag and drop it on the drag points.

Dashboard open questions and comments

- Where do I need to store files and images for implementation? (Instructions Widget, Message of the Day Widget).
- Sites and Holes Widget: In projects with different sites I would like to see the different sites in different Widgets.

The Sidebar

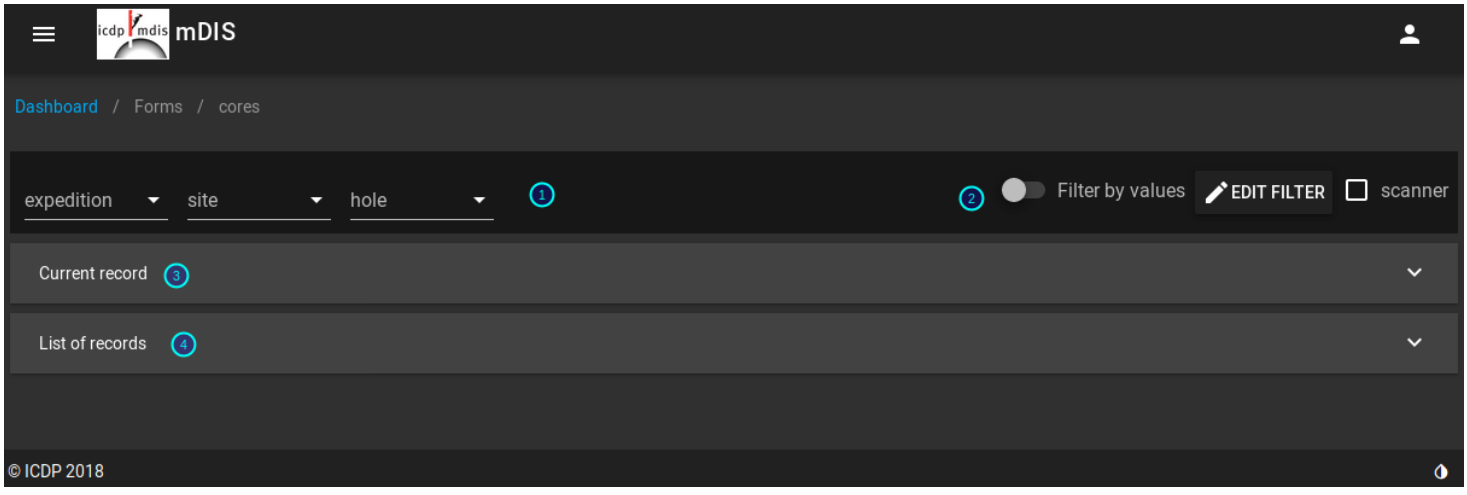
To see the data stored in the mDIS, a click on the burger menu in the upper left corner (Fig. 1) of the dashboard opens an overlay with several forms.



At the top, the dashboard and setting icons are visible. Furthermore, all forms integrated into the mDIS are listed below sorted after Core, Drilling, Lithology, Project, Repositories and Samples. (The list is sorted alphabetically at the moment. It will be changed into a hierarchical sorting). At the bottom is also a files icon, where you can import files (see...later).

Forms

Forms comprise a central part of the DIS application. They can be used to insert, edit, delete, find, view, and export/print data. The user interface of a form consists of several parts:



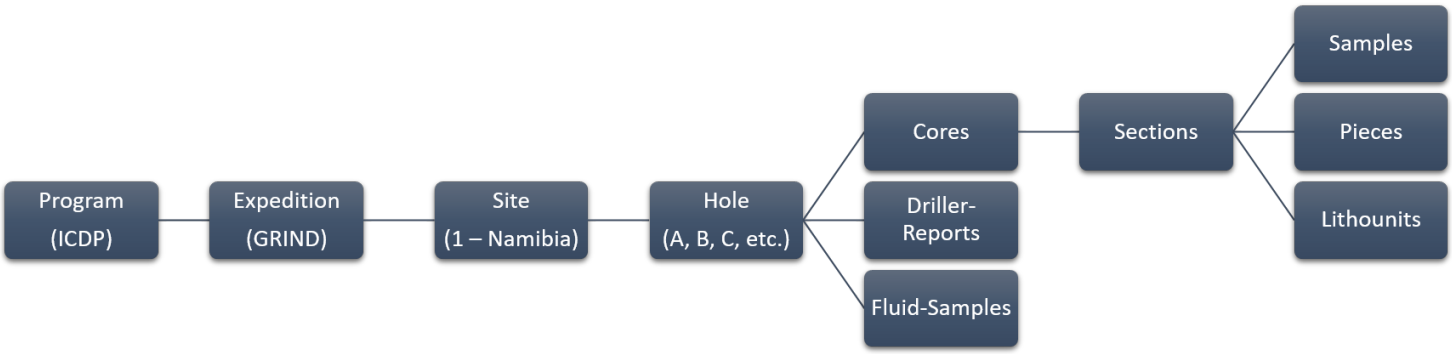
- Filter Bar (1): Located on top of any page, the filter bar allows us to select records based on the data hierarchy (expedition, site, hole, core, ...), or (2) to apply a custom filter.
- Current Record (3): The Current Record panel can be collapsed and expanded to show the data of one selected record and allows to manipulate it.
- List of Records (4): The List of Records panel can be collapsed and expanded to show the all records, or only a subset of records, if a filter was applied.

- Navigation Bar and Button Bar (not shown above, but see **Pagination**): Additionally, the Current Record collapsible panel and the List of Records panel contain a Navigation Bar and an array of buttons. The Navigation Bar can be used to move through the record set. The Button Bar enables users to perform various data manipulation tasks. Its content is context dependent.

If you only want to look at some data you might want to collapse the "Current Record" panel. If you are on a data entry spree, i.e. you have to add many new records at a time, you might prefer to collapse the List of Records temporarily.

