FAIR Wizard

Release 4.5

FAIR Wizard Team

May 10, 2024

ABOUT

The FAIR Wizard is an enterprise-ready and user-friendly application for creating Data Management Plans. It does not only provide all necessary tools for researchers and data stewards to create data management plans (DMPs) easily, efficiently, and in a FAIR manner, but it is a suite of tools to incorporate Data Management Planning process with needs of institutions.

Data stewards can easily capture the knowledge, including required project data and decisions in knowledge models that are then turned into per-project questionnaires to be filled by researchers. The questionnaires guide researchers through the process using recommendations, FAIR metrics indications, and only showing relevant questions based on previous answers.

Once the questionnaire is completed, a DMP can be easily generated using a selected template and output format. The document is then stored in FAIR Wizard for easy access and future reference. This is especially helpful because many funding agencies now require a DMP for their application process.

But the benefits of using FAIR Wizard go beyond just creating a DMP. Institutions can use FAIR Wizard to manage their Data Management Planning processes, using powerfull tools provided within the FAIR Wizard. This includes advanced user management, custom integrations with data and more.

https://youtu.be/oU6z9V4jW0Q

CHAPTER ONE

STRUCTURE OF THE GUIDE

The guide sections are organized into three categories:

- **About** contains an introduction to the FAIR Wizard and its content to gain quick insight into how it works at a high level.
- **Applications** is structured the same way as FAIR Wizard's applications menu, so it is easy to quickly find the relevant sections about how to use a specific part of the application:
  - Data Management Planner
  - Admin Center
  - Integration Hub
  - Reporting
- **More** contains all additional information related to FAIR Wizard, such as changelog or development of various extensions.

Here are some recommended sections where to start based on the role:

Researcher	Data Steward	Admin
Introduction	Introduction	Introduction
Overview	*Overview*	*Overview*
Project	*Knowledge Model*	
	Document Template	
Applications	Applications	Applications
Data Management Planner	*Data Management Planner*	*Admin Center*
	Integration Hub	*Reporting*
Features	Features	Features
Projects	*Knowledge Models*	*Users*
Documents	*Document Templates*	*User Groups*
	User Groups	*Administration*
	Project Templates	*Settings*
	Value Integrations	*Audit Log*
	More	More
	Document Template Development	*Submission Service*
	Integration Questions	Changelog
	Project Importers	

1.1 Introduction

This section will cover the essentials of the FAIR Wizard, including its key components and how they operate together to facilitate effective data management for researchers and data stewards.

1.1.1 Overview

Different components in the FAIR Wizard are connected to create a data management plan and help with data management in general. Different components are typically made and used by different user roles. Data stewards work on preparing content (such as **knowledge models** or **document templates**) for researchers that they can use to work on their data management plans while filling in the **questionnaires** and exporting **documents**.

Data Management Planner

Data Management Planner is the core of the FAIR Wizard. It enables creation of Data Management Plans (DMPs) and other documents based on the answers to a questionnaire. The DMPs are created using a knowledge model and a document template. The knowledge model defines the structure of the questionnaire, and the document template defines the structure of the resulting document.

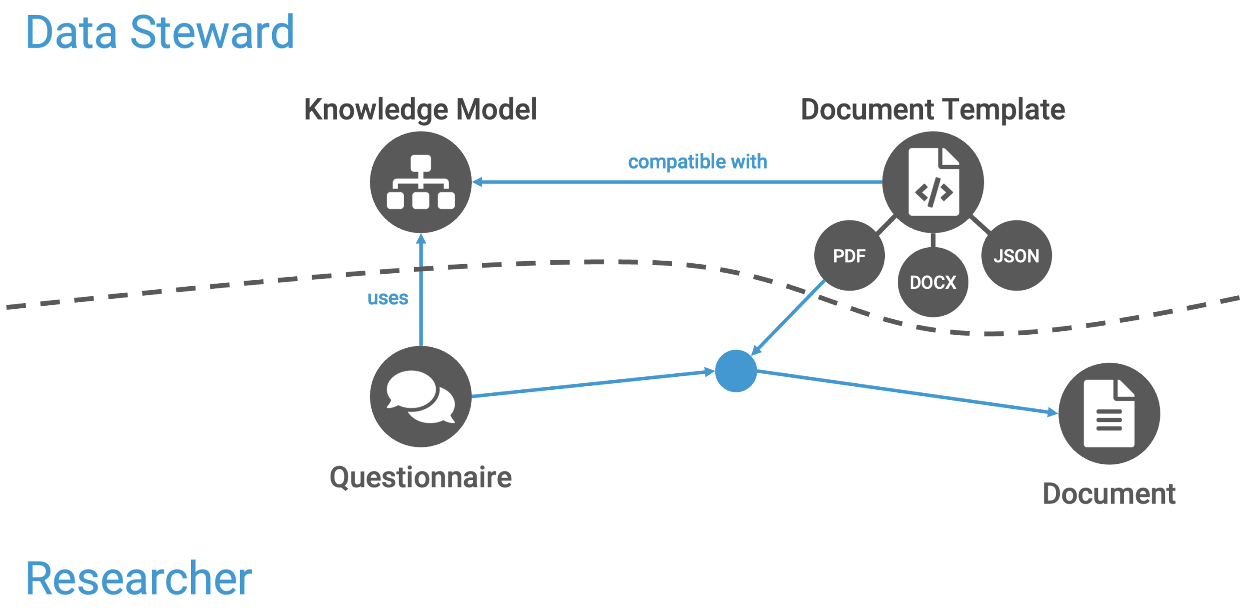


Fig. 1: Different components of the FAIR Wizard and roles interacting with them.

Knowledge Model

The knowledge model refers to something like a template for the questionnaire. However, it is not linear but has a tree-like structure with different branches based on the previous answer. Therefore, even if it can be complex overall, only the specific questions are used in the questionnaire.

Learn more about knowledge models →

Document Template

While the knowledge model defines the structure of the questionnaire, it does not specify how the resulting document (such as the DMP) will look. We use document templates for that. They transform the answers into documents such as PDF, MS Word, or machine-actionable RDF. This way, we can only answer once and produce different documents.

Learn more about document templates →

Questionnaire

A questionnaire is part of a *project* where researchers fill in their answers regarding their particular research. It uses a specific knowledge model that defines its structure.

Learn more about questionnaires →

Document

Documents are produced from the questionnaire answers and a document template. The document template understands the knowledge model structure and knows how to transform the questionnaire answers into a specific document in the selected format. The documents are saved within a project where they were created from the questionnaire.

Learn more about documents →

Admin Center

The Admin Center is an application in which can administrators manage the FAIR Wizard application. This includes management of users, their roles and user groups and various imports. It also includes the management of the application itself. User groups can also be managed by data stewards.

Users

Users are people who use the FAIR Wizard. They can be researchers, data stewards, or administrators. Each user has a role that defines what they can do in the application. Users can be organized into user groups.

Learn more about users →

Imports

Imports are used to import data into the FAIR Wizard. This can be used to import users and projects.

Learn more about imports →

Audit Log

Audit Log is a log of all the actions that have been done in the FAIR Wizard. It is used to track changes and actions done by users.

Learn more about audit log →

Integration Hub

The Integration Hub is an application that is used to create integrations. At the moment it allows creation of Value Integrations.

Value Itegrations

Value Integrations are used to create integrations from data stored in CSV format. These integrations can then be used in Knowledge Models.

Learn more about value integrations →

Reporting

Reporting is an application that is used to generate reports. Reports can be done on Projects and Users.

1.1.2 Knowledge Model

The knowledge model is a tree-like structure of chapters, questions, answers, and other entities that servers as a template for the questionnaire. All the different questions, their possible answers and follow-up questions, metrics and more is defined there.

While all the possibilities are defined in the knowledge model, when researchers use it to create their project, they don't see everything, but only the top-level questions and more detailed questions are only asked if relevant to their use case.

Knowledge models are created by data stewards in the *knowledge model editor*.

Knowledge Model Structure

Knowledge model consists of several entities connected together. You can see how they are connected in the following diagram and read more details about them below.

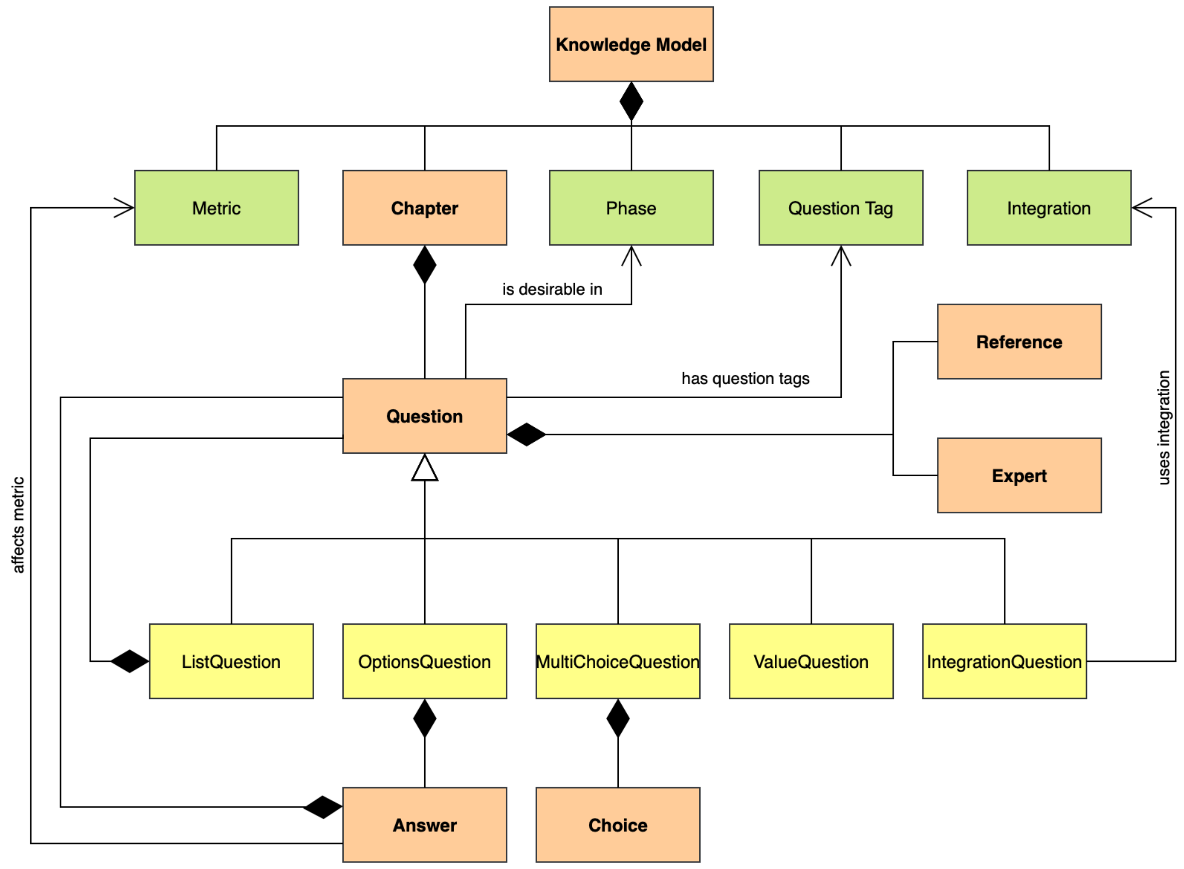


Fig. 2: Knowledge model schema

Knowlege Model

At the top level, the knowledge model contains *chapters*, and entities refered to elsewhere from the knowledge model: *metrics*, *phases*, *question tags*, and *integrations*.

Chapter

The knowledge model consists of chapters at the top level. Each chapter has a **name**, a **description** and a list of *questions*. Usually, chapters are used to group the questions on the same topic together.

Question

Questions are used to collect the answers from users. Each question has a **title** (the actual question), a **description**, a **phase** when it becomes desirable, a list of *references* and *experts*, and a selection of *question tags*.

Then there are some additional settings based on the **question type**.

Options Question

The options question contains a closed list of *answers* where users can pick one. Answers can have some follow-up questions that are only presented to the user when they pick the answer. So the options question can be used for questionnaire branching.

List Question

The list question is used when there are multiple answers and we want to ask more details about those. For example, we can ask about different datasets that will be produced – users will have multiple datasets but we want to ask the same questions for each of those. For that, we configure the **item template**, which defines the questions for each item.

Value Question

The value question asks for a single value that users type in. There are many different types of the value question that can be used:

- String
- Number
- Date
- Date Time
- Time
- Text
- Email
- URL
- Color

The input field differs based on the value type (simple input for string, date picker for date, etc.). Some of these have a check whether the entered value is valid (such as valid email or URL) and displays a warning if not.

Integration Question

The integration question is connected to an external resource where the users can pick the answer from. We need to select an *integration* that the question uses and sometimes additional properties, based on the integration configuration.

Users can then search the external resource through the questionnaire and choose the answer. The advantage is that the answer is not only the text but also a link or PID of the selected item making it more FAIR.

If the desired answer is not present in the external resource, users can still fill in a text answer themselves.

Multi-Choice Question

The multi-choice question has a list of *choices*. Users can then pick as many of those choices as they wish. There are, however, no follow-up questions available for this question type.

Answer

An answer is used with *options questions*. It contains a **label** which is the answer itself. Then an **advice** which is visible only if the answer is selected. We can use this when users pick answer that is not great to provide them further guidance on how to improve.

Answers can have **follow-up questions** that are only visible if the answer is selected. We can use this to ask only relevant questions based on the previous answers.

If there are some *metrics* created in the knowledge model, we can configure how each answer affects them. The result for each metric is eventually calculated as a weighted average of all answers affecting that metric. Therefore, we need to configure:

- **weight** [0..1] - how important the answer is (0 = not important at all, 1 = very important)
- **measure** [0..1] - how it affects the metric (0 = bad, 1 = good)

Choice

A choice is used with *multi-choice questions*. It only contains a **label** which is presented to the user.

Reference

We can provide some additional references for *questions* to help users better understand it or learn more details. There are more types of references.

URL Reference

A URL reference is a simple link to any website. It has **URL** which is the actual link and a **label** that describes what the reference is about.

Book Reference

Warning: Book references are deprecated.

Resource Page Reference

Warning: Resource page references are not yet implemented.

Expert

We can provide a contact information to an expert for some *questions*. An expert has a **name** and an **email**. We can use this, for example, if there is an expert for a specific topics in our institution and we want to make it easy to find out in our customized knowledge model.

Metric

We can define metrics for each knowledge model based on our needs. Each metric has a **title**, an **abbreviation**, and a **description**. Once the metric is defined, we can configure which *answers* affect it and how.

This can be use, for example, to define the FAIR metrics:

- **F** - Findability
- **A** - Accessibility
- **I** - Interoperability
- **R** - Reusability

And then define which answers affect which FAIR metrics to provide more feedback to the researchers.

Phase

We can create phases to reflect the workflow. Such as: *Before submitting the proposal*, *Before submitting the DMP*, etc. Each phase has a **title** and a **description**.

Once we have phases defined, we can assign them to *questions* to indicate where each question become desirable. The phases implicitly follow the order in which they are in the knowledge model and the question is considered desirable from the defined phase and on. So for example, if a question is desirable in *Phase 2*, it is implicitly desirable in *Phase 3*, *Phase 4*, etc.

Question Tag

We can define question tags on the knowledge model and then assign them to different *questions*. This can be used to group together questions on the same topic or for the same purpose.

When researchers create a new project from the knowledge model, they can only choose the question groups they are interested in for their research. So we can use this to create a very rich knowledge model but researchers will be able to use only the parts relevant to them.

Integration

Integrations define a connection to an external service or resource where we can get the answers from. They are used with *integration questions*. For each integration we configure some basic information, such as **ID**, **Name**, or **Logo URL**. Other configuration varies based on the integration type. More information about how to configure integration is available under the *integration questions documentation*.

API Integration

API integration connects to an external service API to search for the answers. We need to provide some **request** and **response** configuration, so FAIR Wizard can use the API.

Widget Integration

Widget integration doesn't use an API but a widget implemented using the DSW Integration SDK. Then we need to configure the **widget URL** where the widget is deployed.

Annotations

Annotations are arbitrary key value pairs that can be assigned to any entity in the knowledge model. These can provide some additional information for the document templates.

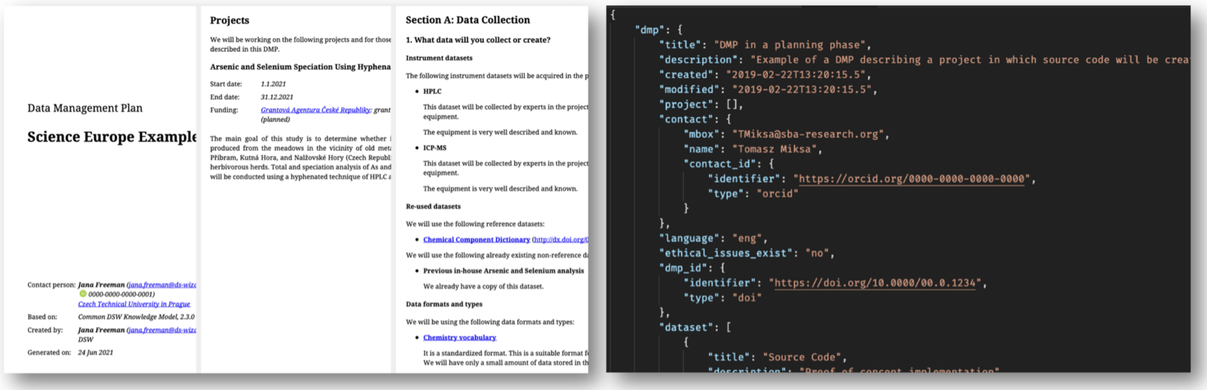
Knowledge Model Customizations

A knowledge model doesn't have to be created from scratch. Instead, it can be created as a customization of an existing knowledge model.

We can choose any existing knowledge model and customize it to our needs. We can add, modify, or remove any entities. If there are newer changes in the parent knowledge model, it is possible to get them into our child knowledge model using the knowledge model migration.

1.1.3 Document Template

Document templates transform the answers from a questionnaire to a document of a specific format. The document template usually follow some standard template, such as Horizon Europe DMP, and can support different formats, such as PDF or MS Word. The formats can be basically any text-based format, so it can also be for example a JSON or XML. It follows that we can easily use them to crate a machine-actionable output, too.



Data Management Plan

Science Europe Example

Contact person: *Jana Freeman* (jana.freeman@ds-wiz
0000-0000-0000-0001)
Czech Technical University in Prague
Based on: *Common DSW Knowledge Model, 2.3.0*
Created by: *Jana Freeman* (jana.freeman@ds-wiz
DSW
Generated on: *24 Jun 2021*

Projects

We will be working on the following projects and for those
described in this DMP.

Arsenic and Selenium Speciation Using Hyphena

Start date: *1.1.2021*
End date: *31.12.2021*
Funding: Grantová Agentura České Republiky*: grant*
(planned)

The main goal of this study is to determine whether i
produced from the meadows in the vicinity of old met
Příbram, Kutná Hora, and Nalžovské Hory (Czech Republ
herbivorous herds. Total and speciation analysis of As and
will be conducted using a hyphenated technique of HPLC a

Section A: Data Collection

1. What data will you collect or create?

Instrument datasets

The following instrument datasets will be acquired in the p

- **HPLC**

  This dataset will be collected by experts in the project
  equipment.

  The equipment is very well described and known.

- **ICP-MS**

  This dataset will be collected by experts in the project
  equipment.

  The equipment is very well described and known.

Re-used datasets

We will use the following reference datasets:

- **Chemical Component Dictionary** (http://dx.doi.org/0

We will use the following already existing non-reference da

- **Previous in-house Arsenic and Selenium analysis**

  We already have a copy of this dataset.

Data formats and types

We will be using the following data formats and types:

- **Chemistry vocabulary**

  It is a standardized format. This is a suitable format f
  We will have only a small amount of data stored in th

```
{
    "dmp": {
        "title": "DMP in a planning phase",
        "description": "Example of a DMP describing a project in which source code will be crea
        "created": "2019-02-22T13:20:15.5",
        "modified": "2019-02-22T13:20:15.5",
        "project": [],
        "contact": {
            "mbox": "TMiksa@sba-research.org",
            "name": "Tomasz Miksa",
            "contact_id": {
                "identifier": "https://orcid.org/0000-0000-0000-0000",
                "type": "orcid"
            }
        },
        "language": "eng",
        "ethical_issues_exist": "no",
        "dmp_id": {
            "identifier": "https://doi.org/10.0000/00.0.1234",
            "type": "doi"
        },
        "dataset": [
            {
                "title": "Source Code",
                "description": "Proof of concept implementation"
```

Fig. 3: Document templates can transform the answers from questionnaires to different formats, such as human-readable PDF or machine-actionable JSON.

It is important to note that the document template always defines which *knowledge models* it is compatible with because in order to transform the answers to the document it needs to understand what the questions are and how the answers should be composed into a document.

We can get existing document templates in the *FAIR Wizard Registry*, or create our own using *document template editor* or *DSW Template Development Kit (TDK)*.

1.1.4 Project

A project is a central part of where we work on our data management plans. It is based on a *knowledge model* and uses one or more *document template* for generating documents.

Projects are fully collaborative, so we can share them with other researchers and work together.

Questionnaire

The questionnaire is part of the project where we fill in our answers. It is generated based on the selected knowledge model. It shows only the questions that are relevant based on our previous answers. There are many more features to provide guidance and embrace collaboration, such as FAIR metrics, comments, TODOs, or version history.

Learn how to work with the questionnaire →

Documents

The outcome of our endeavors in FAIR Wizard should be the data management plan, the document. Once we have enough answers to the questionnaire, we can generate a document using a document template. These documents are saved within the project. We can create as many as we wish with different document templates and formats. We can also set up a default document template for the project and quickly see a preview of the document.

Learn how to work with the documents →

1.2 Data Management Planner

Data Management Planner is an application for collaborative and effective creation of Data Management Plans. This application is mainly used for creating Data Management Plans by **researchers**, but it also includes creation and editing of Knowledge Models and creation and editing of *Document Templates* done by **data stewards**.

1.2.1 Knowledge Models

This section covers how to manage existing *knowledge models* and how to create new ones and make them available for researchers.

Knowledge Model List

As **data stewards** and **admins**, we can manage the knowledge models that are in our FAIR Wizard. Then, **researchers** are using and browsing the knowledge models. We can access the list of knowledge model from the main menu via *Knowledge Model*. The list can be filtered and sorted by name or creation date.

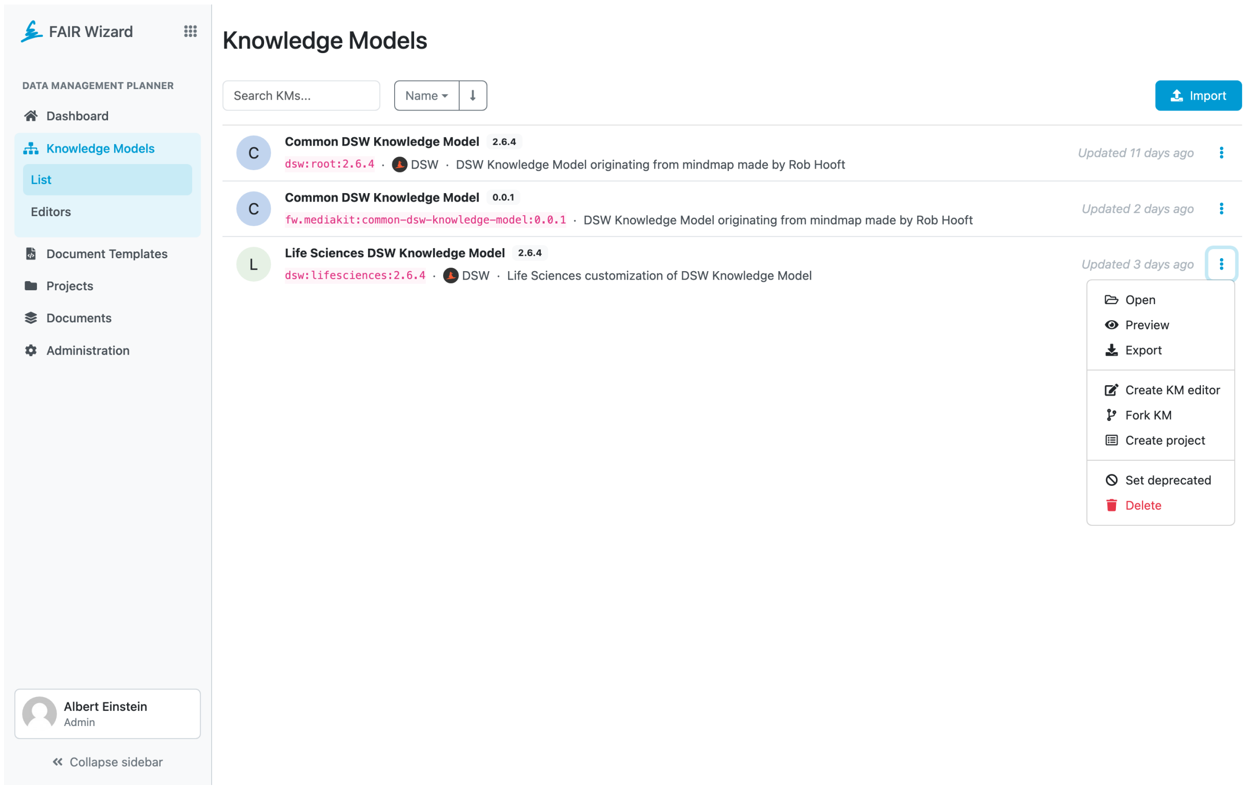
For each knowledge model (KM), we can see the latest version in the list. If we want to read more about a specific KM or see the older versions, we need to access the *Knowledge Model Detail* by clicking the name of KM or clicking *Open* from the right item menu (three dots). There are also other options for each item:

- *Preview* to see how *Projects* generated using this KM would look like.
- *Export* for exporting the latest version of the KM as a file.
- *Create KM editor* is a shortcut for *Create Knowledge Model Editor* for creating a new version.
- *Fork KM* is again a shortcut for *Create Knowledge Model Editor* for to create a fork (some more specific KM based on this one).
- *Create project* is a shortcut to *Create Project* with this KM.
- *Set deprecated* or *Restore* for setting a KM deprecated when we no longer want the **researchers** to use it.
- *Delete* for all versions of the KM (possible only if is not used in any projects or linked in other KMs and editors).

Note: The options in the item menu are based on the role of a current user, e.g. a **researcher** cannot create KM editor.

For **data stewards** and **admins**, *update available* may appear if there is a newer version of the knowledge model in the FAIR Wizard Registry (and if configured).

Finally, there is an option to *Knowledge Model Import* by click the *Import* button in the top right part of the screen.



Common DSW Knowledge Model 2.6.4
dsw:root:2.6.4 · DSW · DSW Knowledge Model originating from mindmap made by Rob Hooft
Updated 11 days ago

Common DSW Knowledge Model 0.0.1
fw.mediakit:common-dsw-knowledge-model:0.0.1 · DSW Knowledge Model originating from mindmap made by Rob Hooft
Updated 2 days ago

Life Sciences DSW Knowledge Model 2.6.4
dsw:lifesciences:2.6.4 · DSW · Life Sciences customization of DSW Knowledge Model
Updated 3 days ago

Fig. 4: List of all knowledge models with actions.

Knowledge Model Import

We can import an existing knowledge model by navigating to *Knowledge Model List* (*Knowledge Models*) in the main menu and then clicking on *Import* button.

From FAIR Wizard Registry

The FAIR Wizard is connected to the FAIR Wizard Registry. That allows us to import knowledge models from it by entering the **knowledge model ID** of desired template (e.g. `dsw:lifesciences:2.6.3`) and pressing the *Import* button.

Note: In case of knowledge model present in the FAIR Wizard Registry, we will be notified about the available upgrades.

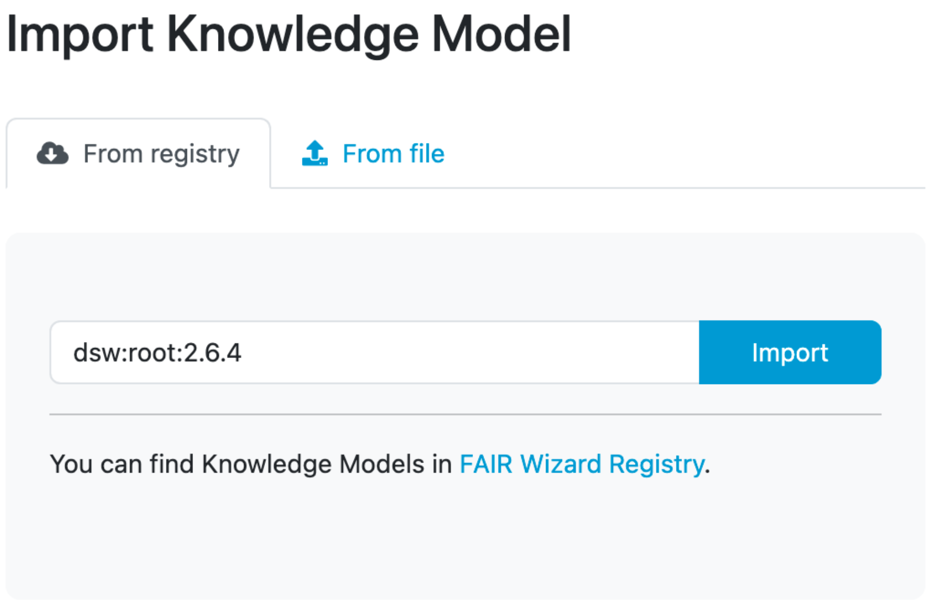


Fig. 5: Input for importing a knowledge model from FAIR Wizard Registry.

From file

We can import a knowledge model as a KM file. Such a file can be created as an export from FAIR Wizard (from *Knowledge Model List* or *Knowledge Model Detail*).

Knowledge Model Detail

We can visit a knowledge model detail by clicking on a desired KM in the *Knowledge Model List* (or selecting *View detail* from the right item menu). The detail shows basic information about the knowledge model such as its name, ID, version, license, metamodel version, or (if applicable) what is the parent knowledge model.

The main part of the detail is the README of the KM that should contain basic information and changelog. In the right panel under the basic information, we can navigate to other versions of the KM or navigate to the FAIR Wizard Registry (if the KM is present there).

In the top bar, we can *Export* the knowledge model as a KM file or *Delete* this version of the knowledge model (only if it is not already used for some projects or other KMs and editors).

In the top pane, we can see the options based on our role:

- *Preview* can be used to check the content of the KM via the *Knowledge Model Preview* feature.
- *Export* for exporting the latest version of the KM as a file.
- *Create KM editor* is a shortcut for *Create Knowledge Model Editor* for creating a new version.
- *Fork KM* is again a shortcut for *Create Knowledge Model Editor* for to create a fork (some more specific KM based on this one).
- *Create project* is a shortcut to *Create Project* with this KM.
- *Set deprecated* or *Restore* for setting a KM deprecated when we no longer want the **researchers** to use it.

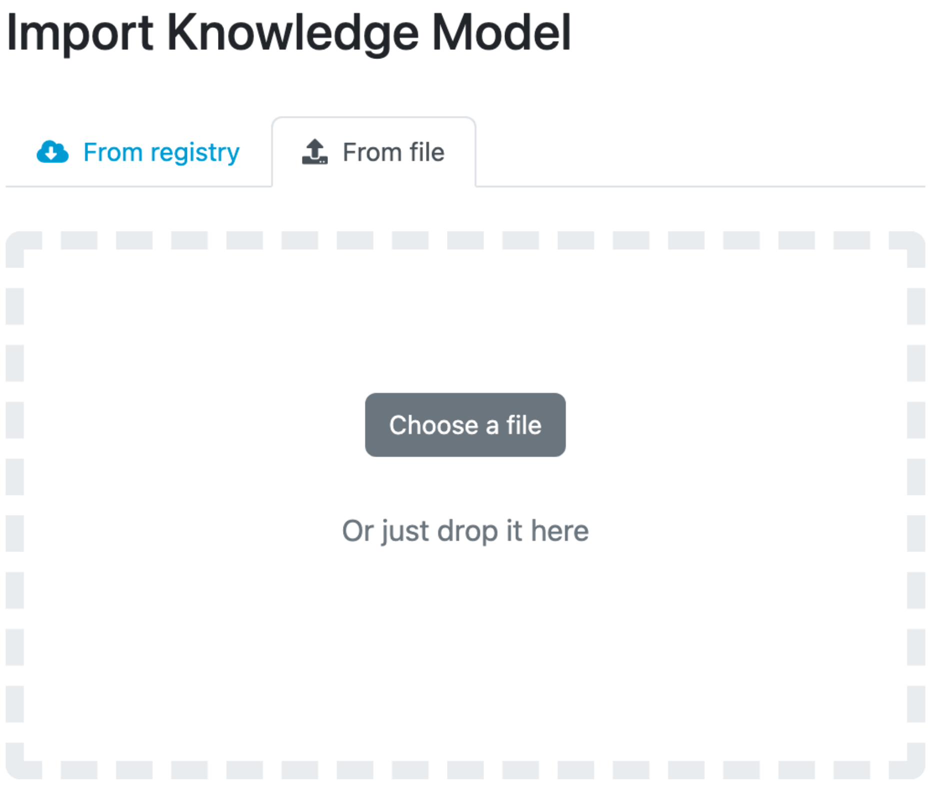
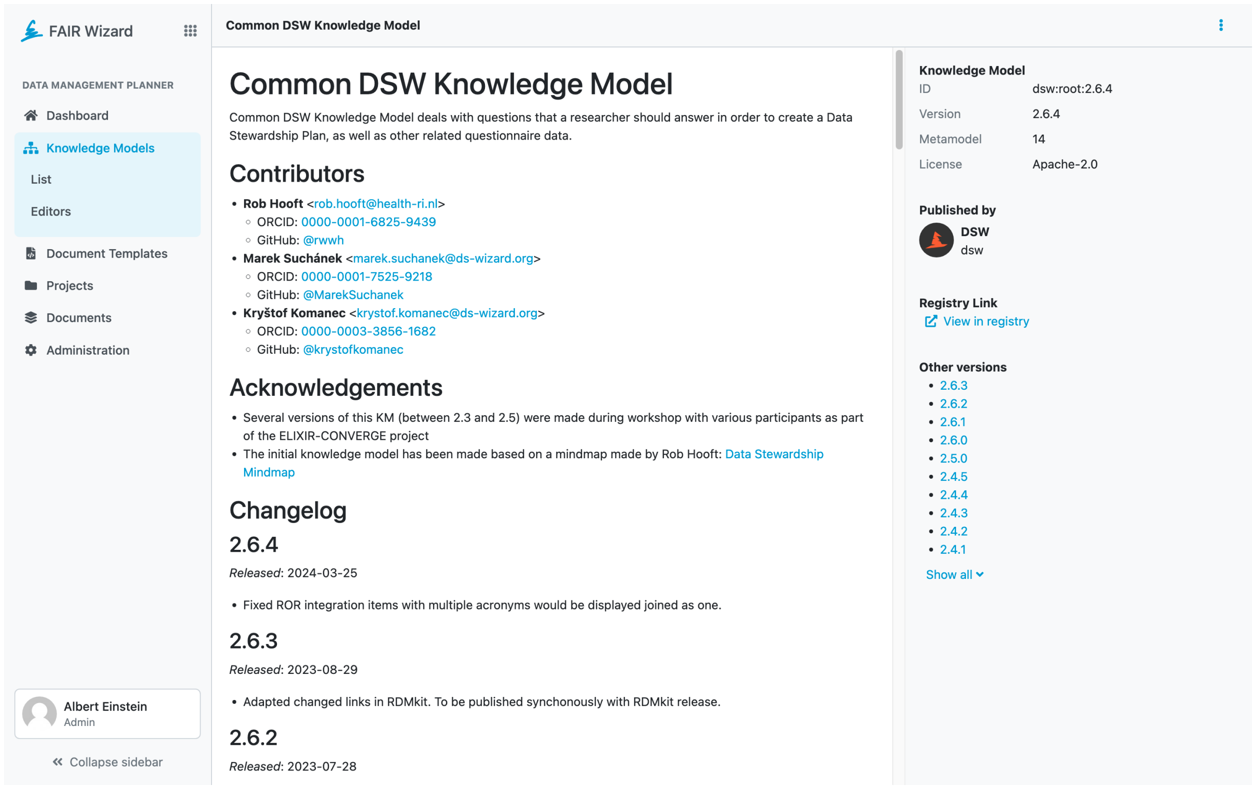


Fig. 6: Input for importing a knowledge model using a KM package.

- *Delete* the specific version of the KM (possible only if is not used in any projects or linked in other KMs and editors).

If we are not seeing the latest version of the KM, a warning message is shown in the top. Similarly, we will see a notification that update is available if there is a newer version in the FAIR Wizard Registry (if configured).



Common DSW Knowledge Model

Common DSW Knowledge Model deals with questions that a researcher should answer in order to create a Data Stewardship Plan, as well as other related questionnaire data.

Contributors

- **Rob Hooft** <rob.hooft@health-ri.nl>
  - ORCID: 0000-0001-6825-9439
  - GitHub: @rwwh
- **Marek Suchánek** <marek.suchanek@ds-wizard.org>
  - ORCID: 0000-0001-7525-9218
  - GitHub: @MarekSuchanek
- **Kryštof Komanec** <krystof.komanec@ds-wizard.org>
  - ORCID: 0000-0003-3856-1682
  - GitHub: @krystofkomanec

Acknowledgements

- Several versions of this KM (between 2.3 and 2.5) were made during workshop with various participants as part of the ELIXIR-CONVERGE project
- The initial knowledge model has been made based on a mindmap made by Rob Hooft: Data Stewardship Mindmap

Changelog

2.6.4

Released: 2024-03-25

- Fixed ROR integration items with multiple acronyms would be displayed joined as one.

2.6.3

Released: 2023-08-29

- Adapted changed links in RDMkit. To be published synchonously with RDMkit release.

2.6.2

Released: 2023-07-28

Knowledge Model

ID	dsw:root:2.6.4
Version	2.6.4
Metamodel	14
License	Apache-2.0

Published by

DSW
dsw

Registry Link

View in registry

Other versions

- 2.6.3
- 2.6.2
- 2.6.1
- 2.6.0
- 2.5.0
- 2.4.5
- 2.4.4
- 2.4.3
- 2.4.2
- 2.4.1

Show all

Fig. 7: Detail of a knowledge model.

Knowledge Model Preview

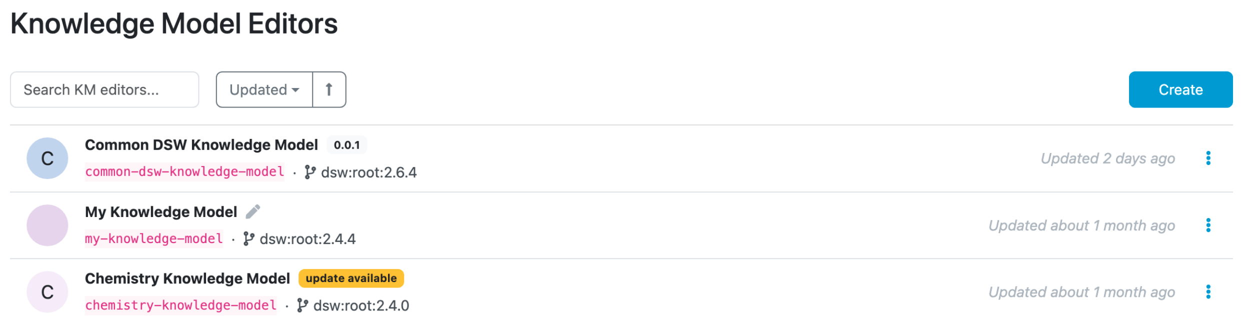
The preview feature for knowledge models allows us to check the content, i.e. how the questionnaire looks like. It can be used to navigate just through the top-level questions. However, with a know UUID of the question, we can create a direct link to a certain question:

For a nested question, all needed question above in the tree will be automatically prefilled. The KM preview may be also available to non-logged-in users, if configured by administrators in *Knowledge Models Settings*.

A user can directly create a new project from the KM preview. Again, this can be available also for non-logged-in users if anonymous projects are enabled by administrations in *Projects Settings*.

Knowledge Model Editors

Here, we can see a list of all knoweldge model editors. Everyone with the data steward role assigned can see all the knowledge model editors.


Knowledge Model Editors

Common DSW Knowledge Model 0.0.1
common-dsw-knowledge-model · dsw:root:2.6.4
Updated 2 days ago

My Knowledge Model
my-knowledge-model · dsw:root:2.4.4
Updated about 1 month ago

Chemistry Knowledge Model update available
chemistry-knowledge-model · dsw:root:2.4.0
Updated about 1 month ago

Fig. 8: List of knowledge model editors.

We can use the search field to search for a specific KM editor. The editors are sorted by when they were last updated but we can change that.

We can *create a new knowledge model editor* by clicking the *Create* button.

By clicking the triple dots on each of the item in the list we can access some actions:

- **Open Editor** - simply open the *editor detail*
- **Upgrade** - if there is a newer version of parent knowledge model, we can use upgrade action to start a *knowledge model migration*, otherwise the action is not visible
- **Publish** - to *publish* a new version of the knowledge model
- **Delete** - to delete the knowledge model editor (cannot be undone)

If there is an ongoing *knowledge model migration*, there are different actions:

- **Continue migration**
- **Cancel migration**

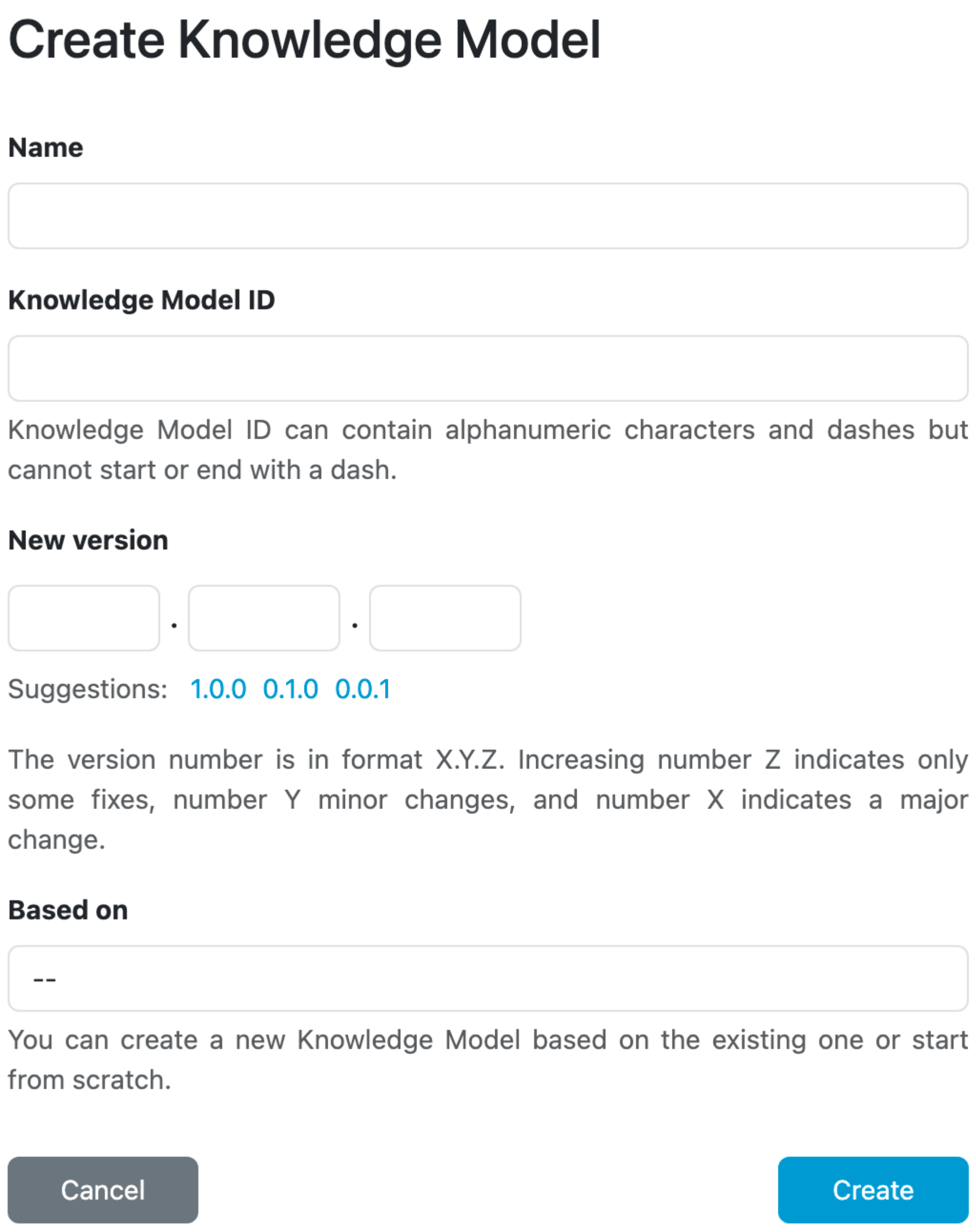
Create Knowledge Model Editor

We can create a new knowledge model editor by navigating to *Knowledge Models* → *Editors* in the main menu and then clicking the *Create* button.

Every knowledge model needs to have a **name**, a **knowledge model ID** and **version**. The name should be something descriptive to help users understand what the knowledge model is about. The knowledge model ID is used for the identification together with the *organization ID* and knowledge model version after it is published. So the identifier of the knowledge model is:

```
<organizationId>:<knowledgeModelId>:<version>
```

We can create a new project either from scratch, i.e. the new knowledge model will be empty and we will build it all ourselves, or based on an existing knowledge models, which means that everything from the chosen knowledge model will be copied to ours. We can start from there and add, delete, or modifiy the existing entities in there. We just need to choose the original knowledge model in the **based on** field. Alternatively, we can open the *knowledge model detail* and click on *Fork KM* there.



Create Knowledge Model

Name

Knowledge Model ID

Knowledge Model ID can contain alphanumeric characters and dashes but cannot start or end with a dash.

New version

. .

Suggestions: 1.0.0 0.1.0 0.0.1

The version number is in format X.Y.Z. Increasing number Z indicates only some fixes, number Y minor changes, and number X indicates a major change.

Based on

--

You can create a new Knowledge Model based on the existing one or start from scratch.

Fig. 9: Form for creating a new knowledge model.

We can only have one knowledge model editor with the same knowledge model ID. If we deleted the editor but want to continue working on that knowledge model, we can create a new editor with the same knowledge model ID. Or we can open the *knowledge model detail* and click on *Create KM editor* there to have the editor create form prefilled.

Knowledge Model Editor

Knowledge model editor is where we build knowledge models. In this section, we will see what entities we can add there, how they are connected, how to work with the editor and how to publish the knowledge model.

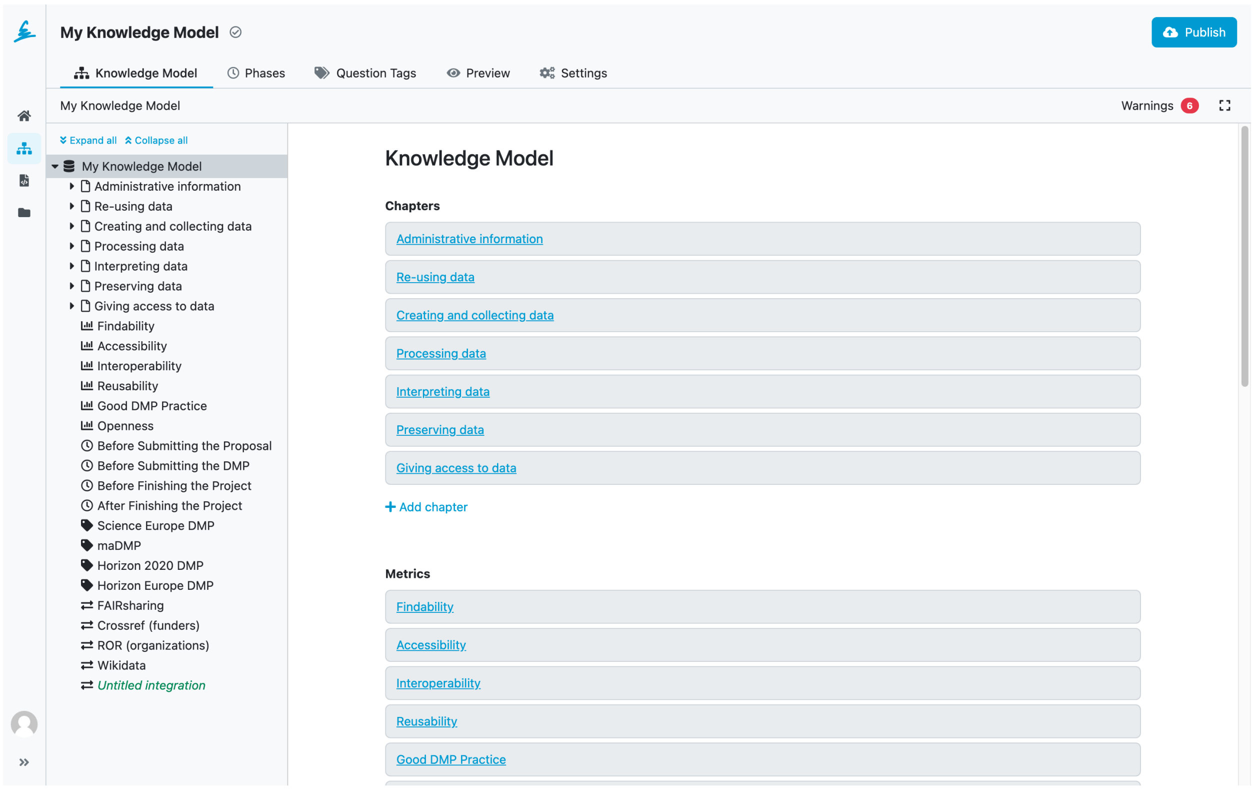


Fig. 10: Knowledge model editor.

Knowledge Model

The *Knowledge Model* tab is where we work on the entities (such as questions or answers) that appear in the knowledge model. We define the structure here.

Navigation

In the top row, we can see the breadcrumbs that show us where we are within the knowledge model. This allows us to quickly jump back up in the hierarchy.

On the lefthand side of the editor, there is a navigation tree reflecting the structure of the knowledge model. We can click on the arrows to expand or collapse individual entities, or click on *Expand all* or *Collapse all*. By clicking on the entity name, we open its editor.

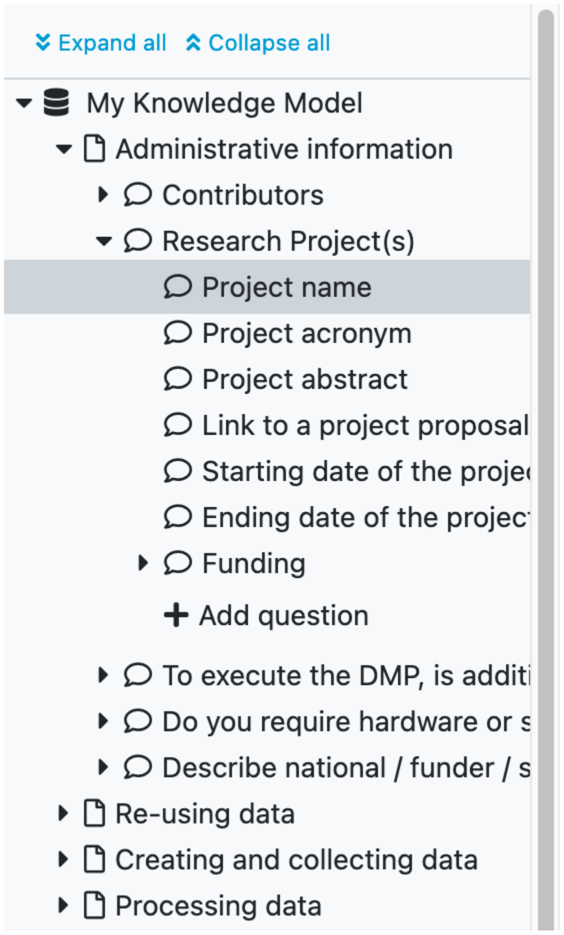

Expand all
Collapse all
My Knowledge Model
Administrative information
Contributors
Research Project(s)
Project name
Project acronym
Project abstract
Link to a project proposal
Starting date of the proje
Ending date of the projec
Funding
Add question
To execute the DMP, is additi
Do you require hardware or s
Describe national / funder / s
Re-using data
Creating and collecting data
Processing data

Fig. 11: Knowledge model editor navigation.

Tip: There is no search, however, if we want to quickly find an entity by name we can click on *Expand all* in the navigation tree and use search functionality of our web browser.

Editors

The main area of the *Knowledge Model* tab is the actual editor. The form fields change based on the entity we edit, but there are some shared actions:

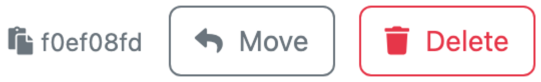


Fig. 12: Editor action buttons.

- **Copy UUID** - every entity has a generated UUID, we can use this button to copy it. We usually need it for *document template development*.
- **Move** - we can move entities around the knowledge model. However, not every entity can be put under everything. We can open the move modal window and see where the current entity could be move to.
- **Delete** - delete is simply used for deleting the entities. This action cannot be undone, so we need to be careful what we delete.

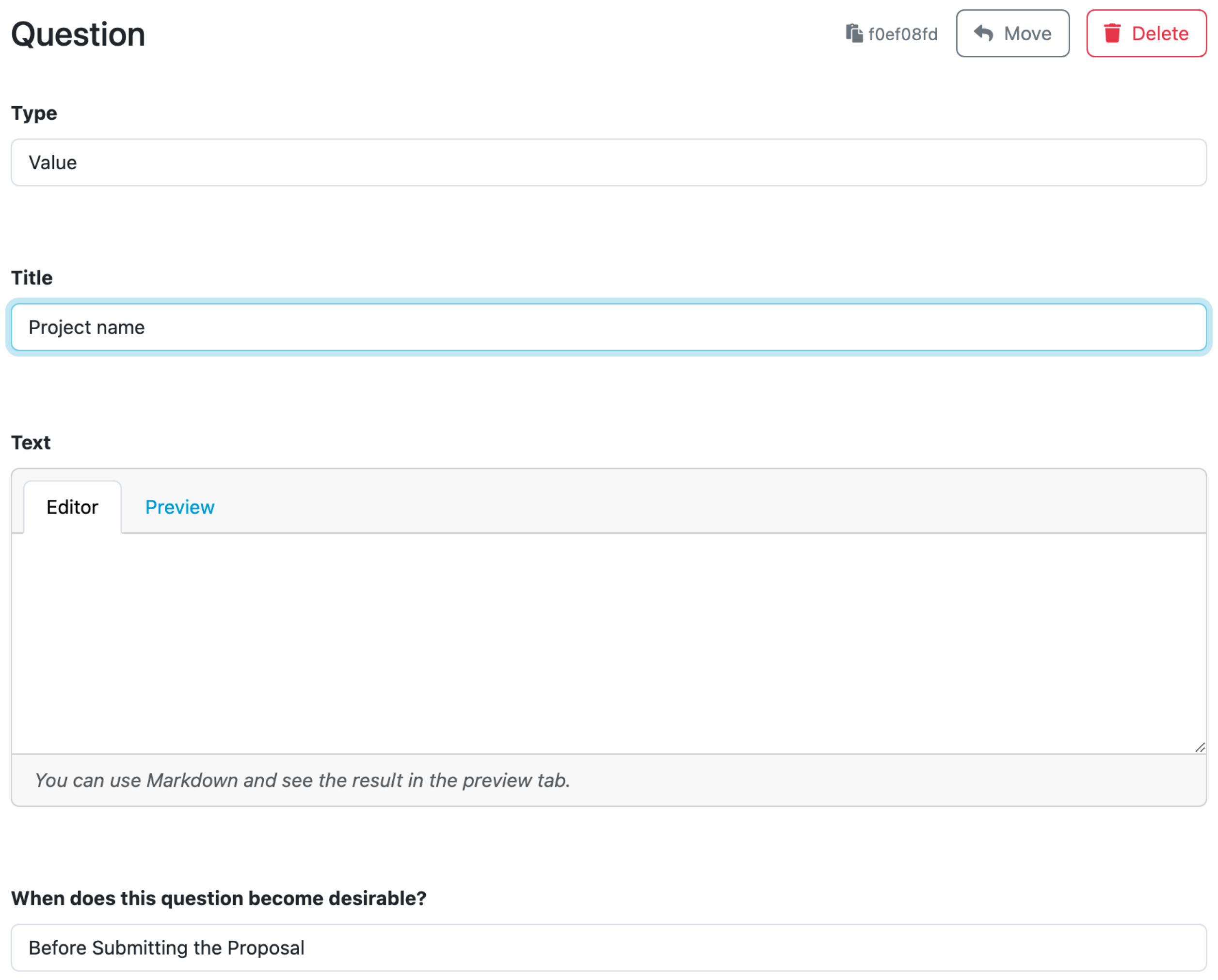
There are different entities we can edit in the knowledge model, the editor shows different fields based on what we edit:

- *Knowledge Model*
- *Chapter*
- *Question*
- *Answer*
- *Choice*
- *Reference*
- *Expert*
- *Metric*
- *Phase*
- *Question Tag*
- *Integration*

Besides their own fields, each entity has so called **Annotations**. They are arbitrary key value pairs that can be assigned to the entity and used later, when *developing a document template*.

Warnings

The editor checks for some possible problems, such as empty title for a chapter or no answers for an options question. If there are any, the *Warnings* tab appear and we can quickly navigate to those problems and fix them.


Question

f0ef08fd

Type

Value

Title

Project name

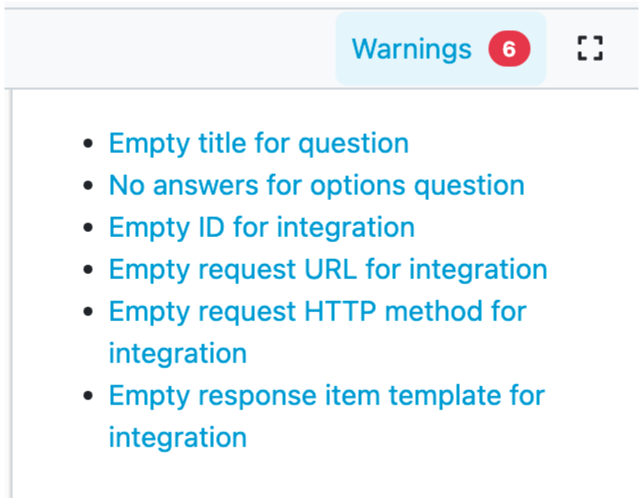
Text

You can use Markdown and see the result in the preview tab.

When does this question become desirable?

Before Submitting the Proposal

Fig. 13: Example of question editor form.



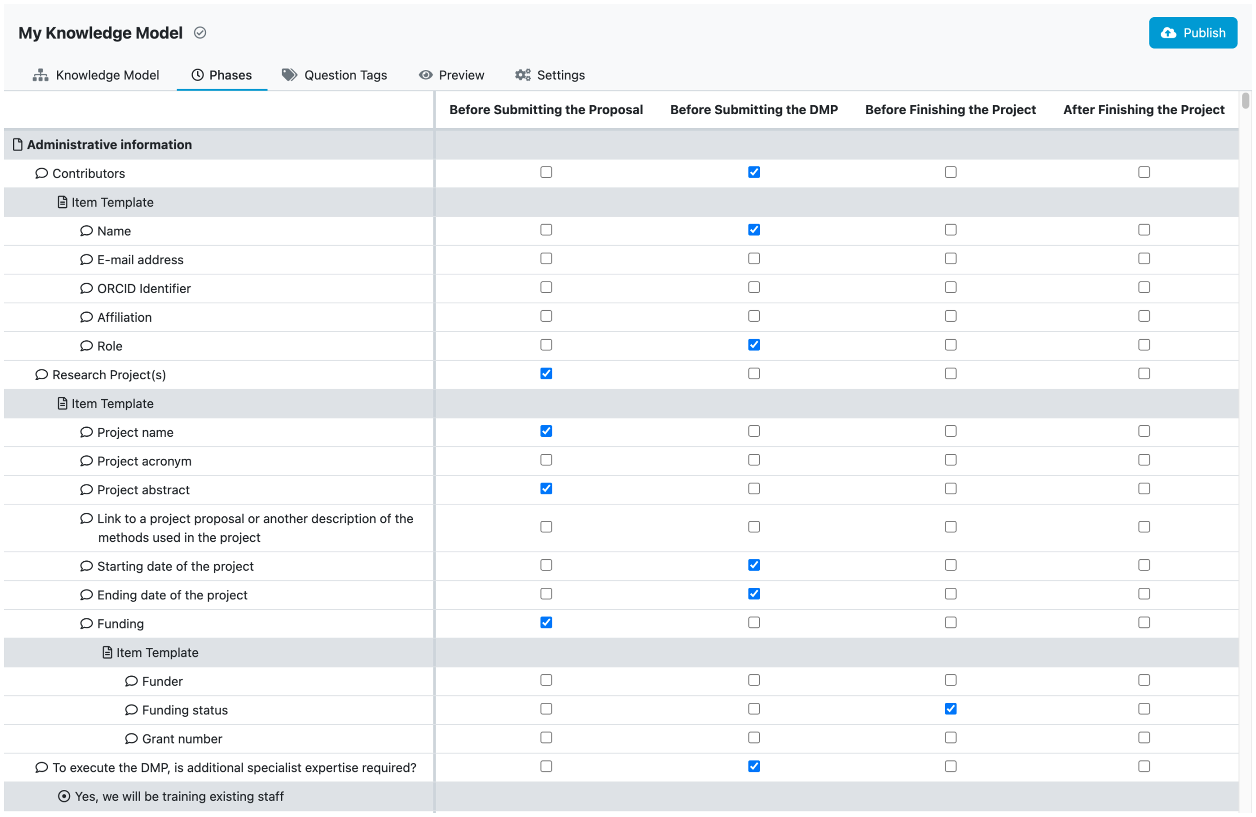
- Empty title for question
- No answers for options question
- Empty ID for integration
- Empty request URL for integration
- Empty request HTTP method for integration
- Empty response item template for integration

Fig. 14: Warnings in the knowledge model editor.

Phases

If there are any *project phases* set up in the knowledge model, we can use the *Phases* tab to see an overview of which questions were assigned to be completed in each phase and to change the phase assignments.

We can simply click to the checkbox in the corresponding phase - question table cell to assign/unassign the phase to that particular question.



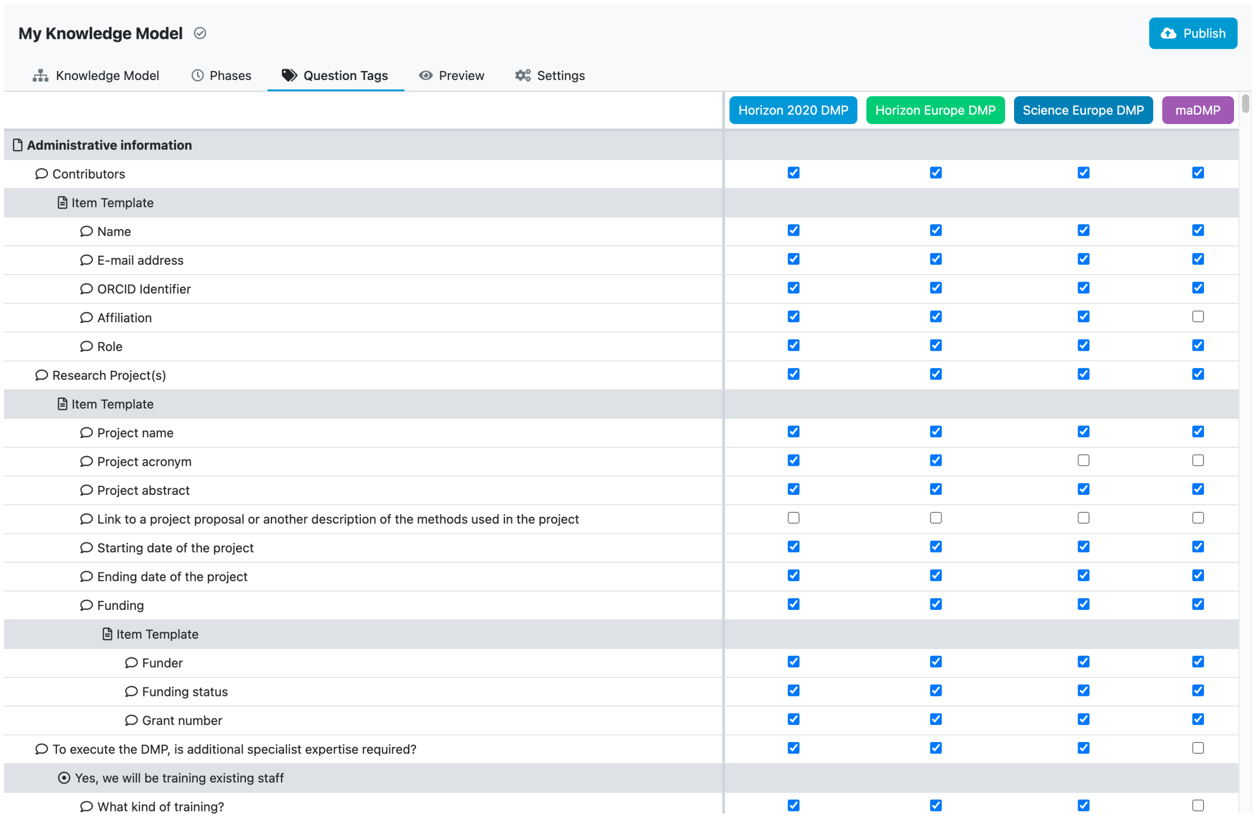
	Before Submitting the Proposal	Before Submitting the DMP	Before Finishing the Project	After Finishing the Project
Administrative information				
Contributors	☐	☑	☐	☐
Item Template				
Name	☐	☑	☐	☐
E-mail address	☐	☐	☐	☐
ORCID Identifier	☐	☐	☐	☐
Affiliation	☐	☐	☐	☐
Role	☐	☑	☐	☐
Research Project(s)	☑	☐	☐	☐
Item Template				
Project name	☑	☐	☐	☐
Project acronym	☐	☐	☐	☐
Project abstract	☑	☐	☐	☐
Link to a project proposal or another description of the methods used in the project	☐	☐	☐	☐
Starting date of the project	☐	☑	☐	☐
Ending date of the project	☐	☑	☐	☐
Funding	☑	☐	☐	☐
Item Template				
Funder	☐	☐	☐	☐
Funding status	☐	☐	☑	☐
Grant number	☐	☐	☐	☐
To execute the DMP, is additional specialist expertise required?	☐	☑	☐	☐
Yes, we will be training existing staff				

Fig. 15: Phases editor where we can assign questions to phases.

Question Tags

If there are any *question tags* set up in the knowledge model, we can use the *Question Tags* tab to see the overview of what questions were these tags assigned to, and to assign the question tags to the questions.

We can simply click to the checkbox in the corresponding question tag - question table cell to assign/unassign the question tag to that particular question.

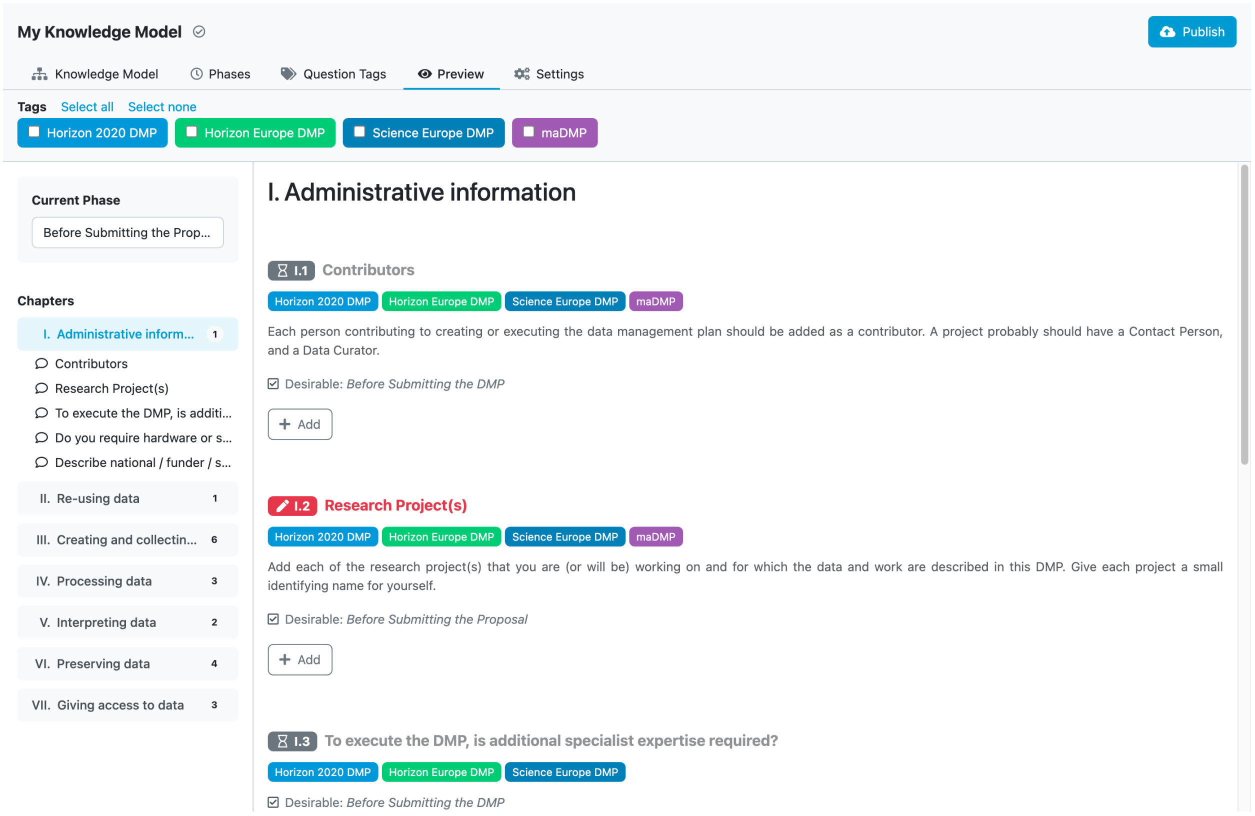


	Horizon 2020 DMP	Horizon Europe DMP	Science Europe DMP	maDMP
Administrative information				
Contributors	✓	✓	✓	✓
Item Template				
Name	✓	✓	✓	✓
E-mail address	✓	✓	✓	✓
ORCID Identifier	✓	✓	✓	✓
Affiliation	✓	✓	✓	☐
Role	✓	✓	✓	✓
Research Project(s)	✓	✓	✓	✓
Item Template				
Project name	✓	✓	✓	✓
Project acronym	✓	✓	☐	☐
Project abstract	✓	✓	✓	✓
Link to a project proposal or another description of the methods used in the project	☐	☐	☐	☐
Starting date of the project	✓	✓	✓	✓
Ending date of the project	✓	✓	✓	✓
Funding	✓	✓	✓	✓
Item Template				
Funder	✓	✓	✓	✓
Funding status	✓	✓	✓	✓
Grant number	✓	✓	✓	✓
To execute the DMP, is additional specialist expertise required?	✓	✓	✓	☐
Yes, we will be training existing staff				
What kind of training?	✓	✓	✓	☐

Fig. 16: Question tag editor where we can assign question tags to questions.

Preview

We can use *Preview* tab to quickly see how the resulting questionnaire would look like. If there are any *question tags* set up, we can also try different combinations of selected tags and observe the changes in the questionnaire.



I. Administrative information

I.1 Contributors

Horizon 2020 DMP · Horizon Europe DMP · Science Europe DMP · maDMP

Each person contributing to creating or executing the data management plan should be added as a contributor. A project probably should have a Contact Person, and a Data Curator.

Desirable: *Before Submitting the DMP*

I.2 Research Project(s)

Horizon 2020 DMP · Horizon Europe DMP · Science Europe DMP · maDMP

Add each of the research project(s) that you are (or will be) working on and for which the data and work are described in this DMP. Give each project a small identifying name for yourself.

Desirable: *Before Submitting the Proposal*

I.3 To execute the DMP, is additional specialist expertise required?

Horizon 2020 DMP · Horizon Europe DMP · Science Europe DMP

Desirable: *Before Submitting the DMP*

Fig. 17: Preview of the resulting questionnaire in the knowledge model editor.

Settings

The *Settings* tab, allows us to adjust atributes of the the knowledge model:

- **Name** - should be short name of the Knowledge Model.
- **Description** - this should be really short and descriptive.
- **Knowledge Model ID** - is an unique Knowledge Model identifier.
- **Version** - this is a number indicating which is the latest version of the Knowledge Model, because it can change over time.
- **License** - this is used when we want to share the knowledge model with other people so they know how they can do that. We recommend using a license identifier from SPDX Licens List.
- **Readme** - this is where we can describe everything we need about the knowledge model. We can, for example, include a changelog of what changed in which version, etc. We can use *Markdown* in this field to provide some nice formatting.

Note: **Name**, **Description** and **Version** are all visible to the researcher, when they select a Knowledge Model for their project. So the **Name** and **Description** should provide them with enough information to select a correct one.

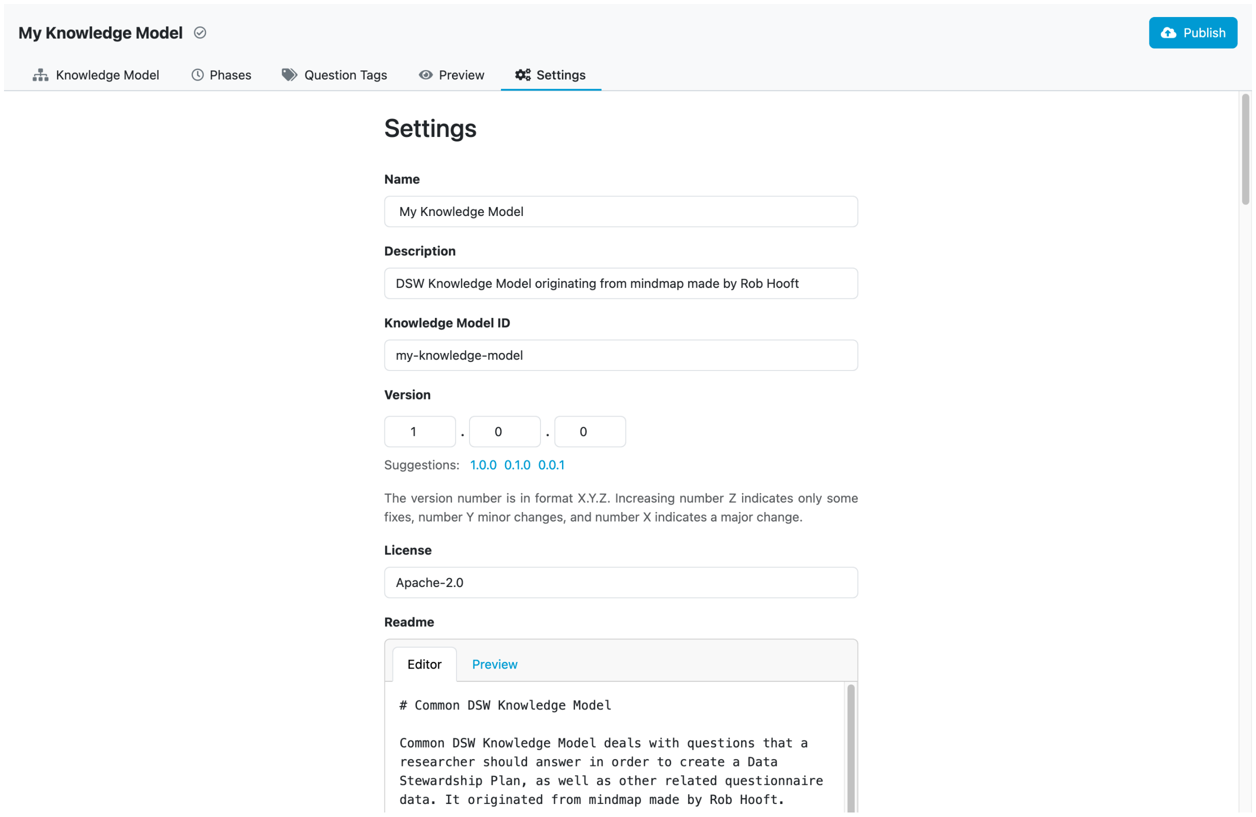
Knowledge Model ID together with the *organization ID* and knowledge model version after it is published are used for the identification. So the identifier of the knowledge model is:

```
<organizationId>:<knowledgeModelId>:<version>
```

For the version number we recommended using similar approach as in semantic versioning. So when we have a version `<major>.<minor>.<patch>`, change in the major version number indicates some breaking changes (deleting questions, significant changes in the questionnaire structure, etc.), change in minor version number indicates some new changes that are backwards compatible (i.e., adding a new question), and change in the patch version number indicate some fixes (such as fixing some typos).

If the knowledge model was based on another knowledge model, we can also see the **Parent Knowledge Model** in the settings.

In the **Danger Zone** we can delete the knowledge model. Once it is deleted it can **no longer be recovered**.



Settings

Name: My Knowledge Model

Description: DSW Knowledge Model originating from mindmap made by Rob Hooft

Knowledge Model ID: my-knowledge-model

Version: 1 . 0 . 0

Suggestions: 1.0.0 0.1.0 0.0.1

The version number is in format X.Y.Z. Increasing number Z indicates only some fixes, number Y minor changes, and number X indicates a major change.

License: Apache-2.0

Readme

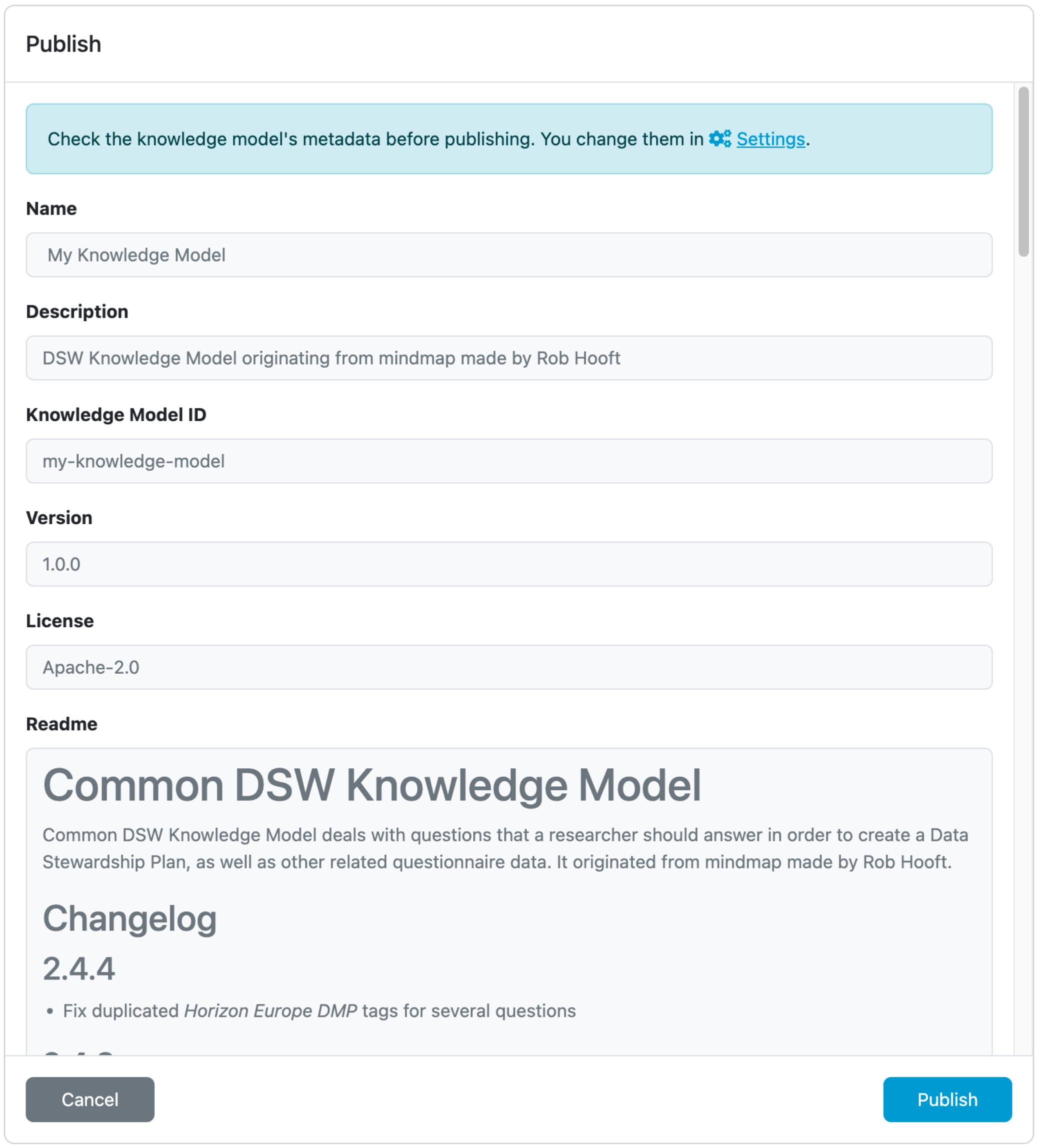
```
# Common DSW Knowledge Model

Common DSW Knowledge Model deals with questions that a
researcher should answer in order to create a Data
Stewardship Plan, as well as other related questionnaire
data. It originated from mindmap made by Rob Hooft.
```

Fig. 18: Knowledge Model settings.

Publish

Before the knowledge model can be used by researchers in their projects, we need to publish it. We can do that by clicking the *Publish* button in the top right corner of the *Knowledge Model Editor*.

If we click the button, we are prompted with the metadata details to check them before publishing. We cannot change anything here, so if we want to change it, we have to press Cancel and edit the details on the *Settings* tab of the *Knowledge Model Editor*.


Publish

Check the knowledge model's metadata before publishing. You change them in Settings.

Name

My Knowledge Model

Description

DSW Knowledge Model originating from mindmap made by Rob Hooft

Knowledge Model ID

my-knowledge-model

Version

1.0.0

License

Apache-2.0

Readme

Common DSW Knowledge Model

Common DSW Knowledge Model deals with questions that a researcher should answer in order to create a Data Stewardship Plan, as well as other related questionnaire data. It originated from mindmap made by Rob Hooft.

Changelog

2.4.4

- Fix duplicated *Horizon Europe DMP* tags for several questions

Fig. 19: Publish dialog where we can confirm or cancel publishing of the Knowledge Model.

If we confirm the publishing of the Knowledge Model by clicking *Publish* in the modal window, the Knowledge Model becomes available to all users and is accessible in *Knowledge Model List*.

The Knowledge Model Editor will remain in the *Knowledge Model Editors* list and will be available for any future changes.

Knowledge Model Migration

Knowledge model can be created either from scratch or based on existing one and further modified. If that happened, we can call the original **Parent KM** and the modification **Child KM**. We can create the Child KM any *published version* of the Parent KM, make some modification and publish a version of the Child KM. However, the Parent KM can evolve, and at some point we might want to have those changes in our Child KM, too.

That is what the knowledge model migration is for. Once a new version of the Parent KM is published, we can start the KM migration where we go through these changes. We can choose whether we want to apply or reject these modification to our Child KM during migration. At the end, we publish a new version of the Child KM with all the selected changes.

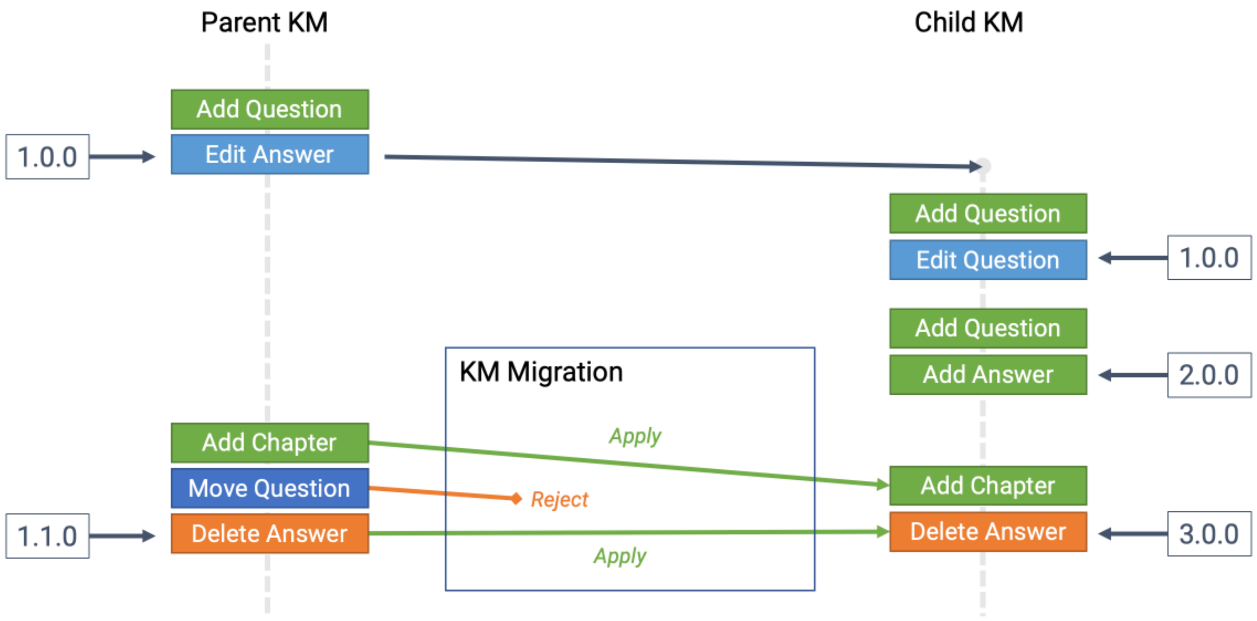


Fig. 20: Schematic representation of KM migration.

Creating a Knowledge Model Migration

We can start the knowledge model migration from the *list of knowledge model editors*. If there is a newer version of the Parent KM available for particular knowledge model editor, we can see the *update available* badge next to the name of the editor.

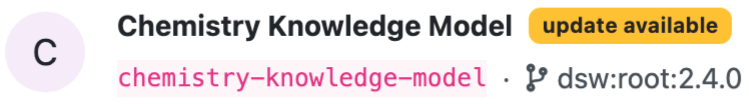

C
Chemistry Knowledge Model
update available
chemistry-knowledge-model · dsw:root:2.4.0

Fig. 21: Badge indicating that there is a newer version of the Parent KM.

We can either click on the badge directly, or choose *Upgrade* option from the dropdown menu. It will open a modal window where we can choose a new version of the Parent KM that we want to migrate our Child KM to. Usually, we want to pick the latest.

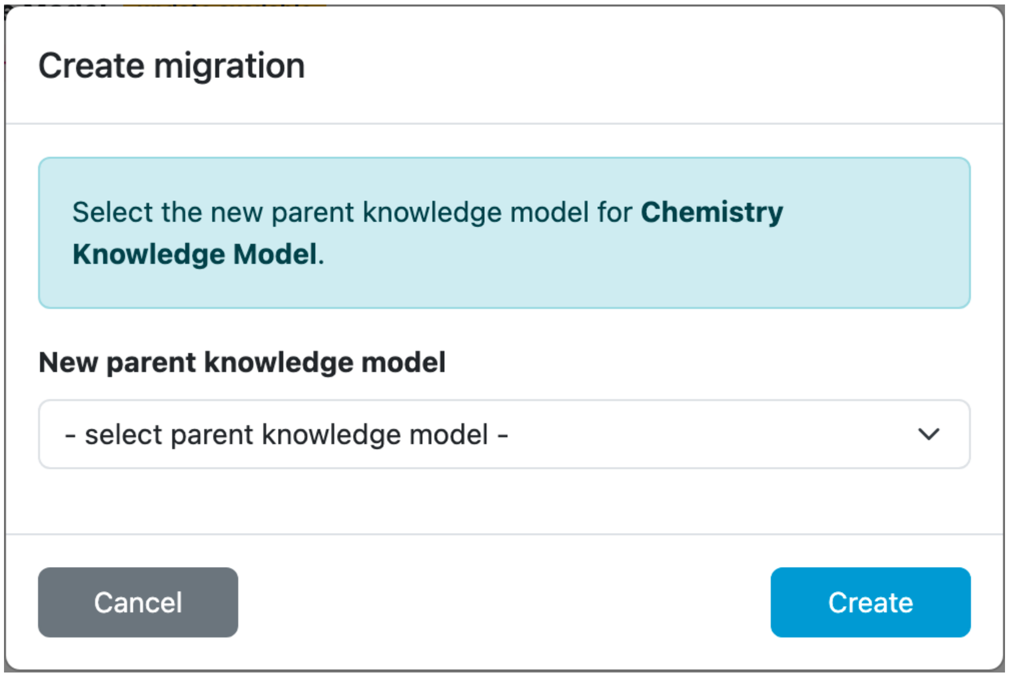

Create migration
Select the new parent knowledge model for **Chemistry Knowledge Model**.
New parent knowledge model
- select parent knowledge model -
Cancel
Create

Fig. 22: Modal window to create a new knowledge model migration.

Knowledge Model Migration

During the migration, we can see all the changes one by one and can choose whether we want to *Apply* or *Reject* the change. We can also choose to *Apply all* if we simply want everything.

Cancelling a Knowledge Model Migration

We can cancell the knowledge model migration at any point before we publish the new version of the Child KM. We need to navigate to the *list of knowledge model editors* and choose *Cancel migration* from the dropdown menu for the desired KM editor.

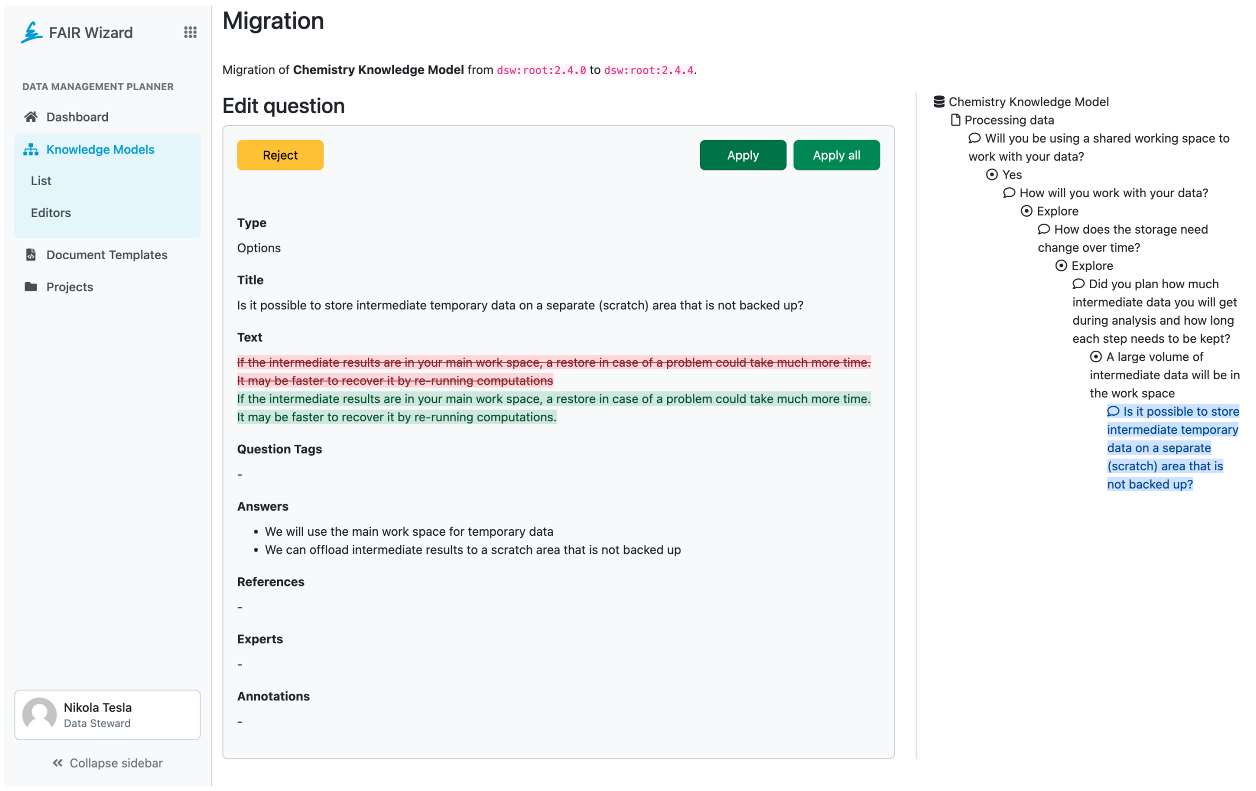
Finishing a Knowledge Model Migration

After we resolve all the changes, we are ready to publish the new version of the Child KM. To do that, we need to click on the *Publish* → button. This will open the Publish new version screen where we need to provide additional information for the new version of the Knowledge Model.

The publish screen shows us some information about the knowledge model, such as it's **Knowledge Model Name** and **Knowledge Model ID**. We need to choose the new **version number**.

Note: We recommended using similar approach as in semantic versioning. So when we have a version `<major>.<minor>.<patch>`, change in the major version number indicates some breaking changes (deleting questions, significant changes in the questionnaire structure, etc.), change in minor version number indicates some new changes that are backwards compatible (i.e., adding a new question), and change in the patch version number indicate some fixes (such as fixing some typos).

Then we need to add some additional metadata (these fields are pre-filled if there was a previous version published):


FAIR Wizard
DATA MANAGEMENT PLANNER
Dashboard
Knowledge Models
List
Editors
Document Templates
Projects
Nikola Tesla
Data Steward
Collapse sidebar
Reject
Apply
Apply all
Chemistry Knowledge Model
Processing data
Will you be using a shared working space to work with your data?
Yes
How will you work with your data?
Explore
How does the storage need change over time?
Explore
Did you plan how much intermediate data you will get during analysis and how long each step needs to be kept?
A large volume of intermediate data will be in the work space
Is it possible to store intermediate temporary data on a separate (scratch) area that is not backed up?

Migration

Migration of **Chemistry Knowledge Model** from `dsw:root:2.4.0` to `dsw:root:2.4.4`.

Edit question

Type

Options

Title

Is it possible to store intermediate temporary data on a separate (scratch) area that is not backed up?

Text

~~If the intermediate results are in your main work space, a restore in case of a problem could take much more time. It may be faster to recover it by re-running computations~~

If the intermediate results are in your main work space, a restore in case of a problem could take much more time. It may be faster to recover it by re-running computations.

Question Tags

-

Answers

- We will use the main work space for temporary data
- We can offload intermediate results to a scratch area that is not backed up

References

-

Experts

-

Annotations

-

Fig. 23: During the migration we can apply or reject the changes form the Parent KM.

- **License** - this is used when we want to share the knowledge model with other people so they know how they can do that. We recommend using a license identifier from SPDX Licens List.
- **Description** - this should be really short and descriptive. It is used, for example, in select boxes when creating a new project to help researchers choose the best knowledge model for their use case.
- **Readme** - this is where we can describe everything we need about the knowledge model. We can, for example, include a changelog of what changed in what version, etc. We can use *Markdown* in this field to provide some nice formatting.

1.2.2 Document Templates

This section is about managing the document templates and document template editors. Similarly to Knowledge Models, we can manage *Document Template List* or work on new or customise existing templates using *Document Template Editors*.

Document Template List

As **data stewards** and **admins**, we can check and manage all document templates from the list accessible from the main menu via *Document Templates*. The list can be filtered and sorted by name.

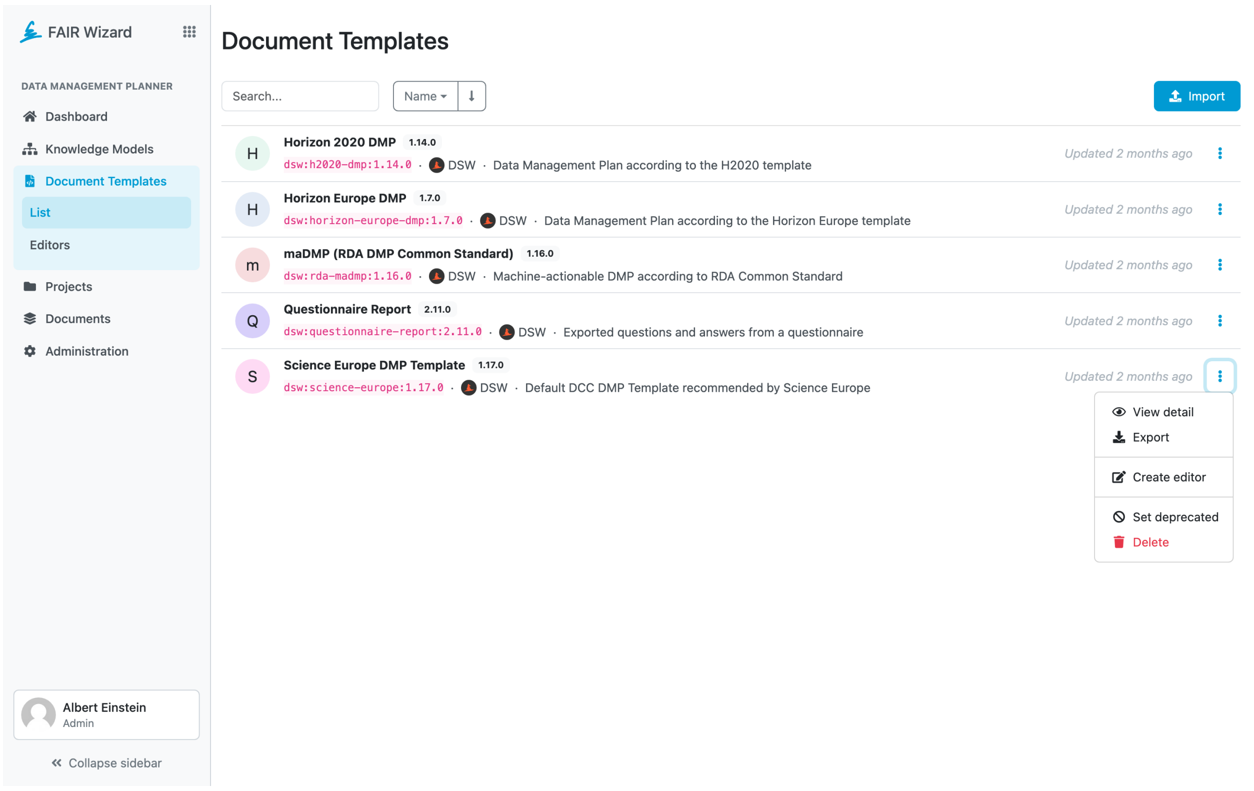
For each document template, we can see the latest version present; however, we can see all the versions by accessing the *Document Template Detail* by clicking the name of the template or via *View detail* from the right dropdown item menu. The dropdown menu also offers *Export* of the latest document template or *Delete* of entire template (all its versions).

Note: A document template can be deleted only if it is not used already for documents, projects, or settings.

Each item may be marked with *unsupported metamodel* when the document template is not compatible with the version of FAIR Wizard. *Metamodel Schemas* are used to define structures that developers can interact with. If the template originates from the FAIR Wizard Registry, *update available* will appear.

If your template is not from the registry, you will have to *Publish* a new version via template editor, which will increase the metamodel version automatically.

Finally, we can use *Import* new document templates by clicking the top right button (see *Document Template Import* for details).



Horizon 2020 DMP 1.14.0
dsw:h2020-dmp:1.14.0 · DSW · Data Management Plan according to the H2020 template
Updated 2 months ago

Horizon Europe DMP 1.7.0
dsw:horizon-europe-dmp:1.7.0 · DSW · Data Management Plan according to the Horizon Europe template
Updated 2 months ago

maDMP (RDA DMP Common Standard) 1.16.0
dsw:rda-madmp:1.16.0 · DSW · Machine-actionable DMP according to RDA Common Standard
Updated 2 months ago

Questionnaire Report 2.11.0
dsw:questionnaire-report:2.11.0 · DSW · Exported questions and answers from a questionnaire
Updated 2 months ago

Science Europe DMP Template 1.17.0
dsw:science-europe:1.17.0 · DSW · Default DCC DMP Template recommended by Science Europe
Updated 2 months ago

Fig. 24: List of all document templates with actions.

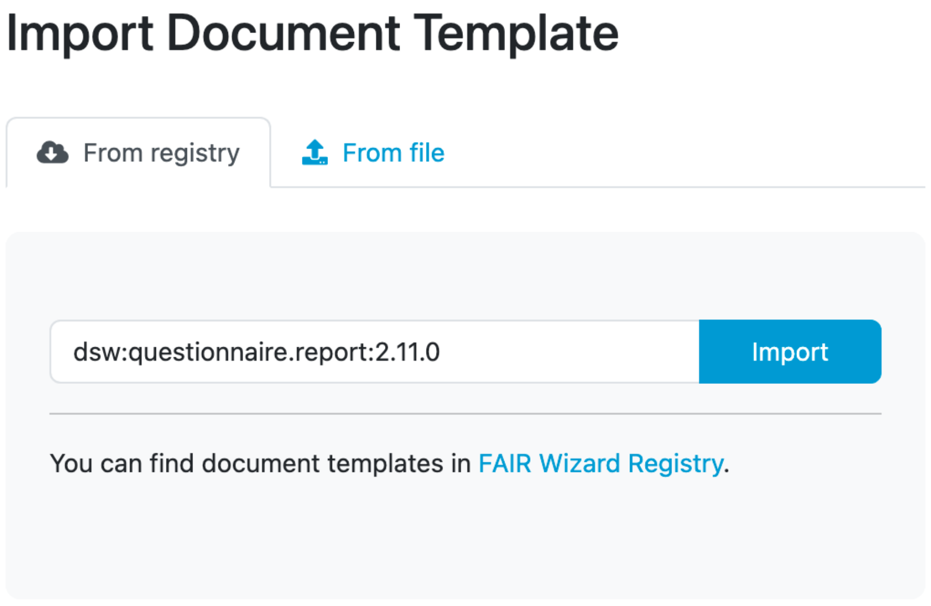
Document Template Import

We can import an existing document template by navigating to *Document Template List* (*Document Templates*) in the main menu and then clicking on *Import* button on the list of document templates.

From FAIR Wizard Registry

The FAIR Wizard is connected to the FAIR Wizard Registry. That allows us to import document templates from it by entering the **document template ID** of desired template (e.g. `dsw:questionnaire-report:2.11.0`) and pressing the *Import* button.

Note: In case of document templates present in the FAIR Wizard Registry, we will be notified about the available upgrades.



You can find document templates in FAIR Wizard Registry.

Fig. 25: Input for importing a document template from FAIR Wizard Registry.

From file

We can import a document template as a ZIP package. Such a package can be created as an export from FAIR Wizard or using the Template Development Kit (see *Document Template Development*).