Navigating by Hierarchy

The mDIS is structured hierarchically starting on the highest level with Project -> Program. The example shows the hierarchy levels of the GRIND expedition implemented in the mDIS from the top level program to the lowest levels samples, pieces and lithounits.



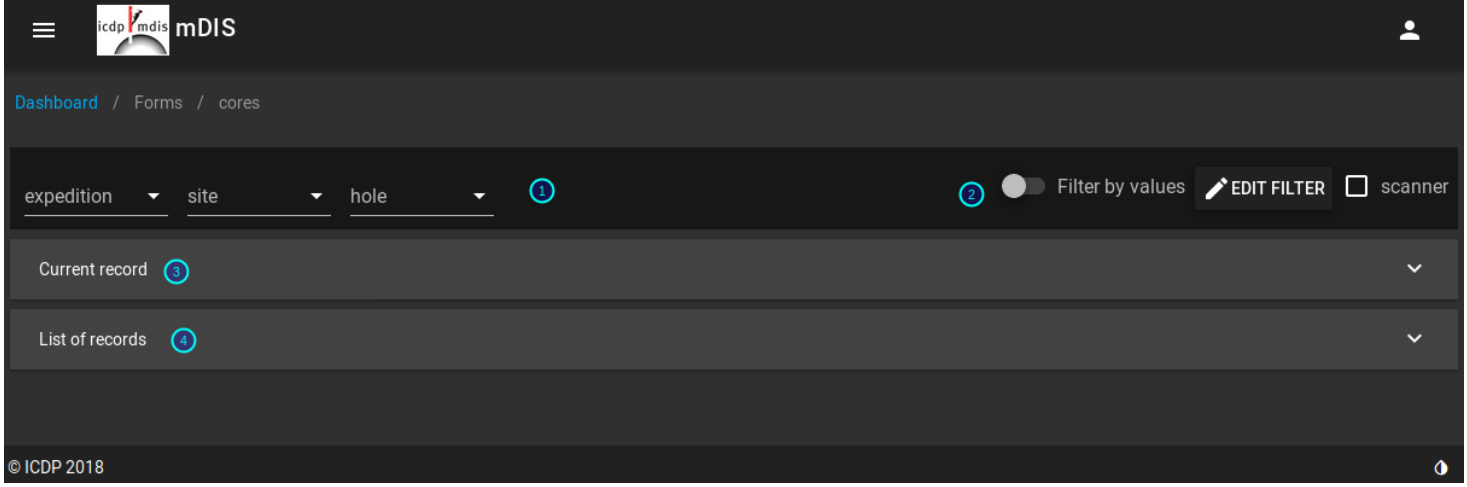
All forms are hierarchically linked by a parent, siblings and child relationship. For example, Hole is the parent to Cores, whereas cores is a child to Hole and a sibling to Driller-Reports and Fluid-samples.

Click on Program in the sidebar. A new window will open.

Two programs are implemented. You can choose between GESEP and ICDP. Click on ICDP. At the top the Program Details appear together with several colourful buttons.

User Interface and Modification Buttons

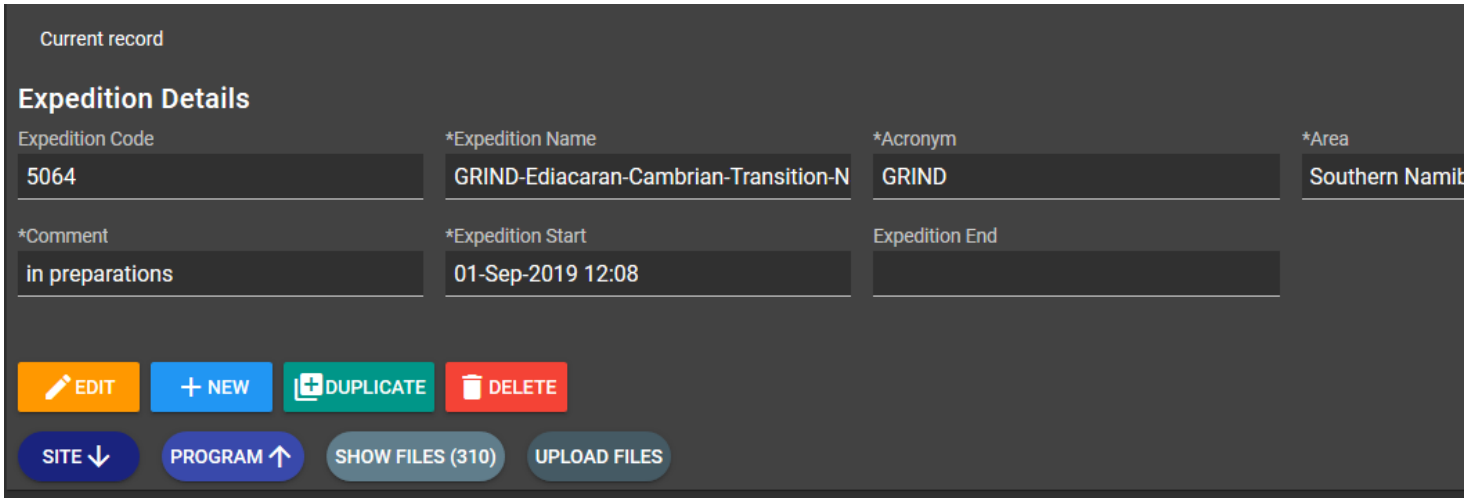
There are two Navigation Bars, one for navigating the *Current record* section of the form, and another one for paginating the *List of records*. The user interface of a form consists of:



- Filter Bar (1): Located on top of any page, the filter bar allows us to select records based on the data hierarchy (expedition, site, hole, core, ...), or (2) to apply a custom filter.
- Current Record (3): The Current Record panel can be collapsed and expanded to show the data of one selected record and allows to manipulate it.
- List of Records (4): The List of Records panel can be collapsed and expanded to show the all records, or only a subset of records, if a filter was applied.