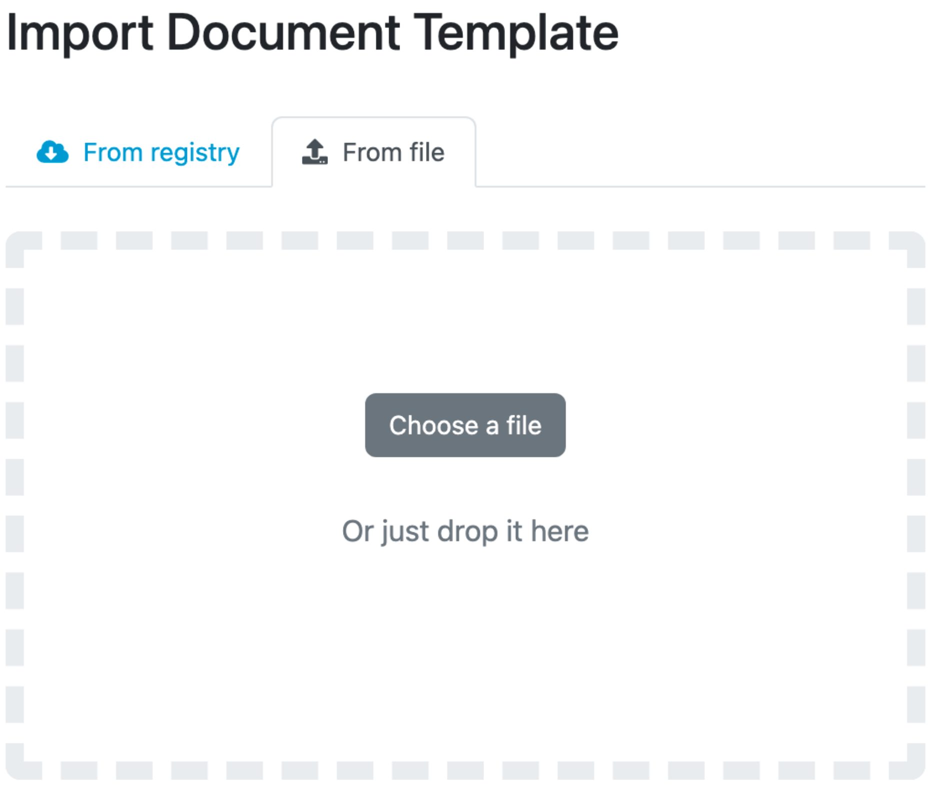


Fig. 26: Input for importing a document template using a ZIP package.

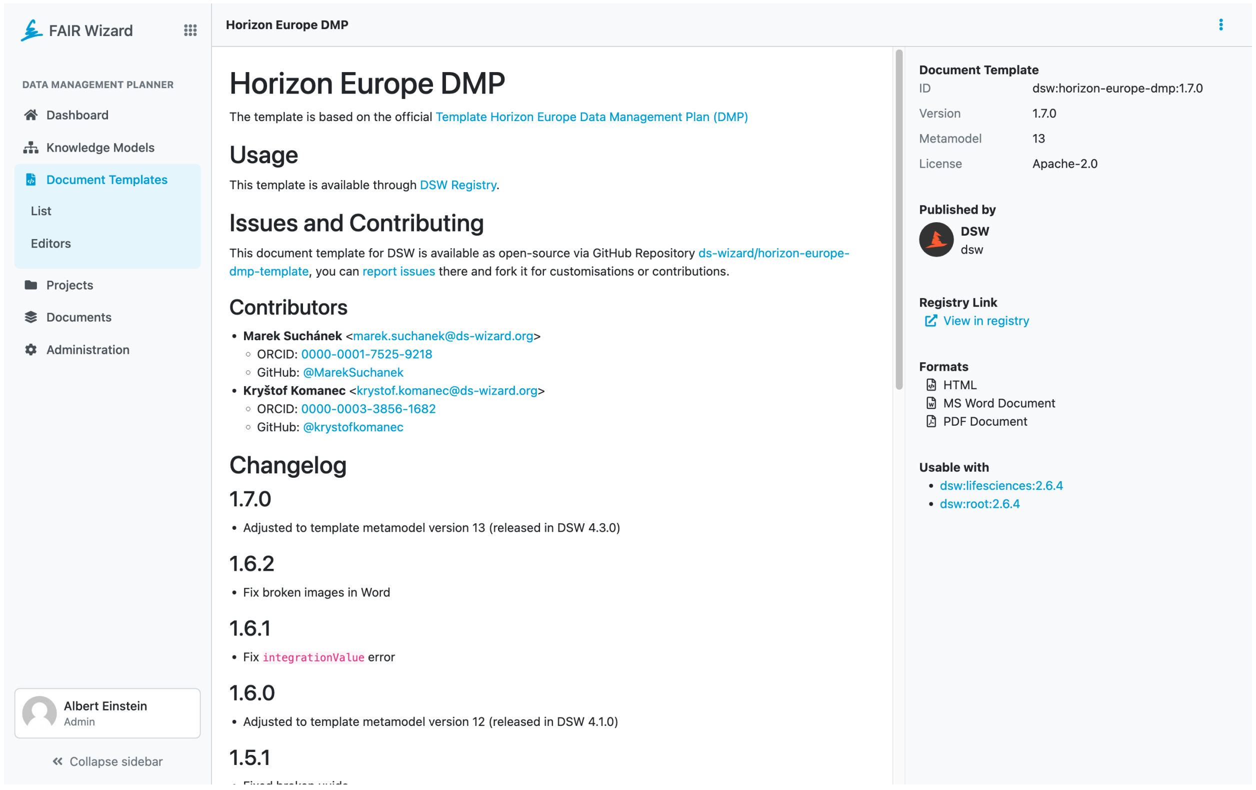
Document Template Detail

We can check a document template detail by clicking on a desired template in the *Document Template List* (or selecting *View detail* from the right dropdown). The detail shows basic information about the template such as its name, ID, version, license, metamodel version, or supported document formats.

The main part of the detail is the README of the template that should contain basic information and changelog for the template. In the right panel under the basic information, we can navigate to other versions of the document template, navigate to the FAIR Wizard Registry (if the template is present there), or check compatible knowledge model with the template.

In the top bar, we can *Export* the template as a ZIP package or *Delete* this version of the template (only if it is not already used for some documents). We can also quickly navigate to *Create Document Template Editor* by clicking *Create editor*; it will prepare editor creation for a new version of this document template. Finally, there is the possibility *Set deprecated* which will change the state of the document template so it is no longer usable by researchers in their projects (it becomes unavailable).

If we are not seeing the latest version of the template, a warning message is shown in the top. Similarly, we will see a notification that update is available if there is a newer version in the FAIR Wizard Registry (if configured).



Horizon Europe DMP

The template is based on the official Template Horizon Europe Data Management Plan (DMP)

Usage

This template is available through DSW Registry.

Issues and Contributing

This document template for DSW is available as open-source via GitHub Repository ds-wizard/horizon-europe-dmp-template, you can report issues there and fork it for customisations or contributions.

Contributors

- **Marek Suchánek** <marek.suchanek@ds-wizard.org>
  - ORCID: 0000-0001-7525-9218
  - GitHub: @MarekSuchanek
- **Kryštof Komanec** <krystof.komanec@ds-wizard.org>
  - ORCID: 0000-0003-3856-1682
  - GitHub: @krystofkomanec

Changelog

1.7.0

- Adjusted to template metamodel version 13 (released in DSW 4.3.0)

1.6.2

- Fix broken images in Word

1.6.1

- Fix `integrationValue` error

1.6.0

- Adjusted to template metamodel version 12 (released in DSW 4.1.0)

1.5.1

Document Template

ID	dsw:horizon-europe-dmp:1.7.0
Version	1.7.0
Metamodel	13
License	Apache-2.0

Published by
DSW
dsw

Registry Link
View in registry

Formats
HTML
MS Word Document
PDF Document

Usable with
- dsw:lifesciences:2.6.4
- dsw:root:2.6.4

Fig. 27: Detail of a document template.

Document Template Editors

On this page, we can see a list of all document template editors. Everyone with the data steward role assigned can see all the document template editors and use them.

We can use the search field to search for a specific document template editor. The editors are sorted by when they were last updated but we can change that.

We can *Create Document Template Editor* by clicking the *Create* button.

By clicking the triple dots on each of the item in the list we can access some actions:

- **Open Editor** - simply open the *Document Template Editor* (we can also click the name of the editor)
- **Delete** - to delete the document template editor (cannot be undone)

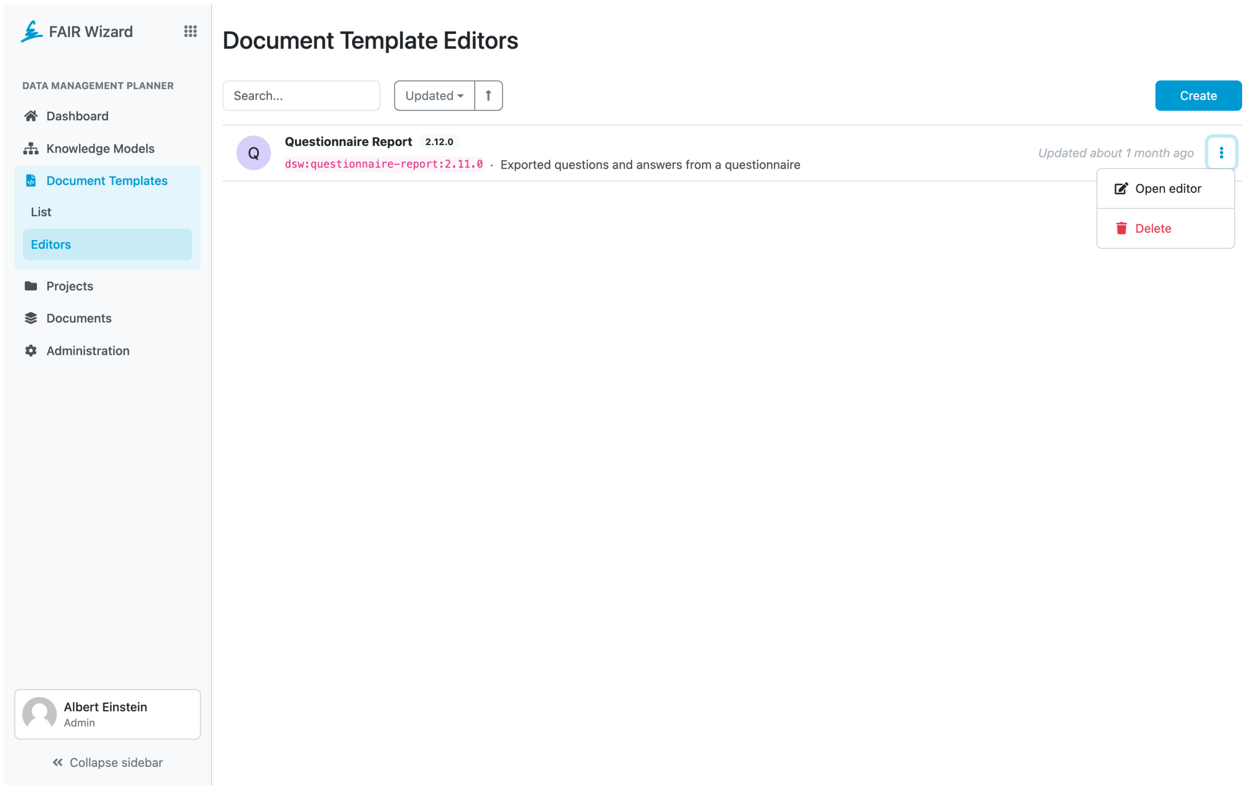


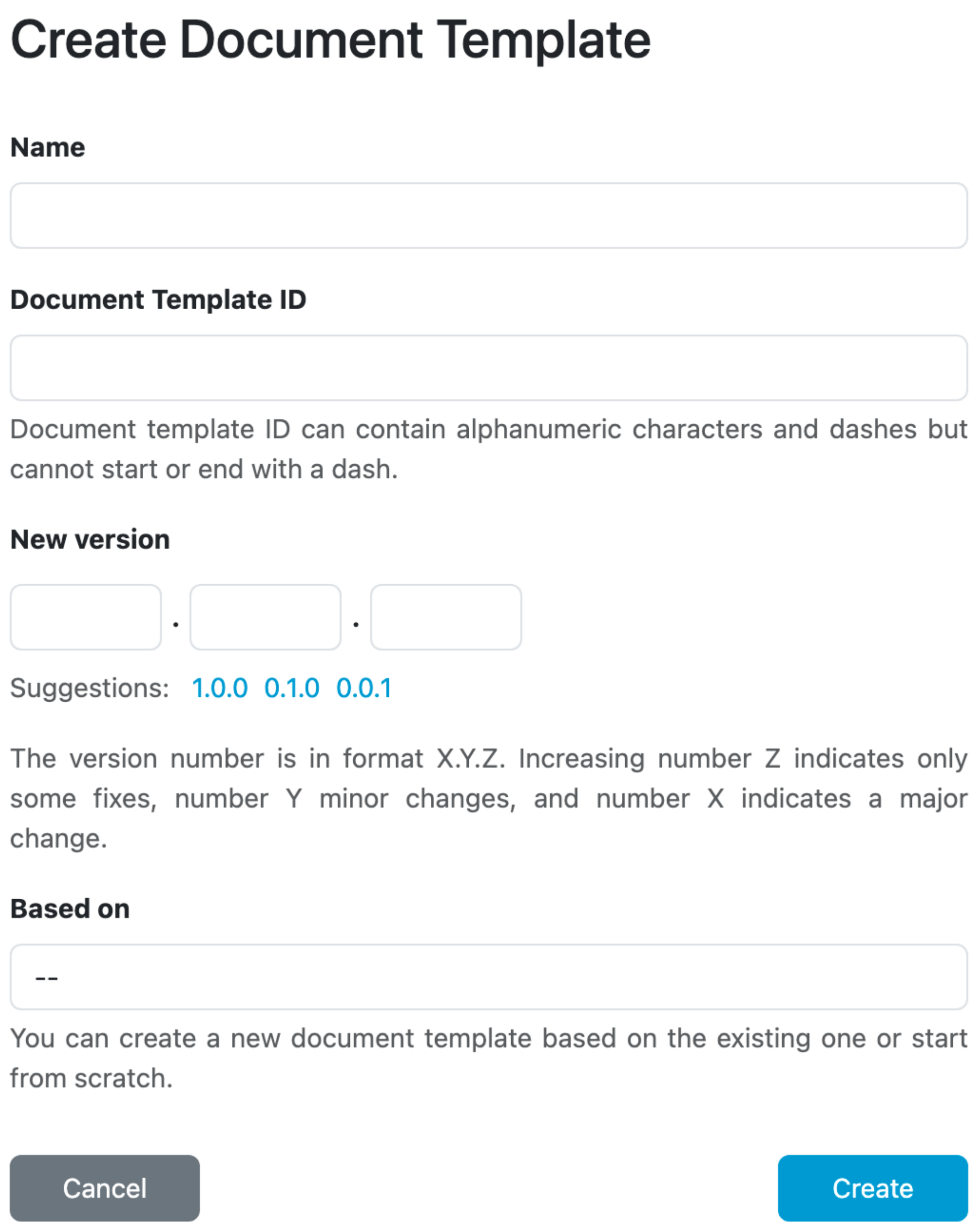
Fig. 28: List of document template editors with actions.

Create Document Template Editor

We can create a new document template editor by navigating to *Document Templates* → *Editors* in the main menu and then clicking the *Create* button.

Every document template needs to have a **Name**, a **Document Template ID** and **version**. The name should be something descriptive to help users understand what the document template is about. The Document Template ID is used for the identification together with the *organization ID* and document template version that we have to fill as a **New Version**. So the identifier of the document template is:

```
<organizationId>:<documentTemplateId>:<version>
```



Create Document Template

Name

Document Template ID

Document template ID can contain alphanumeric characters and dashes but cannot start or end with a dash.

New version

. .

Suggestions: 1.0.0 0.1.0 0.0.1

The version number is in format X.Y.Z. Increasing number Z indicates only some fixes, number Y minor changes, and number X indicates a major change.

Based on

--

You can create a new document template based on the existing one or start from scratch.

Fig. 29: Form for creating a new document template editor.

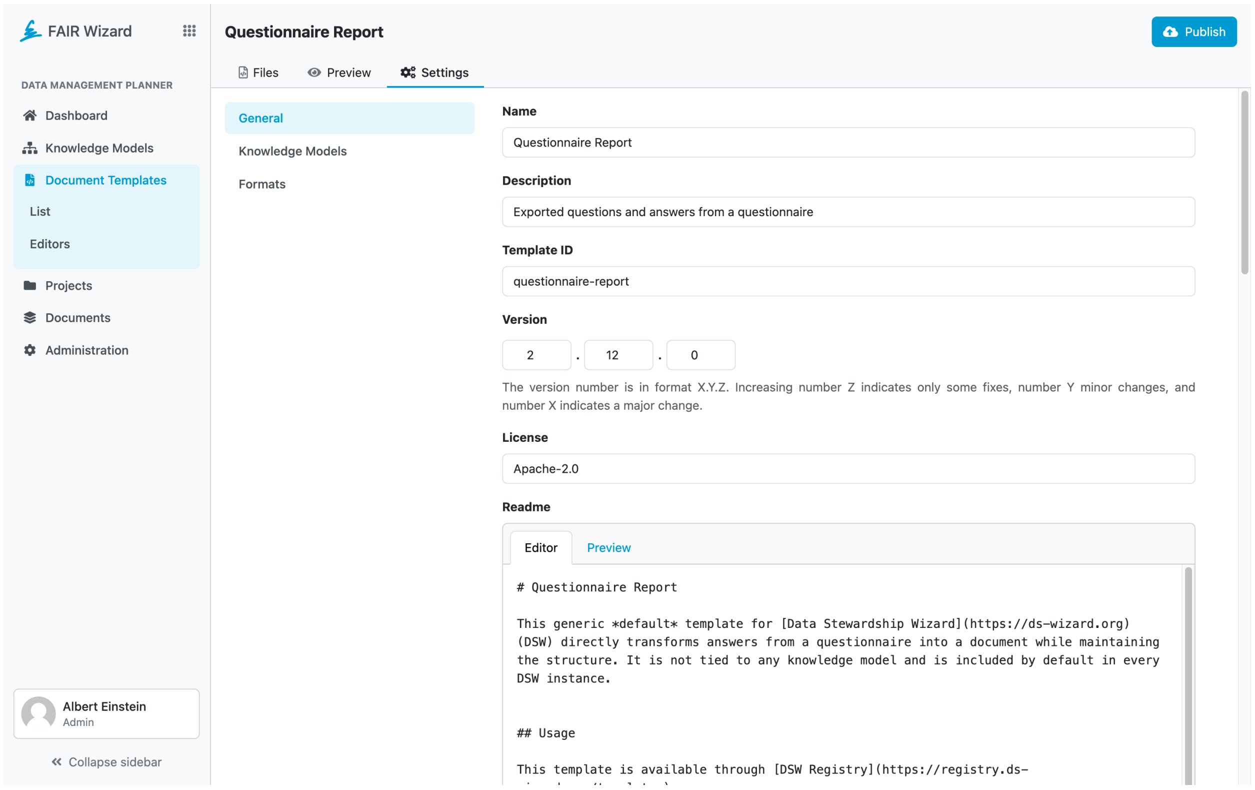
We can create a new editor either from scratch, i.e. the new document template will be empty and we will build it all ourselves, or based on an existing document template, which means that everything from the chosen document template (files as well as configuration) will be copied to ours. We just need to choose the original document template in the **Based on** field. Alternatively, we can click on *Create editor* from *Document Template Detail*.

We can only have one document template editor with the same document template ID. If we deleted the editor but want to continue working on that document template, we can create a new editor with the same document template ID. Alternatively, we would have to use a different document template ID.

Document Template Editor

A document template editor allows us to edit both configuration and all files of a document template. We can manage the configuration (such as template metadata, formats, and steps) on the *Settings* tab. The files including directories can be managed (created, edited, deleted, or uploaded) on the *Files* tab. Finally, the *Preview* tab allows us to quickly check how the document template works for a certain project and format and how the resulting document looks like.

There is also option to *Publish* the document template via *Publish* buttom in the top right corner.



Name
Questionnaire Report

Description
Exported questions and answers from a questionnaire

Template ID
questionnaire-report

Version
2 . 12 . 0

The version number is in format X.Y.Z. Increasing number Z indicates only some fixes, number Y minor changes, and number X indicates a major change.

License
Apache-2.0

Readme

```
# Questionnaire Report

This generic *default* template for [Data Stewardship Wizard](https://ds-wizard.org)
(DSW) directly transforms answers from a questionnaire into a document while maintaining
the structure. It is not tied to any knowledge model and is included by default in every
DSW instance.


## Usage

This template is available through [DSW Registry](https://registry.ds-
```

Fig. 30: Document template editor.

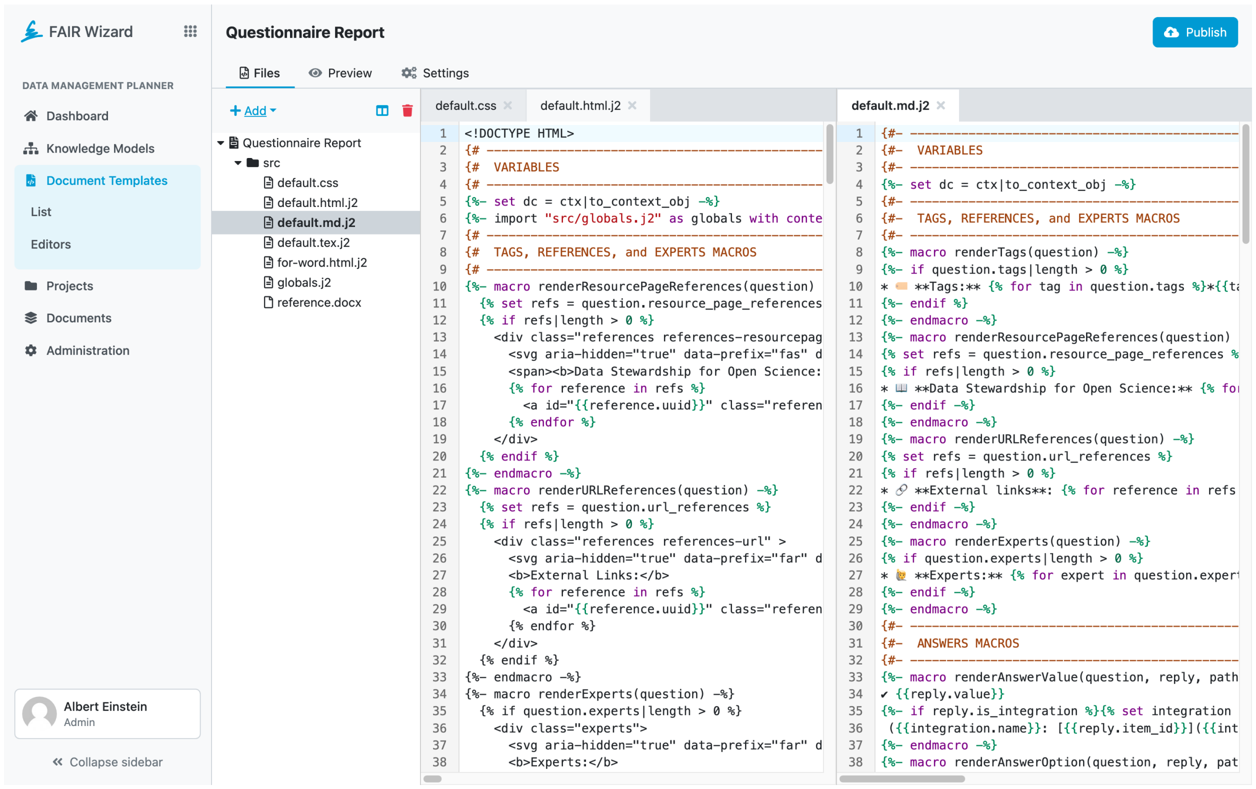
Files

The *Files* tab in *Document Template Editor* allows us to manage the files and directories (or folders) of the document template. There is a file tree on the left side whereas the main part is containing the built-in text editor. We can create a new folder or text file by clicking *Add* on the of top the file tree and then *File* or *Folder*. Once we enter a desired name, the folder or file is created in the active directory. Alternatively, we can upload a file by clicking *Add* and then *Upload*.

If we select a text file in the file tree, it is opened in the built-in text editor. We can also open multiple files (the editor supports tabs). Moreover, if we have more files opened, we can see a *split view* icon on the top of the file tree. When we click it, it will split the view and move the opened file to the other group. If there are already two groups, it switches the file between the two groups. We can close the file by clicking cross icon in its tab.

We can easily delete a file or a folder; when it is selected in the file tree, we can simply click the red trash icon above the file tree. The deletion must be confirmed in the prompt so we will not delete something by accident as it is not reversible operation.

If we make some changes in a file, the asterisk (or star) symbol will appear by the name in the tree view as well as in the tab (if opened). The changes must be then saved using *Save* button or discarded using *Discard* button in the top right corner. Those buttons will appear instead of *Publish* as it is not possible to publish a document template with unsaved changes. After saving the changes and switching to *Preview* (or refreshing it), the document will be re-generated using the newly changed document template.



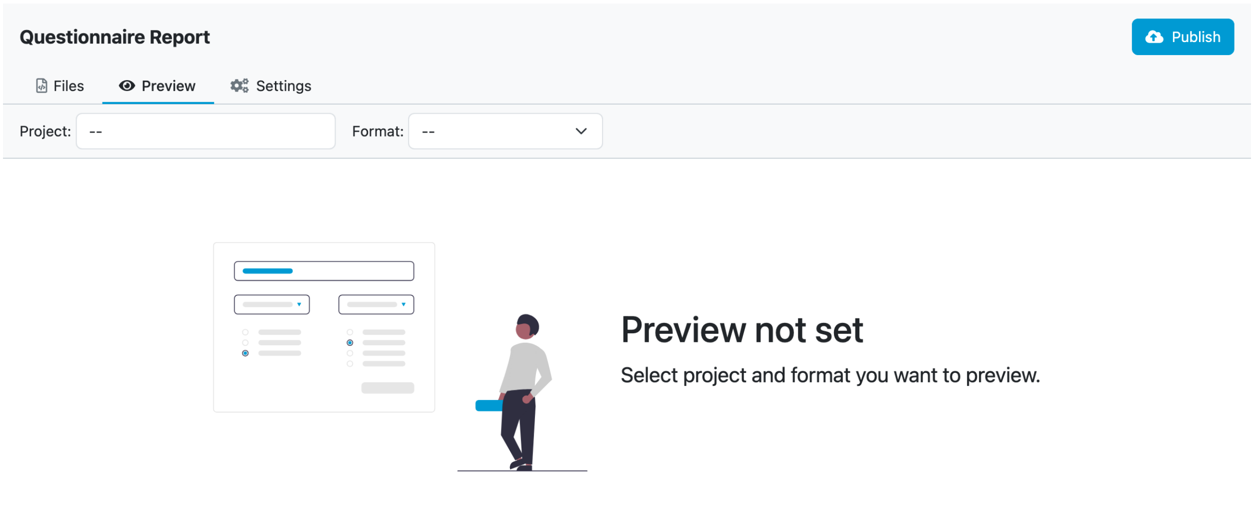
```
<!DOCTYPE HTML>
{# ----------------------------------------------
{#  VARIABLES
{# ----------------------------------------------
{%- set dc = ctx|to_context_obj -%}
{%- import "src/globals.j2" as globals with conte
{# ----------------------------------------------
{#  TAGS, REFERENCES, and EXPERTS MACROS
{# ----------------------------------------------
{%- macro renderResourcePageReferences(question)
  {% set refs = question.resource_page_references
  {% if refs|length > 0 %}
    <div class="references references-resourcepag
      <svg aria-hidden="true" data-prefix="fas" d
      <span><b>Data Stewardship for Open Science:
      {% for reference in refs %}
        <a id="{{reference.uuid}}" class="referen
      {% endfor %}
    </div>
  {% endif %}
{%- endmacro -%}
{%- macro renderURLReferences(question) -%}
  {% set refs = question.url_references %}
  {% if refs|length > 0 %}
    <div class="references references-url" >
      <svg aria-hidden="true" data-prefix="far" d
      <b>External Links:</b>
      {% for reference in refs %}
        <a id="{{reference.uuid}}" class="referen
      {% endfor %}
    </div>
  {% endif %}
{%- endmacro -%}
{%- macro renderExperts(question) -%}
  {% if question.experts|length > 0 %}
    <div class="experts">
      <svg aria-hidden="true" data-prefix="far" d
      <b>Experts:</b>
```

```
{#- ---------------------------------------------
{#-  VARIABLES
{#- ---------------------------------------------
{%- set dc = ctx|to_context_obj -%}
{#- ---------------------------------------------
{#-  TAGS, REFERENCES, and EXPERTS MACROS
{#- ---------------------------------------------
{%- macro renderTags(question) -%}
{%- if question.tags|length > 0 %}
* 🏷️ **Tags:** {% for tag in question.tags %}*{{ta
{%- endif %}
{%- endmacro -%}
{%- macro renderResourcePageReferences(question)
{% set refs = question.resource_page_references %
{% if refs|length > 0 %}
* 📖 **Data Stewardship for Open Science:** {% fo
{%- endif -%}
{%- endmacro -%}
{%- macro renderURLReferences(question) -%}
{% set refs = question.url_references %}
{% if refs|length > 0 %}
* 🔗 **External links**: {% for reference in refs
{%- endif -%}
{%- endmacro -%}
{%- macro renderExperts(question) -%}
{% if question.experts|length > 0 %}
* 🙋 **Experts:** {% for expert in question.expert
{%- endif -%}
{%- endmacro -%}
{#- ---------------------------------------------
{#-  ANSWERS MACROS
{#- ---------------------------------------------
{%- macro renderAnswerValue(question, reply, path
✔ {{reply.value}}
{%- if reply.is_integration %}{% set integration
 ({{integration.name}}: [{{reply.item_id}}]({{int
{%- endmacro -%}
{%- macro renderAnswerOption(question, reply, pat
```

Fig. 31: Files editor with tabs and split view.

Preview

The *Preview* tab of *Document Template Editor* allows us to quickly check how our document template works and how the resulting document looks like for a selected project and document format. We can select any **Project** that we are allowed to view. Then, we select the desired **Format** (one of those specified on the *Settings* tab of the editor). Any change in the template configuration and its files will trigger re-generation of the preview (we can simply click the *Preview* tab again).



Preview not set

Select project and format you want to preview.

Fig. 32: Preview in a document template editor (without selected **Project** and **Format**).

Note: A good practice is to open the document template editor in a different browser tab than editor opened on the *Files* or *Settings* tab. Moreover, we can also open the project that we use for preview.

Settings

The *Settings* tab of *Document Template Editor* allows us to adjust the configuration and metadata of the document template. It is split to the following three parts:

General

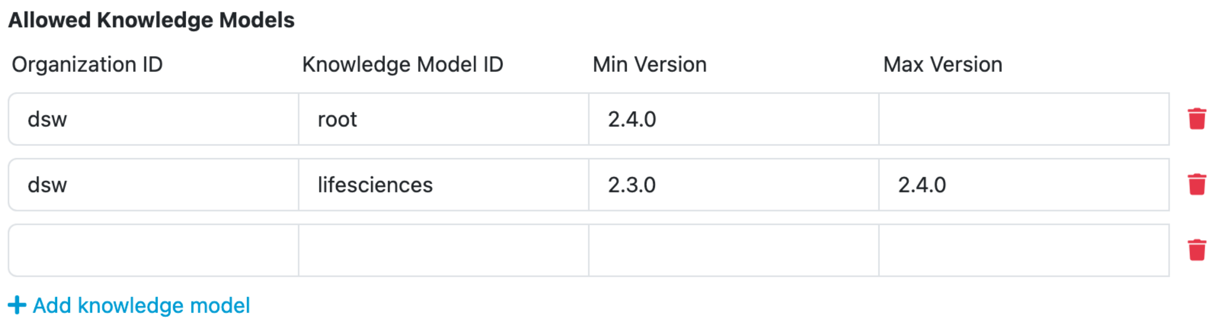
This part allows us to change the metadata about the document template:

- **Name** should be short name of the template.
- **Description** should be short and descriptive (users will see it while selecting a document template).
- **Template ID** is the document template ID (as explained for *Create Document Template Editor*).
- **Version** of the document template.
- **License** should contain a name of used license (e.g. *Apache-2.0* or *unlicensed*).
- **Readme** can contain a longer description, acknowledgements, notes how to use the template, links to resources, and a changelog.

Knowledge Models

We can specify here what knowledge models are compatible with our document template. This is useful to capture if our template is usable only with a certain knowledge model(s) which will guarantee us some content (specific chapters, questions, answers, etc.). For each entry, we are prompted to specify the **Organization ID**, **Knowledge Model ID**, **Min Version**, and **Max Version**. Any of these value can be left empty which means *any value*.

For example, if we set **Organization ID** to *fw*, **Knowledge Model ID** to *root*, **Min Version** to *2.4.0* and leave **Max Version** empty, it will mean knowledge model *fw:root:2.4.0* and any higher version. So that example would work for *fw:root:2.4.4* or *fw:root:2.5.0* but not for *fw:root:2.3.0* nor *acme:base-km:2.6.0*



Organization ID	Knowledge Model ID	Min Version	Max Version
dsw	root	2.4.0	
dsw	lifesciences	2.3.0	2.4.0

Fig. 33: Allowed knowledge models specification for a document template.

Formats

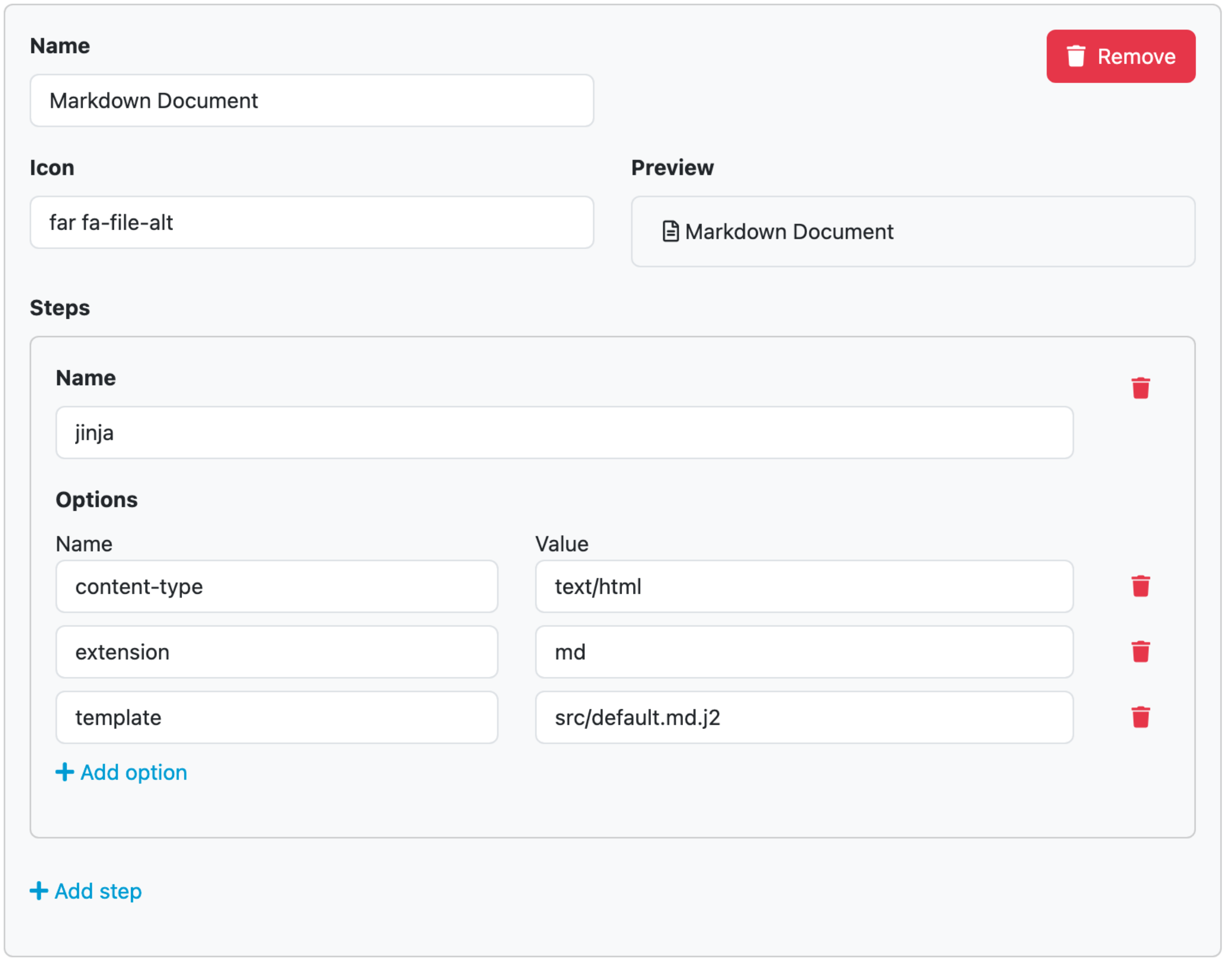
Each document template can support multiple file formats and users will be able to select which one they want to use to generate (or preview a document). We can add a new format by clicking *Add* button. Then, each format must have a **Name** and **Icon** (by using Font Awesome).

Each format has some steps which capture how a file for that format is produced. There are different steps defined such as `json`, `jinja2`, `pandoc`, or `wkhtmltopdf` which is used as its **Name**. Then, the step may have certain configuration **Options**. For example, `jinja2` must have `content-type`, `extension`, and `template` specified. All the possible steps and their options are further described in the *Document Template Development*.

Publish

Once we are ready with our document template in the editor, we can publish a new version by using the *Publish* button located in the top right corner of *Document Template Editor*. If we click the button, we can check the metadata details and confirm the publishing. We cannot change anything here, so if we want to make some changes, we have to press *Cancel* and edit the details on the *Settings* tab of the *Document Template Editor*.

If we confirm the publishing of the document template by clicking *Publish* in the modal window, the document template becomes available to all users and is accessible in *Document Template List*. Moreover, the document template editor disappears (as the state of the document template changed). If we want to directly continue in developing a new version, we have to *Create Document Template Editor*.



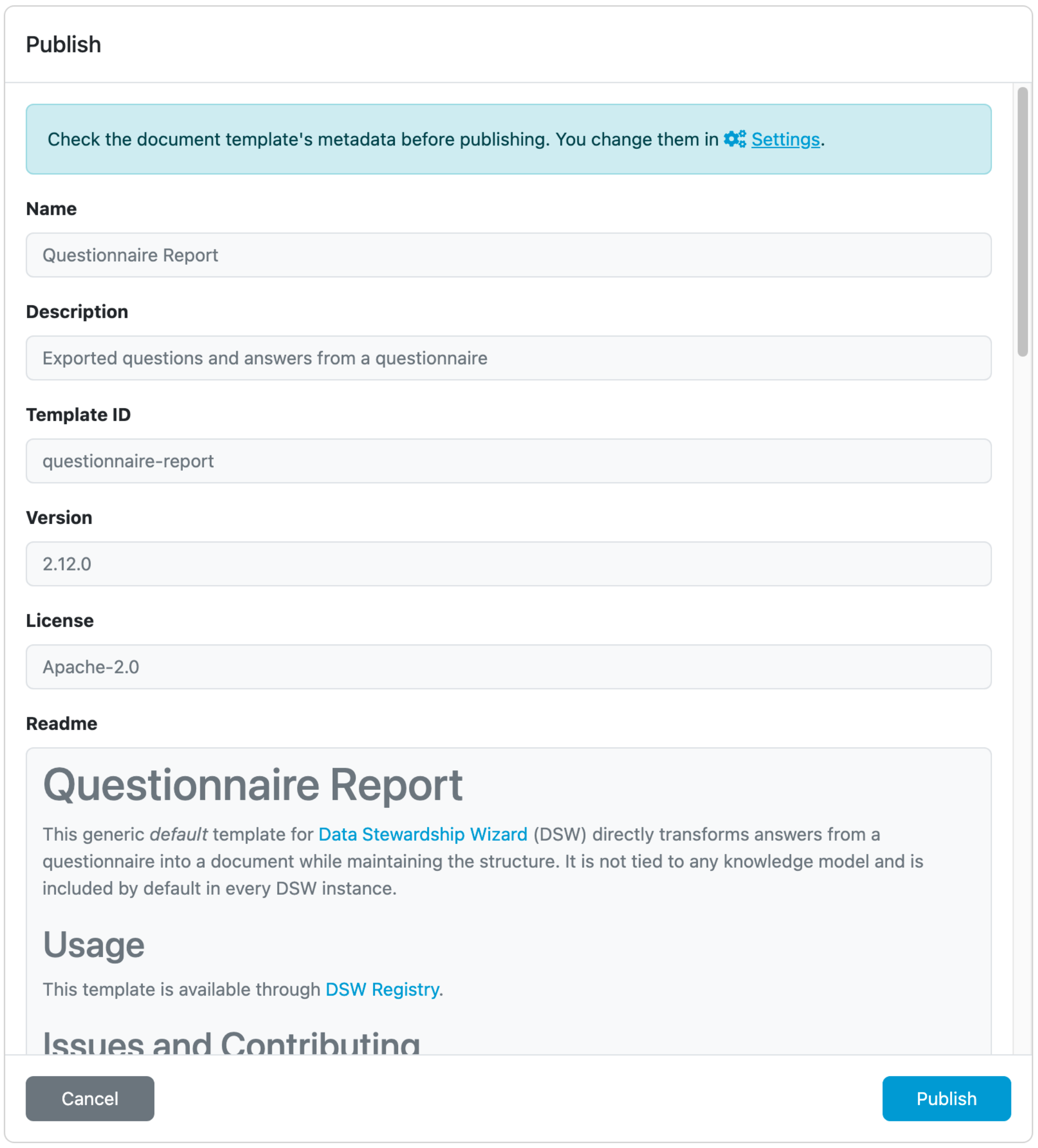
Name	Markdown Document
Icon	far fa-file-alt

Step Name: jinja

Options

Name	Value
content-type	text/html
extension	md
template	src/default.md.j2

Fig. 34: Markdown format specification with `jinja` step.



Publish

Check the document template's metadata before publishing. You change them in Settings.

Name

Questionnaire Report

Description

Exported questions and answers from a questionnaire

Template ID

questionnaire-report

Version

2.12.0

License

Apache-2.0

Readme

Questionnaire Report

This generic *default* template for Data Stewardship Wizard (DSW) directly transforms answers from a questionnaire into a document while maintaining the structure. It is not tied to any knowledge model and is included by default in every DSW instance.

Usage

This template is available through DSW Registry.

Issues and Contributing

Fig. 35: Confirmation of document template publishing.

1.2.3 Projects

This section is about *projects* – how to create them, how to work and collaborate on them, how to generate documents, how to upgrade projects, what are the project templates and more.

Projects are mainly used by the researchers. We have the introduction video about how to create project, fill the questionnaire and get the documents.

https://youtu.be/usCrhfG_OXE

Project List

In the project list, we can see a list of all projects we have access to. Those are the projects where we were assigned to with any role or that are visible by all other logged-in users in the *project sharing settings*. The projects are filtered to those we are explicitely assigned to by default.

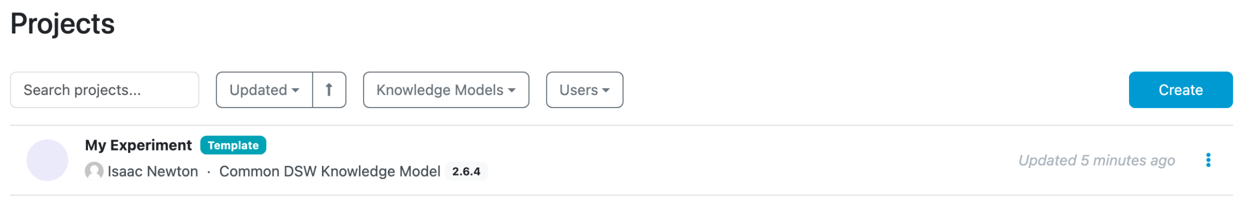


Fig. 36: Project list with filters.

We can search for specific projects using the search field or filter them using additonal filters:

- **Project Template** - if we want to see only *project templates* or regular projects
- **Project Tags** - show only the projects that has specific tags assigned
- **Knowledge Models** - show only the projects created from a specific knowledge model
- **Users** - filter only the projects with specifc users assigned to them

Note: Note that some of the filters can be disabled based on the FAIR Wizard settings or user role.

We can *create a new project* by clicking the *Create* button.

By clicking the triple dots on each of the item in the list we can access some actions:

- **Open project** - will simply open the project
- **Create project from this template** - will create a new project from the selected project template (this is only available if the project is also a project template)
- **Clone** - will create the exact copy of the project
- **Create migration** - will start a *project migration*
- **Delete** - will delete the project (this action cannot be undone)

If there is an ongoing *project migration*, there are different actions:

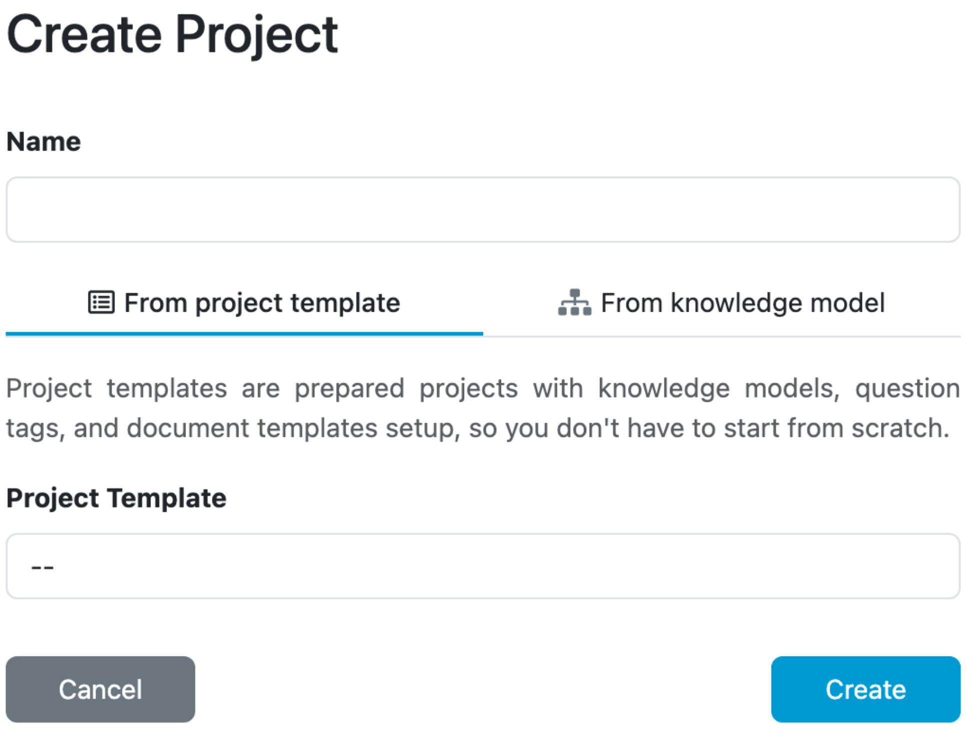
- **Continue migration** - to come back to the ongoing project migration
- **Cancel migration** - to cancel the ongoing project migration

Create Project

We can create a new project by navigating to *Projects* in the main menu and then clicking on *Create* button on the project list.

Note: If we have **Researcher** role we also have a **Create Project** widget on our dashboard right after logging in. We can click on *Create* button there, too.

Based on our configuration, we can create the new project from a project template, custom, or both.



Create Project

Name

From project template | From knowledge model

Project templates are prepared projects with knowledge models, question tags, and document templates setup, so you don't have to start from scratch.

Project Template

--

Fig. 37: Different options how to create a project.

From Project Template

There are many options how to create and configure a project, such as what *knowledge model* or *document template* to use. *Project templates* are prepared projects by data stewards with possibly pre-selected knowledge models and document templates, they can have some pre-filled answers, comments and TODOs as well.

So creating such templates can be used by our researchers to have a smoother start of project. We just need to give our project a **name** and choose the **project template** from offered options.

Custom

If there are no project templates available or we don't want to use them, we can choose to create a custom project. We need to give our project a **name** again, but this time we choose a *knowledge model* from offered options.

If the knowledge model has *question tags*, we can either choose to create questionnaire with all available questions or filter them by the question tags.

This will create an empty project with only the selected knowledge model and we need to configure everything (such as a document template) ourselves.

Project Detail

Project detail is where we work on our data management plan. In this section, we explore different features from filling in the answers to collaborating with other people or generating documents.

Questionnaire

Questionnaire is the first tab of the project detail. This is the most imporant part where we fill in all the details about our project.

Current Phase

If the *knowledge model* we use for the *project* has *phases* defined, we can see a phase selection in the questionnaire detail. Different questions become desirable based on the selected phase. For example, some should be answered before submitting the proposal, while others can be filled later.

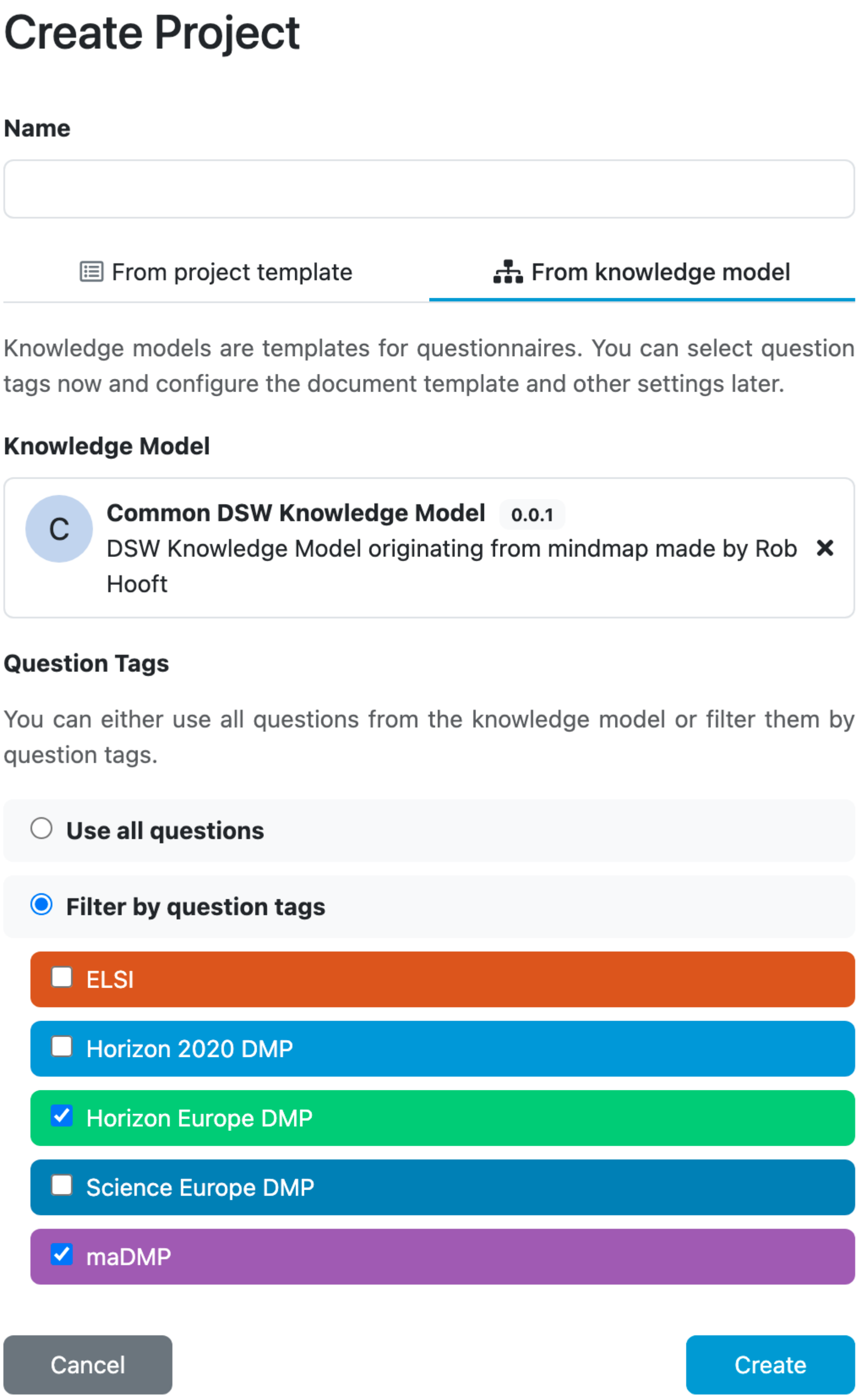
By clicking the phase selection we open the modal window where we can choose the current phase.

We can see the desirability of questions based on the phase we are currently in. We can also see the number of questions that still need to be answered in this phase for each chapter in the chapter list.

There are three desirability states the question can be in:

- **red, with a pen icon** - this question must be answered in the current phase
- **light grey, with an hourglass icon** - this question will have to be answered in some later phase
- **green, with a checkmark icon** - this question has already been answered

Note: If there is no phase defined on the knowledge model, the current phase selection is not visible in the questionnaire detail.



Create Project

Name

From project template | From knowledge model

Knowledge models are templates for questionnaires. You can select question tags now and configure the document template and other settings later.

Knowledge Model

Common DSW Knowledge Model 0.0.1
DSW Knowledge Model originating from mindmap made by Rob Hooft

Question Tags

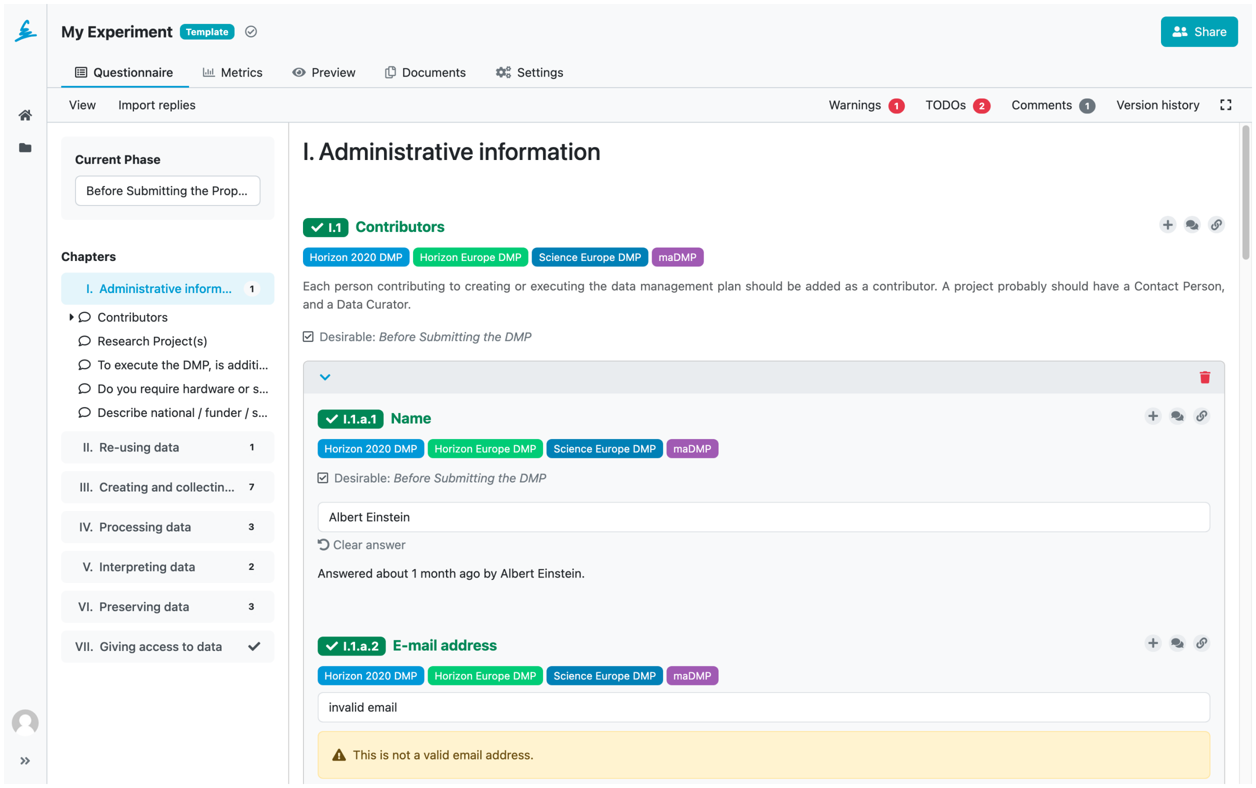
You can either use all questions from the knowledge model or filter them by question tags.

( ) Use all questions

(●) Filter by question tags

- [ ] ELSI
- [ ] Horizon 2020 DMP
- [x] Horizon Europe DMP
- [ ] Science Europe DMP
- [x] maDMP

Fig. 38: Creating custom project with question tag selection.



I. Administrative information

I.1 Contributors

Horizon 2020 DMP | Horizon Europe DMP | Science Europe DMP | maDMP

Each person contributing to creating or executing the data management plan should be added as a contributor. A project probably should have a Contact Person, and a Data Curator.

Desirable: *Before Submitting the DMP*

I.1.a.1 Name

Horizon 2020 DMP | Horizon Europe DMP | Science Europe DMP | maDMP

Desirable: *Before Submitting the DMP*

Albert Einstein

Clear answer

Answered about 1 month ago by Albert Einstein.

I.1.a.2 E-mail address

Horizon 2020 DMP | Horizon Europe DMP | Science Europe DMP | maDMP

invalid email

This is not a valid email address.

Fig. 39: Project detail opened on the questionnaire page.

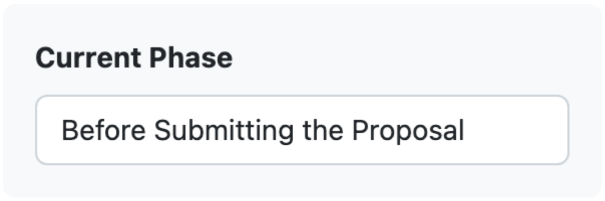
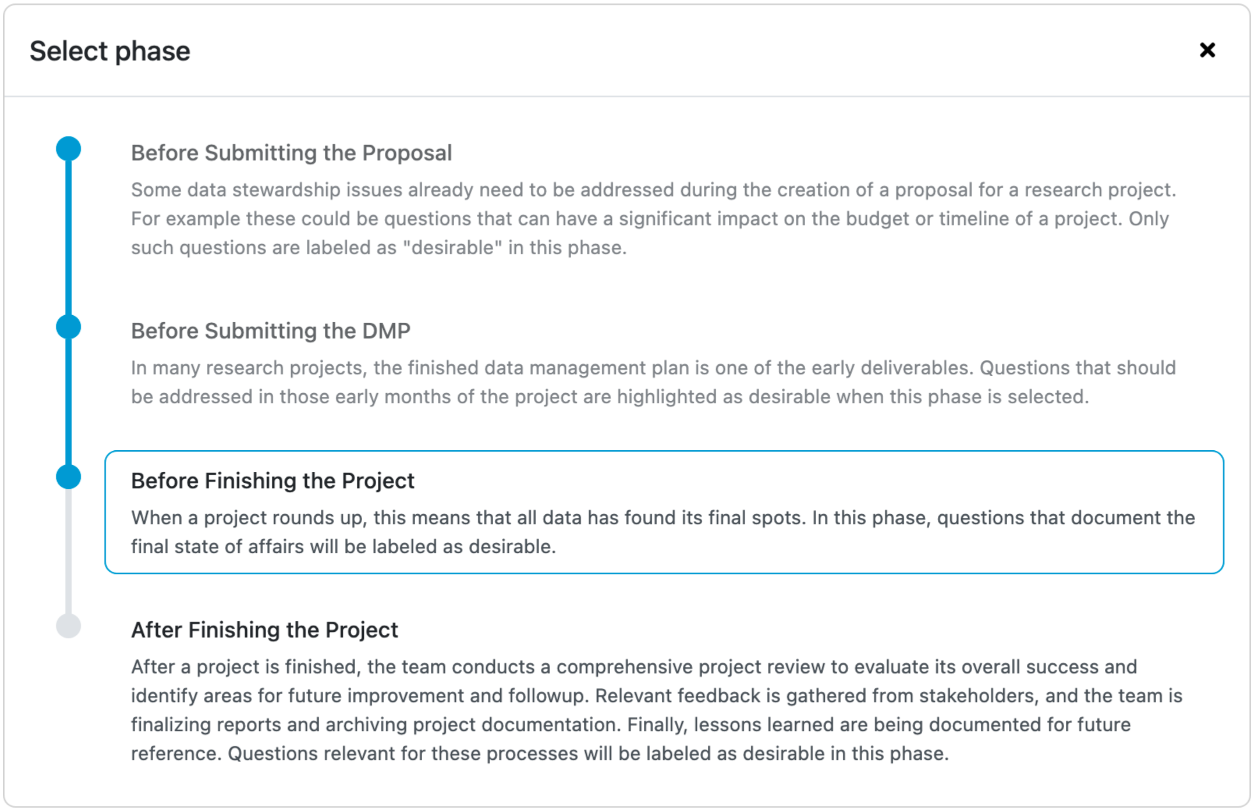


Fig. 40: Phase selection widget.


Select phase

Before Submitting the Proposal

Some data stewardship issues already need to be addressed during the creation of a proposal for a research project. For example these could be questions that can have a significant impact on the budget or timeline of a project. Only such questions are labeled as "desirable" in this phase.

Before Submitting the DMP

In many research projects, the finished data management plan is one of the early deliverables. Questions that should be addressed in those early months of the project are highlighted as desirable when this phase is selected.

Before Finishing the Project

When a project rounds up, this means that all data has found its final spots. In this phase, questions that document the final state of affairs will be labeled as desirable.

After Finishing the Project

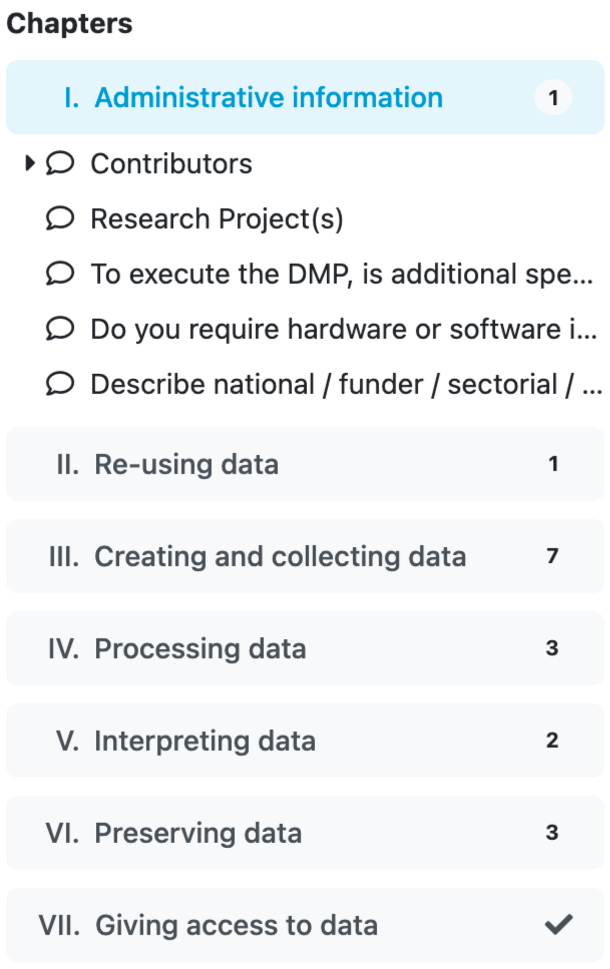
After a project is finished, the team conducts a comprehensive project review to evaluate its overall success and identify areas for future improvement and followup. Relevant feedback is gathered from stakeholders, and the team is finalizing reports and archiving project documentation. Finally, lessons learned are being documented for future reference. Questions relevant for these processes will be labeled as desirable in this phase.

Fig. 41: Phase selection modal window.

Chapters

Below the current phase selection, we have a list of *chapters*. We can see the number of questions that are yet to be answered in current phase (or overall if there are no phases defined in the knowledge model). We can use this list to navigate freely between chapters.

For the opened chapter, we can see a navigation tree for the chapter structure, showing the questions, follow-up questions, items, etc. We can use this tree to quickly navigate to a specific question in the chapter.


Chapters

I. Administrative information 1

Contributors

Research Project(s)

To execute the DMP, is additional spe...

Do you require hardware or software i...

Describe national / funder / sectorial / ...

II. Re-using data 1

III. Creating and collecting data 7

IV. Processing data 3

V. Interpreting data 2

VI. Preserving data 3

VII. Giving access to data ✓

Fig. 42: Chapter list showing the also the questions for the opened chapter.

Questionnaire Area

The questionnaire area fills the most space in the questionnaire screen. It displays the questions and answers from the opened chapter.

Each question has an identifier which indicates the chapter it belongs to, as well as its order and nesting within the chapter. For example, **I.1.a.5**, where the Roman numeral represents the number of the chapter, and the remaining numbers indicate the order and nesting of the question. Then there is also the question name.

Some additional information can also be part of the question:

- **Question tags** - can indicate some additional groupping of questions, for example what DMP templates is this question used for
- **Description** - additional information explaining the question
- **Desirability** - what phase this question become desirable in
- **List of references** - links to additional external resources related to the question
- **List of experts** - whom to contact when help is needed with answering the question

Based on our role in the project and specific settings, there are some additional actions besides answering the question:

- *Add TODO*
- *Add comment*
- Provide feedback for the question

We can get more information on how various collaboration tools work and can be used in *Sharing*.

The most important part is, however, answering the question. The way of how to answer the question differs based on the question type.

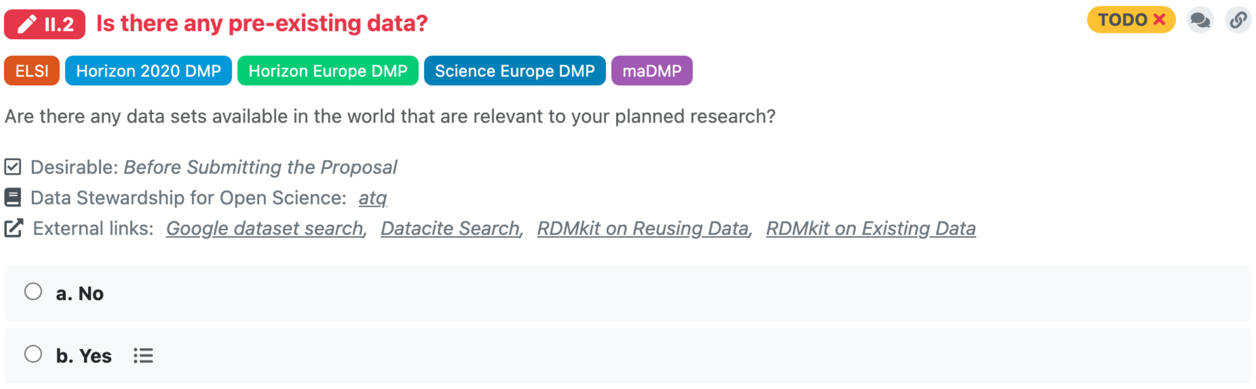
The following video tutorial explains questions and different question types in more detail.

https://youtu.be/jdKooHVZEhI

Options Question

Options question has a list of pre-set answers and we can choose one from those. There can be some follow-up questions (indicated by the icon by the answer). These questions are displayed only if we select that answer.

If there are some merics set for the answers, we can see labels with the metrics by the answer as well. The color of the label indicates how good or bad the answer is (red means bad, green good, yellow something in between).


II.2 Is there any pre-existing data?

Are there any data sets available in the world that are relevant to your planned research?

Desirable: *Before Submitting the Proposal*
Data Stewardship for Open Science: atq
External links: Google dataset search, Datacite Search, RDMkit on Reusing Data, RDMkit on Existing Data

a. No
b. Yes

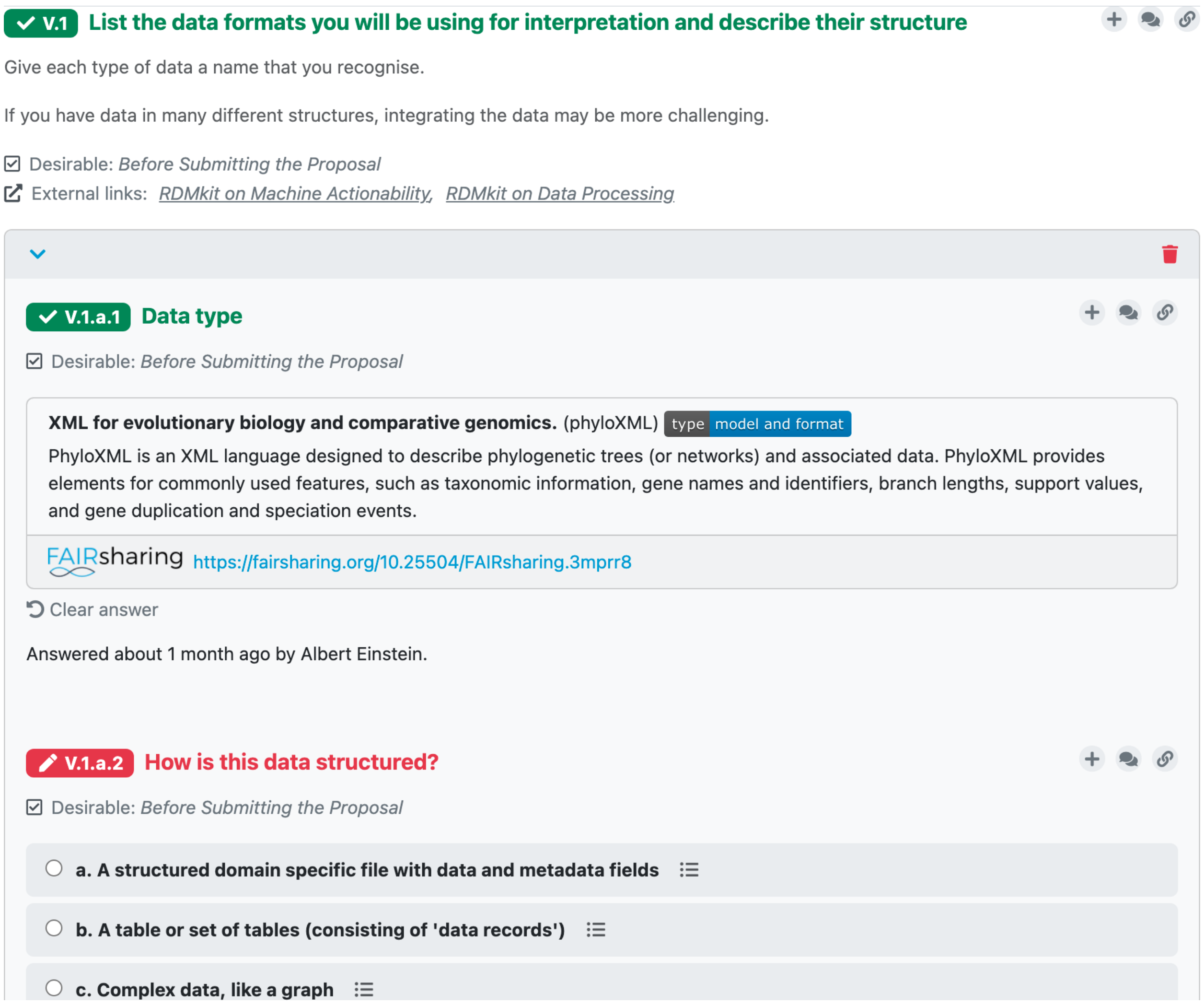
Fig. 43: Options question with a closed set of answers.

List Question

List question doesn't have a simple answer but a list of items. Each of the items has the same set of subquestions. For example, a list question asking about the project contributors where each item represent one contributor with questions about their name, role, etc.

We can simply click on *Add* button under the question to add a new item. Then, we can answer the questions for the item. If the item has a lot of questions, we can use the arrow icon in the item's top left corner to **fold/unfold** the item.

There is a trash bin icon in the item's top right corner that we can use to **delete** the item. If there are more than one item, there are also arrow icons that we can use to **change the order** of the items.


✓ V.1 List the data formats you will be using for interpretation and describe their structure

Give each type of data a name that you recognise.

If you have data in many different structures, integrating the data may be more challenging.

Desirable: *Before Submitting the Proposal*

External links: *RDMkit on Machine Actionability,* *RDMkit on Data Processing*

✓ V.1.a.1 Data type

Desirable: *Before Submitting the Proposal*

XML for evolutionary biology and comparative genomics. (phyloXML) type model and format

PhyloXML is an XML language designed to describe phylogenetic trees (or networks) and associated data. PhyloXML provides elements for commonly used features, such as taxonomic information, gene names and identifiers, branch lengths, support values, and gene duplication and speciation events.

FAIRsharing https://fairsharing.org/10.25504/FAIRsharing.3mprr8

Clear answer

Answered about 1 month ago by Albert Einstein.

V.1.a.2 How is this data structured?

Desirable: *Before Submitting the Proposal*

a. A structured domain specific file with data and metadata fields

b. A table or set of tables (consisting of 'data records')

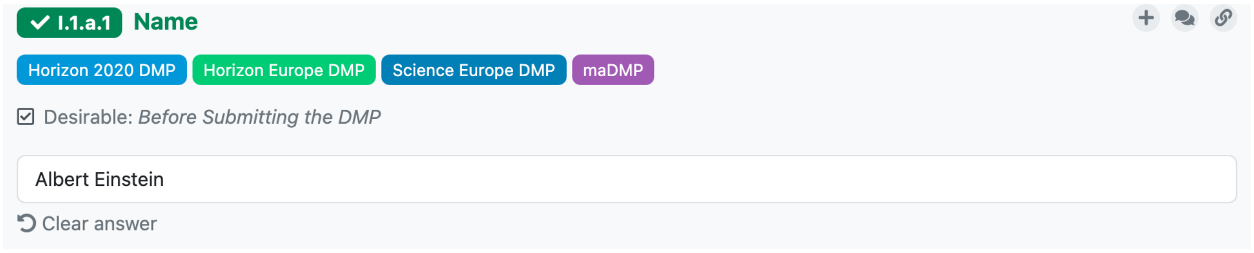
c. Complex data, like a graph

Fig. 44: List question with a single item.

Value Question

Value question contains an input field for our answer. This can be a simple text field (such as asking for a project contributor's name), or some additional widget, for example a date picker.

Some of the value types contains a validation (e.g., email or URL). We can still type in an invalid answer, but it will display a warning and also show it in the list of *warnings*.



Desirable: *Before Submitting the DMP*

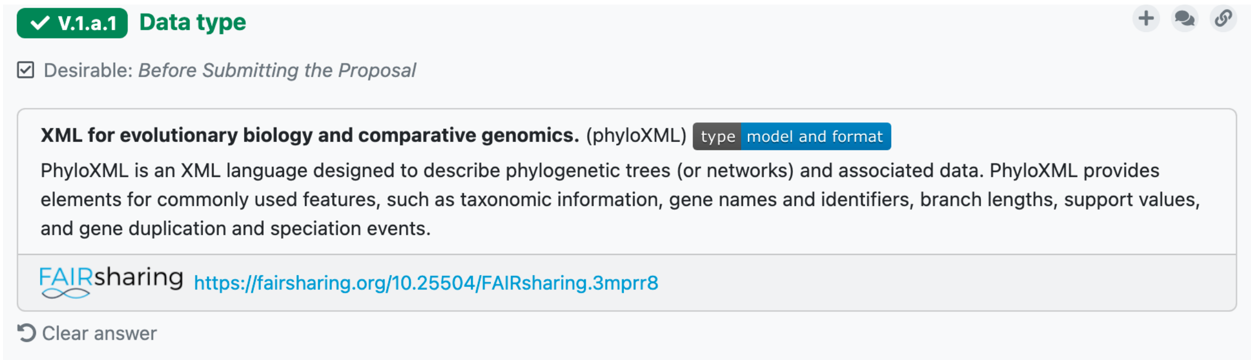
Albert Einstein

Fig. 45: Value question with a simple text input.

Integration Question

Integration question is connected to an external resource where it searches for the answers. The input field works as a search field, so when we start typing something, it will search the external resource and offers us a list of possible answers.

When we pick an answer from the list, we not only have the answer but also **a link to the selected item in the external service**. If the answer we searched for is not there, we can simply keep what we have written in the input field. We just won't have the link with this answer.



Desirable: *Before Submitting the Proposal*

XML for evolutionary biology and comparative genomics. (phyloXML) type model and format

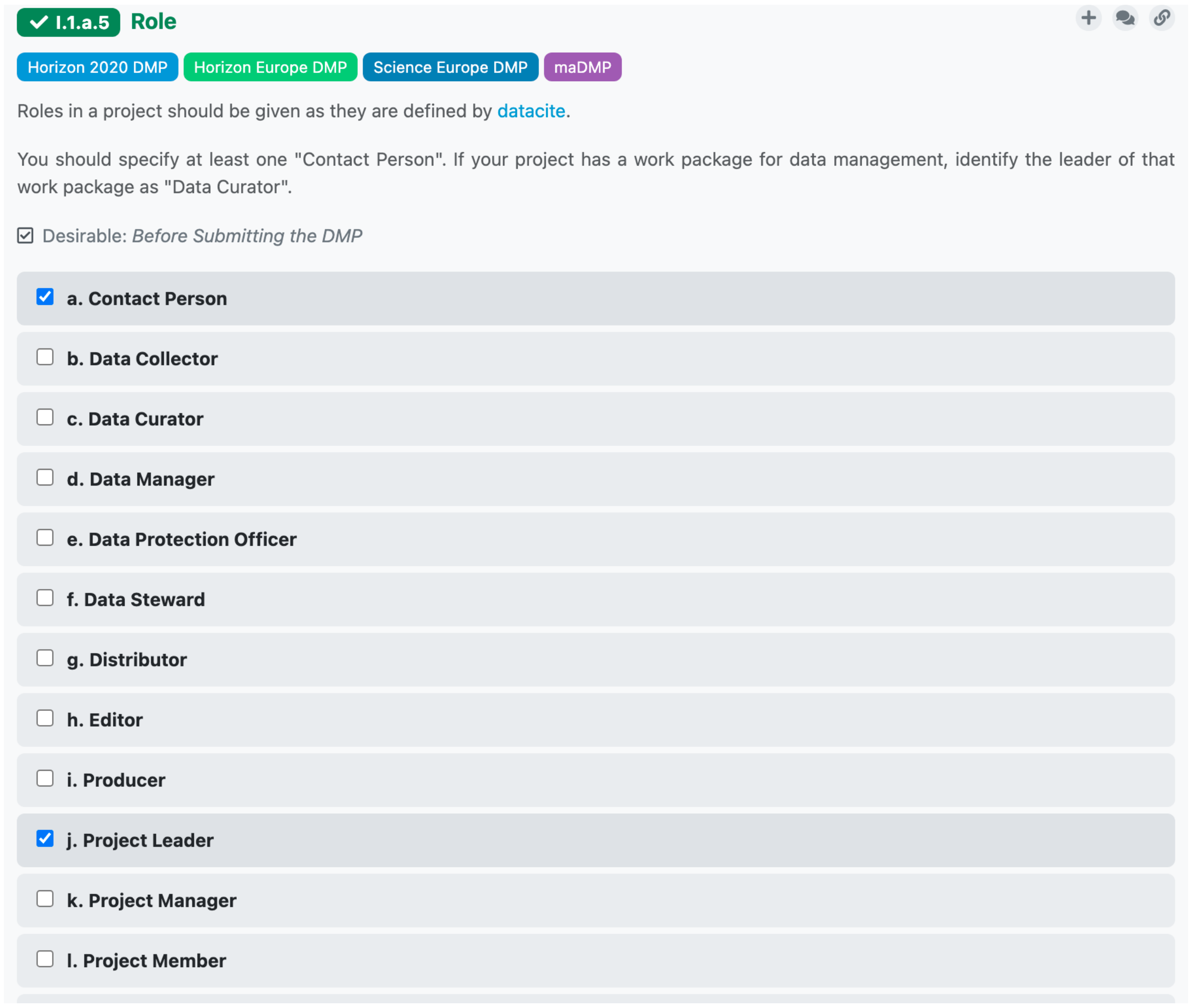
PhyloXML is an XML language designed to describe phylogenetic trees (or networks) and associated data. PhyloXML provides elements for commonly used features, such as taxonomic information, gene names and identifiers, branch lengths, support values, and gene duplication and speciation events.

https://fairsharing.org/10.25504/FAIRsharing.3mprr8

Fig. 46: Integration question with a response from FAIRsharing containing also a link.

Multi-Choice Question

Multi-choice question is similar to the options question, however we can choose more there one answer and there are no follow-up questions.



Role

Roles in a project should be given as they are defined by datacite.

You should specify at least one "Contact Person". If your project has a work package for data management, identify the leader of that work package as "Data Curator".

Desirable: *Before Submitting the DMP*

- [✓ a. Contact Person]
- b. Data Collector
- c. Data Curator
- d. Data Manager
- e. Data Protection Officer
- f. Data Steward
- g. Distributor
- h. Editor
- i. Producer
- [✓ j. Project Leader]
- k. Project Manager
- l. Project Member

Fig. 47: Multi-choice question with many choices.

View settings

In the questionnaire tab, there is a menu bar with various options. The first one is *View*, where we can show or hide some question details:

- *Answered by* - show/hide who and when answered questions
- *Phases* - show/hide what phase the questions are desirable in
- *Question tags* - show/hide the question tags
- *Non-desirable questions* - show/hide questions that are not desired to be filled in current phase
- *Metric values* - show/hide value of metrics for accessibility purpose

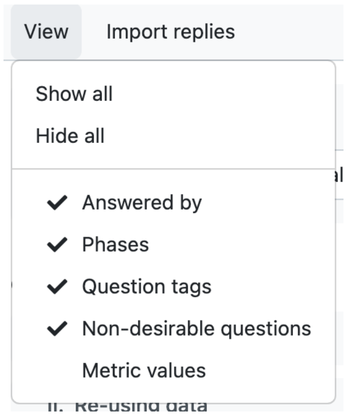


Fig. 48: Different view options accessible form the questionnaire toolbar.

Import answers

Questionnaire answers can be imported for various sources using *project importers*.

If there are some project importers available for the project, there is the *Import answers* button in the questionnaire menu bar. We can choose one of the available importers there and then follow the instructions in the importer window.

Warnings

Some value questions (such as email or URL) validates the answer written there. If it is an invalid value, we will see *Warnings* tab in the questionnaire menu bar with a badge showing the number of warnings. If we click on it, we can see a list of all questions that has a warning and we can click on it to navigate quickly to that question.

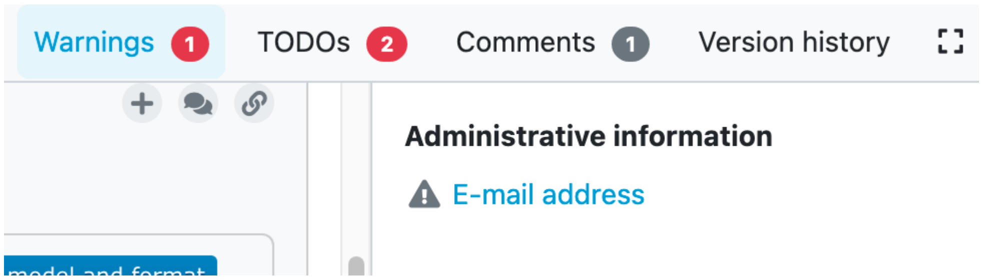
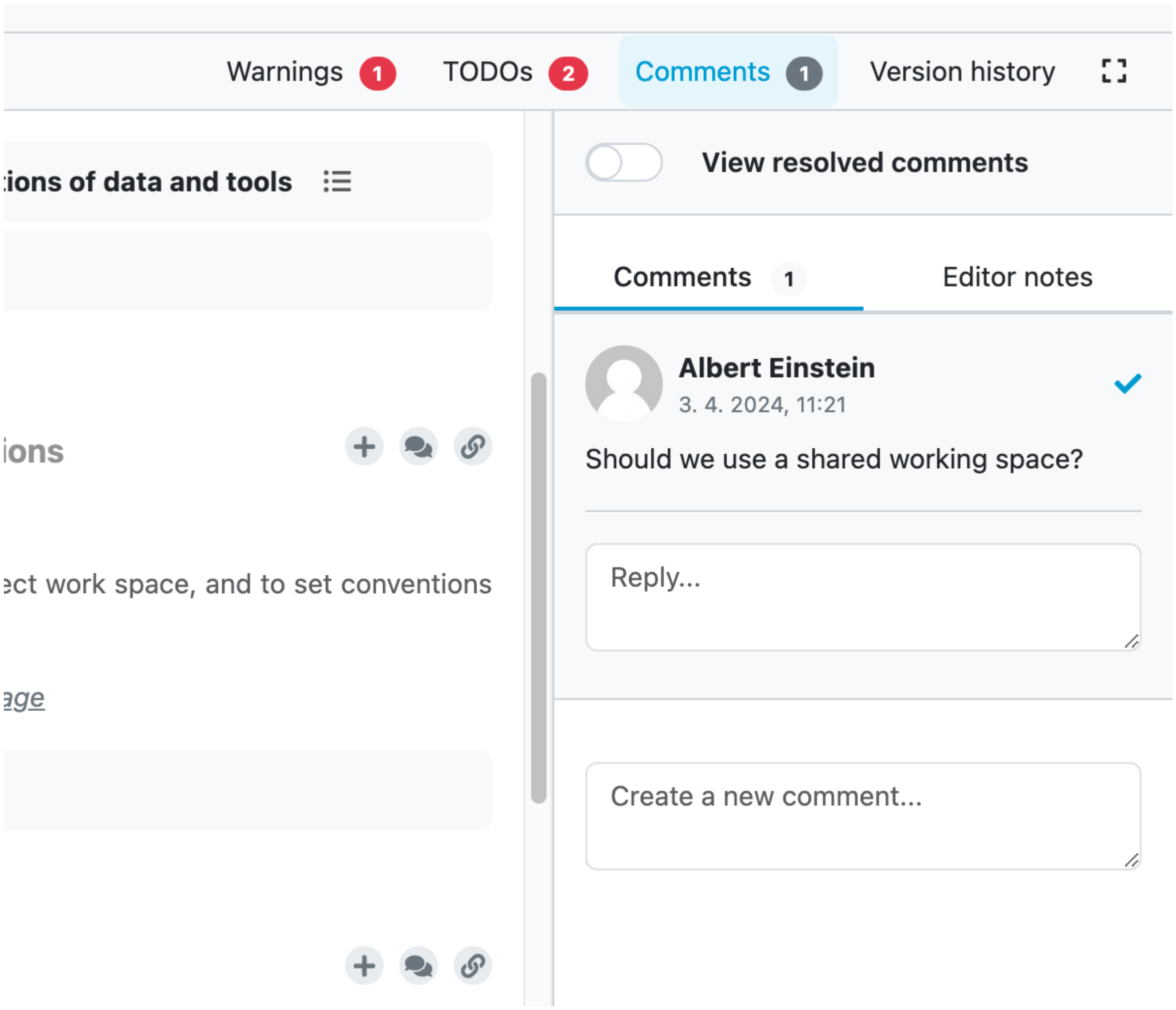


Fig. 49: Warnings refering to questions with invalid values.

Comments

We sometimes want to leave comments to discuss things with our team or just reminders for ourselves. We can write comments to each question in the questionnaire.



Albert Einstein
3. 4. 2024, 11:21
Should we use a shared working space?

Fig. 50: Example of a comment.

Add Comment

To add a comment, we need to click on the comments icon by the question which opens a side panel with all the comments related to that question. Then, we can write the comment into the text box and submit.

View Comments

When there are any comments for a question, the comments icon is changed. It has a yellow color and shows the number of unresolved comments for that question.

In the questionnaire menu bar, there is also a *Comments* tab, showing a badge with the number of comments everywhere in the questionnaire. If we open the tab, we can see a list of questions for each chapter where there are some comments. Clicking on the question there will bring us to that question and open the comments side panel.

Comment Threads

Comments are organized into comment threads for better clarity. We can either start a new thread or reply in an existing thread if our comment is on the same topic.

When the thread is resolved, we can click on the ✓ icon in order to resolve it. Resolved threads can be later viewed by selecting *View resolved comments*. They can also be reopened if needed.

Editor Notes

Besides comments there are also editor notes which work the same way as comments but they are visible only to project editors and owners. We can use editor notes to internal communication with our team while working on the DMP and then comments to gather the feedback from supervisor or reviewer.

TODOs

When we are filling in the questionnaire, we can stumble upon a question that we don't know how to answer yet, but we don't want to forget to come back to that question. We can click on + *Add TODO* to add a TODO to the question.

We can then open the *TODOs* tab from the questionnaire menu to see the list of all questions with assigned TODO in the questionnaire. By clicking on a question there, we can quickly jump back to that question and fill it.

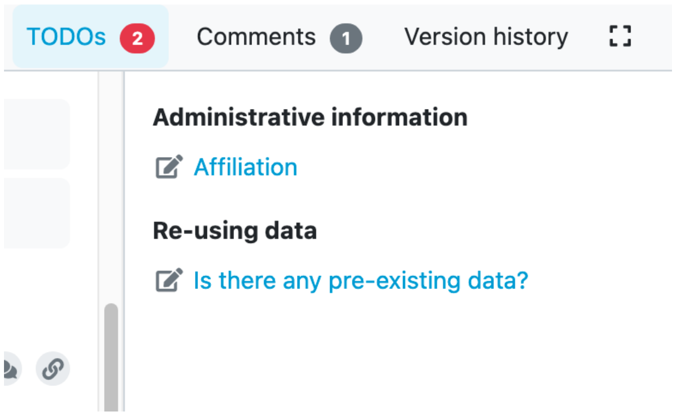


Fig. 51: List of TODOs.

Version History

When we open the *Version history* tab from the questionnaire menu bar we can see the list of all the changes that happened in that questionnaire. We can see who and when made what changes groupped by months and days.

Name a Version

At any point (also retroactively) we can name a version. Click on the triple dots on any event we want to name and choose *Name this version*. Then we just fill in name and description of that version. If the event already has a named version, we can choose *Rename this version* instead.

When we have some named versions, we can choose *Named versions only*. Then, we don't see every single change but only the important versions we gave a name to.

View Questionnaire in a Version

Thanks to the version history, we can see how the questionnaire was filled at any point in the past. We simply find the event in the version history and choose *View questionnaire* from the event menu.

Create Document from an Older Version

Sometimes, we might want to create a document from an older version. For example, we created only a PDF document, but later we find out that we also needed a Word document. To do that, we simply find that version in the version history and select *Create document*. Then, we just fill in the details in the form and create the document.

Revert to an Older Version

We can also revert a questionnaire to an older version. We can simply find the desired version in the version history and choose *Revert to this version* from the event menu.

Warning: Reverting to an older version cannot be undone. It is therefore recommended to create a copy of the project before reverting.

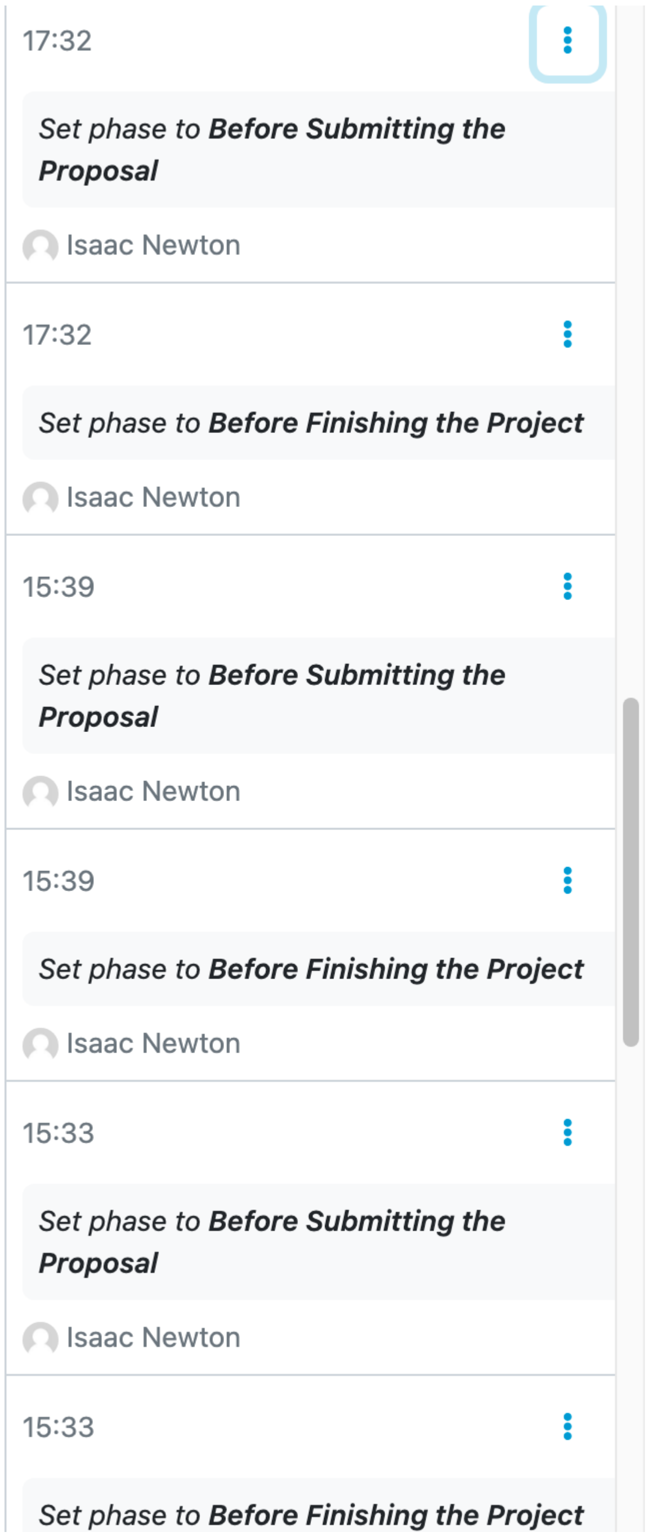
Metrics

In the *Metrics* tab in the project detail we can see a **Summary Report** for the whole questionnaire and then the same details for each individual chapter.

The report shows how many answered questions out of how many are there for the current phase (if there are phases in the knowledge model) and overal.

If there are any metrics in the knowledge model, the report also shows the score for each metric. The score is calculated as a weighted average of all the answers affecting that metric and is always between 0 and 1. If there are at least 3 metrics present, a spider chart is also displayed.

There is also a metrics description at the bottom of the page to better understand what exactly each metric means.



17:32

*Set phase to **Before Submitting the Proposal***

Isaac Newton

17:32

*Set phase to **Before Finishing the Project***

Isaac Newton

15:39

*Set phase to **Before Submitting the Proposal***

Isaac Newton

15:39

*Set phase to **Before Finishing the Project***

Isaac Newton

15:33

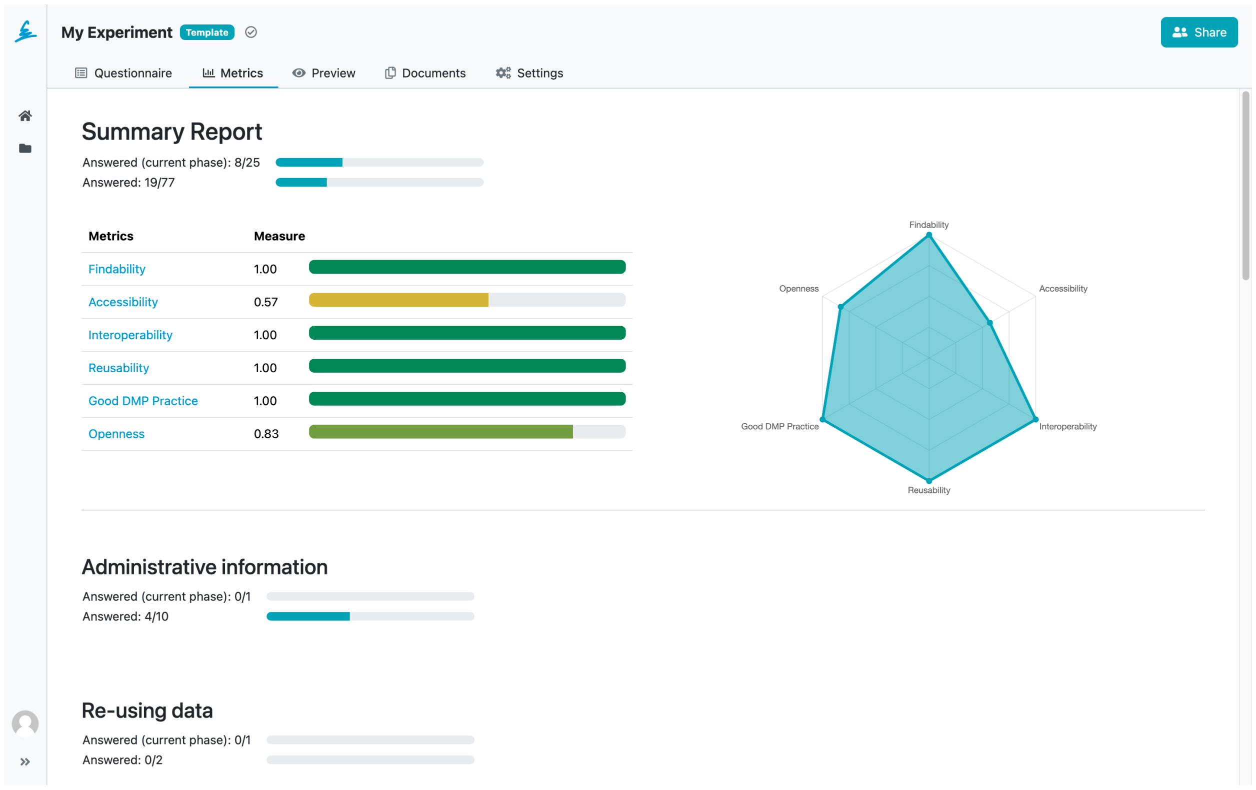
*Set phase to **Before Submitting the Proposal***

Isaac Newton

15:33

*Set phase to **Before Finishing the Project***

Fig. 52: Version history shows all events changing the project.



Summary Report

Answered (current phase): 8/25

Answered: 19/77

Metrics	Measure
Findability	1.00
Accessibility	0.57
Interoperability	1.00
Reusability	1.00
Good DMP Practice	1.00
Openness	0.83

Administrative information

Answered (current phase): 0/1

Answered: 4/10

Re-using data

Answered (current phase): 0/1

Answered: 0/2

Fig. 53: Summary report of project metrics.

Preview

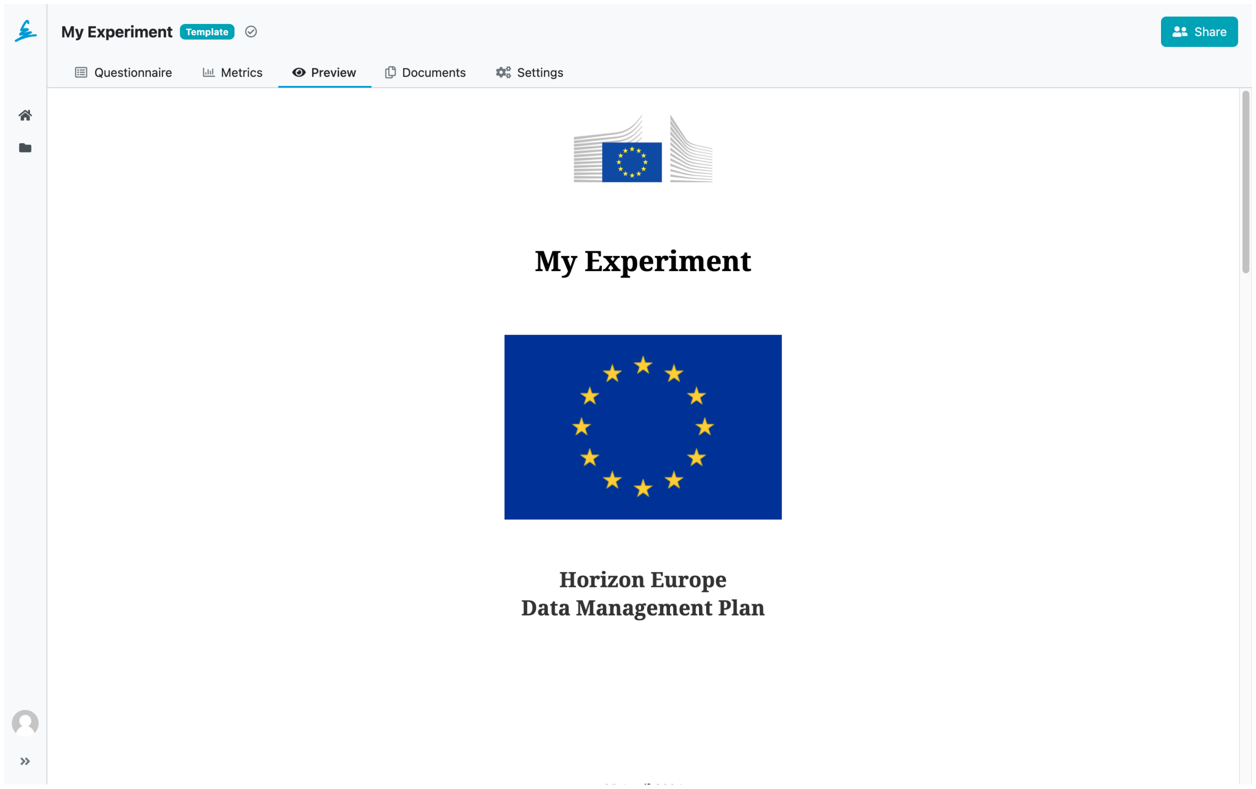
We can quickly see how the resulting document would look like in the *Preview* tab.

Default Document Template Not Set

The preview uses the default document template. If there is no default document template set in the *project settings*, we need to set it first.