If you only want to look at some data you might want to collapse the *Current Record* panel. If you are on a data entry spree, i.e. you have to add many new records at a time, you might prefer to collapse the List of Records temporarily.

If the *Current Record* panel is expanded, it is possible to edit records by different colorful modification buttons.



It is possible to either edit, insert a new, duplicate or delete a program.

Edit (orange button):

To edit a record, you have to select a current record i.e. by clicking on one in the list of records. Activating the edit records changes the color of the form fields to that of the edit button and shows two additional buttons Save and Cancel. If you leave the record without saving, no modifications will be recorded. If you press Cancel, any modifications will be discarded. On several fields, the entered value will be validated, as soon as you leave the field. If the value is not valid, label and description of the field will be displayed in red

color. On some fields the validation can only take place when trying to save the record. On saving the record, you should see a confirmation message at the bottom of the screen.

New (light blue button):

To create a new record, all hierarchy filter conditions have to be selected. I.e. for a core otherwise that new core could not be assigned to the correct hole. The fields of the new record are mostly empty, only a few fields are prefilled. Enter all required fields and press Save to store the new record.

Smart Copy (green button):

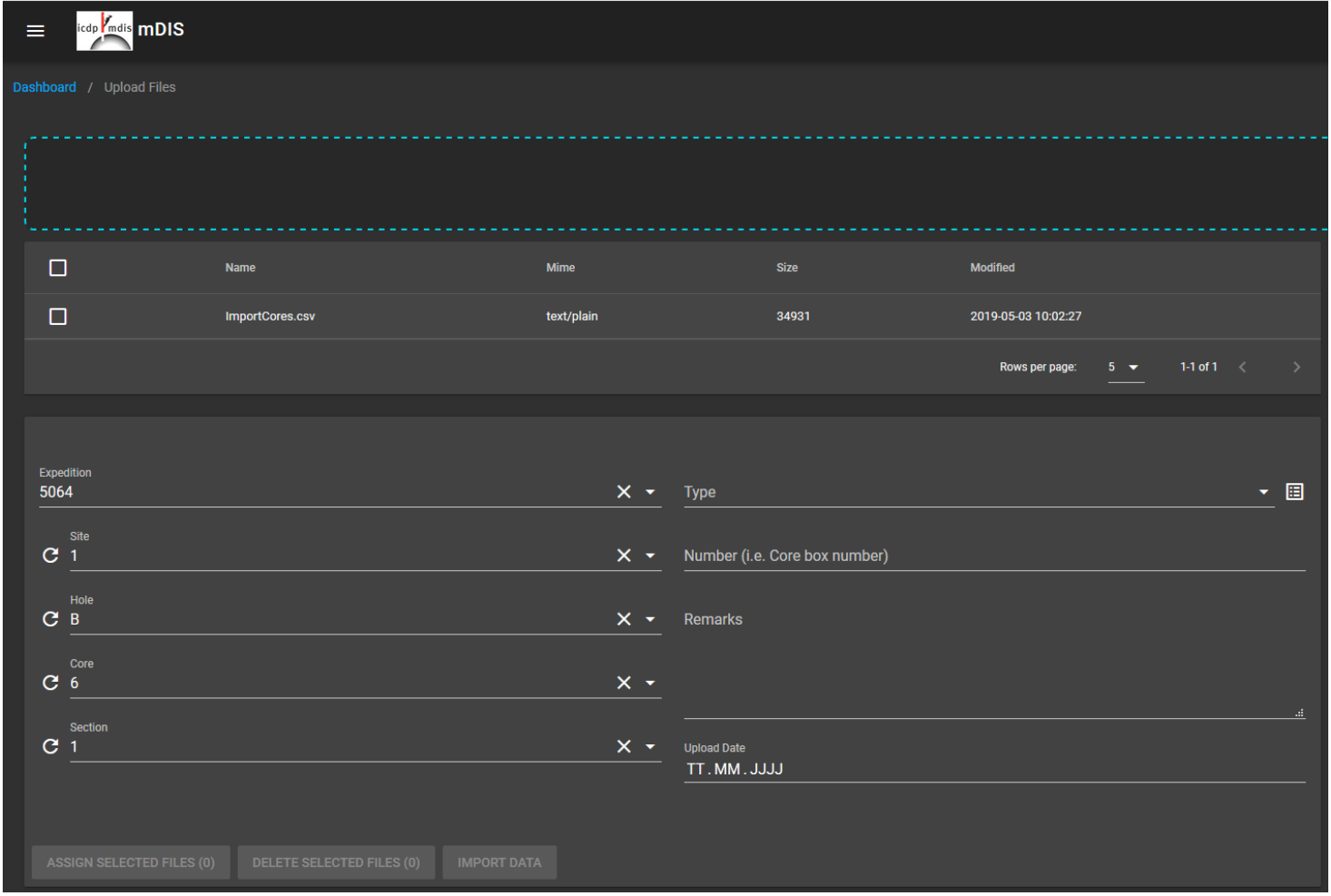
If you need to enter several similar records in sequence, it can be easier to copy an existing record instead of entering all field values again. Select an existing record and click "Smart Copy". Most fields of the current record are being copied, some will also be incremented, and some field will be empty. This happens when the system cannot guess a follow-up value, thus the field will have to be entered anew.

Delete (red button):

A record can only be deleted, if no other referencing records do exist. You can delete a core only if no sections belonging to it exist. Press the button and confirm the dialog. If the record cannot be deleted, you will get a popup error message.