Download Preview

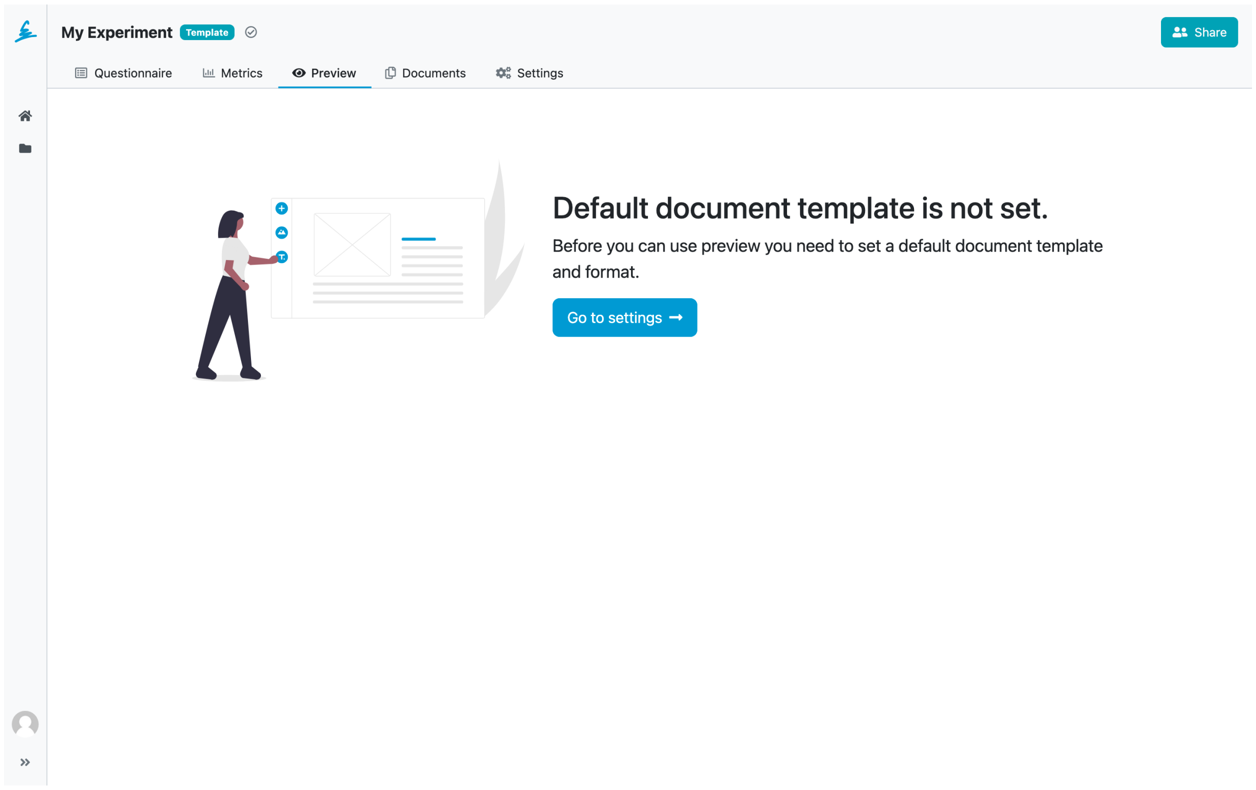
Not all formats can be displayed in the web browser, for example a MS Word document. In that case, we'll be able to download the document instead of previewing it.



My Experiment

Horizon Europe
Data Management Plan

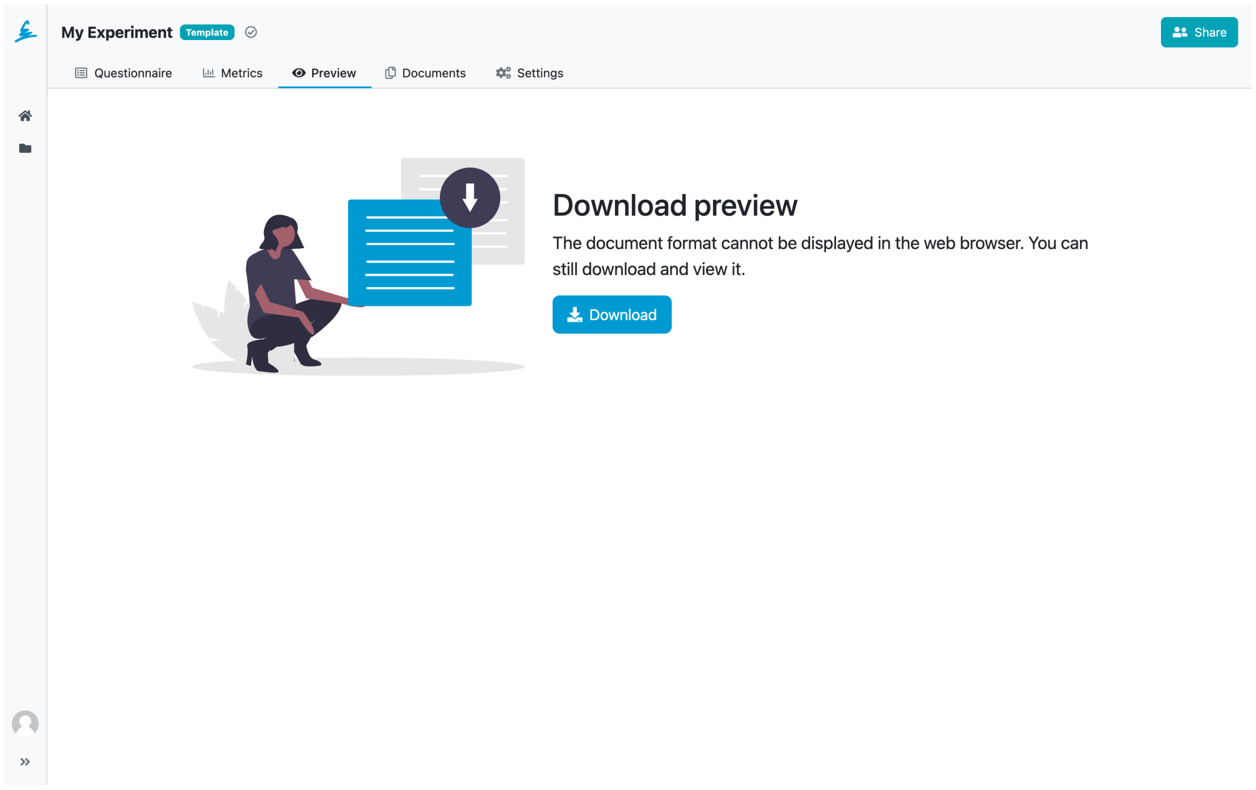
Fig. 54: Document preview in the project detail.



Default document template is not set.

Before you can use preview you need to set a default document template and format.

Fig. 55: Preview only works when the default document template is set.



Download preview

The document format cannot be displayed in the web browser. You can still download and view it.

Fig. 56: Some formats cannot be displayed in the web browser.

Documents

In the *Documents* tab in the project detail we can manage the documents related to the project. We can see a list of all the documents with their **format**, **file size**, and **document template** used to create that document.

While we can quickly see how the current state of the questionnaire looks like in the document in the *preview* tab, the documents created here are persistent. That means that the once the document is created, it is immutable and you can always download it later, after you have changed the questionnaire, and it will still be the same.

New document

We can click on *New document* when we want to create a new document. We need to give a name to our new document (project name is prefilled) and choose the **document template** and **format**. If there is a *default document template and format* set for the project, they are prefilled in this form. However, we can change them to whatever we want before creating the document. Once we hit *Create*, we are taken back to the document list and we'll see the new document there (it might take while before it is generated though).

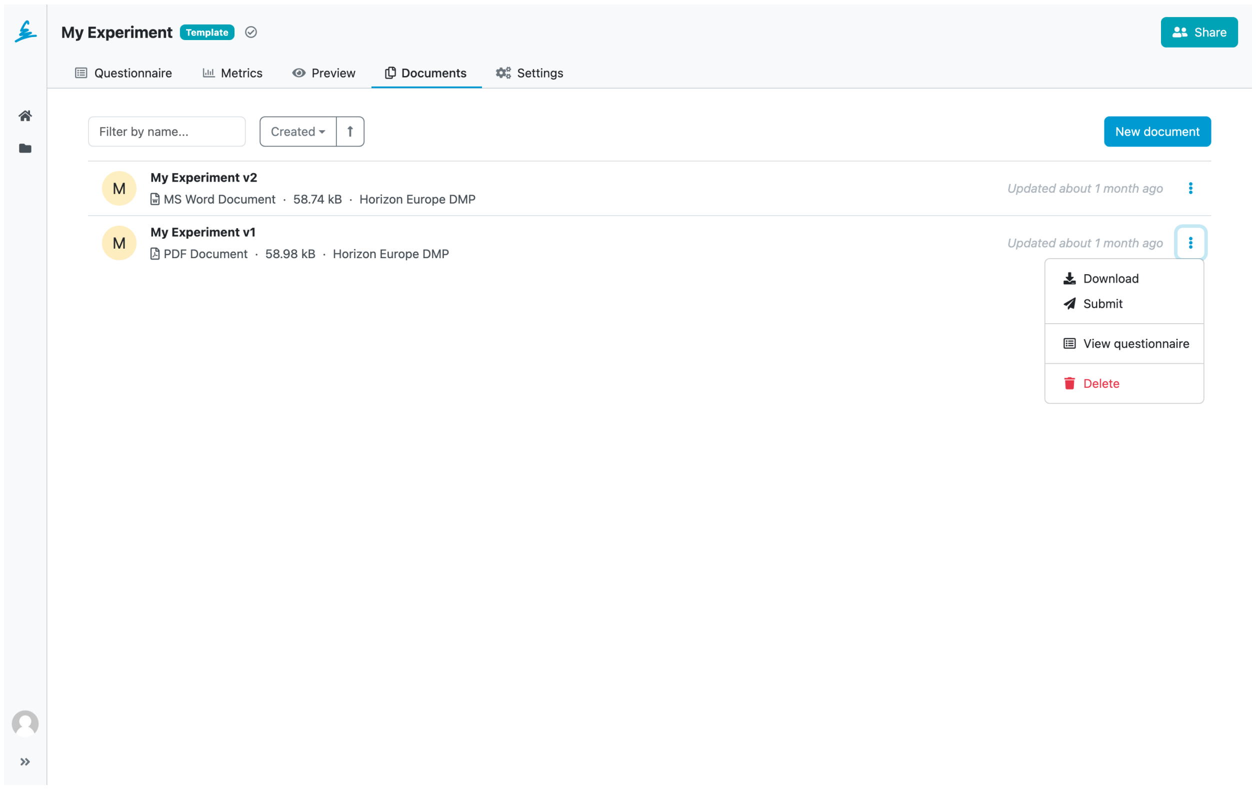


Fig. 57: List of the documents on the project.

Document Submission

Submission services can be used to quickly submit the document directly from FAIR Wizard to some external service.

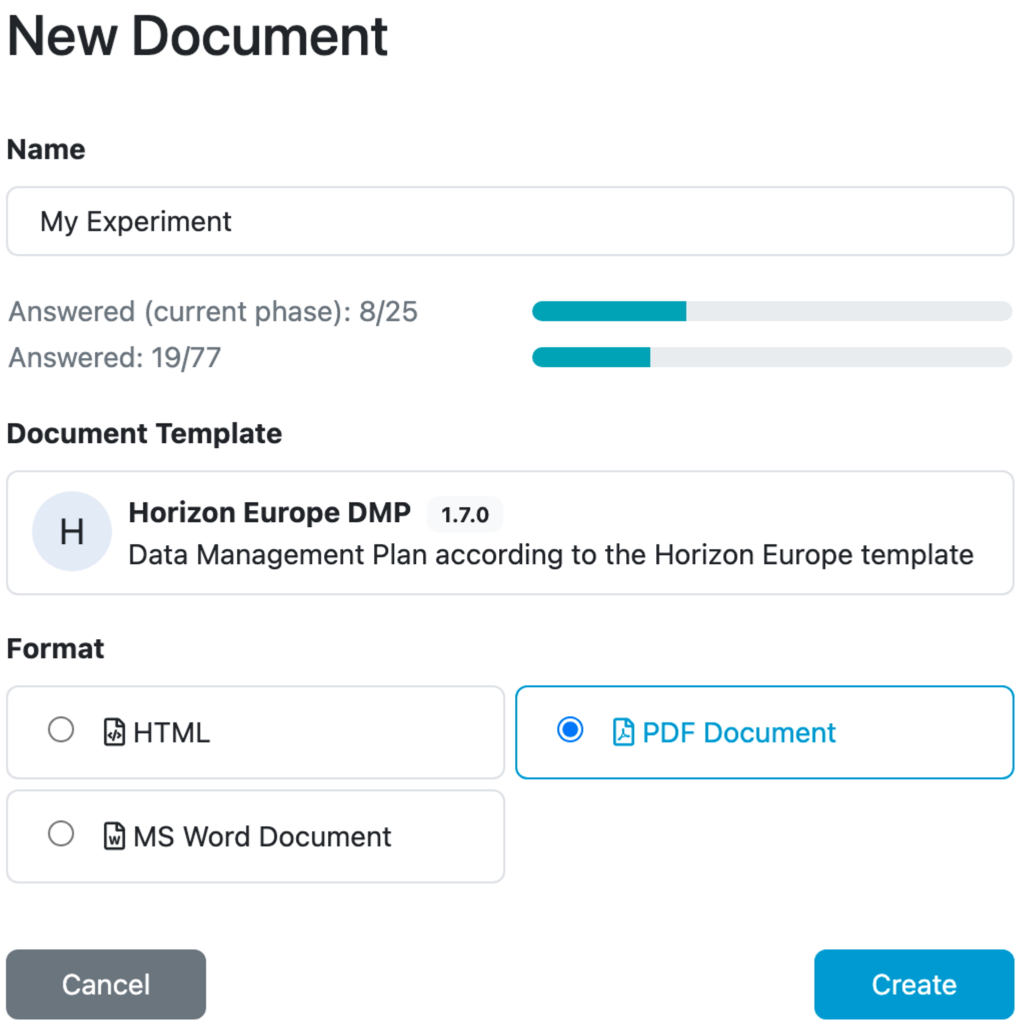
If there are any document submission services configured for our FAIR Wizard, we should be able to see *Submit* option when we open document menu in the document list by clicking on the triple dots. Then we choose the desired from the list and click on *Submit*.

We can see all the submissions for each document in the document list as well.

Settings

In the *Settings* tab, we can configure some details about the project. First we have a **name** and a **description** to identify the project.

Next, we have **Project Tags**. These can be used for providing some metadata or categorization of the project. In the project list, we can filter the projects by these tags. We can write any text we want as a project tag and FAIR Wizard will suggest us the tags that are there already in use so we can keep them consistent.



New Document

Name

My Experiment

Answered (current phase): 8/25

Answered: 19/77

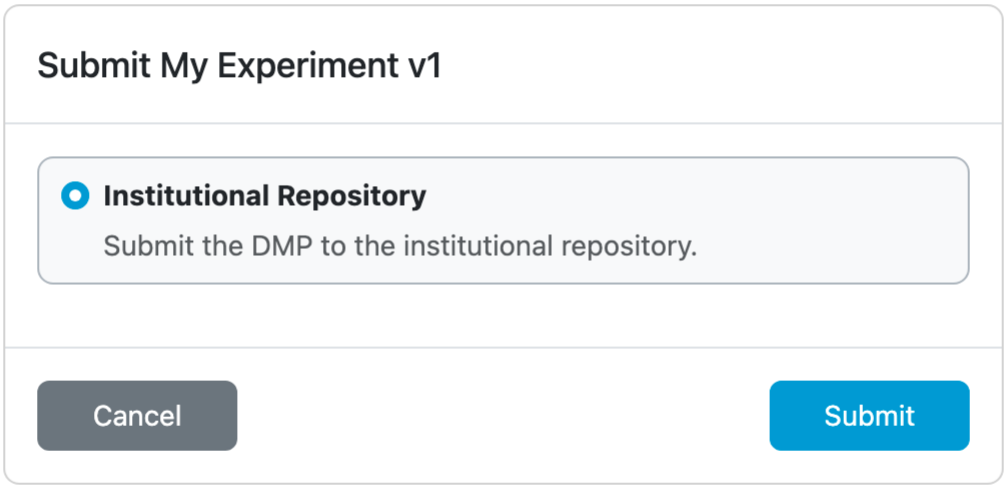
Document Template

Horizon Europe DMP 1.7.0
Data Management Plan according to the Horizon Europe template

Format

HTML [✓ PDF Document] MS Word Document

Fig. 58: We can choose any compatible document template and format when creating a new document on the project.



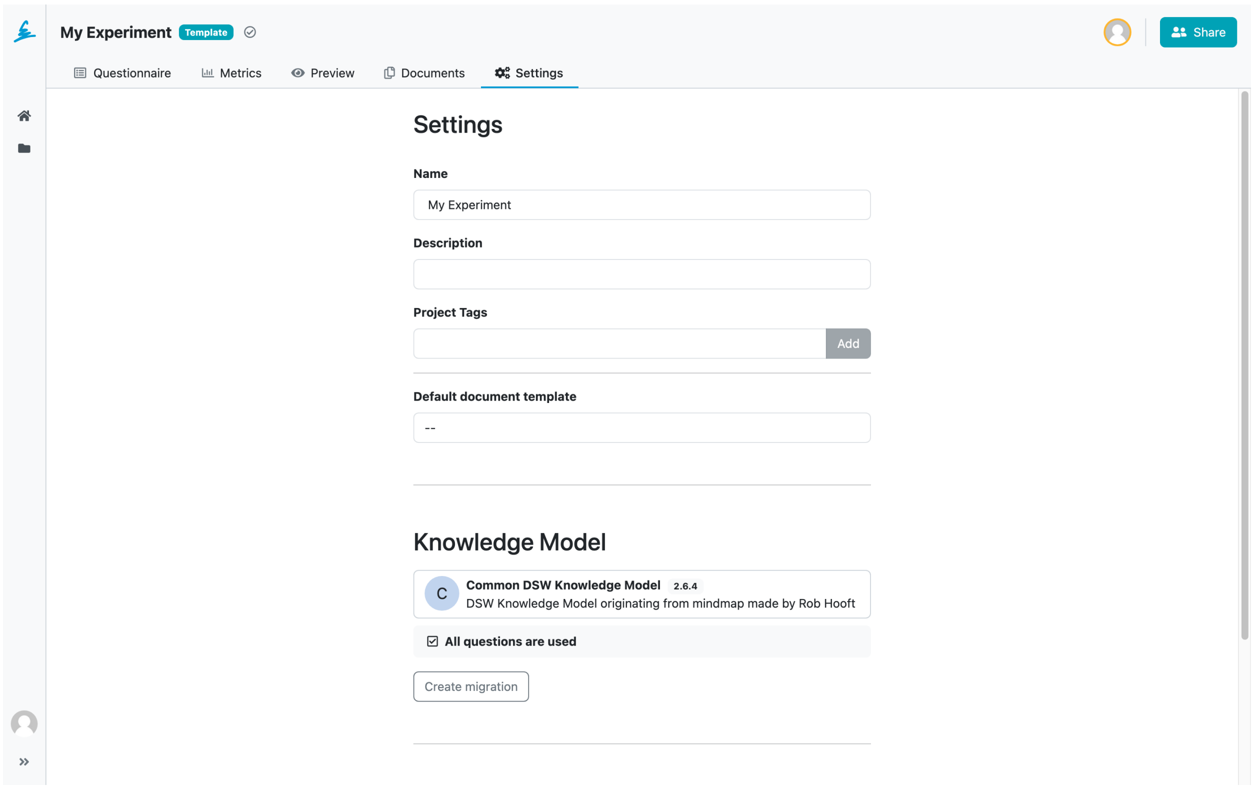
Submit My Experiment v1

Institutional Repository

Submit the DMP to the institutional repository.

Cancel Submit

Fig. 59: Submission service selection for a document.



My Experiment Template

Questionnaire Metrics Preview Documents Settings

Share

Settings

Name

My Experiment

Description

Project Tags

Add

Default document template

--

Knowledge Model

Common DSW Knowledge Model 2.6.4

DSW Knowledge Model originating from mindmap made by Rob Hooft

☑ All questions are used

Create migration

Fig. 60: Project settings.

Default Document Template

We can set a **default document template** and a **default document format**. These are then used in the *preview* tab and also pre-selected when creating a *new document*.

Project Template

We can use the project as a *project template*. If we enable this option, researchers can use it when creating a new project *from project template*. Note that we also need to make it visible for other logged-in users in *sharing* settings.

Note: Project template options are visible only for Data Stewards or Admins.

Unsupported metamodel can appear, when the document template is not compatible with the version of FAIR Wizard. Researchers should contact their Data Steward or Administrator in this case.

Knowledge Model

We can see the *knowledge model* and its tags used for creating the project. If we want to change that, we can simply create a *project migration*.

Danger Zone

In the danger zone, we can delete the project. Once the project is deleted it can **no longer be recovered**.

Sharing

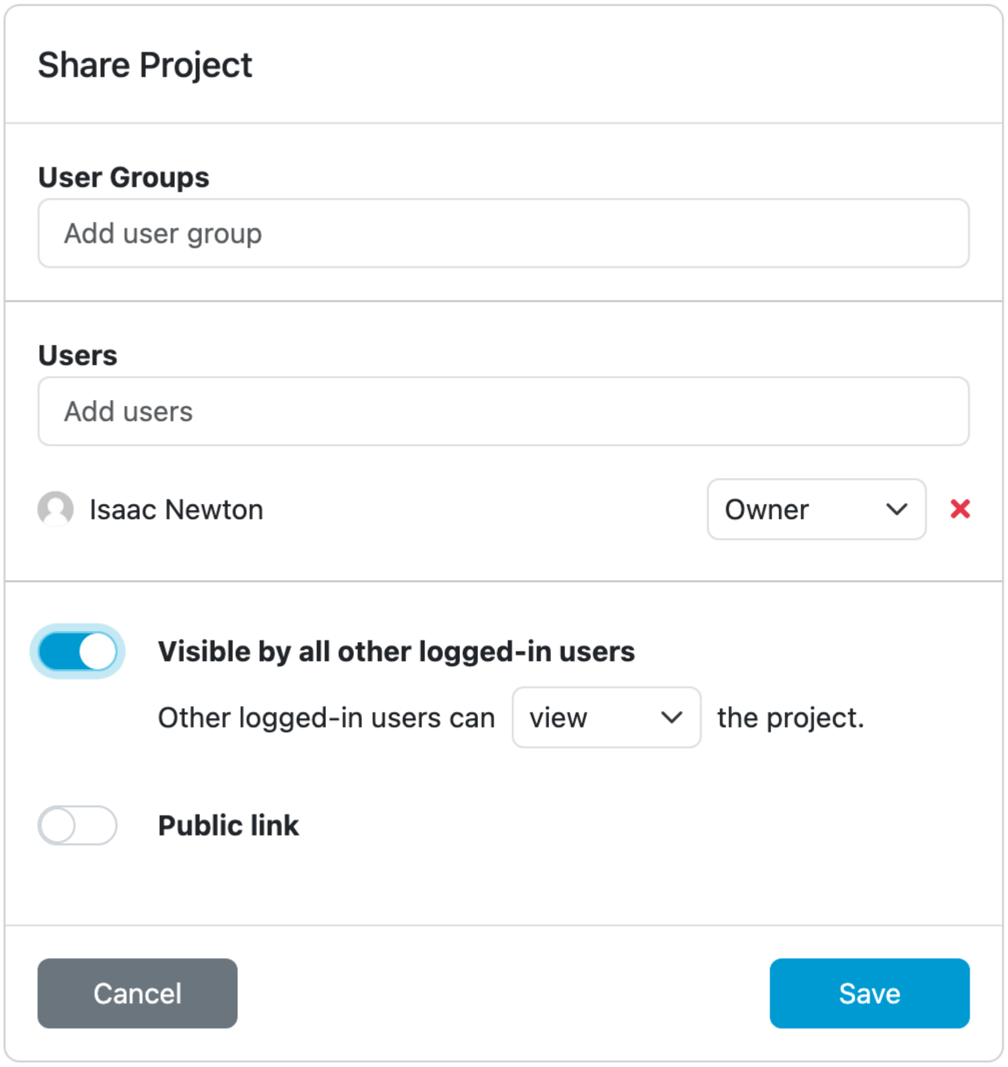
We can share project with other FAIR Wizard users or even external collaborators. We can access all the sharing option by clicking the *Share* button in the top right.

There are different roles in the project that can access different features:

The following video tutorial explains and showcases sharing options and tools that can be used while collaborating with others. Some features mentioned in the video are also explained in the *project questionnaire*.

https://youtu.be/Yzg6I6epcXQ

Note: Some of the following options might be globally disabled in the application settings for the whole FAIR Wizard, therefore not visible on the project level.


Share Project

User Groups

Add user group

Users

Add users

Isaac Newton — Owner

Visible by all other logged-in users [✓]

Other logged-in users can view the project.

Public link

Fig. 61: Project sharing settings.

Users

We can choose specific users from the FAIR Wizard and their role on the project to grant them access to project features based on the table above. This is a good way to add other collaborators that work together with us on the project. Also, this is the only way to add other project owners.

Visible by all other logged-in users

We can enable the *Visible by other logged-in users* toggle to grant access to the project for all other users without the need to explicitly list them. Then, we can also choose what the users can do – **view**, **comment**, or **edit** the project. We cannot grant **owner** access this way though.

This can be useful, when we want to have an example project accessible by everybody. We can simply enable this and choose that other users can **view** the project. We also want to set this up when we create a *project template*.

Public link

We can enable the *Public link* toggle to grant access to the project to anyone who has the link. We can again choose what they can do – **view**, **comment**, or **edit** the project. Also, the public link is visible there, so we can simply copy it and send to whoever we want to collaborate with. Then, they don't need a FAIR Wizard account and still be able to access our project.

Project Migration

Every project is based on a specific *knowledge model*, its version, and selected tags. Sometimes, we might want to change the knowledge model to a different version (for example, when a new version is released), change the knowledge model (for example, when a new customization is created), or just change the tag selection. Project migration is a process where we can do this.

The following video explains all aspects of Project Migration.

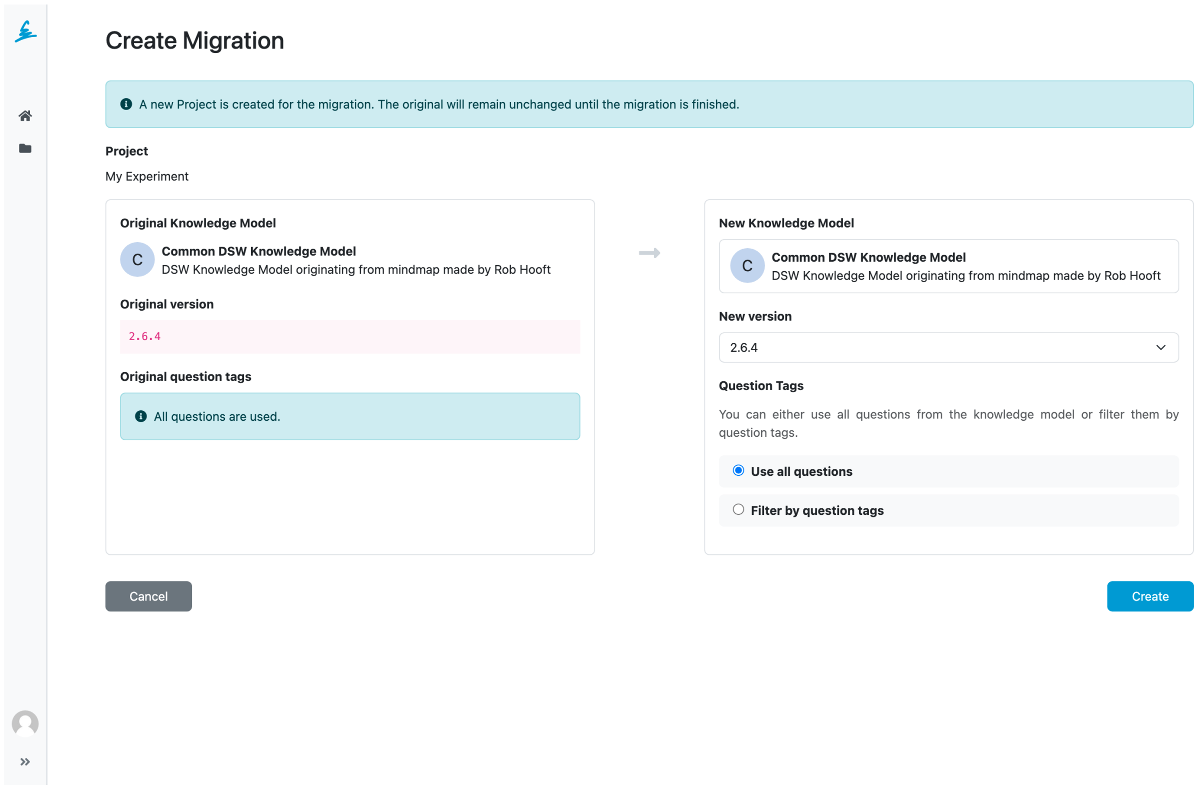
https://youtu.be/536IqCCfLyk

Creating a Project Migration

We can start a project migration either from the *project list*, or from the *project settings*. Sometimes, when there is a newer version of the knowledge model available, we can see a tag *update available* next to the project name. We can click on the tag to start the migration as well.

We can see the **original knowledge model**, its **version**, and selected **question tags** on the left side. On the right side we can choose new values for all of these. After we are satisfied with our selection we can click on *Create* button.

Note that the original project will remain unchanged until the migration is finished. So we can cancel it anytime before it is finsihed without affecting the project.


Create Migration

A new Project is created for the migration. The original will remain unchanged until the migration is finished.

Project
My Experiment

Original Knowledge Model
Common DSW Knowledge Model
DSW Knowledge Model originating from mindmap made by Rob Hooft

Original version
2.6.4

Original question tags
All questions are used.

New Knowledge Model
Common DSW Knowledge Model
DSW Knowledge Model originating from mindmap made by Rob Hooft

New version
2.6.4

Question Tags
You can either use all questions from the knowledge model or filter them by question tags.

[✓ Use all questions]
Filter by question tags

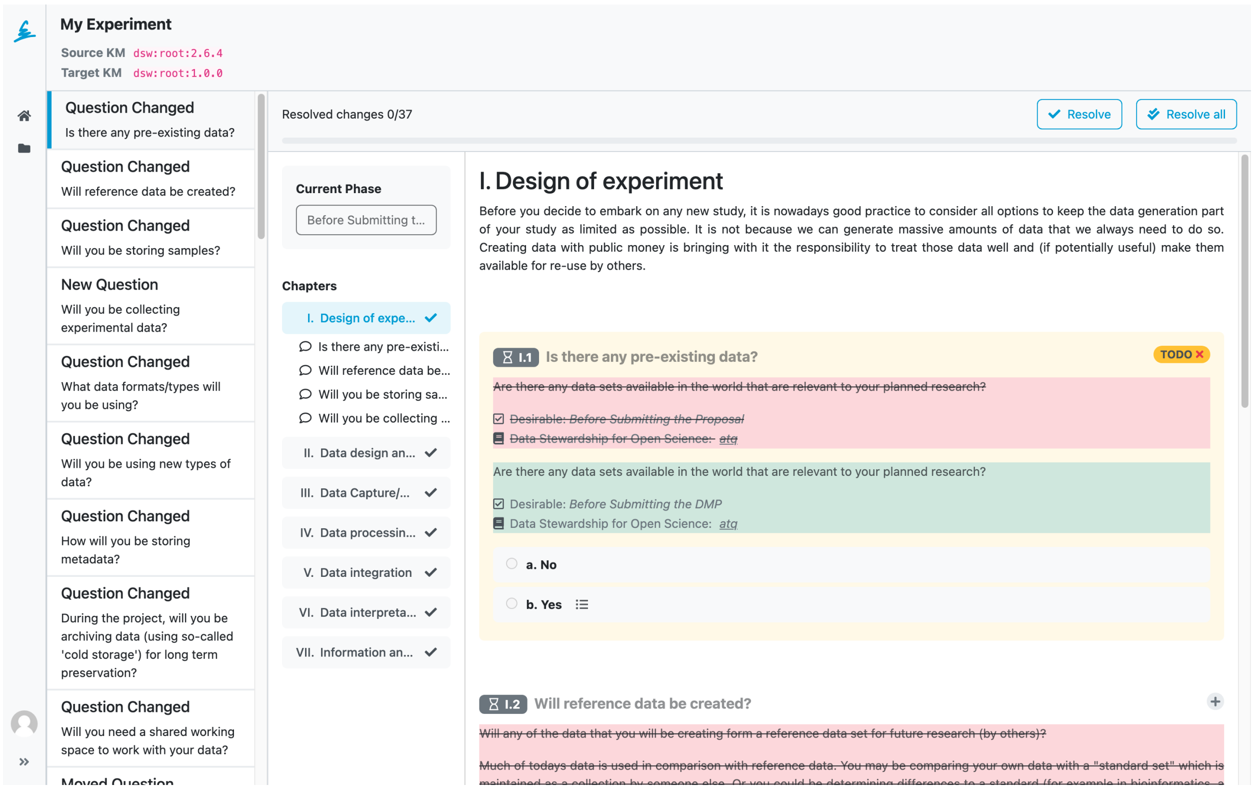
Fig. 62: Choosing a new knowledge model for the project when creating a project migration.

Project Migration

The next screen is the project migration itself. We can go through all the changes between the orignal and the new knowledge model that affects our answers. During this process, we can also add *todos* if we want to come back to a specific question later, after the migration.

It is possible that there are no changes to review. This can happen when we don't have all the answers in the questionnaire yet and those we have are not affected by the changes, i.e., all of the questions that we answered are in the original and in the new knowledge model.

We can leave the migration at any point now and come back to it later. We will see the project twice in the *project list*, one of them tagged as *migrating*. If we open the migrating one, we can come back to the project migration. If we open the other one, we can access the original project, however, only in the read-only mode until the migration is finished or cancelled.


My Experiment

Source KM dsw:root:2.6.4

Target KM dsw:root:1.0.0

Question Changed
Is there any pre-existing data?

Question Changed
Will reference data be created?

Question Changed
Will you be storing samples?

New Question
Will you be collecting experimental data?

Question Changed
What data formats/types will you be using?

Question Changed
Will you be using new types of data?

Question Changed
How will you be storing metadata?

Question Changed
During the project, will you be archiving data (using so-called 'cold storage') for long term preservation?

Question Changed
Will you need a shared working space to work with your data?

Moved Question

Resolved changes 0/37

Current Phase: Before Submitting t...

Chapters

I. Design of expe...
- Is there any pre-existi...
- Will reference data be...
- Will you be storing sa...
- Will you be collecting ...

II. Data design an...

III. Data Capture/...

IV. Data processin...

V. Data integration

VI. Data interpreta...

VII. Information an...

I. Design of experiment

Before you decide to embark on any new study, it is nowadays good practice to consider all options to keep the data generation part of your study as limited as possible. It is not because we can generate massive amounts of data that we always need to do so. Creating data with public money is bringing with it the responsibility to treat those data well and (if potentially useful) make them available for re-use by others.

I.1 Is there any pre-existing data? TODO

~~Are there any data sets available in the world that are relevant to your planned research?~~

~~Desirable: *Before Submitting the Proposal*~~

~~Data Stewardship for Open Science: *atg*~~

Are there any data sets available in the world that are relevant to your planned research?

Desirable: *Before Submitting the DMP*

Data Stewardship for Open Science: *atg*

a. No

b. Yes

I.2 Will reference data be created?

~~Will any of the data that you will be creating form a reference data set for future research (by others)?~~

~~Much of todays data is used in comparison with reference data. You may be comparing your own data with a "standard set" which is maintained as a collection by someone else. Or you could be determining differences to a standard (for example in bioinformatics, a~~

Fig. 63: Reviewing changes during the project migration.

Cancelling a Project Migration

At any point before we finalize the migration, we can decide that we actually don't want to do the migration. We can simply navigate to the *project list* and choose the *Cancel migration* action on the project tagged as *migrating*. This will cancel the migration and the original project will remain unaffected.

Finishing a Project Migration

After we resolve all the changes (or if there are no changes to review), we can click on *Finalize migration*. This will complete the project migration applying all the knowledge model changes, and unlocking the project from the read-only mode.

Project Templates

Note: Only the data stewards or admins can create project templates.

When creating a new project, we need to choose a knowledge model and optionally select some question tags. After the project is created, we should also choose a *default document template* and format to enable *preview*. It can be overwhelming for new researchers to set up everything when they are new to all this.

Project templates are special type of projects where we can set up everything – choose a knowledge model and question tags, set up default document template, pre-fill some answers, add TODOs, comments or editor notes. Researchers can then pick from these project templates when *creating a new project*. The new project will be the exact copy of the project template so they don't have to set those things themselves and they have an easier start to their data management planning.

When we want to turn a project into a project template we need to go to the *project settings* and check the *Project Template* checkbox.

Sharing the Project Templates

Project templates follow the same rules for sharing as regular projects. Therefore, to make it available for other users, we need to set up proper *sharing settings*.

We can either share them with specific users only, or we can simply enable that the project template is **visible by all other logged-in users** and that they can only **view** it (as we do not want them to change it).

External Resources

- Project Templates in FAIR Wizard

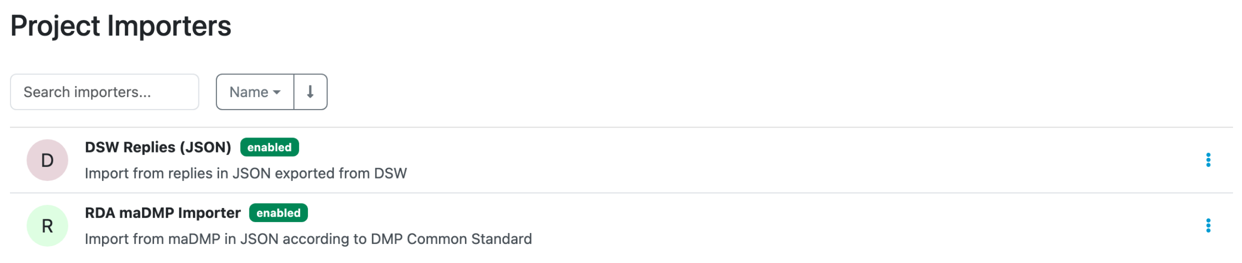
Project Importers

Warning: Project importers are an experimental feature.

We can use project importers to import data from different FAIR Wizard or even different applications to FAIR Wizard. Each has a set of supported knowledge models defined. This is because each knowledge model has a different structure and the importer needs to understand it so it can import the answers to the correct questions.

Note: Only data stewards or admins can access project importers.

If we navigate to *Projects* → *Importers*, we can see the list of all available importers. We can enable or disable them by clicking on the triple dots icon and choosing the appropriate action.


Project Importers

Search importers...

Name ↓

D DSW Replies (JSON) enabled
Import from replies in JSON exported from DSW

R RDA maDMP Importer enabled
Import from maDMP in JSON according to DMP Common Standard

Fig. 64: List of project importers where we can enable or disable them.

More information about how to develop project importers is available on the *project importers development* page.

1.2.4 Documents

As admins, we can quickly browse all documents stored in the FAIR Wizard by navigating to *Documents* from the main menu. It is possible to search for a document by name or sort them using the name or creation timestamp.

Each document has name, format, and certain size (if the generation is finished). It can be directly downloaded or deleted from the list. Moreover, we can quickly navigate to the project from which the document is created.

In case that there is a document that was not generated due to an error, we can check the error message. Also, if there are some documents got stuck in *Queued* or *In Progress* status, we should check the deployment (especially of the document worker component).

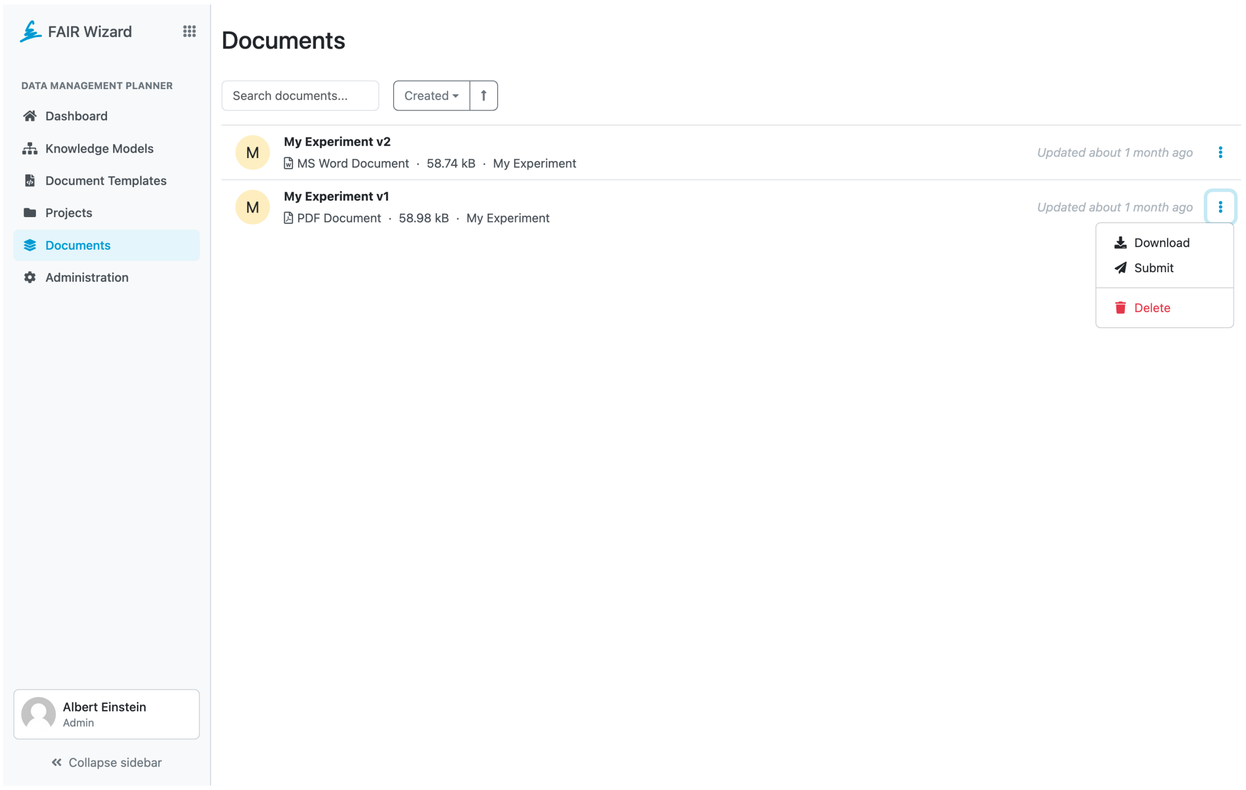


Fig. 65: List of all documents across projects.

1.2.5 Administration

Administration section serves admins with managing the FAIR Wizard. By navigating via *Administration* item from the main menu, we can manage the things listed below.

Settings

This section covers different settings available to admin users. The settings are categorized as listed below.

Note: Do not forget press *Save* button to save the changes.

System Settings

System settings allow us to configure basics about the organization running the FAIR Wizard.

Organization Settings

On this page, we can configure **Name**, **Description**, and **Organization ID** for our FAIR Wizard. The organization ID can contain alphanumeric characters and dot symbol (but cannot start or end with dot).

Note: It is recommended to use period as for domain name or to use it for capturing organizational structure in the ID, for example, `faculty.university` or `group.faculty.university`.

User Interface Settings

User Interface (UI) settings allow us to manage how the Data Management Planner app looks like: dashboard shown when user logs in, announcements and custom menu links.

Dashboard Settings

The dashboard settings allows us to adjust what users will see after they log in, i.e. on the application initial page called the dashboard.

Dashboard Style

We can select the **Dashboard Style** whether the user should see a standard **welcome** screen which just greets the user in the application, or a **role-based** dashboard which contains widgets based on current user's role (see *User Roles*):

- **Researcher**
  - **Recent Projects Widget** contains a list of recent projects of the user for a quick navigation.
  - **Create Project Widget** lets the user quickly start a new project.
- **Data Steward**
  - **Create KM / Project Template Widgets** let the user to quickly start a new knowledge model editor or project template.

  - **Outdated KM / Document Templates Widgets** allow to quickly see outdated packages and document templates in case the FAIR Wizard Registry connection is configured.
  - **Import KM / Document Template Widgets** allow to proceed easily to import of a knowledge model or a document template in case the FAIR Wizard Registry connection is configured.
- **Administrator**
  - **Outdated KM / Document Templates Widgets** allow to quickly see outdated packages and document templates in case the FAIR Wizard Registry connection is configured.
  - **Usage Widget** summarises the usage just as is also possible to see in the *Usage*.
  - **Configure Organization Widget** quickly navigates to *Organization Settings* if it is not yet done.

Announcements

Another option to adjust the dashboard and/or the login screen is to add Announcements. Announcements are displayed above the main content in the login screen. In dashboard, they are also displayed above the main content for both **welcome** and **role-based** dashboard style. There are three levels of Announcements:

- **Info** - light blue color for sending informations to the users.
- **Warning** - yellow to warn the users about something.
- **Critical** - red to signalize the Announcement is critical and it needs attention.

The content of the Announcement can be edited using Markdown. There are also two additional switches which determine, where the Announcement is displayed. The Announcement can be set up to be displayed either on the dashboard after users log in or on the login screen before the users log in. It is also possible to display the same Announcement in both places. Number of Announcements is not limited.

Menu Settings

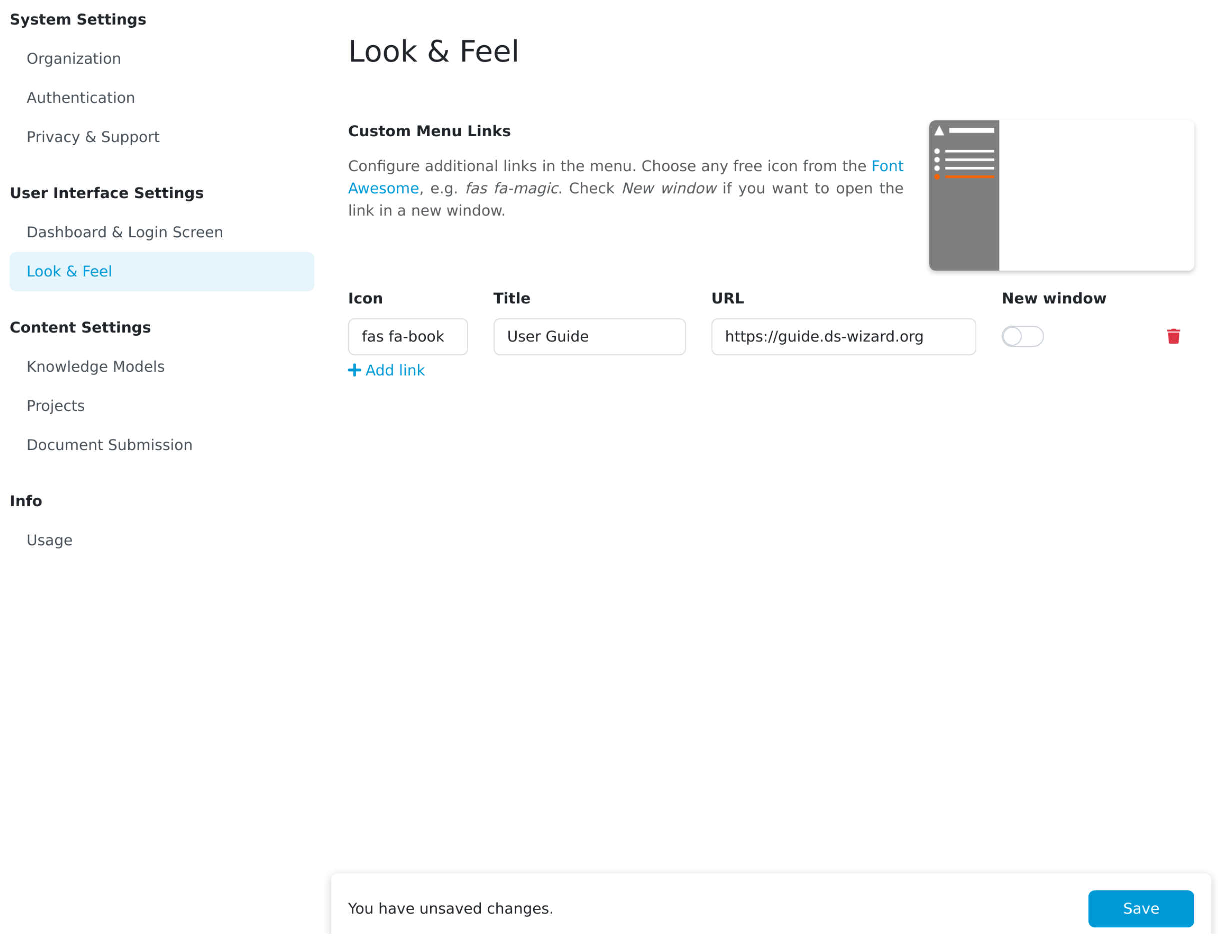
This part of settings allows us to adjust how the FAIR Wizard menu looks like.

Custom Menu Links

We can easily add custom links to the main (left) menu by clicking *Add* under **Custom Menu Links**. For each link, we can set **Icon** (from Font Awesome), **Title** and the target **URL**. We can also set whether the link should open in **New Window** (if not, it will nagivate user directly in the same window/tab from FAIR Wizard). Once the links are there, we can manage them or delete them at this place.

Content Settings

This part of settings allows us to configure various content-related things such as Knowledge Models, Projects, and Document Submissions as listed below.



Look & Feel

Custom Menu Links

Configure additional links in the menu. Choose any free icon from the Font Awesome, e.g. *fas fa-magic*. Check *New window* if you want to open the link in a new window.

Icon	Title	URL	New window
fas fa-book	User Guide	https://guide.ds-wizard.org	

You have unsaved changes.

Fig. 66: Example configuration of a custom menu link.

Knowledge Models Settings

Public Knowledge Models

If we want to let users to see and browse certain Knowledge Models (specifically, visit the *KM detail* and the *KM preview*) even if not logged in, we can enable **Public Knowledge Models**. Then, we need to specify **Allowed Packages**, e.g. which ranges of verions of a certain knowledge model will be publicly available. A blank value serves as *any value*, for example, if we fill the **Organization ID** and **Knowledge Model ID** but leave **Min Version** and **Max Version**, it will result in all version of that knowledge model to be public.

Integration Config

The integrations specified in Knowledge Models can use configuration values (typicaly secrets such as API keys or tokens) from YAML configuration specified here under **Integration Config**. The value here can be for example:

```
dbase:
    apiKey: topSecretDBaseApiKey
    apiUrl: https://api.dbase.example:10666
```

Note: This configuration value is encrypted in the database.

Projects Settings

Project Visibility

If we want to let users select visibility of their projects within the FAIR Wizard, we can enable **Project Visibility** feature. If it is disabled, the new projects will have the **Default Project Visibility** which is used when creating a new project:

- **Private** = the project is visible only to the users with explicit access to the project.
- **Visible - View** = the project is visible in view-only mode to all logged-in users, i.e. all users will be able to see the project in their *projects list* and access it (but not edit or comment anything unless they are invited with different permissions).
- **Visible - Comment** = the project is visible in comment mode to all logged-in users, i.e. all users will be able to see the project in their *projects list*, access it and comment it (but not edit anything unless they are invited with different permissions).
- **Visible - Edit** = the project is visible in edit mode to all logged-in users, i.e. all users will be able to see the project in their *projects list*, access it, comment it, and also edit it (e.g. answer questions or editor notes).

Project Sharing

If we want to let users select sharing option of their projects within the FAIR Wizard, we can enable **Project Sharing** feature. If it is disabled, the new projects will have the **Default Project Sharing** which is used when creating a new project:

- **Restricted** = only logged-in users can access the project depending on the project visibility (no public access for anonymous users).
- **View with the link** = anyone with the link to the project may open it in view mode and browse it.
- **Comment with the link** = anyone with the link to the project may open it in comment mode, i.e. browse it and comment on questions.
- **Edit with the link** = anyone with the link to the project may open it in edit mode, i.e. browse it, comment on questions, and also edit it (e.g. answer questions or editor notes).

Anonymous Projects

If we have enabled *Public Knowledge Models*, we can also allow anonymous users to create projects where they will be able to fill questionnaires by enabling **Anonymous Projects**. These anonymous project then work as any other projects with public link set to edit permissions. However, if a logged-in user accesses such a project, then such a user may claim the ownership by clicking *Add to my projects* button. Anonymous users cannot create new documents, for that they must register and open the project as a logged-in user.

Feedback

In case we want to allow users to provide feedback specific to questions directly from *questionnaires*, we can enable **Feedback** and configure a GitHub repository which will be used as an issue tracker. We should have a bot/service account created with access to the GitHub repository and obtain Personal Access Token. This account will be used to create the GitHub issues in the repository. Then, we need to simply fill **GitHub Repository Owner**, **GitHub Repository Name**, and the **Access Token**.

Note: The **Access Token** value is encrypted in the database.

Project Tagging

If enabled, users will be able to tag their projects (using so-called **Project Tags**) and then use those tags to filter the *projects list*. The users will be always able to write their own tags but we can provide a list of pre-defined **Default Project Tags** (one per line).

Document Submission Settings

If we enable the **Document Submission** feature, users will be able to see *Submit* option for their documents. After selecting it, they will be prompted to select a service compatible with the document where they want to submit it.

Each service must have its own **ID** (recommended is to use lowecase alphanumeric symbols and dash symbols). Then, we can set human-readable **Name** and **Description** (Markdown-enabled) to clarify for users what is this service and in what cases they should use it. We also need to specify the **Supported Formats**, for each we need to select a document template, its version, and the desired format. Finally, we configure how the document is sent to the external service, the request may contain some **User Properties** (users will be able to set values for them in their user profiles) and it is a HTTP request with a specific **Method**, sent to the **URL**, possibly with HTTP **Headers**. The very last option is to check whether the file should be sent as **Multipart** (with its own **File Name**) instead of plainly in the request body. Most of this configuration should be specified by the external submission service.

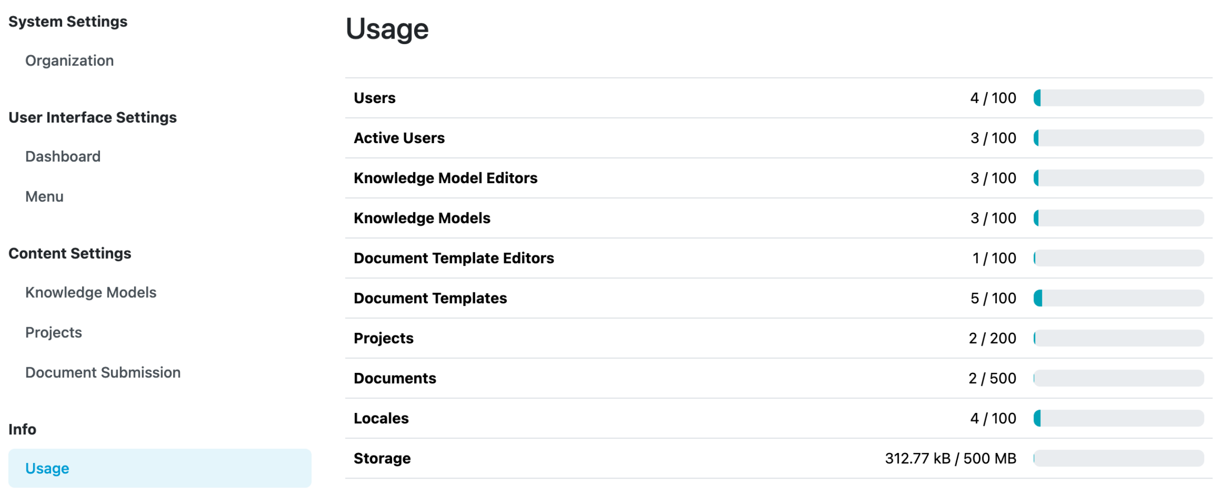
Note: In case we will update the document template used in **Supported Formats**, we should verify that it is still suitable for the submission service and if yes, then add it as a new entry under **Supported Formats**.

Info

The info section of settings simply serves to let us check various information and statistics about the FAIR Wizard instance.

Usage

Usage allows us quickly see numbers of entities in the FAIR Wizard such as number of users, active users, knowledge models, KM editors, templates, projects, or documents. Moverover, we can see storage usage, i.e. how much capacity is being used by documents and templates (their files). We cannot perform any actions on this page.



Usage

Users	4 / 100
Active Users	3 / 100
Knowledge Model Editors	3 / 100
Knowledge Models	3 / 100
Document Template Editors	1 / 100
Document Templates	5 / 100
Projects	2 / 200
Documents	2 / 500
Locales	4 / 100
Storage	312.77 kB / 500 MB

Fig. 67: Usage statistics.

Locales

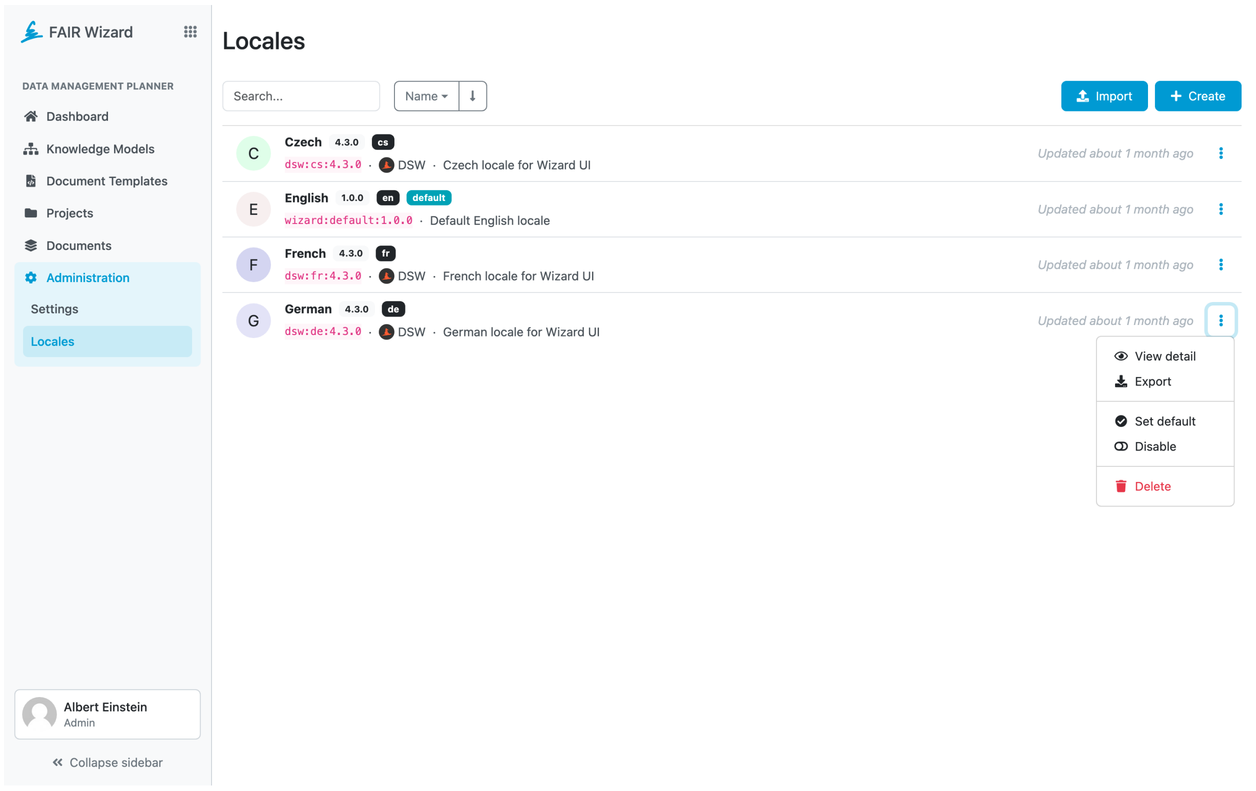
After navigating to *Locales* (under *Administration*), we can browse and manage a list of locales in the FAIR Wizard. Similarly to knowledge models and document templates, each locale has its unique identifier and version. Moreover, each locale has a language code specified. In the list we see the latest version and can quickly navigate to *Locale Detail* (which includes also older versions) by clicking the locale name or selecting *View detail* from the right item menu of the desired row.

There is always the **English** locale (`wizard:default:1.0.0`) which is embedded and cannot be deleted. For others, we can use *Export* and *Delete* options from the right item menu.

Another option is to switch other locale to be the default one using *Set default* action. The default locale will be used in case no available locale tgat matching user's preferences (explicit or implicit from the web browser). We can *Disable* or *Enable* locales except the default one (which must be enabled).

If there is a locale with newer version available in the FAIR Wizard Registry (and if configured), *update available* clickable badge may appear. Finally, we can use *Import* to *Import Locale* and *Create* to *Create Locale*.

Note: We support community of DSW translators by managing the repository ds-wizard/wizard-client-locales and service for translating using web browser localize.ds-wizard.org.



Czech 4.3.0 cs
dsw:cs:4.3.0 · DSW · Czech locale for Wizard UI — Updated about 1 month ago

English 1.0.0 en default
wizard:default:1.0.0 · Default English locale — Updated about 1 month ago

French 4.3.0 fr
dsw:fr:4.3.0 · DSW · French locale for Wizard UI — Updated about 1 month ago

German 4.3.0 de
dsw:de:4.3.0 · DSW · German locale for Wizard UI — Updated about 1 month ago

Fig. 68: List of locales.

Warning: For now, only Data Management Plannner has localization.

Import Locale

We can import an existing locale by navigating to *Settings* → *Locales* in the main menu and then clicking on *Import* button on the list of locales.

From FAIR Wizard Registry

The FAIR Wizard instance is connected to the FAIR Wizard Registry. That allows us to import locales from it by entering the **locale ID** of desired template (e.g. `fw:cs:4.3.0`) and pressing the *Import* button.

Note: In case of locales present in the FAIR Wizard Registry, we will be notified about the available upgrades.

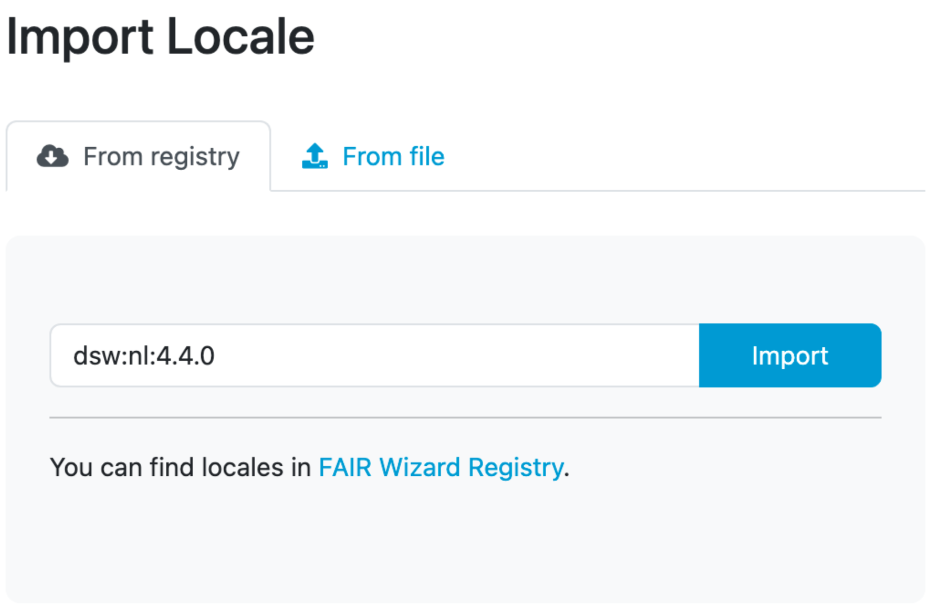


Fig. 69: Input for importing a locale from FAIR Wizard Registry.

From file

We can import a locale as a ZIP package. Such a package can be created as an export from FAIR Wizard.

Create Locale

We can create a new locale directly in FAIR Wizard by pressing *Create* from *Locales*. We need to fill the details about the new locale such as name, description, language code (RFC5646, e.g. `en` or `en-GB`), locale ID, locale version, license, README (with Markdown syntax), and recommended app version. The recommended app version captures for which version of the FAIR Wizard is this locale intended and compatible with (it can be used in other versions as well but may have some untranslated texts).

Finally, a PO file is requested from us. We can create such PO file in a standard (gettext-based) way. The needed POT file is always part of the release attachments of the wizard client application (select the desired release and there is `wizard.pot` asset).

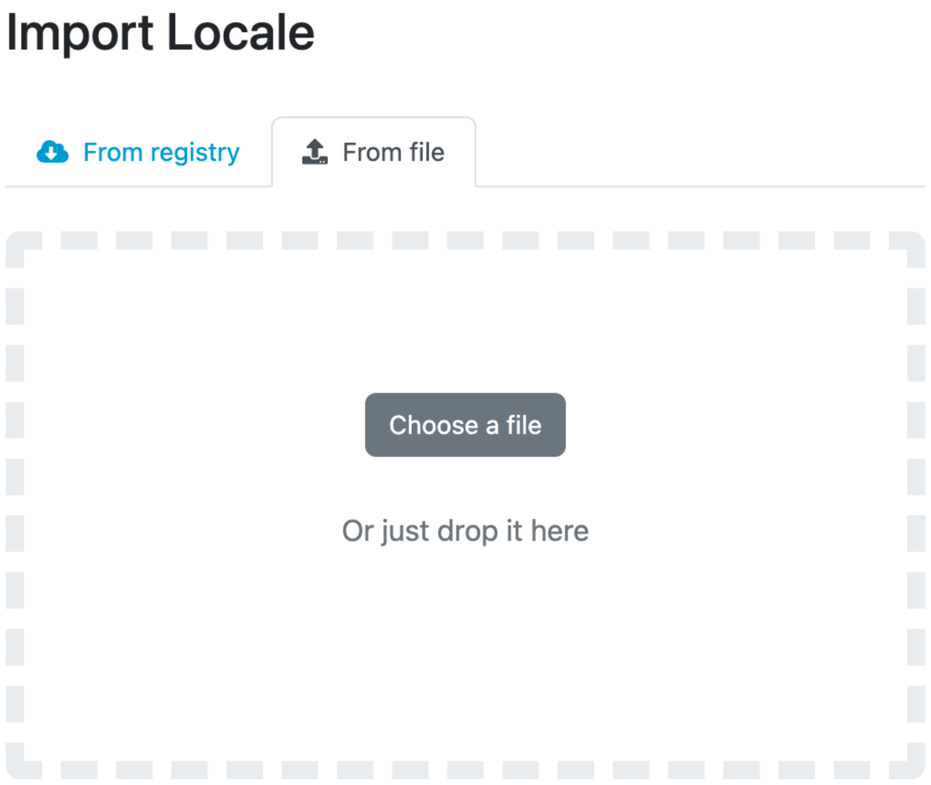


Fig. 70: Input for importing a locale using a ZIP package.

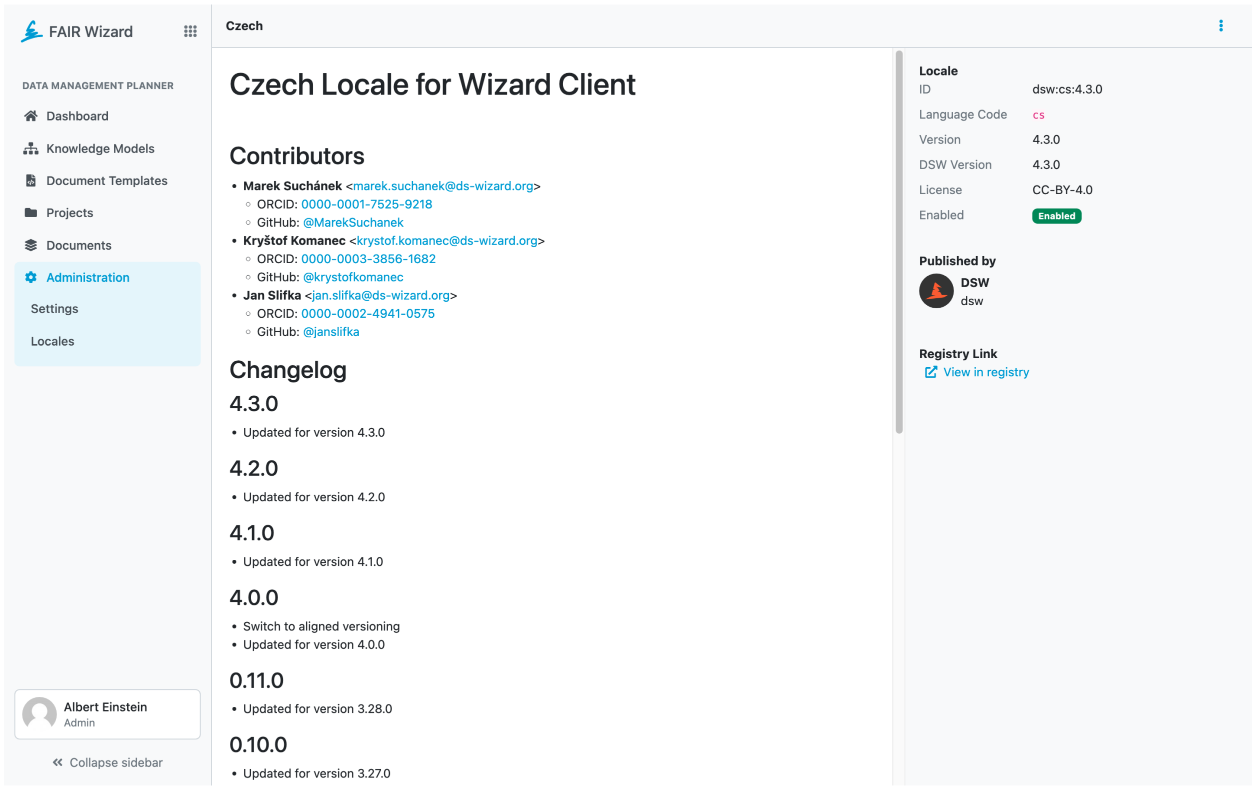
Locale Detail

The detail of a locale provides us information about a locale after navigating to it from *Locales*. The detail shows basic information about the locale such as its name, ID, language code (RFC5646, e.g. `en` or `en-GB`), version, recommended compatible FAIR Wizard version, license, and indication whether the locale is enabled or not.

The main part of the detail is the README of the locale that should contain basic information and changelog. In the right panel under the basic information, we can navigate to other versions of the locale.

In the top bar, we can *Export* the locale as a ZIP package, set or unset it as default, *Enable* or *Disable* it, or *Delete* this version of the locale.

Note: The default locale must be always enabled as it serves for users that did not have any preference or do not request matching locale directly using the browser configuration.



Czech Locale for Wizard Client

Contributors

- Marek Suchánek <marek.suchanek@ds-wizard.org>
  - ORCID: 0000-0001-7525-9218
  - GitHub: @MarekSuchanek
- Kryštof Komanec <krystof.komanec@ds-wizard.org>
  - ORCID: 0000-0003-3856-1682
  - GitHub: @krystofkomanec
- Jan Slifka <jan.slifka@ds-wizard.org>
  - ORCID: 0000-0002-4941-0575
  - GitHub: @janslifka

Changelog

4.3.0
- Updated for version 4.3.0

4.2.0
- Updated for version 4.2.0

4.1.0
- Updated for version 4.1.0

4.0.0
- Switch to aligned versioning
- Updated for version 4.0.0

0.11.0
- Updated for version 3.28.0

0.10.0
- Updated for version 3.27.0

Locale

ID	dsw:cs:4.3.0
Language Code	cs
Version	4.3.0
DSW Version	4.3.0
License	CC-BY-4.0
Enabled	Enabled

Published by
DSW
dsw

Registry Link
View in registry

Fig. 71: Detail of a locale.

1.2.6 Profile

As every logged-in user, we can manage our own profile. We can see the options by hovering over the box with our name and avatar in the lower part of the left sidebar with main menu (only avatar is shown in case of collapsed sidebar). From there, we can navigate to *User Settings*. If locales are configured, we can also *Change Language*.

Moreover, there are also options to *Log out*, see basic information about the FAIR Wizard using *About*, or in case of problems we can use *Report issue* to know how to proceed.

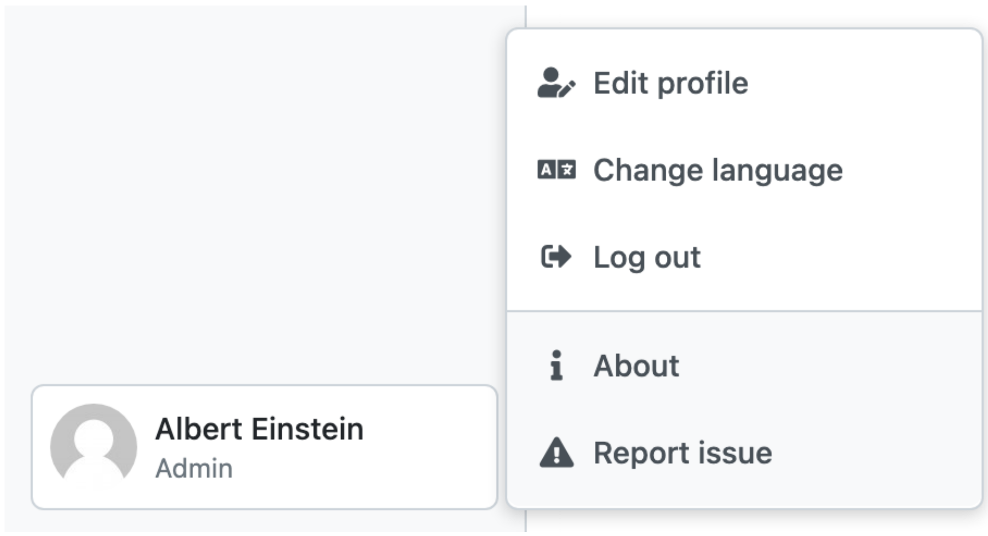


Fig. 72: Profile menu.

User Settings

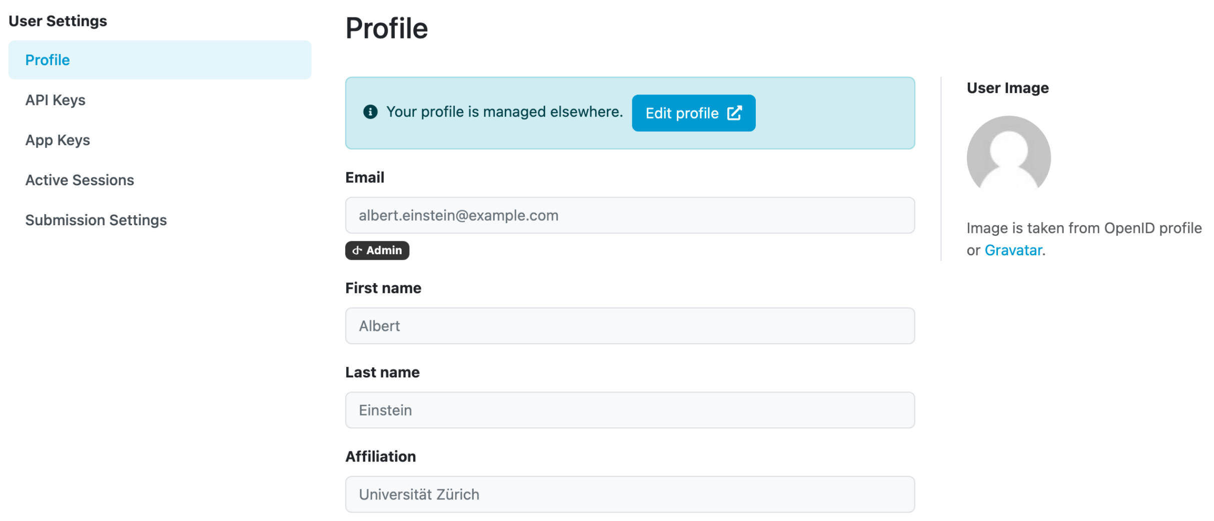
After navigating to *Edit profile* from the *Profile* menu, we open the User Settings. We can view our **Email** address, **First name**, **Last name**, and **Affiliation**. To edit those values, we must first switch to the *Admin App* and continue there.

We can see API Keys we have generated and App Keys used to connect to different apps.

We can also see Active Sessions and revoke them if needed.

In case of configured submission services, there might be additional inputs under **Submission Settings** such as API tokens or other information used for the document submission.

Note: The values of **Submission Settings** are treated as potentially sensitive information; thus are stored encrypted.



Your profile is managed elsewhere.

Email: albert.einstein@example.com

First name: Albert

Last name: Einstein

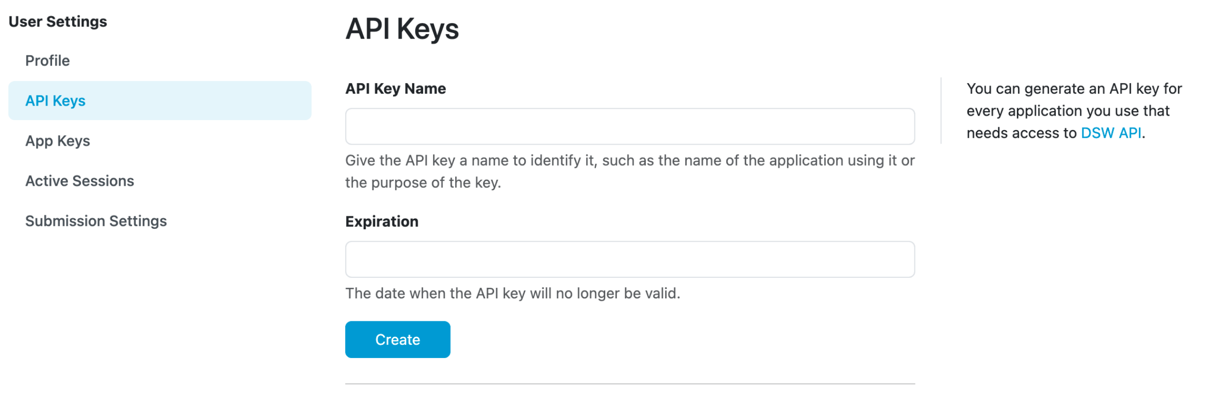
Affiliation: Universität Zürich

Image is taken from OpenID profile or Gravatar.

Fig. 73: Form for editting profile with example submission settings.

API Keys

When we want to access the FAIR Wizard through FAIR Wizard API we have to set up an API Key. The API Key has an *API Key Name* so we can remember for what purpose this Key is used and *Expiration* which is a date from when the API Key will no longer be valid.



Give the API key a name to identify it, such as the name of the application using it or the purpose of the key.

The date when the API key will no longer be valid.

You can generate an API key for every application you use that needs access to DSW API.

Fig. 74: Form for creating an API Key.

After filling out the *API Key Name* and *Expiration*, an **API Key** is generated. In this step we have to copy the API Key, because after seeing it once, it is no longer possible to access it again.

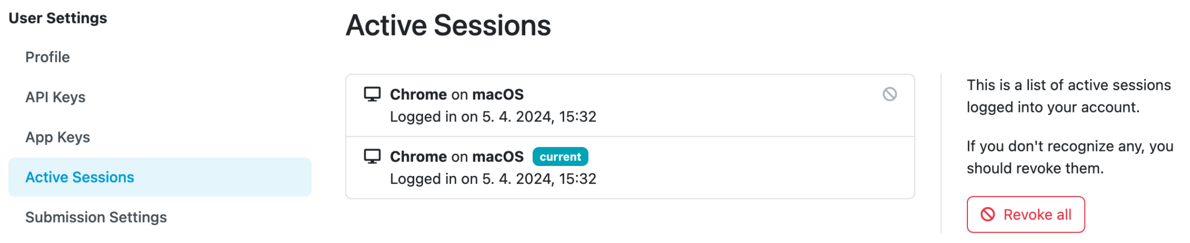
After we click on Done button, the new API Key is hidden and the information about this key is added to the table below, that contains all Active API Keys.

App Keys

When apps of the FAIR Wizard are communicating, they sometimes need to store each other App Keys. We can view them here.

Active Sessions

Here we can see list of active sessions that are logged into our account. In case we don't longer want any of them to have access to our, we can revoke the session.



Chrome on macOS
Logged in on 5. 4. 2024, 15:32

Chrome on macOS current
Logged in on 5. 4. 2024, 15:32

This is a list of active sessions logged into your account.

If you don't recognize any, you should revoke them.

Fig. 75: List of active sessions.

Submission Settings

If there are any Submission Settings that require input from the user, this input is requested here in Submissions Settings.

Change Language

A user can explicitly select a desired language after clicking *Change language* from the *Profile* menu. In case the language becomes unavailable later after the selection, it will fall back to the one marked as **default**.

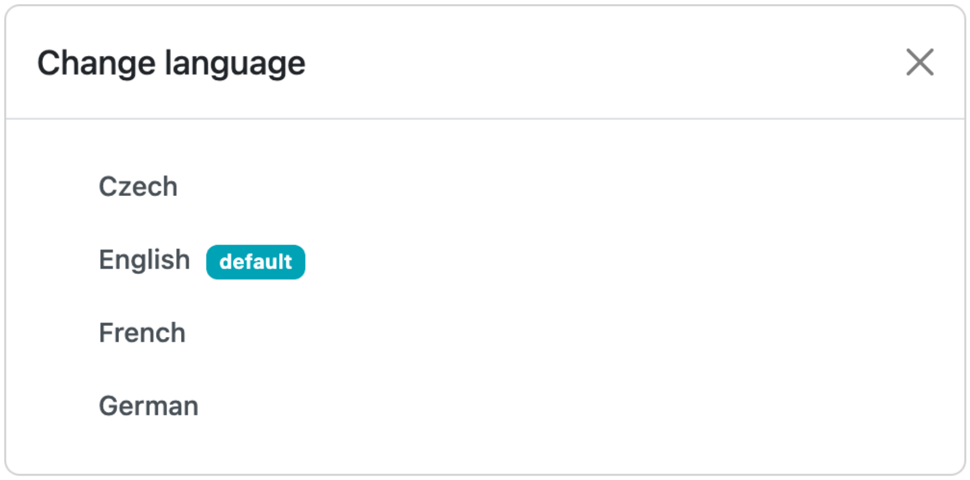


Fig. 76: Modal window with language selection.

Note: The selection of language is saved only locally in the browser (in local storage), so if we log in from different locations, we need to select the language there again.

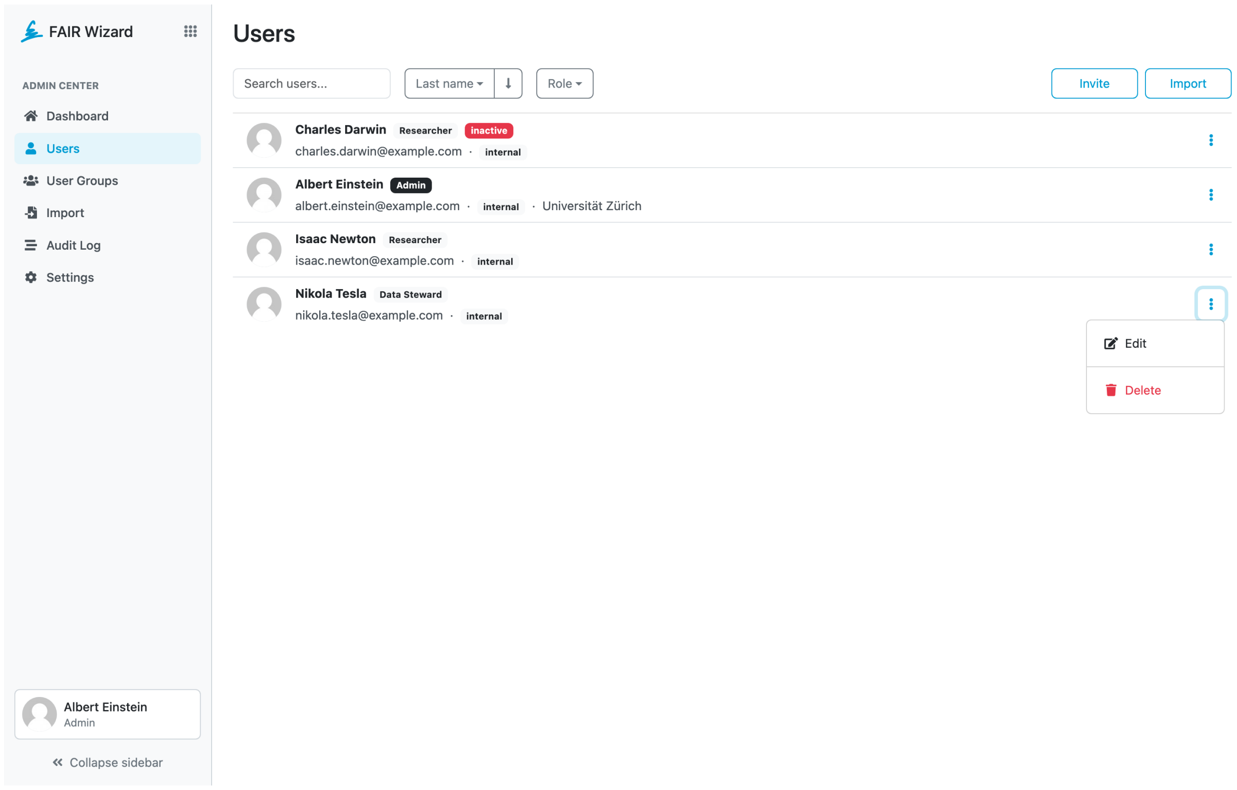
1.3 Admin Center

Admin Center is an application for management of the FAIR Wizard application and users within it. This application is mainly used for administration by **administrators**. It is used for users and user groups management, import of data, and settings management. User groups can also be managed by **data stewards**.

1.3.1 Users

Users list allows administrators to see and manage all users in the FAIR Wizard. The list can be filtered using role, searched using name or email fragment, and sorted via various properties of users. The list shows the role of a user next to its name and also indicates in case the user is inactive. Next to the email, we can quickly see what authentication services the user uses to log-in (*internal* is the internal authentication with email-password credentials, other are based on configured *OpenID services*).

A *user detail* can be opened by clicking the name of a user or by selecting *Edit* in the right dropdown menu for the desired row. There, a user can be also deleted via the *Delete* action. Finally, administrator can *invite a new user* by clicking *Invite*, or *import users* by clicking *Import*.



Charles Darwin Researcher inactive
charles.darwin@example.com · internal

Albert Einstein Admin
albert.einstein@example.com · internal · Universität Zürich

Isaac Newton Researcher
isaac.newton@example.com · internal

Nikola Tesla Data Steward
nikola.tesla@example.com · internal

Fig. 77: List of users.

Invite User

As administrators, we can create new users manually by clicking *Invite* on the *users list* and submitting the form. Each user must have a unique email address, first name and last name and assigned *role*. Optionally, a user can have affiliation specified.

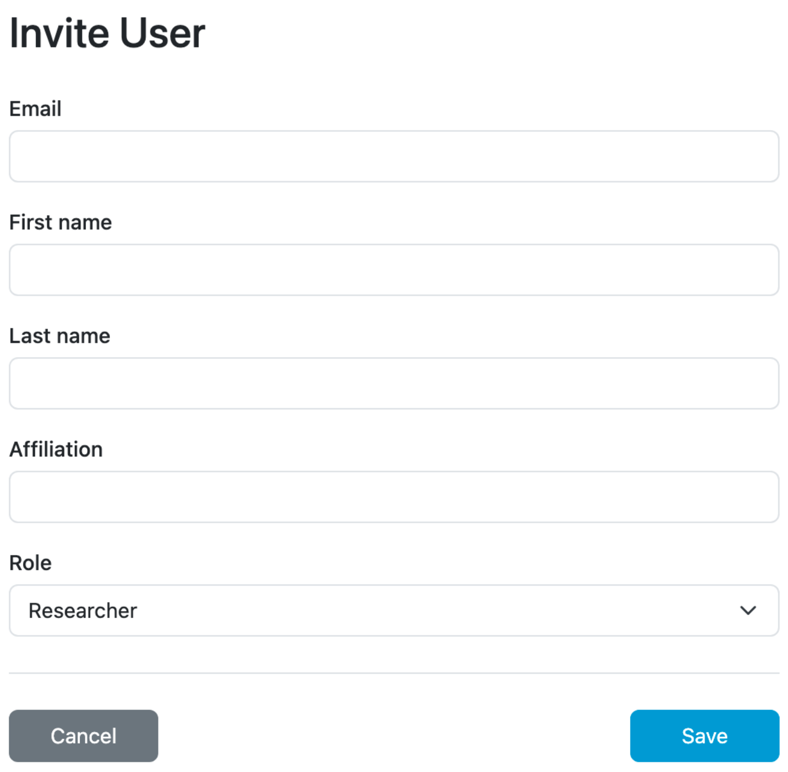


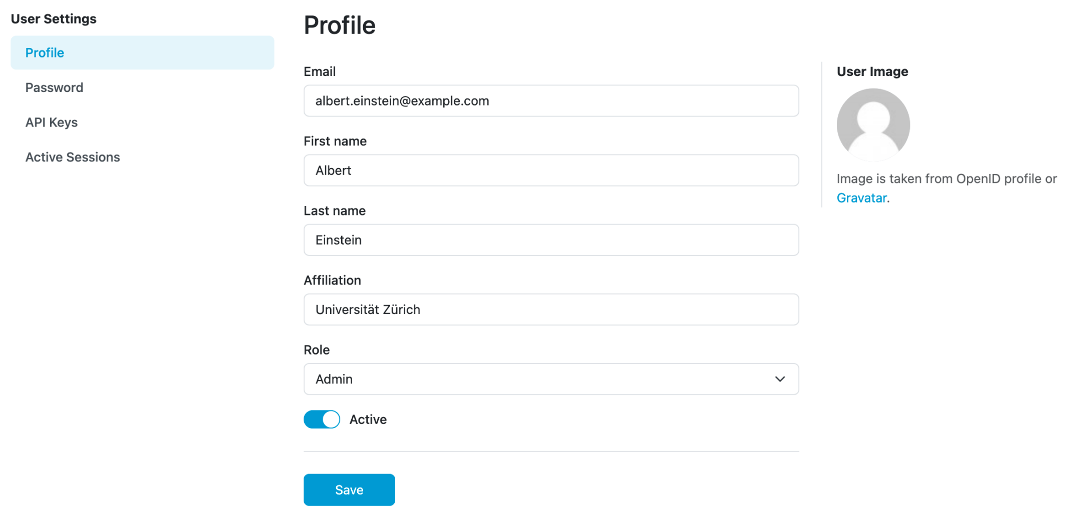
Fig. 78: Invite user.

Note: If the user is created by administrator, the user is activated by default and no email is sent to the user.

User Detail

As administrators, we can edit existing users manually on the detail (selected user from the *users list*). It is possible to change all properties of the user, including possibility to change whether the user account is active or inactive.

The password can be also changed (after selecting *Password* from the left navigation of user settings).



Profile

Email: albert.einstein@example.com

First name: Albert

Last name: Einstein

Affiliation: Universität Zürich

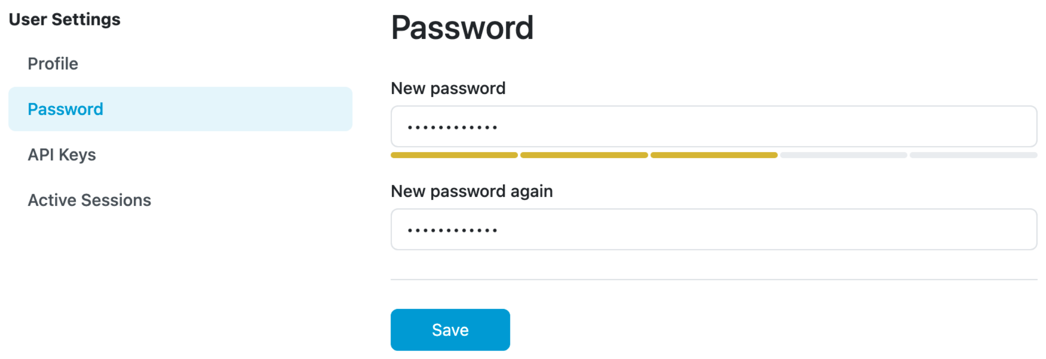
Role: Admin

Active

User Image

Image is taken from OpenID profile or Gravatar.

Fig. 79: Detail of a user profile.



Password

New password: ············

New password again: ············

Fig. 80: Form for changing password of a user.

User Roles

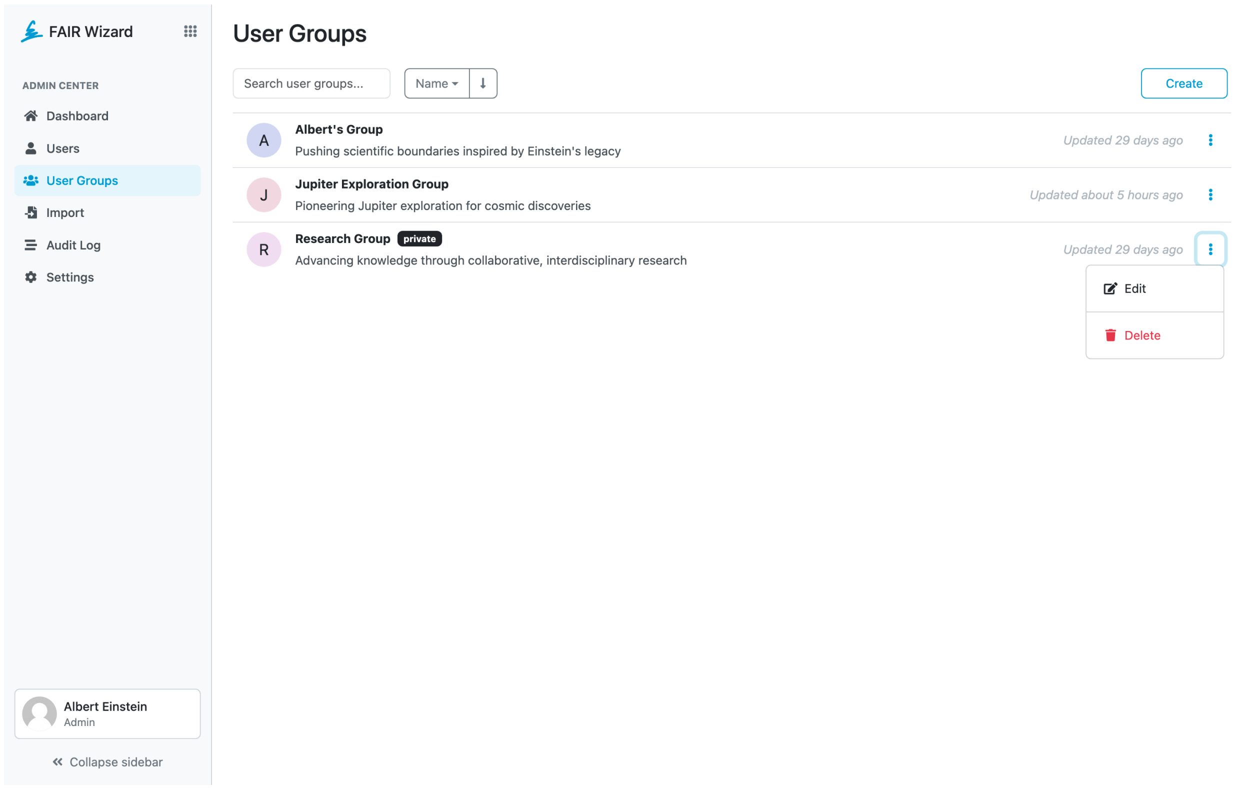
There are three user roles available: researchers, data steward, and admin. Permissions are associated with the roles, basically they affect what the users can do in the FAIR Wizard:

1.3.2 User Groups

As a data steward or an administrator, we can create a user group. The group can be used to add multiple people to a project at once. A group can be selected in *sharing settings* of a project.

User Groups list allows us to see and manage all user groups in the FAIR Wizard. The list can be searched based using name fragment, and sorted via properties of user groups, name and date created.

User Group Settings can be opened by clicking the name of a user group or by selecting *Edit* in the right dropdown menu for the desired row. There, a user group can be also deleted via the *Delete* action. Finally, a new user group can be created by clicking *Create*.



Albert's Group
Pushing scientific boundaries inspired by Einstein's legacy
Updated 29 days ago

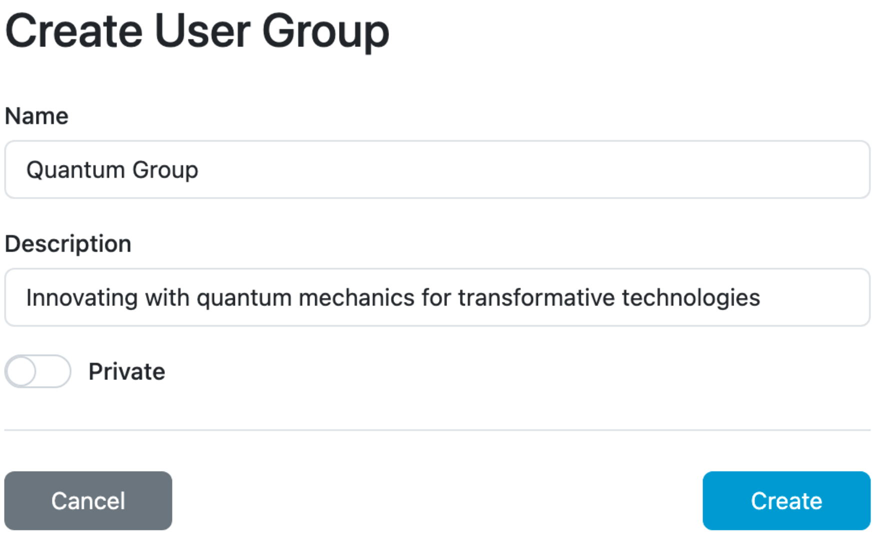
Jupiter Exploration Group
Pioneering Jupiter exploration for cosmic discoveries
Updated about 5 hours ago

Research Group private
Advancing knowledge through collaborative, interdisciplinary research
Updated 29 days ago

Fig. 81: User groups.

Create

As an administrator, we can create a new user group. For the user group we must set its name and optionally its description. We can also set the group as either a public or private group. Public groups are visible to all users, while private groups are only visible to the group members.


Create User Group

Name: Quantum Group

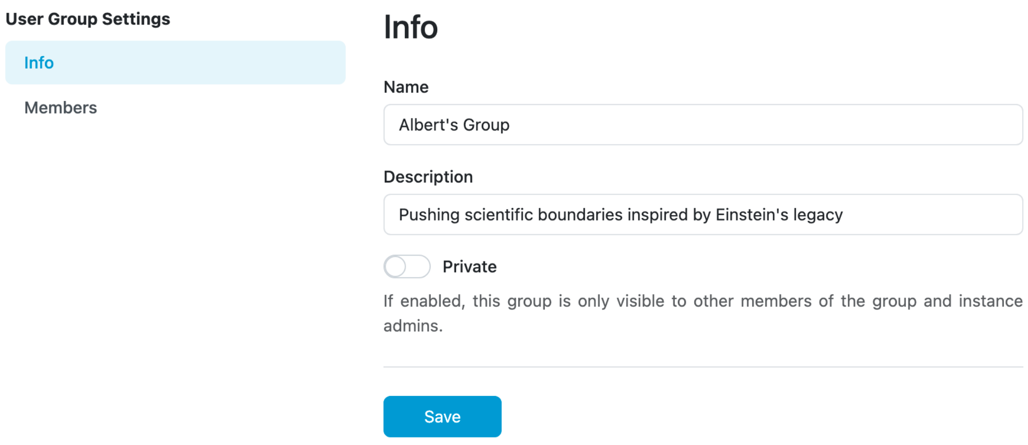
Description: Innovating with quantum mechanics for transformative technologies

Private

Fig. 82: Create user group.

User Group Detail

As an administrator, we can manage existing user group from User Group Settings. We can change name and description of a group and make it private. If a group is private, then only its members and application admins can see it.



Info

Name: Albert's Group

Description: Pushing scientific boundaries inspired by Einstein's legacy

Private

If enabled, this group is only visible to other members of the group and instance admins.

Fig. 83: User group detail.

We can also see list of all members of a group. Each member can be set as either Owner of a group or its member. Member can also be removed from the group by clicking on the x button. We can view users' profile by clicking his name.

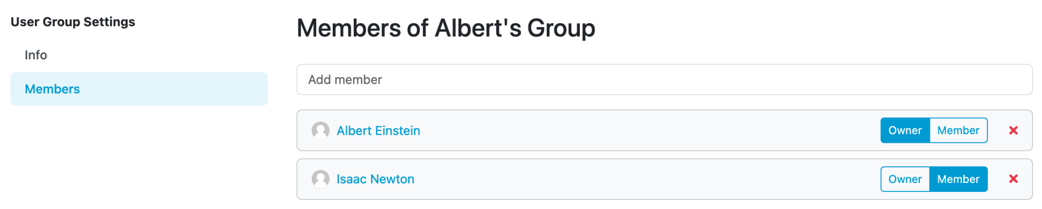


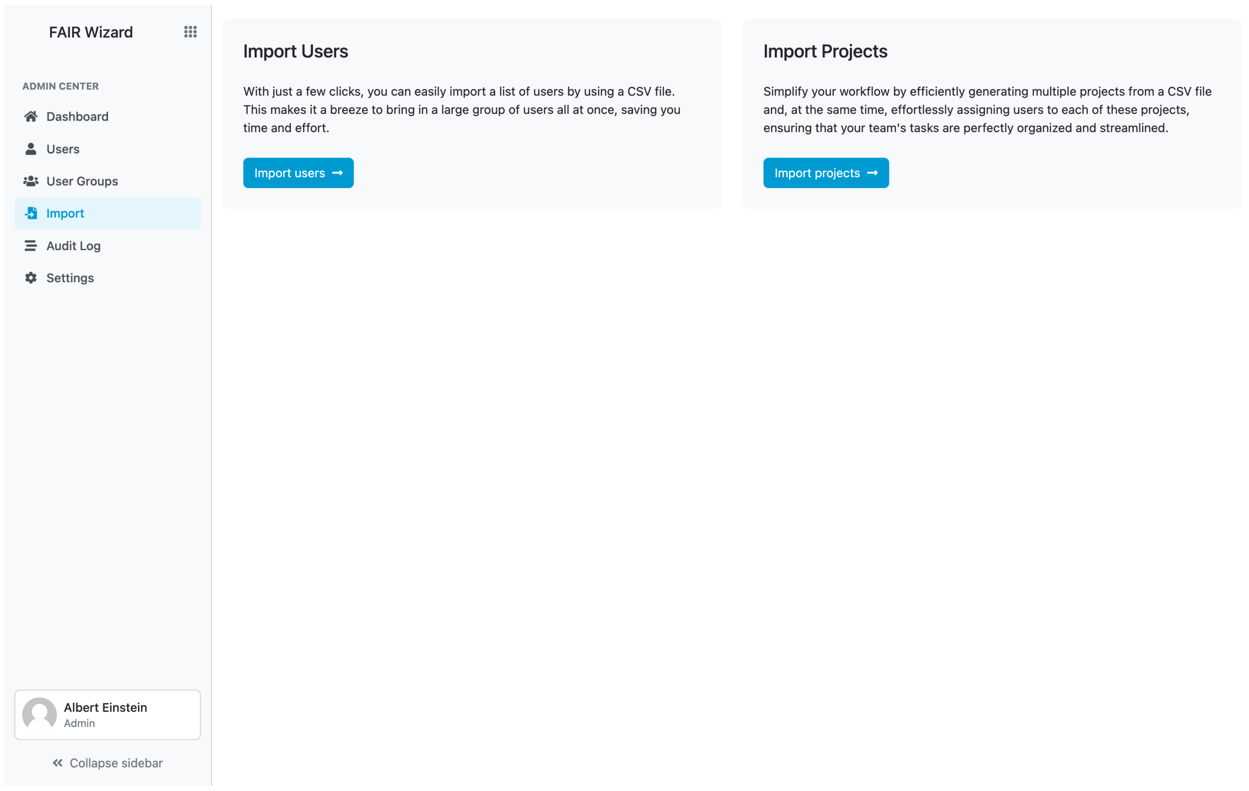
Fig. 84: User group members.