The second row of buttons (rounded edges) links to related forms:

- By using the blue and lighter blue hierarchy buttons you can move to all forms by their relationship.
- In a dark blue button *Expedition* is visible. The arrow with the head pointing down indicates that its hierarchical level is lower than the program details. *Expedition* is a child to *Program*. A click on the button opens the new tab *Expeditions*. It allows the selection of an expedition to work in. Select an expedition.
- Again the corresponding details open above the expeditions. This time it is possible to go further down in the hierarchy to the child *Site* (dark blue button with the arrow pointing down) or go up a level in the hierarchy to the parent *Program* which is indicated by a slightly lighter blue coloured button with an arrow pointing up.
- The *Show Files* button opens a new tab. Click on *List of Files* and all stored files (as pictures, PDFs, Doc-Files) are listed. A click on a file shows its details and its stored files. It is also possible to edit, un-assign or delete a file.
- The *Upload* button allows the upload of different file formats by drag and drop or clicking on the upload area, which is framed by a turquoise dashed line. After choosing the file, the additional file must be assigned to the mDIS-data by filling out the form at the bottom.



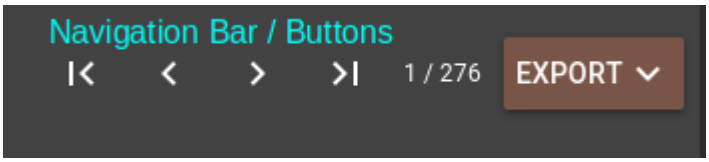
- Another button is visible on the right side of the form called *Export*. Depending on the actual hierarchy level different options for export exist. For example the data can be exported as the list of records opening a new tab with a list of the data or it is possible to choose the export as csv-file or sometimes also as PDF. Also, the printing of a QR code can be possible (**see later**).

Pagination

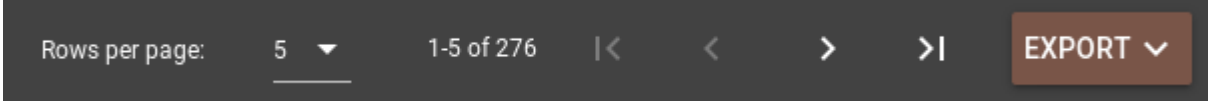
For Pagination of long tables, two Navigation Bars are shown at the bottom of both collapsible sections of any **form**.

You can select how many records should be displayed on a single page or screen. You can see which records are displayed and you can navigate to the previous or next page.

Pagination through long tables



Below the fields on the right, you find the Navigation Areas where you can see which of how many records are being displayed. You can navigate to the first, previous, next and last record in the list of filtered records.



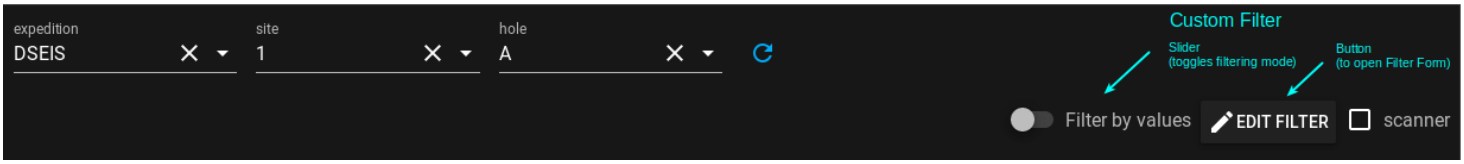
Filtering

Hierarchical Filters

The main method of filtering records in DIS is based on their hierarchy of 1:n relationships (**see figure above**). A *hole* can contain many *cores*. Thus, a *core* always belongs to a certain *hole*, so it is obvious to select that *hole* to find that *core*. Since the *hole* belongs to a *site*, you first have to select that *site*. Hence the filters of a form show the hierarchy of the ancestor elements to select from.

The filter bar "remembers" the values a user has set. However, when you select a *new* value from the list of a filter element, previously selected values of the dependent filter elements (those "on the right" of that list) will be reset. For example, if you change the setting of the filter "site", the list of available values for *hole* and *section* will be reset.

Custom Filters: Filter by values



The "Filter by values" form

For certain items, e.g. *cores*, the table of items can grow to a size which can become inconvenient to display and paginate.

Thus, in addition to the hierarchical filter mentioned above, you can set custom filter conditions, e.g. "Show cores from a depth larger than 400 meters".

To do so, click on the button " `Edit filter` " to open a modal form accepting you custom fiter expressions. The form has the same fields, arranged in the same sequence as the usual data-entry form in the "current record" section, but allows to enter filter expressions in any field, even in calculated fields.

This extra modal dialog is necessary to enter more complex, multi-field filter conditions. Below you find the different methods of filtering values. Press the button "Apply" to filter the records using the entered conditions. Using the `Filter by Values` slider, you can switch the custom filter off and on. The filter bar will "remember" your filter expressions.

For filter expressions spanning multiple fields, values are combined with the `AND` logical connector. The logical `OR` operation is not supported currently.

Comparison Operations

For columns that contain numbers or dates, you can use the operators `>`, `<`, `=`, `!=` to compare values. (These are *not* regular expressions, unlike the 'Text Search' operators mentioned below).

- `< 6.3` : Only values smaller than 6.3. Always use the dot to separate the fraction! Do not use the comma.
- `>=6.3` : Only values equal or bigger than 6.3. You do not have to write a space before the value.
- `!= 3` : Only values different from 3. Identical to `<>3`.
- `> 2017-06-23` : Only dates later than June 23rd, 2017. Always write dates in the form `yyyy-mm-dd`.
- `< 2017-06-23 11:00` : Only dates before 11:00. Please watch that you have to use UTC values for the comparison.

Text Search

For text and date fields you can just enter the value you are looking for. Strings are searched for using **regular expressions**. In mDIS, regular expression searches are by default *case-insensitive*. The following examples illustrate:

- `tube` - The word "tube" has to be contained in the field. It does not matter, if it is written in upper or lower case letters.
- `tube barrel` - The whole text has to be contained in the field.
- `barrels?` - The character before the question mark is optional, so it matches `barrel` or `barrels`.
- `^tube` - The field value has to start with `tube`. `^` stands for the start of the string.
- `^MR$` - The field value is exactly `MR`. `$` stands for the end of the string.
- `MR|HO` - The field contains either the value `MR` or the value `HO` (or both).
- `ICDP5063ECE.001` - `.` stands as a placeholder for one character
- `^ICDP50.*01$` - The field starts with `ICDP50` and ends with `01`. `*` stands for any amount (also none) of the character before it, in this case the placeholder `.`.
- `.` - Not empty. At least one character in the field.
- `2017-10-` - Any dates in October 2017
- `[^BW]` - Field values that do *not start* with `BW`.