We can add new members to the group by picking them from dropdown menu.

Note: Don't forget to click on the Save button to save the changes.

1.3.3 Import

As an administrator, we can import users and projects using provided templates.



Import Users

With just a few clicks, you can easily import a list of users by using a CSV file. This makes it a breeze to bring in a large group of users all at once, saving you time and effort.

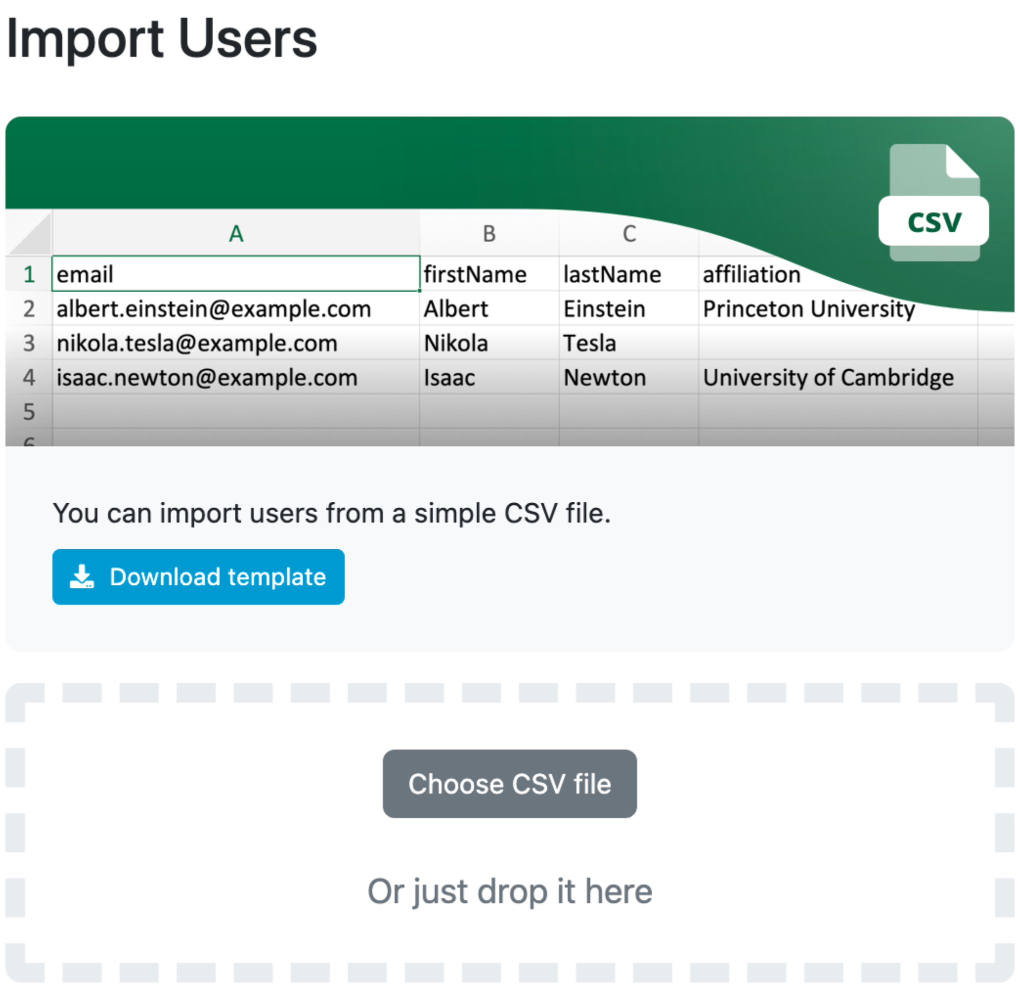
Import Projects

Simplify your workflow by efficiently generating multiple projects from a CSV file and, at the same time, effortlessly assigning users to each of these projects, ensuring that your team's tasks are perfectly organized and streamlined.

Fig. 85: Import options.

Users Import

As an administrator we can import users using provided template. The template has four columns: email, firstName, lastName and affiliation. Once it is filled with data, we can import it back to the FAIR Wizard to populate it with users.



Import Users

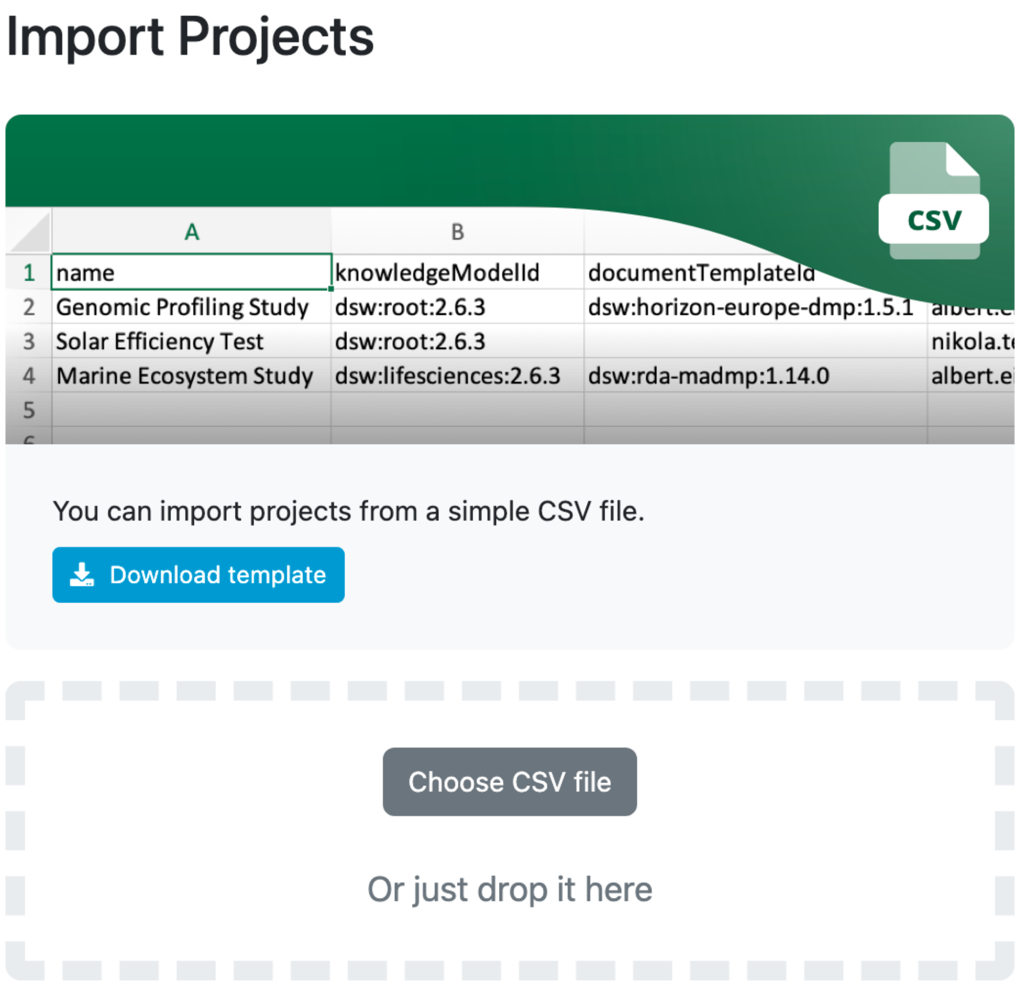
	A	B	C	
1	email	firstName	lastName	affiliation
2	albert.einstein@example.com	Albert	Einstein	Princeton University
3	nikola.tesla@example.com	Nikola	Tesla	
4	isaac.newton@example.com	Isaac	Newton	University of Cambridge
5				

You can import users from a simple CSV file.

Fig. 86: Import users.

Projects Import

As an administrator, we can import projects using provided template. The template has four columns: name, knowledgeModelId, documentTemplateId and emails of users we want to add as owners to a project. Once it is filled with data, we can import it back to the FAIR Wizard to populate it with projects.



Import Projects

	A	B	
1	name	knowledgeModelId	documentTemplateId
2	Genomic Profiling Study	dsw:root:2.6.3	dsw:horizon-europe-dmp:1.5.1
3	Solar Efficiency Test	dsw:root:2.6.3	
4	Marine Ecosystem Study	dsw:lifesciences:2.6.3	dsw:rda-madmp:1.14.0
5			

You can import projects from a simple CSV file.

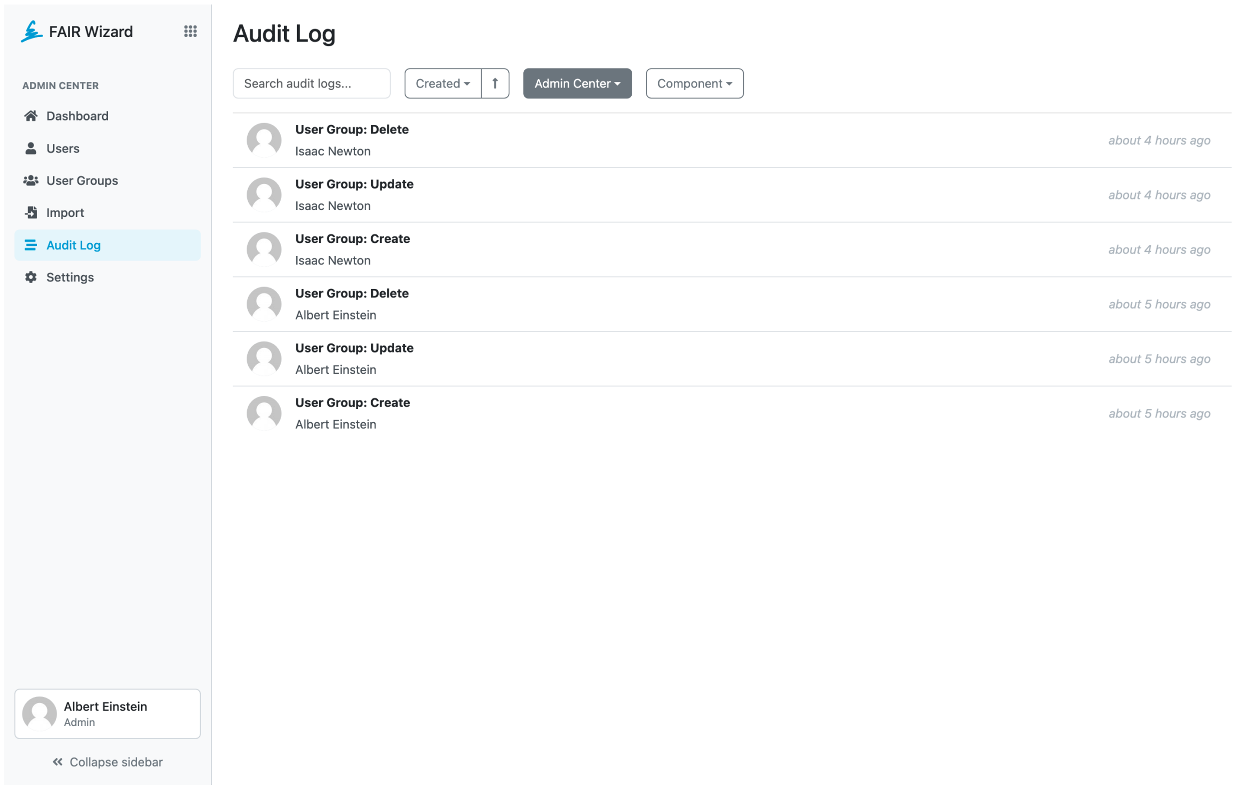
Fig. 87: Import projects.

1.3.4 Audit Log

Audit log is a feature that allows administrators to track everything that is happeningg in the FAIR Wizard.

Note: For now, Audit log is only available for actions done in the Admin Center. Other applications will be added in the future updates.

List can be searched based using content of audit logs. The dropdown menu can be used to select which app logs we want to display. Then we can also select which component we want to see. In the list we can see what particular action has been done, who did it and when it happened.



User Group: Delete
Isaac Newton
about 4 hours ago

User Group: Update
Isaac Newton
about 4 hours ago

User Group: Create
Isaac Newton
about 4 hours ago

User Group: Delete
Albert Einstein
about 5 hours ago

User Group: Update
Albert Einstein
about 5 hours ago

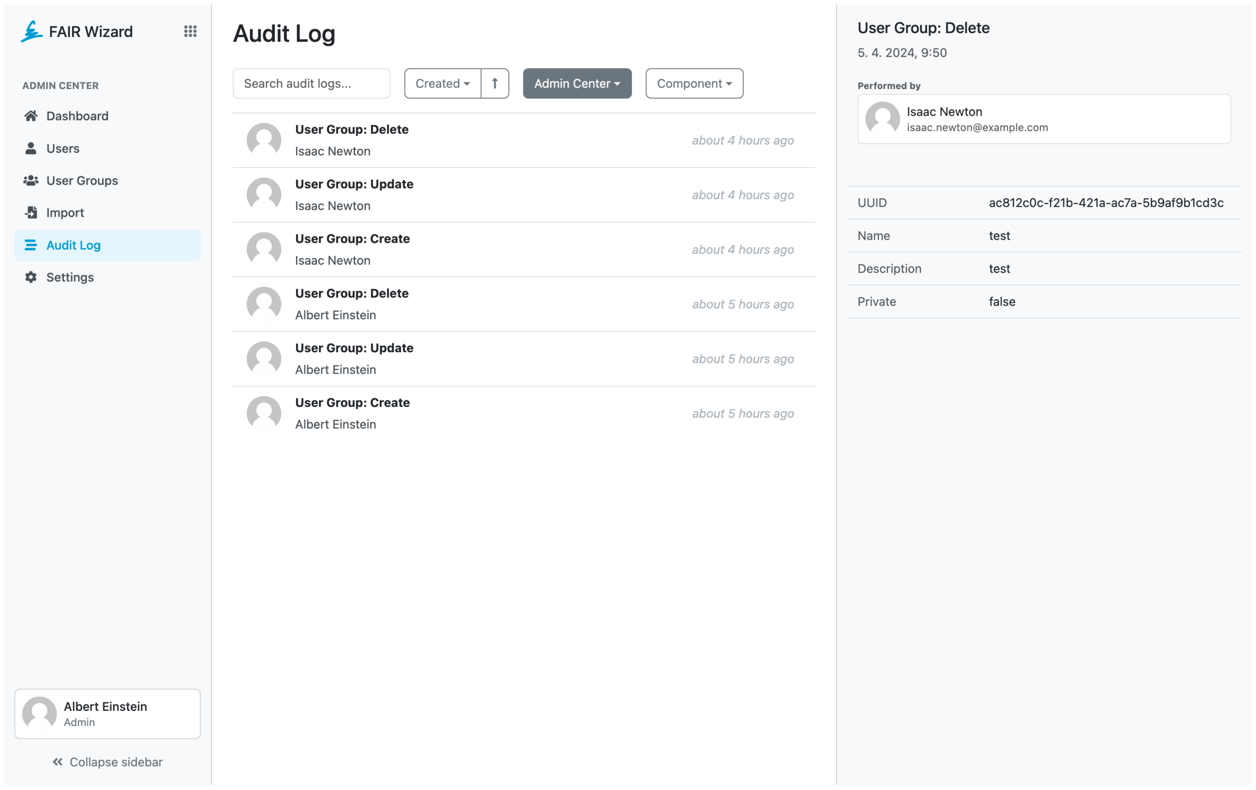
User Group: Create
Albert Einstein
about 5 hours ago

Fig. 88: Audit log.

A log item detail can be opened by clicking any log record. Based on which log record has been opened, detail contains information on what was changed and how. Some things, such as performer of the change can be clicked to open a detail of that user.

1.3.5 Settings

Settings section serves admins with managing the FAIR Wizard, more specificaly its Admin Center. By navigating via *Settings* item from the main menu, we can manage the things listed below.



User Group: Delete
Isaac Newton
about 4 hours ago

User Group: Update
Isaac Newton
about 4 hours ago

User Group: Create
Isaac Newton
about 4 hours ago

User Group: Delete
Albert Einstein
about 5 hours ago

User Group: Update
Albert Einstein
about 5 hours ago

User Group: Create
Albert Einstein
about 5 hours ago

User Group: Delete
5. 4. 2024, 9:50

Performed by
Isaac Newton
isaac.newton@example.com

UUID	ac812c0c-f21b-421a-ac7a-5b9af9b1cd3c
Name	test
Description	test
Private	false

Fig. 89: Audit log detail.

System Settings

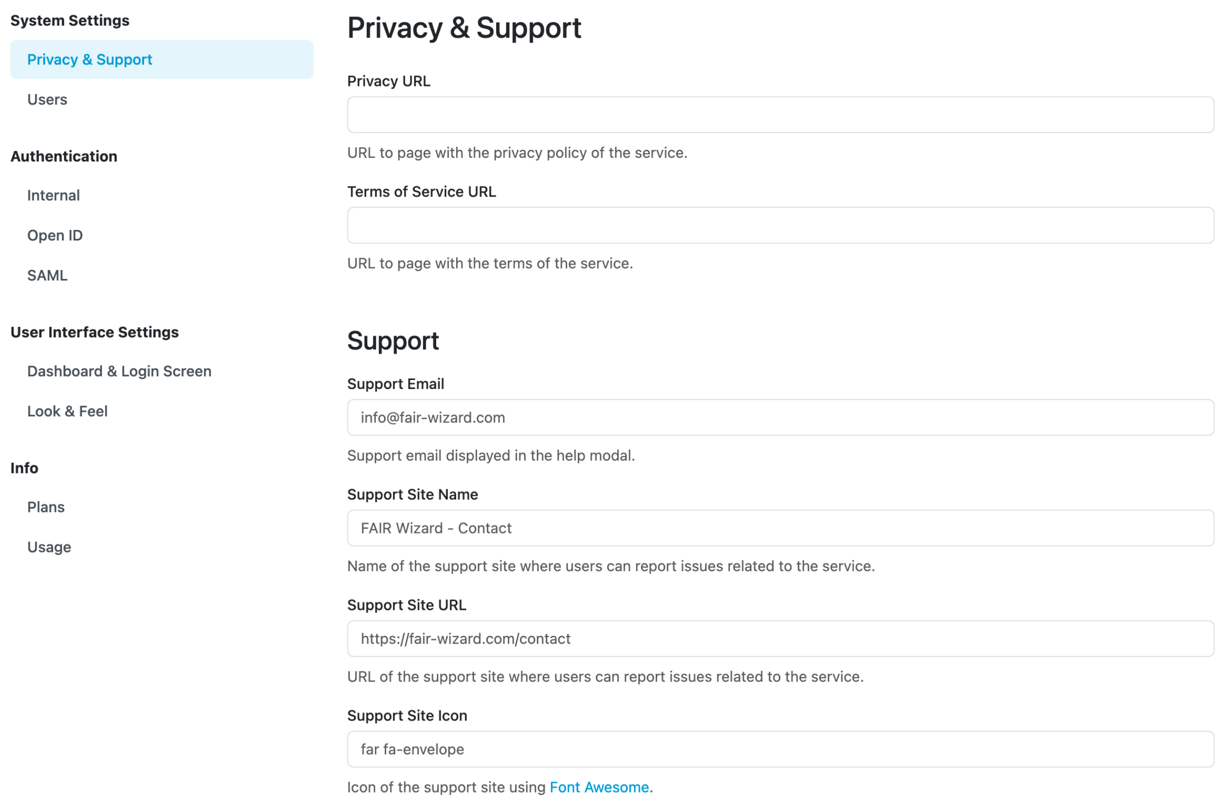
System settings allow us to configure Privacy and Support settings of FAIR Wizard. There is also Users settings.

Privacy & Support Settings

To request users to agree with **Privacy Policy** and/or **Terms of Service** documents, we can configure URLs to locate such documents. Then, when new users register to the service or login for the first time, they will be prompted to agree with the linked documents. Note that usually you should inform already registered users in case you change such documents.

Support

These settings also allow us to configure **Support Email** that users can use to request help or report issues. Similarly **Support Site Name** and **Support Site URL** can be used in case we want users to create tickets in issue tracker of some repository, e.g., on GitHub. **Support Site Icon** can also be configurated, using Font Awesome. These support links together with an icon are then shown in *Report issue* modal window.



Privacy & Support

Privacy URL

URL to page with the privacy policy of the service.

Terms of Service URL

URL to page with the terms of the service.

Support

Support Email

info@fair-wizard.com

Support email displayed in the help modal.

Support Site Name

FAIR Wizard - Contact

Name of the support site where users can report issues related to the service.

Support Site URL

https://fair-wizard.com/contact

URL of the support site where users can report issues related to the service.

Support Site Icon

far fa-envelope

Icon of the support site using Font Awesome.

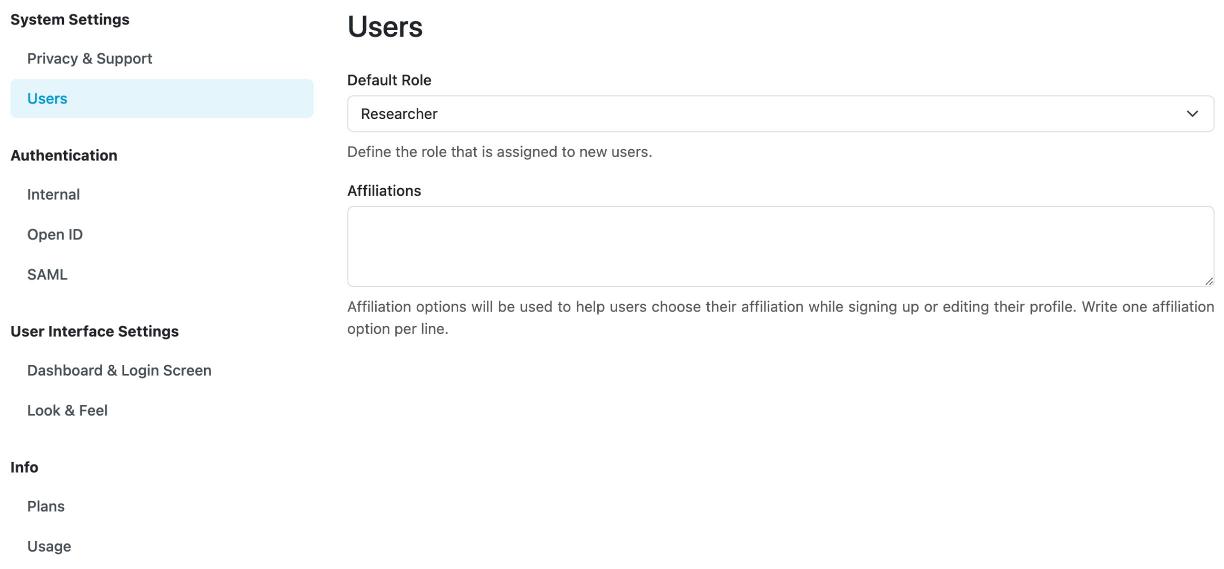
Fig. 90: Privacy and Support settings.

Users Settings

Users Settings allow us to configure default role for new users. It is also possible to set up a list of affiliations that will be suggested to users while they register or update their profiles.

Warning: It is recommended to set up the default role for new users with lowest possible rights, **Researcher**. This way, users will not have access to change any settings.

Another part of settings on this page are **Affiliations**. It servers to pre-define a list of possible affiliations (on per line) that will be suggested to users while they register or update their profiles.



Default Role: Researcher

Define the role that is assigned to new users.

Affiliations

Affiliation options will be used to help users choose their affiliation while signing up or editing their profile. Write one affiliation option per line.

Fig. 91: Users settings.

Authentication Settings

The Authentication Settings page allows you to configure the authentication settings for users. You can configure the following settings:

- **Internal**: This is the default authentication method. You can configure the settings for the internal authentication method.
- **Open ID**: You can configure specific Open ID authentications.
- **SAML**: You can configure specific SAML authentications. However SAML configuration requires certain technical knowledge and cannot be done only in the FAIR Wizard.

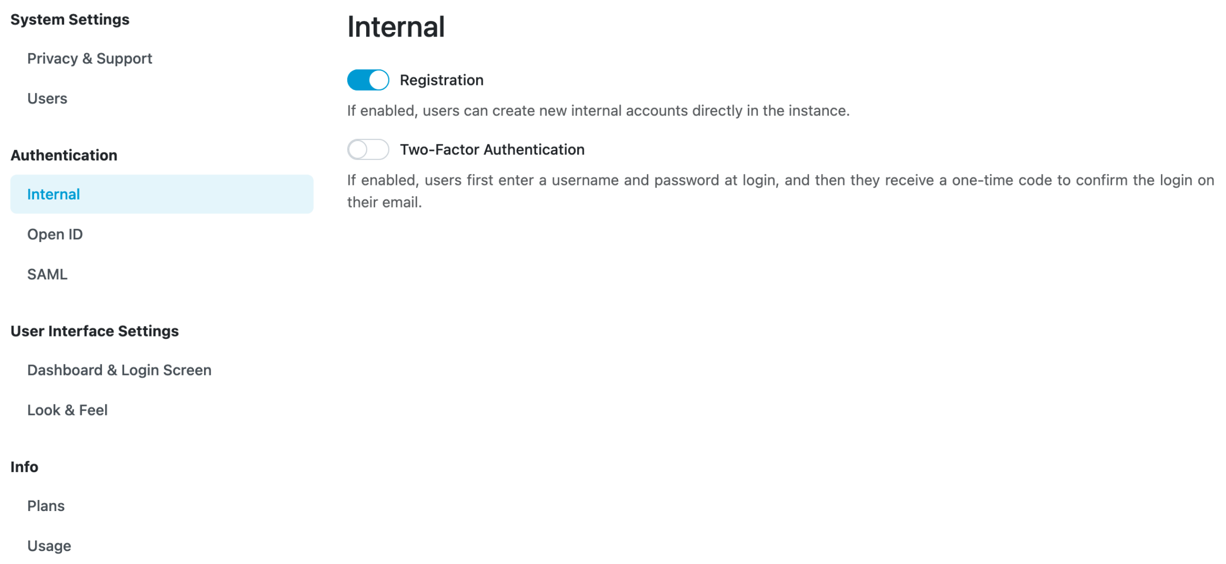
The FAIR Wizard also supports Shibboleth and eduGAIN.

Note: For configuration of SAML, Shibboleth or eduGAIN please contact the FAIR Wizard team.

Internal Settings

Internal Settings allow us to enable or disable registration. Once turned off, users can only register using other ways, e.g. OpenID or SAML.

We can also enable or disable Two-Factor Authentication (2FA) for users. If enabled, users will be required to enter a one-time code sent to their email address after entering their password.



Internal

Registration

If enabled, users can create new internal accounts directly in the instance.

Two-Factor Authentication

If enabled, users first enter a username and password at login, and then they receive a one-time code to confirm the login on their email.

Fig. 92: Internal Settings.

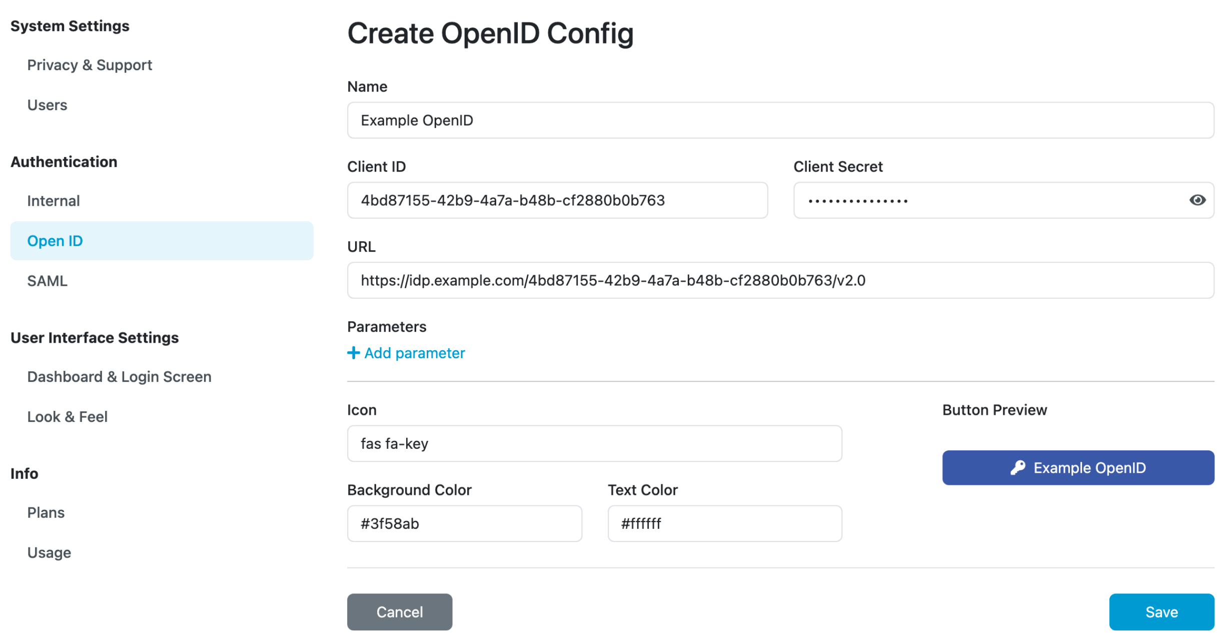
Open ID Settings

Using these settings we can add Open ID configuration to allow logging into the FAIR Wizard instance via external identity provider. First, press *Create* button and fill **Name** of the service (use only lowercase alphanumeric characters or dash symbols). Then, we should prepare the client application on the side of OpenID service:

- Use **Callback URL** (and optionally **Logout URL**) to create the client
- Obtain **Client ID** and **Client Secret**
- Obtain OpenID endpoint **URL** (we may get one ending with `/.well-known/openid-configuration`, if so we just use the part before this suffix)
- Configure the client to have the following claims: `openid`, `profile`, `email`
- Configure the client to provide the following details in ID tokens: `email`, `given_name`, `family_name`

Back in the FAIR Wizard settings, we can fill **Client ID**, **Client Secret**, and **URL** from our OpenID client together with optional **Parameters** (usually not needed). Finally, we can configure how the log-in button will look like by setting **Icon** (by using Font Awesome), **Name**, **Background**, and text/icon **Color**.

Note: After setting a new OpenID service, we should directly test it and verify that the configuration works well. For that, we can simply open our FAIR Wizard instance in a new anonymous window of the web browser.



Name: Example OpenID

Client ID: 4bd87155-42b9-4a7a-b48b-cf2880b0b763

Client Secret: •••••••••••••••

URL: https://idp.example.com/4bd87155-42b9-4a7a-b48b-cf2880b0b763/v2.0

Parameters

Icon: fas fa-key

Background Color: #3f58ab

Text Color: #ffffff

Fig. 93: Example configuration of OpenID service.

SAML Settings

SAML is another option for authentication. However setting up SAML is not trivial and requires certain level of technical knowledge. Please contact the FAIR Wizard team team to assist you with SAML configuration.

Based on SAML, FAIR Wizard supports authentication using Shibboleth and eduGAIN.

Note: For configuration of SAML, Shibboleth or eduGAIN please contact the FAIR Wizard team.

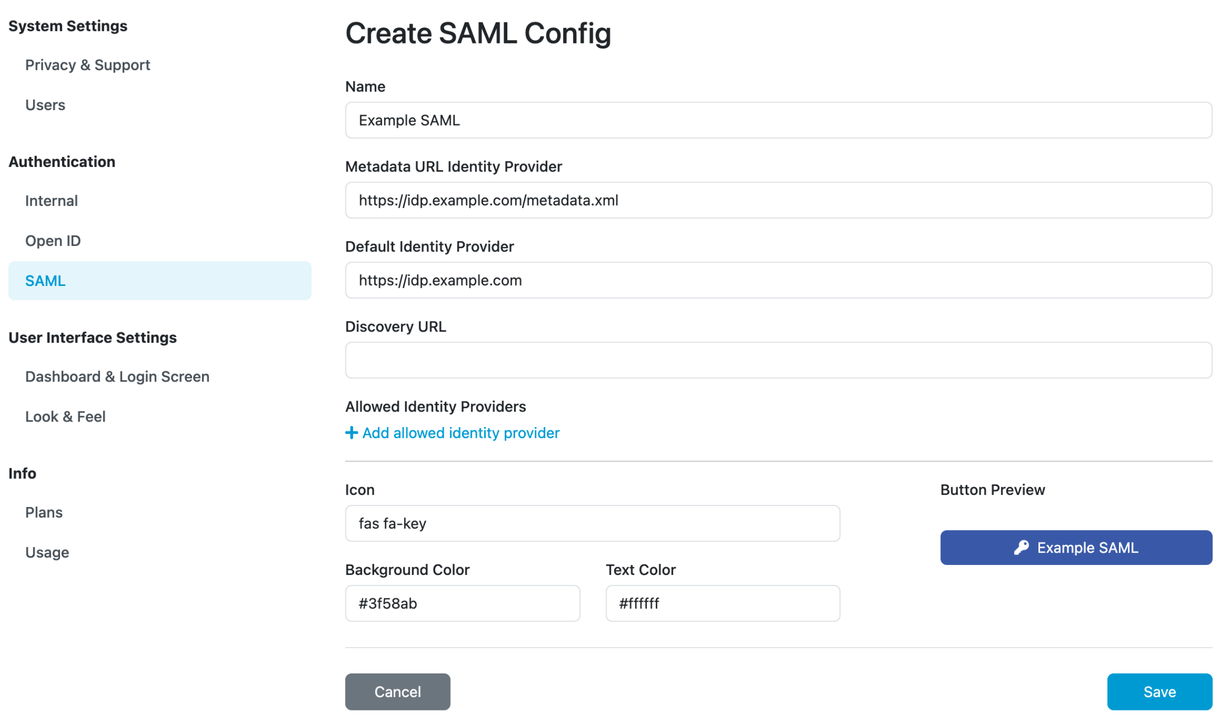
User Interface Settings

User Interface Settings allow us to customize login screen info and various announcements. It also allows us to change FAIR Wizard title, logo and colors. Lastly we can add custom menu links.

Dashboard and Login Screen Settings

Login Info

It is possible to write a message that users will see before logging in the FAIR Wizard, using HTML or Markdown. The Login info is placed in the center of the login screen. We can use it to explain users in what cases they can/should use our FAIR Wizard, how they should log in (e.g. if we have more *authentication services* configured), or if there is any news regarding our FAIR Wizard.



Name: Example SAML
Metadata URL Identity Provider: https://idp.example.com/metadata.xml
Default Identity Provider: https://idp.example.com
Discovery URL:
Allowed Identity Providers
Icon: fas fa-key
Background Color: #3f58ab
Text Color: #ffffff

Fig. 94: Example configuration of SAML service.

Warning: Defining HTML classes in the login info can overwrite FAIR Wizard application classes. It is recommended to use prefixes for classes, if they are used, to avoid conflicts.

Sidebar Login Info

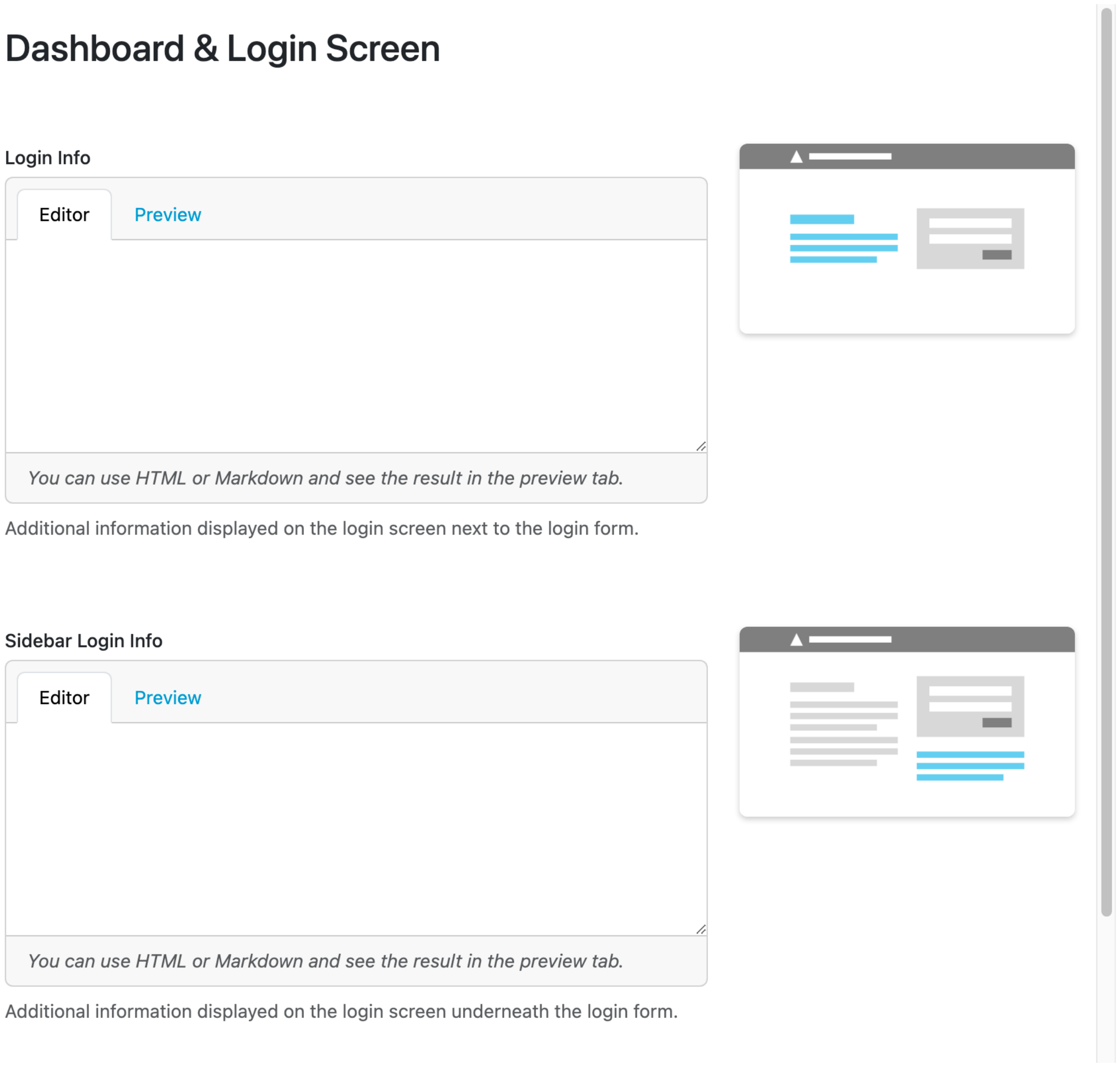
It is also possible to write another message that users will see on the login screen. The Sidebar Login info is placed underneath the login form. We can also use HTML or Markdown as in the Login Info.

Warning: Defining HTML classes in the login info can overwrite FAIR Wizard application classes. It is recommended to use prefixes for classes, if they are used, to avoid conflicts.

Announcements

Another option to adjust the dashboard and/or the login screen is to add Announcements. Announcements are displayed above the main content in the login screen. In dashboard, they are also displayed above the main content for both **welcome** and **role-based** dashboard style. There are three levels of Announcements:

- **Info** - light blue color for sending informations to the users.
- **Warning** - yellow to warn the users about something.
- **Critical** - red to signalize the Announcement is critical and it needs attention.



Dashboard & Login Screen

Login Info

You can use HTML or Markdown and see the result in the preview tab.

Additional information displayed on the login screen next to the login form.

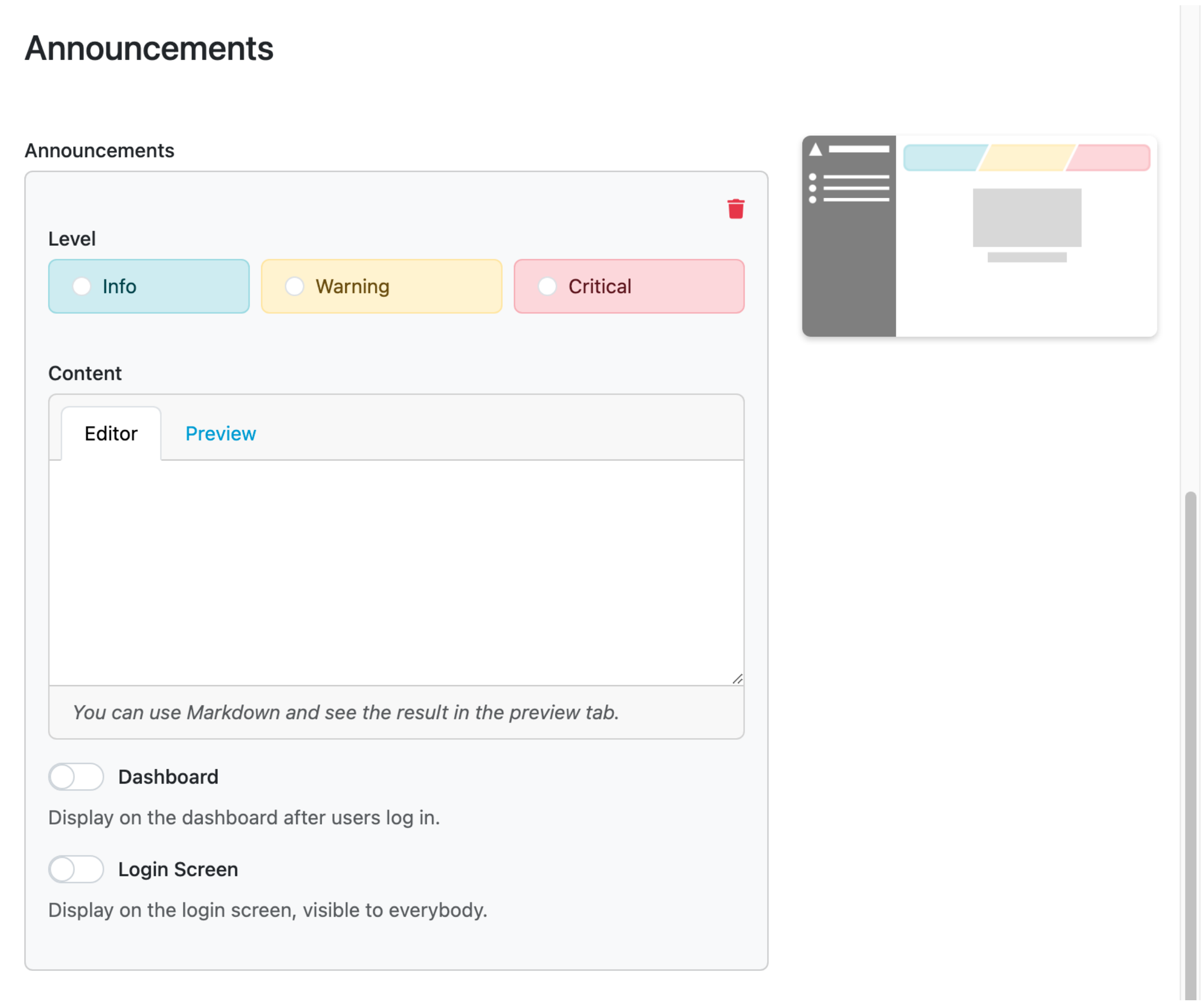
Sidebar Login Info

You can use HTML or Markdown and see the result in the preview tab.

Additional information displayed on the login screen underneath the login form.

Fig. 95: Privacy and Support settings.

The content of the Announcement can be edited using Markdown. There are also two additional switches which determine, where the Announcement is displayed. The Announcement can be set up to be displayed either on the dashboard after users log in or on the login screen before the users log in. It is also possible to display the same Announcement in both places. Number of Announcements is not limited.


Announcements

Announcements

Level

Info | Warning | Critical

Content

Editor | Preview

You can use Markdown and see the result in the preview tab.

Dashboard

Display on the dashboard after users log in.

Login Screen

Display on the login screen, visible to everybody.

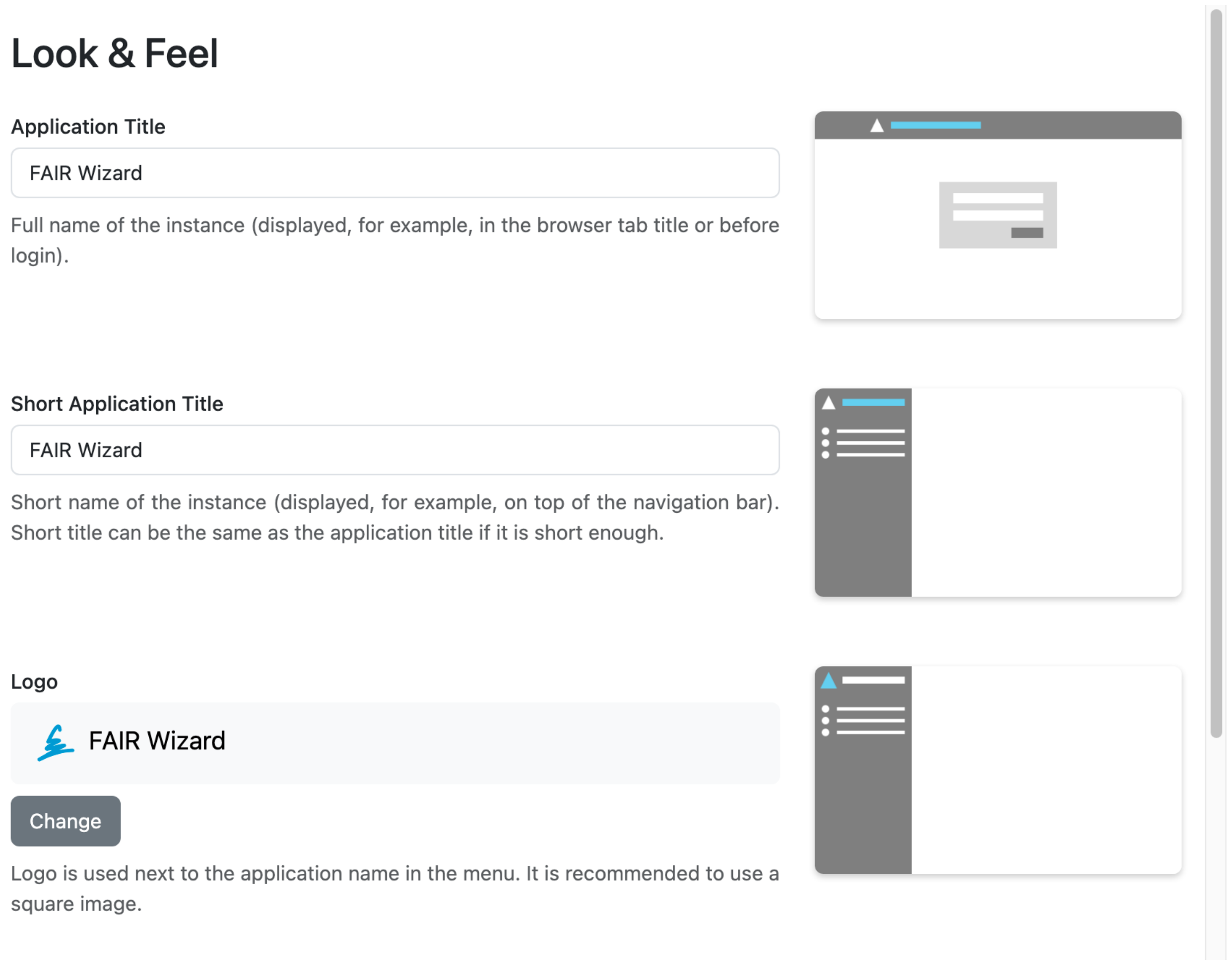
Fig. 96: Announcements settings.

Look and Feel Settings

Look and Feel Settings allow us to configure **Application Title** and **Short Application Title**. They are used together with **Logo** in the FAIR Wizard and also in various email templates (invitation to a project, forgotten password, . . .).

Primary Color and **Illustrations Color** can be used to customize the look of the FAIR Wizard, for example by using company colors.

Lastly we can easily add custom links to the main (left) menu by clicking *Add link* under **Custom Menu Links**. For each link, we can set **Icon** (from Font Awesome), **Title** and the target **URL**. We can also set whether the link should open in **New Window** (if not, it will nagivate user directly in the same window/tab from FAIR Wizard). Once the links are there, we can manage them or delete them at this place.



Look & Feel

Application Title

FAIR Wizard

Full name of the instance (displayed, for example, in the browser tab title or before login).

Short Application Title

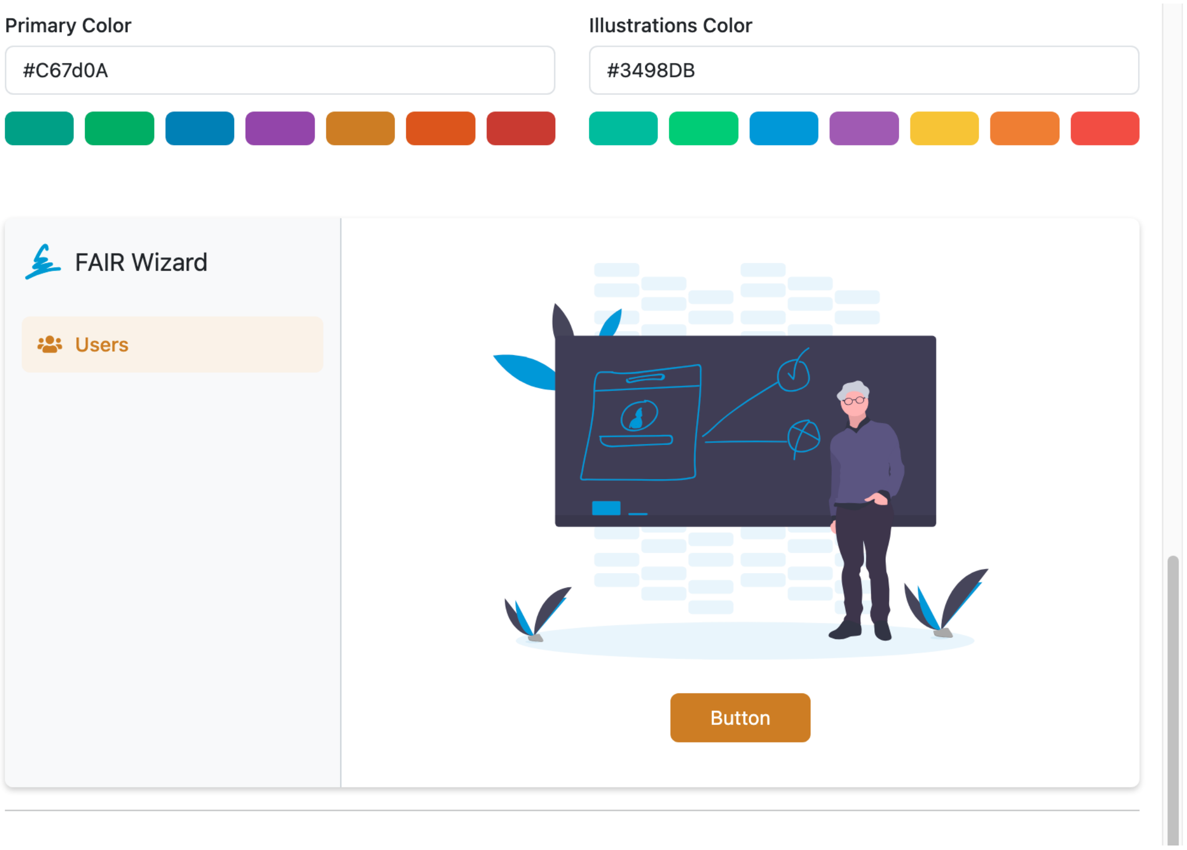
FAIR Wizard

Short name of the instance (displayed, for example, on top of the navigation bar). Short title can be the same as the application title if it is short enough.