The following characters are *Metacharacters*. They have a special meaning when they are used in Regular Expressions.

Metacharacter(s)	Meaning
`^`, `$`	Start, and end, of string
`.`	Placeholder for any character
`*`, `+`	Quantifier of the previous character, `+` stands for 1 to n, `*` for 0 to n.
`(`, `)`	Groups of conditions; i.e. for alternative values in the midst of a string
`|`	(Pipe symbol): signifies OR operator. Alternative values; often used in parenthesizes

Metacharacter(s)	Meaning
`\\`	Used to escape other characters; can be used to escape itself `\\\\`
`[ ]`	character class

For searches for strings that actually contain these characters they must be "escaped", that means here: preceded with two backslashes. So for searching for a literal dot in a field you would enter "\.", for US Dollars you would enter "US\$".

"QR Code Scanner" Filter

On some forms, mDIS has a "QR Code Scanner" feature, visible as a checkbox. Toggling this checkbox activates the "search-by-barcode" filter. This enables mDIS users to send QR codes scanned in via a handheld scanner, to mDIS. Then mDIS preselects the associated item.

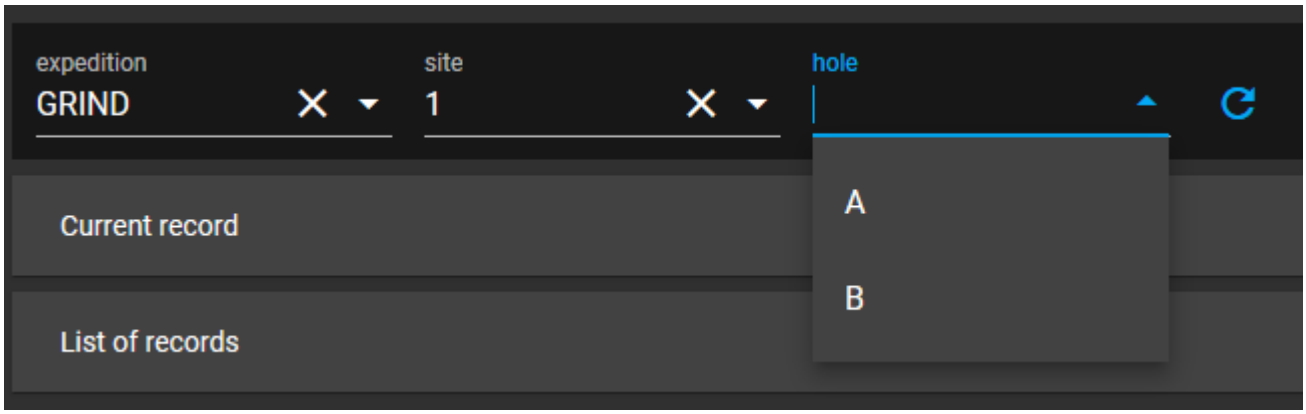
The user must send an IGSN as a filter expression.

This means, this currently works for Cores and Sections only. And the correct form must be already open. You cannot send a section IGSN to the mDIS screen with the form for cores open.

Data Input

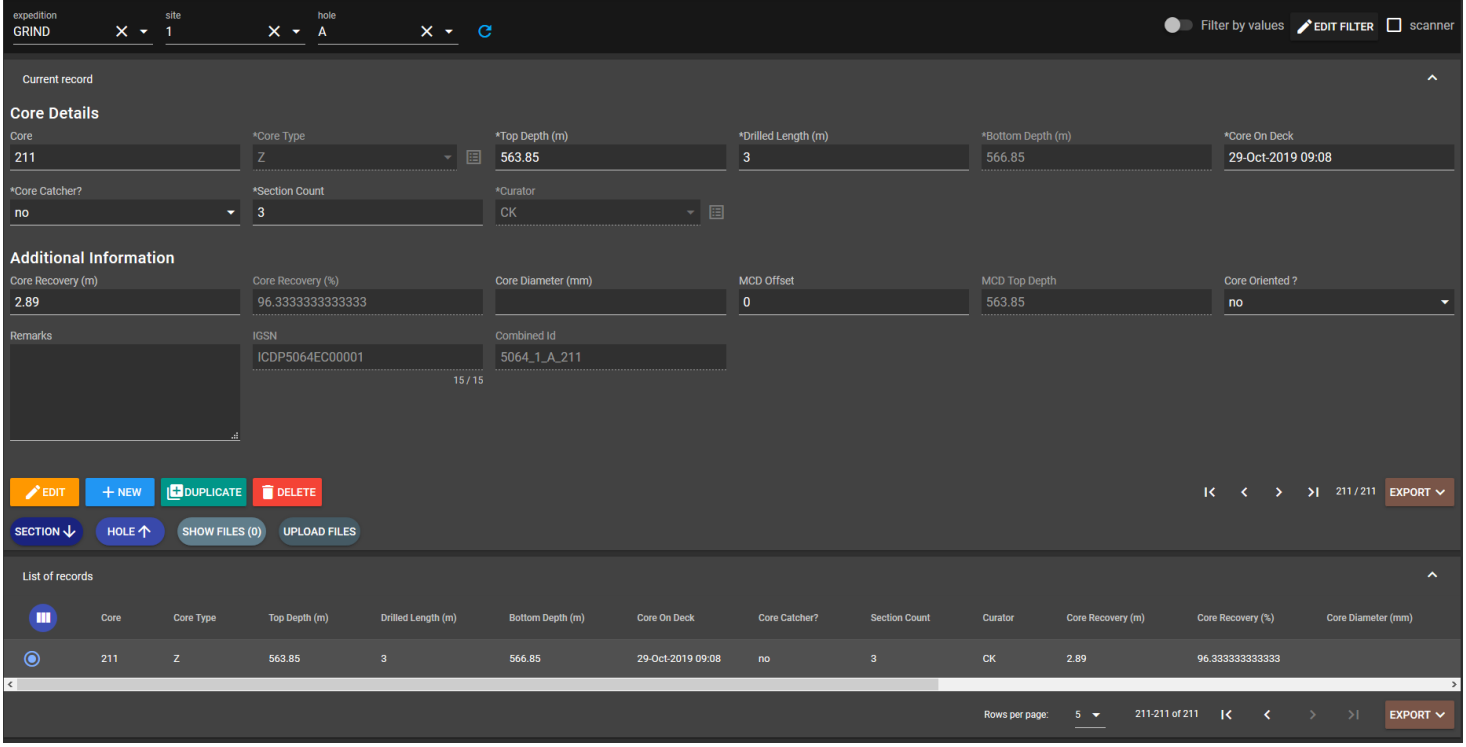
New Core Run

When logging into the mDIS the Dashboard opens. Click on the burger menu in the upper left corner and then on cores. Fill out the expedition, site and hole in the top. Then, the data in the list of records becomes visible for the selected hole.



Select the expedition, site and hole. Afterwards, you see all already existing cores in the List of records and you can type in or modify data under current record.

Under *current record* new cores can be entered. You can collapse or expand this section by clicking on the header. Here, all filtered records are displayed in a table.



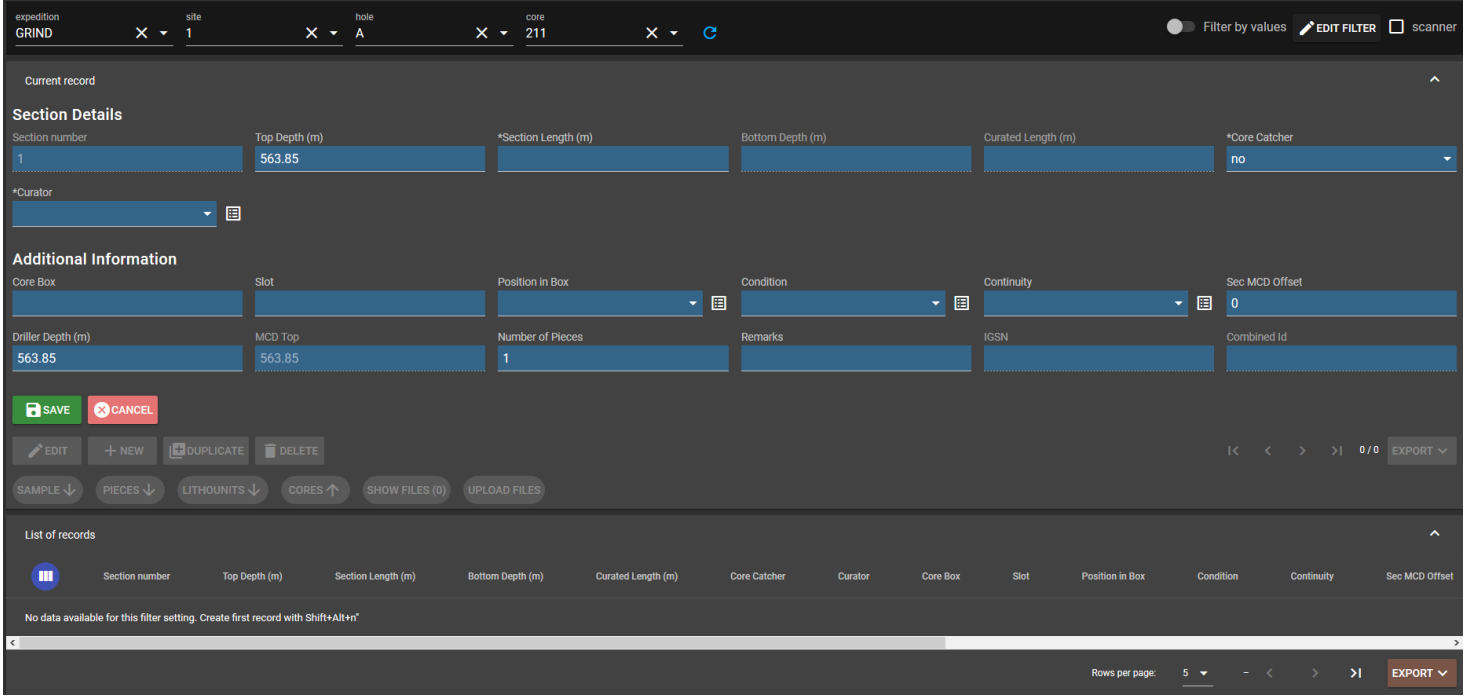
- By clicking the *new* button, a new core is added. The core number and the top depth are calculated from the last core and filled in automatically.
- Fields with a star a mandatory. By entering the drilled length, the bottom depth is calculated automatically.
- For some fields, you can select the value from a list. If you have sufficient permissions, you can modify that list of values by clicking on the icon on the right of the dropdown field. In the modal dialog that opens, you can see all values available for that dropdown list. You can edit them, delete entries or create new ones. Clicking on the cross on the top left closes the dialog and you can go back to the form. Any updates of the list values are visible immediately, without refreshing the page.
- On date fields you can click on the clock to open a date selector.
- The number in section count divides the core into several sections that need to be further characterized on the next lower hierarchy level.
- The green button *save* saves the new entry and an IGSN and a combined ID are generated.

If a dataset needs to be changed, the core in the *List of Records* needs to be selected by clicking on it and then by clicking the orange *edit* button (Fig. 2). After modifying the data, the *save* button must be clicked.

In the *List of Records* all records are shown. You can choose which columns to show in the table by clicking on the blue button with three white rows left in the table head. A dropdown menu will be displayed, where you can select the columns you would like to see, and hide the other columns. If you click on the header of a column in the table, the records will be sorted by that column. You will see a small arrow to indicate the sort column. If you click on the same column header again the records will be sorted in reverse order.

New Sections

A child of a core is a section. With pressing the blue *section* button with the down pointing arrow, the lower level *section* opens in a new tab.



The newly entered core is pre-selected in the top row and the list of records will be opened, but if a new core was just entered no records exist.

- A new record is implemented by clicking on current record and on the button *New*.
- The section number, the top depth (and driller depth) are prefilled.
- The section length, core catcher and curator are mandatory fields, indicated by the star in front of the name.
- By saving, an IGSN and a combined ID are generated.

Repeat for all new sections.

That can be either done by clicking the *New* or *Smart Copy* buttons. The *New* button generates a new empty form and the *Smart Copy* button duplicates the last data entry. It carries over values most likely to remain constant, while intelligently incrementing others.

> ### An Annoyance
>
> Beware that the "top depth" field should be incremented automatically by clicking "Smart Copy", but it is not. It is only updated automatically after the *New* button was pressed. However, as a workaround, a smart-copied record might *still* be updated after some other field is updated, e.g. "Curator". If that still does not work: the top depth, which is the bottom depth of the former section, needs to be entered manually. This completion is due to a misconfigured feature ("Yii Behavior") of the system and will be fixed in a later release.

Samples, Pieces and Lithounits

Select a section in the list of records for further data input. You can choose between different children by clicking on the blue button with the down pointing arrow: *Samples*, *Pieces* or *Lithounits*.

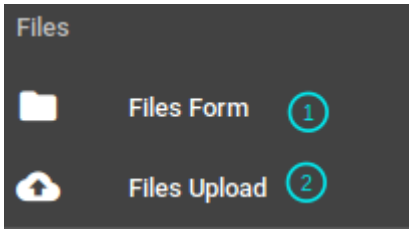
In the *Samples Form* a new sample is created under current record and new. By filling out the form, the top depth is a relative depth within the section in cm.

In the *Pieces Form* a new piece detail is created under current record and new. By filling out the form, the top depth and bottom depth are relative depths within the section in cm.

In the *Lithounits Form* a new lithounit is created under current record and new. By filling out the form, the top depth is a relative depth within the section in cm. Under additional information in the fields color, composition, sedimentary features and secondary features multiple features can be selected.

File Upload

The mDIS Drawer contains two similar items, `Files Form` and `Files Upload`.



1. Files Form

In the `Files Form`, you can browse files uploaded previously. The Files Form has a design similar to all other data input forms, and uses the same mechanisms to assign metadata.

There are two collapsible panels, `Current File` and `List of Files`.

`List of files` is a searchable data table, very similar to **any other data table.**

In the `Current File` panel area, the file metadata are displayed. They can also be edited. If the current file is an image that the system was able to convert automatically (e.g. from .tif to .jpg) during a previous file upload, you will also see a small preview of the image. Above the preview area, there is a link to download/open the file.

If someone has mis-assigned a file, e.g. some photo to a wrong core/section-record, you can remove the assignment. Then the original file will removed from the `Files Form`, be moved back to the `Files Upload` folder (see below), and you can use the `Files Upload` form to re-assign it to a different item.

> **File deletions are final**
>
> If you delete an item, the file in the file system (and the preview, converted versions, etc.) will be deleted as well. This cannot be undone, so be careful.
>
> Form deletions and table/model deletions are also final. They can also *not* be reverted, unless you have a **backup**.

2. Files Upload

You can assign images or other files to other mDIS entities such as expedition, site, hole, core or section. This process is divided in two parts, Uploading Files and Assigning Files.

Uploading the files

Open the file upload form. At the top you can see the upload area in which you can drag-and-drop multiple files from your local hard disc. Alternatively, you can click on that area and select (multiple) files from your computer or mobile device, e.g from the picture gallery of your smartphone. Click on `Upload` to start the upload process. The duration depends on the total file size, bandwidth and other connectivity parameters in your network.

Newly uploaded files will be placed in the `backend/data/upload` folder on the mDIS server.

Assigning the files

Open the `Files Upload` form.

Select one or multiple files from the list of files available in the upload directory. Below the list, select the expedition, site, hole, etc. to which you want to assign the files. Choose only those select-boxes from the Filter Bar that are appropriate. This means, you can assign a file to a site, or even to an expedition. Simply omit those select-boxes you do not need.

Select the type of the file(s), optionally enter a number as an id or a counter. Depending on the file type this could be, for example, a core box number. Optionally, add some remarks.

Then press the button `Assign selected files` .

Images will be converted automatically to JPG, if the file type is *not* JPG (e.g. TIFF). For large-size images a ~200x300px preview image is also created.

After successfully assigning metadata, the files will disappear from the Files Upload form. This is expected behaviour. File upload and file management are two different things, and it is more practical to have two different administration pages for these tasks.

Therefore, newly assigned files will be moved on the mDIS server, in order to keep the `data/upload` directory a bit cleaner. Files will be physically moved from directory `backend/data/upload/` to a new subdirectory with a name according to the item type, e.g. `backend/data/upload/ArchiveBox` .

Switching between `Files Form` and `Files Upload` form

You can get a listing of the newly assigned files by switching to the `Files Form` , mentioned above.

Alternatively, open the collapsible panel of the record of the associated data-input-form for the item just assigned. That means, if you assigned the file to Site 2, open the Site input form, select row 2, expand the

`Current Record` panel.

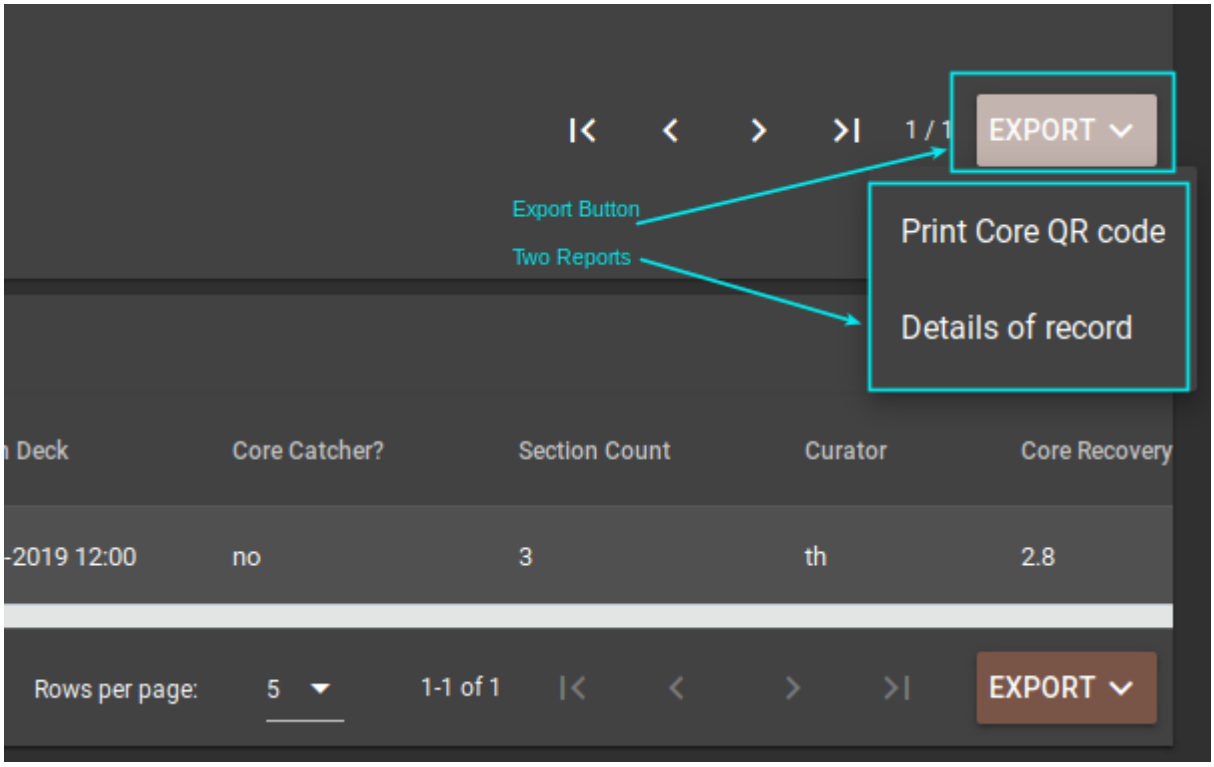
If you have made an error, e.g. by assigning files to the wrong item, you can *unassign* the item in the Files Form. Afterwards, switch back to the Files Upload form, and redo the assignment process.

You can also use the Files Upload form to import external data.

Reports

There are Single-Item-Reports, e.g., a Report to export a single QR-code label, as well as Multi-item Reports, such as exported data tables.

The figure below shows two reports available through the "upper" `Export` button. These are single-item reports. The "lower" `Export` button would display a pull-down-menu of multi-item reports. (That menu is not shown in the figure below).



Details-of-Record

The most basic Single-Item Report.

Prints a simple View of a single table row. The column values get "flipped" to a two-column table of PropertyNames and PropertyValues. It is the *complete* table row. It is *not the subset* of entry-fields of the form which you *display* on the screen. Remember, some columns might have been set to "invisible" with the column chooser icon (located in the left corner of the full datatable). Form-fields invisible on the screen *still* get displayed on the "Details of Record" report.

QR Code "reports"

The "reports" actually represent QR-codes, for stickers and labels.

Export/Print a 2D QR-Code. The look-and-feel (logos etc.) is set by editing the PHP-file of the report and the CSS files in `web/css`. This report is commonly used for creating label stickers, e.g. for marking core-boxes.

Predefined Roles

mDIS Administrators create *users* (with username, password, and email), and then typically assign these users one of a handful of predefined **roles**.

mDIS *Viewer* role

- are only allowed to browse science data using existing forms
- can export/print/download the existing data using existing reports
- cannot modify (edit/delete) any of the existing data
- cannot insert any new data
- TODO Viewer can reset/change own password
- TODO can also set some personal preferences on a "Settings" page

mDIS *Operator* role

- can modify existing forms or generate new forms based on existing data tables
- can modify (edit/delete) any of the existing data
- can insert new data into existing data tables
- can export/print/download the existing data using existing reports
- cannot modify existing data tables, or generate new data tables
- cannot modify user settings
- TODO: can view group memberships and role assignments of other users (but not change them)
- backend: can restart the DIS (otherwise they would reboot the computer or VM if something goes wrong. They need an alternative.)

Sysadmin (sa) Role

For more about the *sa* role, please read the **sysadmin documentation**.

Export

Exporting Data to CSV Files and as Reports

TBC

Backup

Backups of mDIS VirtualBox Guests

By default, there exists a backup job that is scheduled to automatically run daily. The job puts database dumps and form designs into a single zip file. Usually these files are stored in directory `backend/data/upload/backup/mysql`. A typical filename of such a zipfile would be `mysqldb_backup_mdis-GRIND--10.132.0.3--2020-04-03--dis.sql.zip`.

Your system administrator might have changed these settings. Another common location to store those backups is directory `/var/tmp/Upload`.

For details and alternatives to make mDIS backups, read this **extra document** from the sysadmin documentation. For restore- operations, read this also: **mysql backups**.

You should agree with your system administrator on a strategy to make backups of files you create independently, such as photographs, core scans etc.

TBC