Logo

Logo is used next to the application name in the menu. It is recommended to use a square image.

Fig. 97: Titles and logo settings.



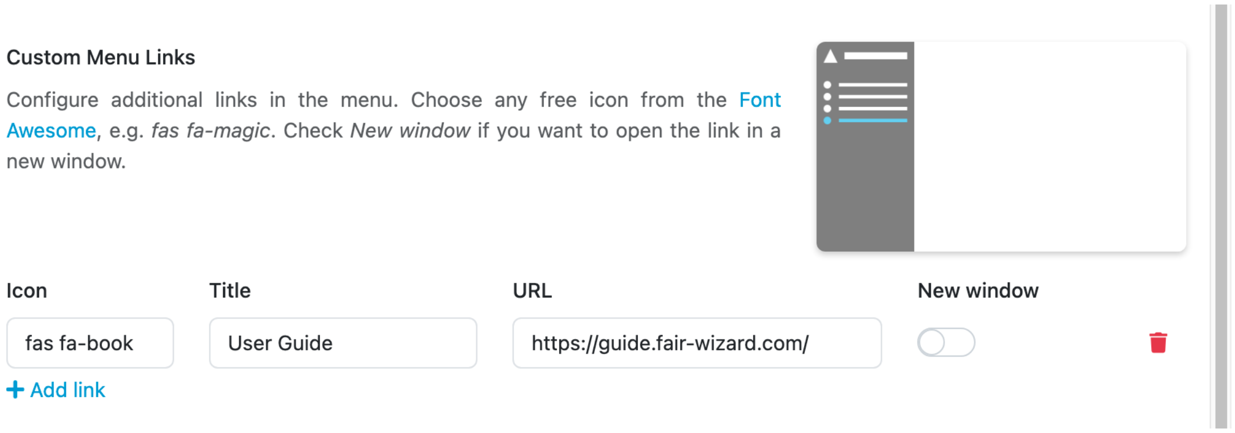
Primary Color

#C67d0A

Illustrations Color

#3498DB

Fig. 98: Colors settings.



Custom Menu Links

Configure additional links in the menu. Choose any free icon from the Font Awesome, e.g. *fas fa-magic*. Check *New window* if you want to open the link in a new window.

Icon	Title	URL	New window
fas fa-book	User Guide	https://guide.fair-wizard.com/	

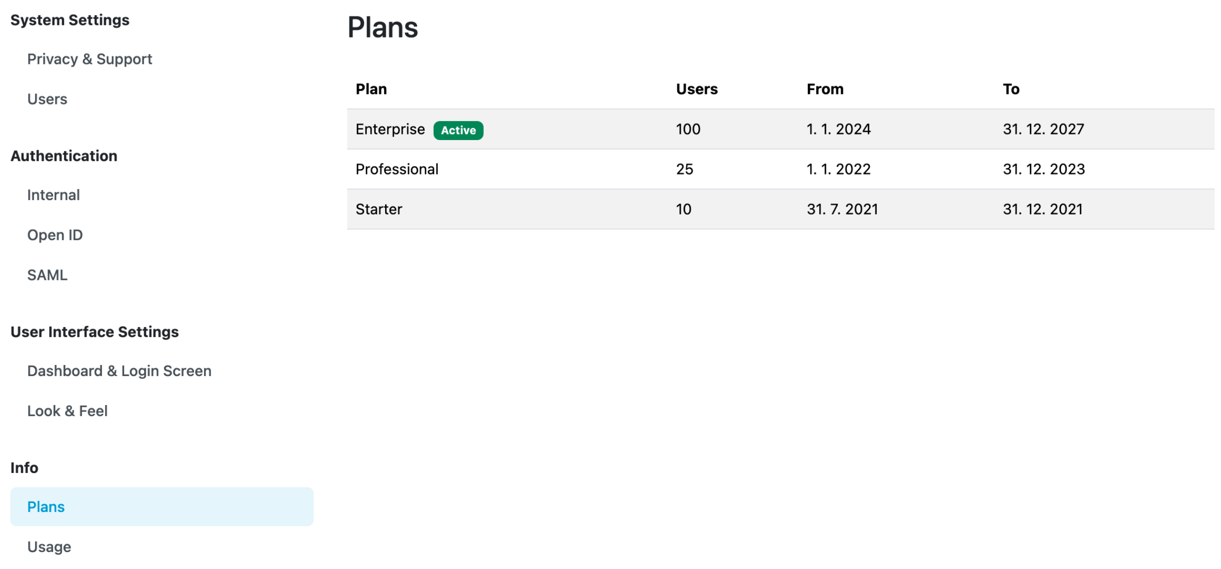
Fig. 99: Example configuration of a custom menu link.

Info

The info section of settings simply serves to let us check various information and statistics about the FAIR Wizard instance.

Plans

Plans provide information about which plan we are currently subscribed to. Each plan specifies the number of users allowed to access the FAIR Wizard, as well as the start and end dates of the plan's activation. Past plans are also displayed here.



Plans

Plan	Users	From	To
Enterprise Active	100	1. 1. 2024	31. 12. 2027
Professional	25	1. 1. 2022	31. 12. 2023
Starter	10	31. 7. 2021	31. 12. 2021

Fig. 100: Plans.

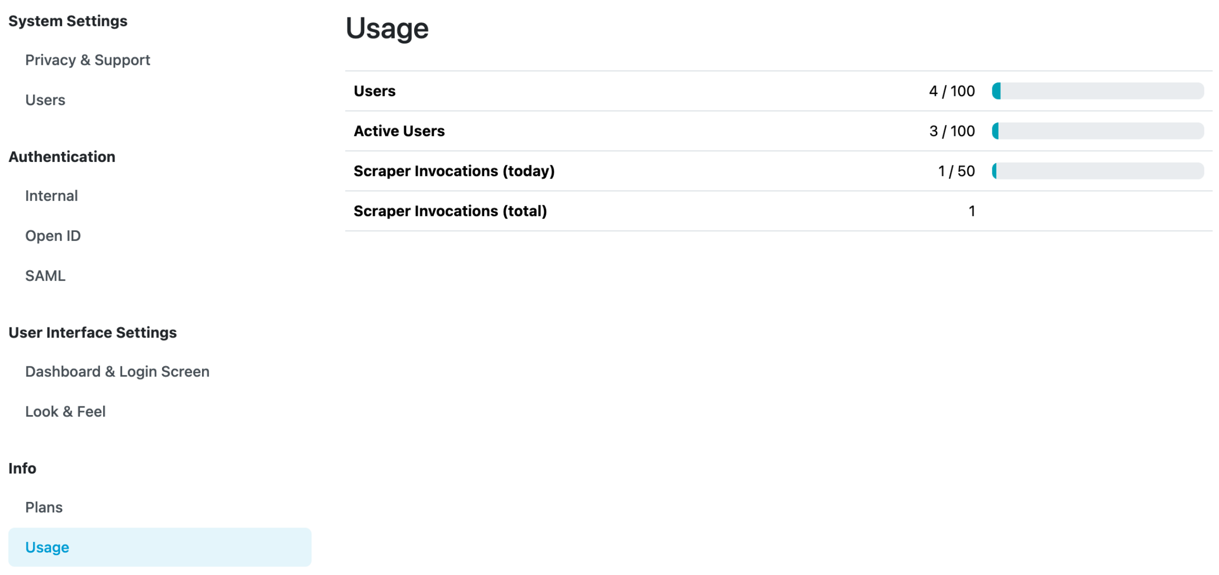
Usage

Usage allows us quickly see numbers of users, active users in the FAIR Wizard such as number of available scraper invocations for reporting. We cannot perform any actions on this page.

1.3.6 Profile

As every logged-in user, we can manage our own profile. We can see the options by hovering over the box with our name and avatar in the lower part of the left sidebar with main menu (only avatar is shown in case of collapsed sidebar). From there, we can navigate to *Edit Profile*.

Moreover, there are also options to *Log out*, see basic information about the FAIR Wizard using *About*, or in case of problems we can use *Report issue* to know how to proceed.



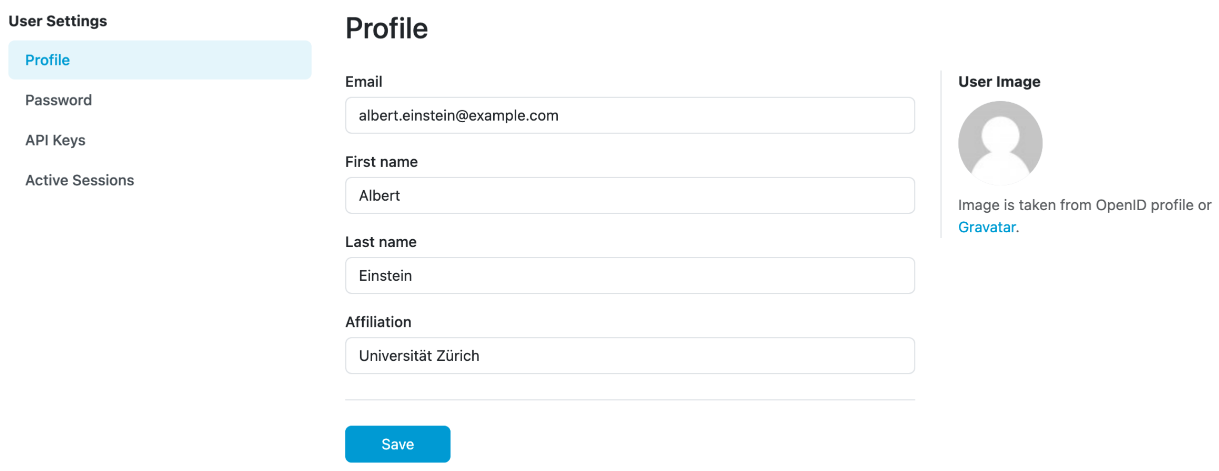
Usage

Users	4 / 100
Active Users	3 / 100
Scraper Invocations (today)	1 / 50
Scraper Invocations (total)	1

Fig. 101: Usage statistics.

Edit Profile

After navigating to *Edit profile* from the *Profile* menu, we open the User Settings. We can edit our **Email** address, **First name**, **Last name**, and **Affiliation**. We need to submit the changes by clicking *Save* button.



Profile

Email	albert.einstein@example.com
First name	Albert
Last name	Einstein
Affiliation	Universität Zürich

User Image

Image is taken from OpenID profile or Gravatar.

Fig. 102: Form for editting profile.

If we want to *Change Password*, we need to switch to *Password* from the left menu titled **User settings**.

Change Password

The password be can changed after navigating using *Password* from the *Edit Profile*. Here we can simply enter new password (it must be strong enough), repeat it again and press *Save* button.

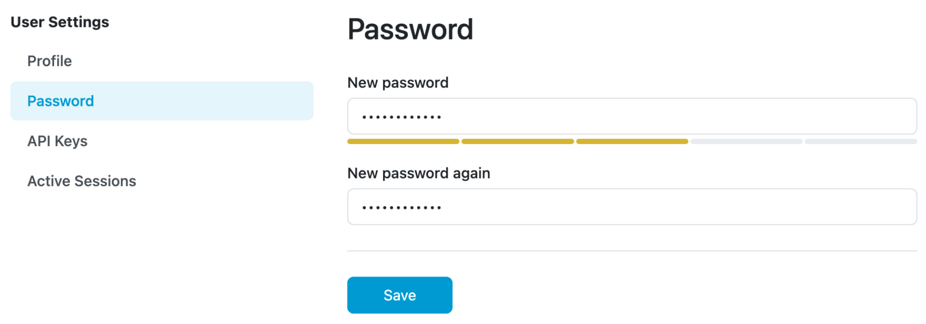
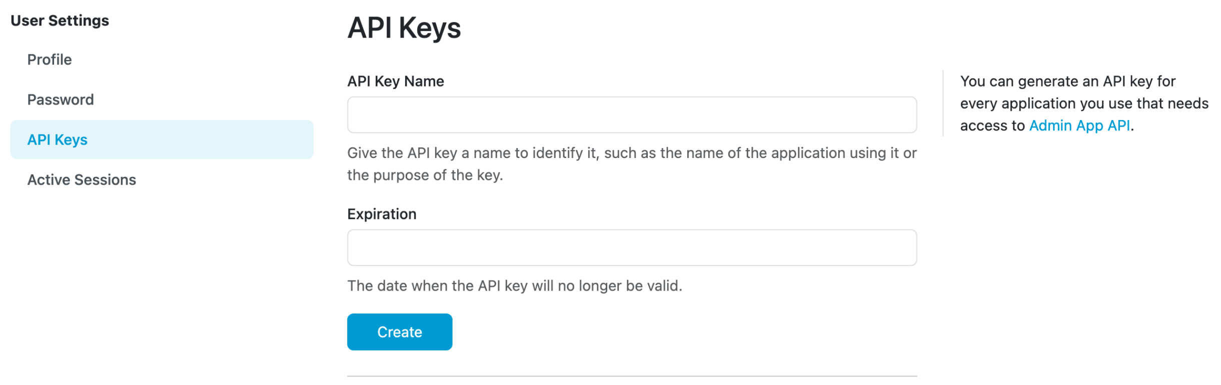


Fig. 103: Form for changing password.

API Keys

When we want to access the FAIR Wizard through FAIR Wizard API we have to set up an API Key. The API Key has an *API Key Name* so we can remember for what purpose this Key is used and *Expiration* which is a date from when the API Key will no longer be valid.



Give the API key a name to identify it, such as the name of the application using it or the purpose of the key.

The date when the API key will no longer be valid.

You can generate an API key for every application you use that needs access to Admin App API.

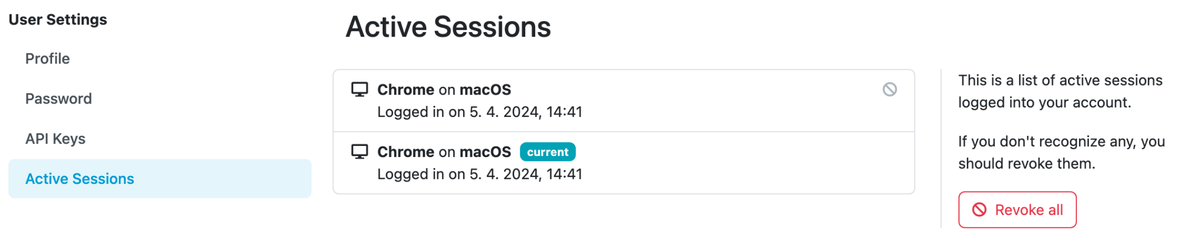
Fig. 104: Form for creating an API Key.

After filling out the *API Key Name* and *Expiration*, an **API Key** is generated. In this step we have to copy the API Key, because after seeing it once, it is no longer possible to access it again.

After we click on Done button, the new API Key is hidden and the information about this key is added to the table below, that contains all Active API Keys.

Active Sessions

Here we can see list of active sessions that are logged into our account. In case we don't longer want any of them to have access to our, we can revoke the session.



Active Sessions

Chrome on macOS
Logged in on 5. 4. 2024, 14:41

Chrome on macOS
Logged in on 5. 4. 2024, 14:41

This is a list of active sessions logged into your account.

If you don't recognize any, you should revoke them.

Fig. 105: List of active sessions.

1.4 Integration Hub

Integration Hub is an application for management of integrations. For now, it offers creation of Value integration.

1.4.1 Value Integrations

As an administrator, we can use value integration to setup an integration to be used by Data Stewards in their Knowledge Models as source of data.

We can start creating a new value integration by clicking on the *Create* button.

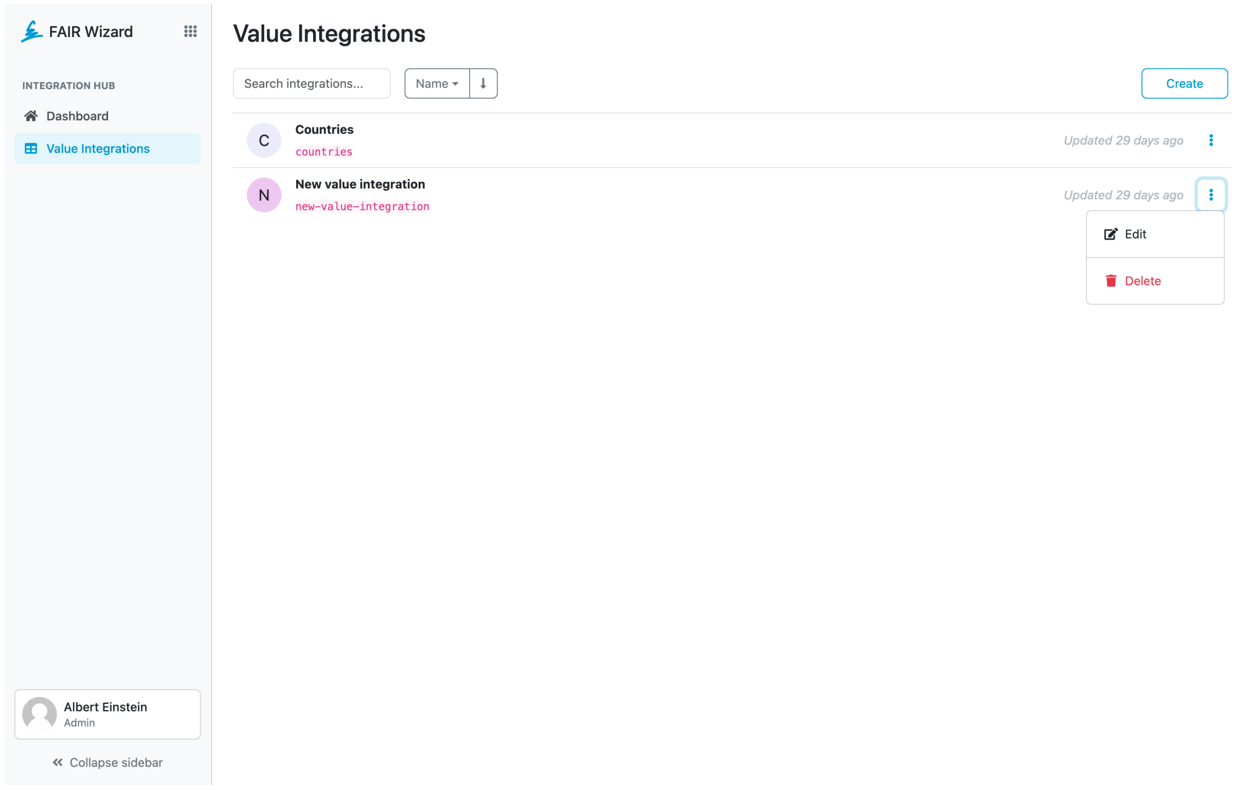
Create

As an administrator, we can create a value integration. To start, we must fill in its name and id. After clicking on *Create* button, we can continue in the *value integration detail*.

Detail

As an administrator, we need to fill in some information about the Integration. We have filled `name` and `ID` in the previous step, now we just need to fill in the `Response Item Template`. This defines how the items from the values will be displayed for the user. You can use the column names as item properties in `Jinja2` notation. You can also use `Markdown` for some formatting. For example, if there is a column called name, you can use `**{{item.name}}**` to display the name in bold.

In Values, we can set up specific values. First, we need to give names to columns and then fill the rows with data. We can do that either manually or use Import functionality. We can also manually delete columns or rows or *Delete all*.

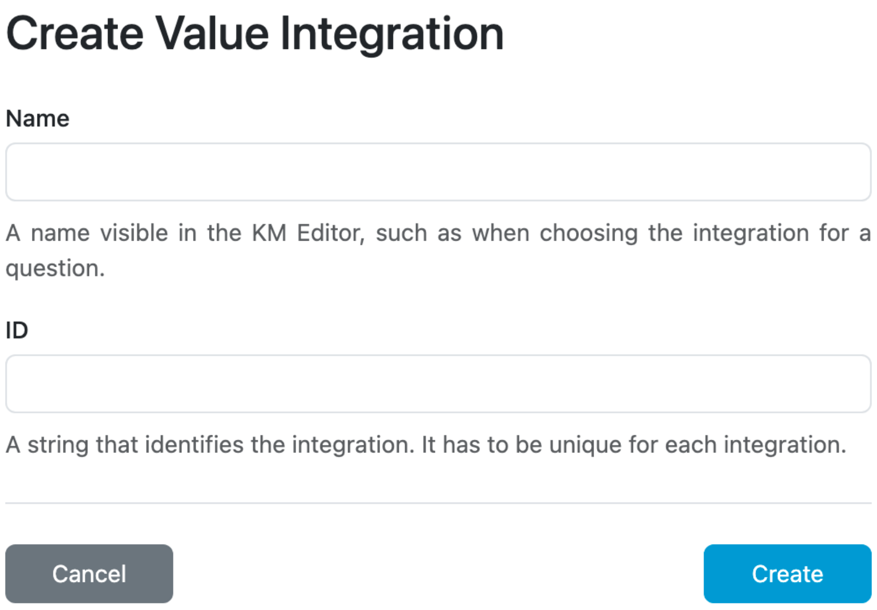


Value Integrations

Countries
countries
Updated 29 days ago

New value integration
new-value-integration
Updated 29 days ago

Fig. 106: Value integrations.



Create Value Integration

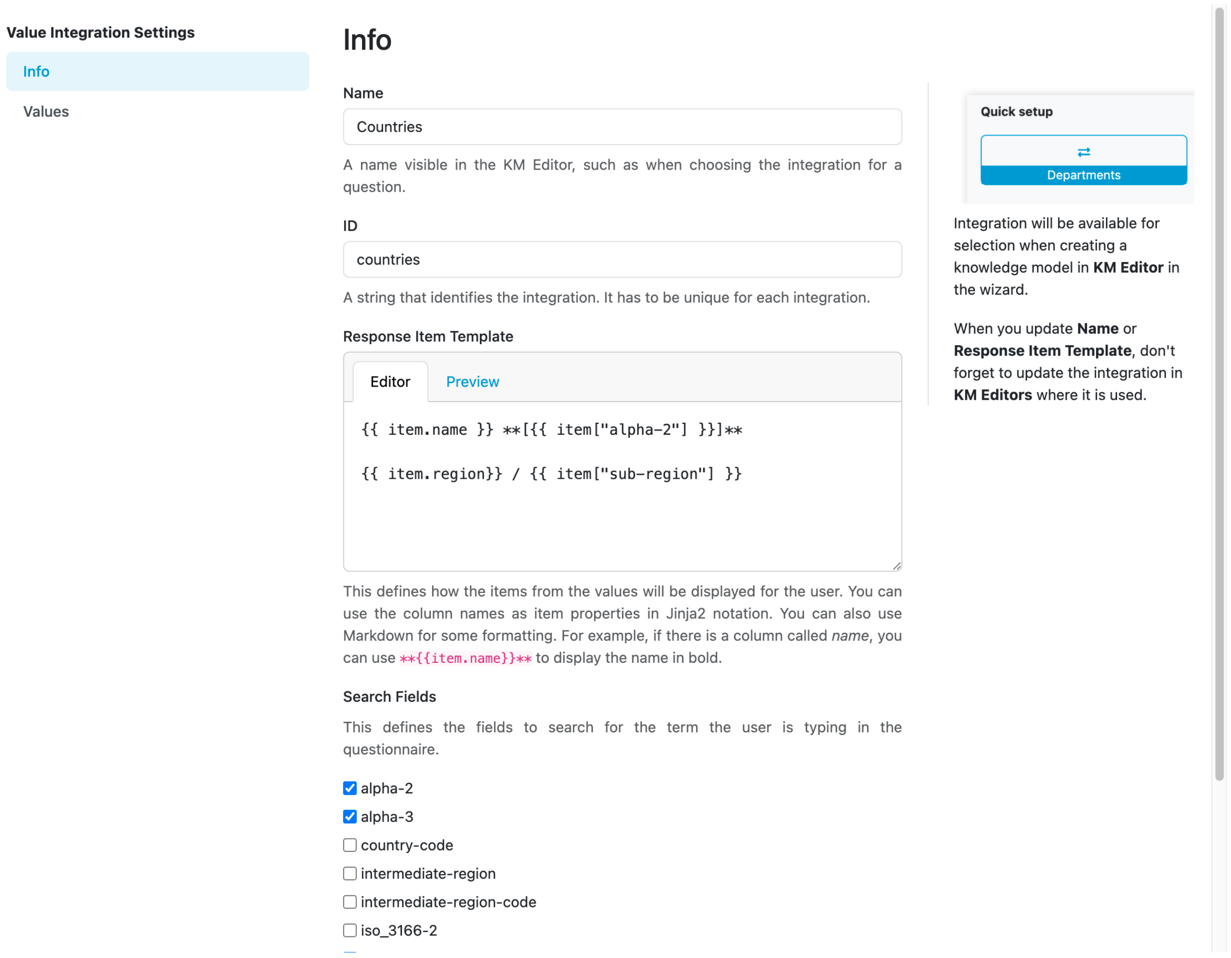
Name

A name visible in the KM Editor, such as when choosing the integration for a question.

ID

A string that identifies the integration. It has to be unique for each integration.

Fig. 107: Create value integration.



Info

Name

Countries

A name visible in the KM Editor, such as when choosing the integration for a question.

ID

countries

A string that identifies the integration. It has to be unique for each integration.

Response Item Template

```
{{ item.name }} **[{{ item["alpha-2"] }}]**

{{ item.region}} / {{ item["sub-region"] }}
```

This defines how the items from the values will be displayed for the user. You can use the column names as item properties in Jinja2 notation. You can also use Markdown for some formatting. For example, if there is a column called *name*, you can use `**{{item.name}}**` to display the name in bold.

Search Fields

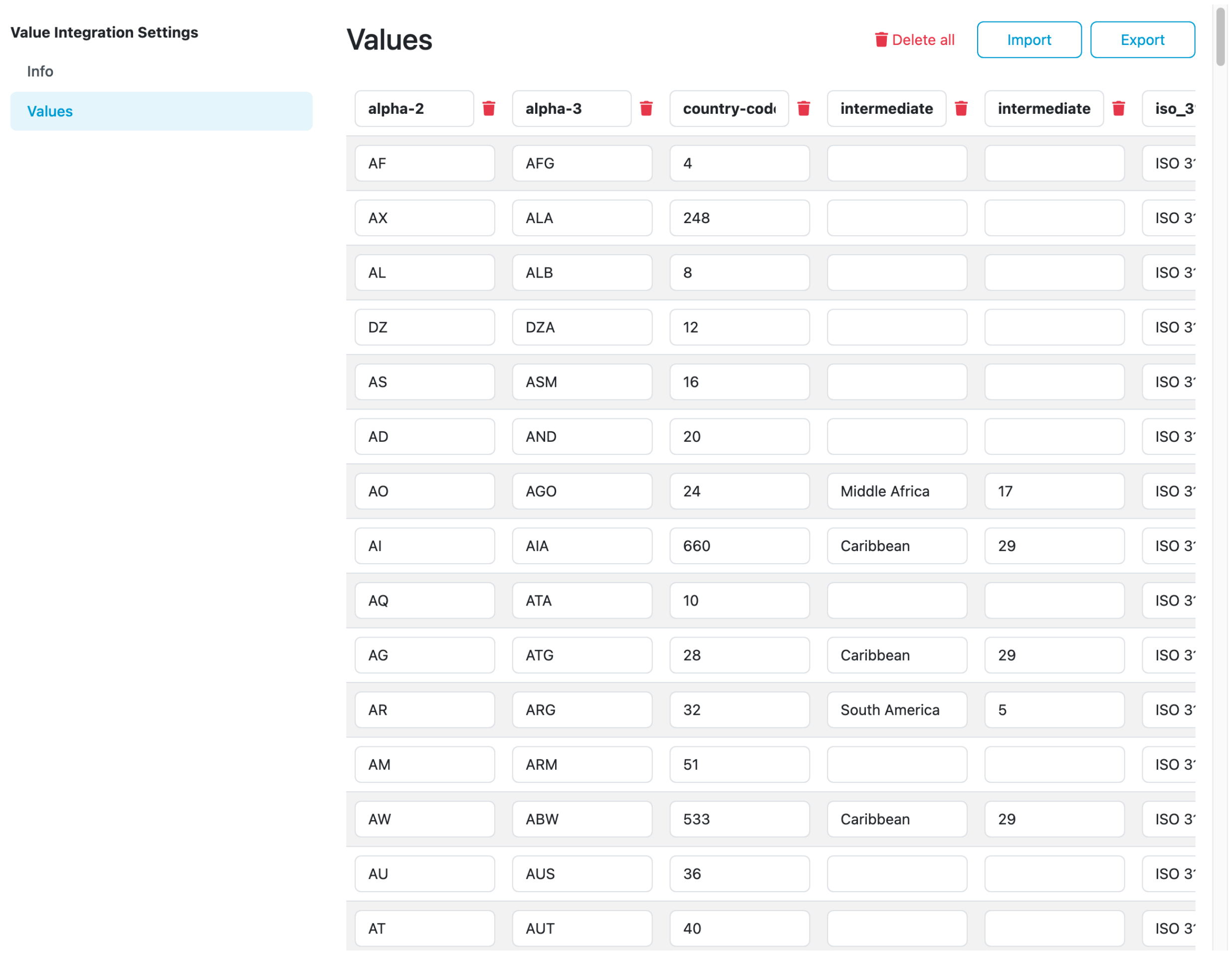
This defines the fields to search for the term the user is typing in the questionnaire.

- [✓] alpha-2
- [✓] alpha-3
- [ ] country-code
- [ ] intermediate-region
- [ ] intermediate-region-code
- [ ] iso_3166-2

Integration will be available for selection when creating a knowledge model in **KM Editor** in the wizard.

When you update **Name** or **Response Item Template**, don't forget to update the integration in **KM Editors** where it is used.

Fig. 108: Value integration detail info.


Value Integration Settings

Info

Values

Values

Delete all | Import | Export

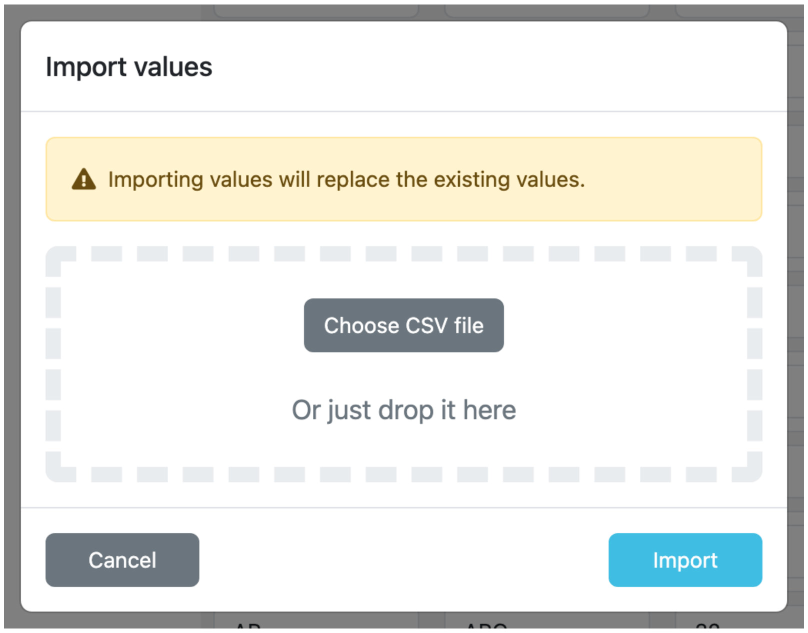
alpha-2	alpha-3	country-cod	intermediate	intermediate	iso_3
AF	AFG	4			ISO 3
AX	ALA	248			ISO 3
AL	ALB	8			ISO 3
DZ	DZA	12			ISO 3
AS	ASM	16			ISO 3
AD	AND	20			ISO 3
AO	AGO	24	Middle Africa	17	ISO 3
AI	AIA	660	Caribbean	29	ISO 3
AQ	ATA	10			ISO 3
AG	ATG	28	Caribbean	29	ISO 3
AR	ARG	32	South America	5	ISO 3
AM	ARM	51			ISO 3
AW	ABW	533	Caribbean	29	ISO 3
AU	AUS	36			ISO 3
AT	AUT	40			ISO 3

Fig. 109: Value integration detail values.

Note: Order of columns can be random as it does not matter for the JSON.

Warning: JSON is case sensitive.

We can import any CSV file. To do that, we click on the *Import* button and select the file. The file should be in CSV format and the first row should contain the column names. The file should not contain any empty rows. If the file is not in the correct format, the system will display an error message.


Import values

Importing values will replace the existing values.

Fig. 110: Value integration import.

If we choose to insert values manually or do some changes to the values, we can always use *Export* button to get the data.

Note: Don't forget to save your changes.

1.5 Reporting

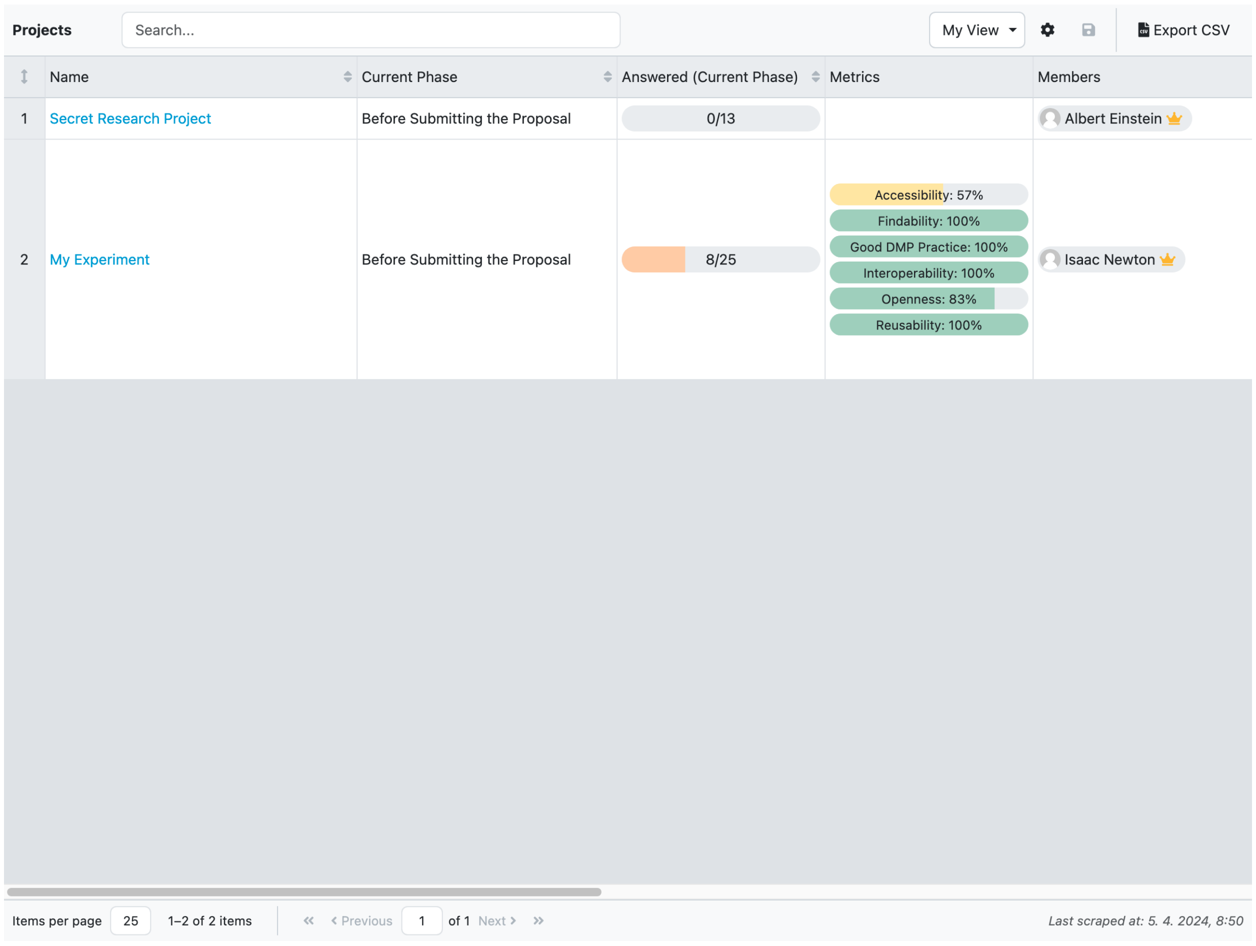
Reporting is an application for management of reports. It allows scraping of data from other apps, and then creating reports based on that data.

Reporting can be currently done on top of *projects* and *users*.

To get latest data, click on *Scrape now* button. This will scrape the data from the other apps and store it in the reporting database.

1.5.1 Projects

As an administrator, we can create a report based on various projects. We can select from many different fields and see how our projects are doing in them. The report can be modified and save to be viewed later. The data of the report can also be exported to a CSV file.



	Name	Current Phase	Answered (Current Phase)	Metrics	Members
1	Secret Research Project	Before Submitting the Proposal	0/13		Albert Einstein
2	My Experiment	Before Submitting the Proposal	8/25	Accessibility: 57% Findability: 100% Good DMP Practice: 100% Interoperability: 100% Openness: 83% Reusability: 100%	Isaac Newton

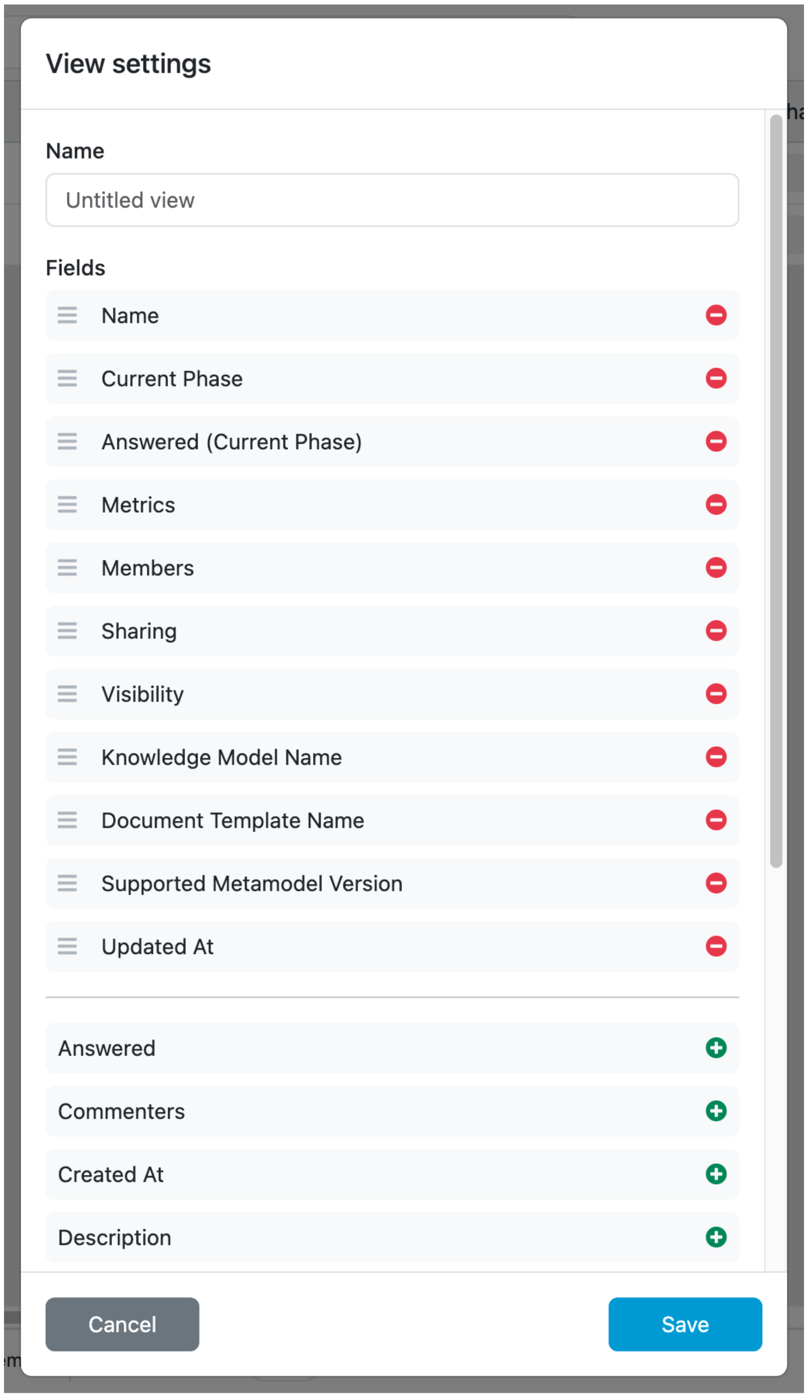
Fig. 111: Projects overview.

New report can be created by clicking on the dropdown menu in the top right corner. Then by clicking on + *Create a new view* we open the view settings. We can give our view a name and select which fields we want to have in there. The report can be saved by clicking on *Save*. We can also delete the report by clicking on *Delete*.

We can resize all rows height by clicking on the double arrow in the top left corner. If we want to edit width or height of individual cells, we can do it using drag-and-drop on the borders. Lastly we can edit how many rows are on the page by clicking on the *Items per page* dropdown menu.

Note: Don't forget to click on *Save* icon after you are done with editing the view.

The report can be exported to a CSV file by clicking on *Export CSV*.



View settings

Name

Untitled view

Fields

- Name
- Current Phase
- Answered (Current Phase)
- Metrics
- Members
- Sharing
- Visibility
- Knowledge Model Name
- Document Template Name
- Supported Metamodel Version
- Updated At

- Answered
- Commenters
- Created At
- Description

Fig. 112: Form for editing reporting view.

1.5.2 Users

As an administrator, we can create a report based on users. We can select from many different fields and see how our users are doing in them. The report can be modified and save to be viewed later. The data of the report can also be exported to a CSV file.

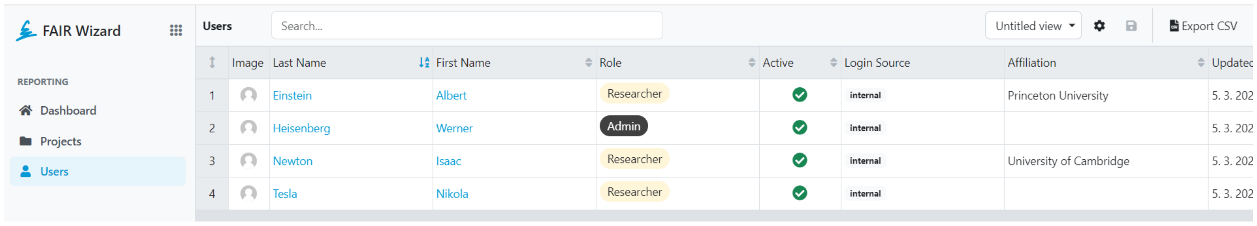


	Image	Last Name	First Name	Role	Active	Login Source	Affiliation	Updatec
1		Einstein	Albert	Researcher	✓	internal	Princeton University	5. 3. 202
2		Heisenberg	Werner	Admin	✓	internal		5. 3. 202
3		Newton	Isaac	Researcher	✓	internal	University of Cambridge	5. 3. 202
4		Tesla	Nikola	Researcher	✓	internal		5. 3. 202

Fig. 113: Users overview.

New report can be created by clicking on the dropdown menu in the top right corner. Then by clicking on + *Create a new view* we open the view settings. We can give our view a name and select which fields we want to have in there. The report can be saved by clicking on *Save*. We can also delete the report by clicking on *Delete*.

We can resize all rows height by clicking on the double arrow in the top left corner. If we want to edit width or height of individual cells, we can do it using drag-and-drop on the borders. Lastly we can edit how many rows are on the page by clicking on the *Items per page* dropdown menu.

Note: Don't forget to click on *Save* icon after you are done with editing the view.

The report can be exported to a CSV file by clicking on *Export CSV*.

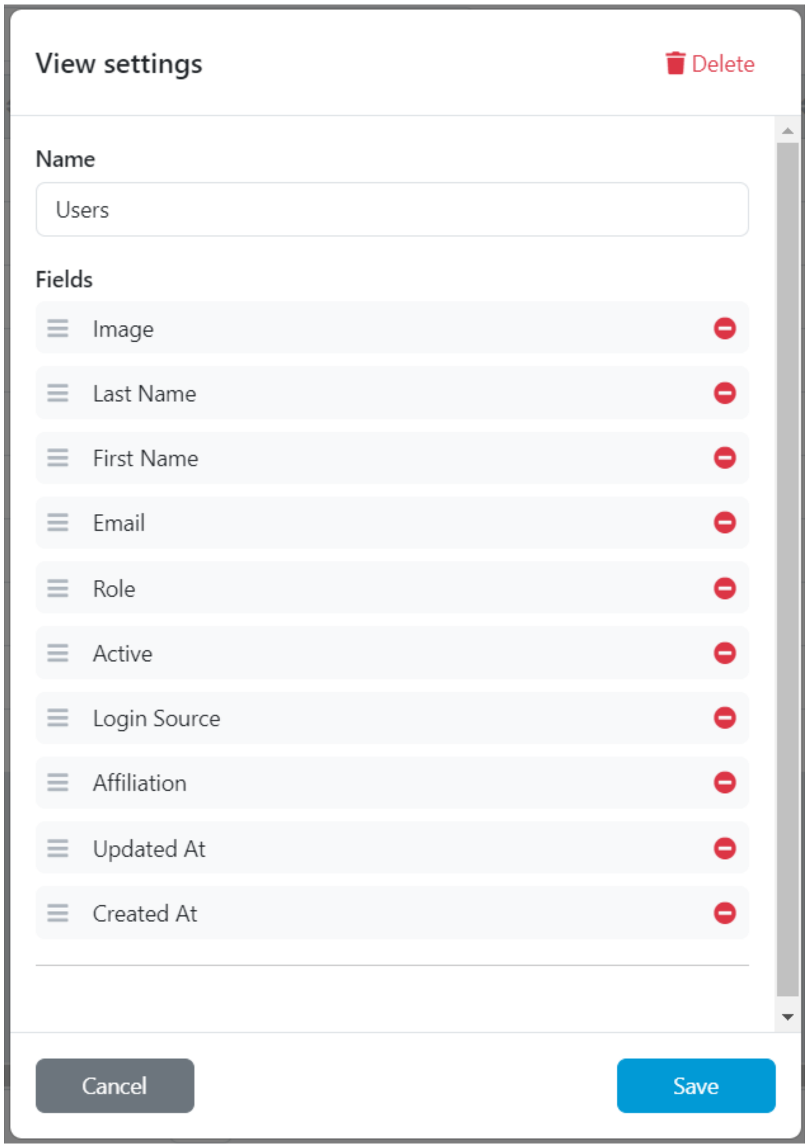
1.6 Development

FAIR Wizard can be extended in many ways and new components and ways of integrations can be developed to support our needs. Besides the API available for everything that can be done in FAIR Wizard, new *integration questions* and *project importers* can be implementet to get data from outside to FAIR Wizard, or new *document templates* and *submission services* can be created to get the data outside of FAIR Wizard in the desired form.

This section provides information on how to develop custom content for FAIR Wizard to fully tailor the tool to our specific requirements.

1.6.1 Metamodel Schemas

As Data Stewardship Wizard evolves, the internal structures may change during the time. To support migration under the hood, we use metamodel versioning for KM and templates.



View settings

Delete

Name

Users

Fields

- Image
- Last Name
- First Name
- Email
- Role
- Active
- Login Source
- Affiliation
- Updated At
- Created At

Cancel

Save

Fig. 114: Form for editing reporting view.

KM Package (.km file)

File for import and export of Knowledge Models is a JSON file that contains all KM packages (lists of change events with additional metadata). The structure of events is versioned using the KM metamodel version number and migrations in FAIR Wizard automatically update the KMs when needed. As said, files according to this schema can be exported from *Knowledge Model List* or *Knowledge Model Detail* and then used for *Knowledge Model Import*.

Metamodel Version	Schema file	Changes (brief)	Since
14	JSON Schema	Optional Integration fields	4.1.0
13	JSON Schema	New question value types	3.12.0
12	JSON Schema	Enhanced integration (e.g. widget type)	3.10.0
11	JSON Schema	Annotations and timestamps for events	3.8.0
10	JSON Schema	Integrations with item template	3.6.0
9	JSON Schema	Annotations	3.5.0
8	JSON Schema	Metrics and phases are part of KM	3.2.0
7	JSON Schema	KM name attribute removed	2.13.0
6	JSON Schema	Multi-choice question type added	2.11.0
5	JSON Schema	Move event	2.0.0
4	JSON Schema	Refactored KM, optional chapter text	1.10.0
3	JSON Schema	Changed integration question fields	1.8.0
2	JSON Schema	Changed phases representation	1.7.0
1	JSON Schema	Initial versioned metamodel	1.6.0

Document Context

Document Context is provided to the document templates by document worker. It contains all relevant data about project/questionnaire with replies, related knowledge model, author, and more. As KM evolves, the context may evolve as well. It is versioned using the Template metamodel version number. A document template must support the metamodel that is in the current FAIR Wizard version. It is needed to know how the document context looks like especially for *Document Template Development*.

Metamodel Version	Schema file	Changes (brief)	Since
13	JSON Schema	Removed states from templates	4.1.0
12	JSON Schema	Optional Integration fields	4.1.0
11	JSON Schema	Change template metadata	3.20.0
10	JSON Schema	New question value types	3.12.0
9	JSON Schema	Enhanced integration (e.g. widget type)	3.10.0
8	JSON Schema	Annotations change	3.8.0
7	JSON Schema	Project tags and description	3.7.0
6	JSON Schema	Integrations with item template	3.6.0
5	JSON Schema	Annotations	3.5.0
4	JSON Schema	Metrics and phases	3.2.0
3	JSON Schema	Project versions	2.12.0
2	JSON Schema	Reply provenance	2.6.0
1	JSON Schema	Initial versioned metamodel	2.5.0

Template (.json file)

Each template has its descriptor file `template.json` which contains all the information about the template, its format(s) and steps how to produce the document(s). It is also versioned by the Template metamodel version number. This file also contains the actual number of the supported version… With local *Document Template Development*, we will need to manage the file according to the schema; however, when *Document Template Editors* are used, we will define it using forms directly in FAIR Wizard.

Note: Between versions 1 and 5, the structure of `template.json` is still the same. Only the document context has been changed.

Metamodel Version	Schema file	Changes (brief)	Since
13	JSON Schema	Removed states from templates	3.20.0
12	JSON Schema	Optional Integration fields	3.12.0
11	JSON Schema	Change template metadata	3.20.0
10	JSON Schema	New question value types	3.12.0
9	JSON Schema	Enhanced integration (e.g. widget type)	3.10.0
8	JSON Schema	Annotations change	3.8.0
7	JSON Schema	Project tags and description	3.7.0
6	JSON Schema	Integrations with item template	3.6.0
5	JSON Schema	Annotations	3.5.0
4	JSON Schema	Metrics and phases	3.2.0
3	JSON Schema	Project versions	2.12.0
2	JSON Schema	Reply provenance	2.6.0
1	JSON Schema	Initial versioned metamodel	2.5.0

1.6.2 Document Template Development

Document templates allows to specify how to export a questionnaire in form of a textual file. It is a highly flexible element of the tool; however, the development requires basic programming skills with Jinja2 templating language. We can develop the document templates either on our local computer (traditional development with text editor or IDE) with use of the *Template Development Kit* (TDK) or directly in FAIR Wizard using *Document Template Editors*.

Every document template is based on the *template specification* and typically uses the *document context* to query information from a project (questionnaire replies, knowledge model, metadata, etc.) to create a document.

Examples

- ds-wizard/questionnaire-report-template
- ds-wizard/madmp-template
- ds-wizard/horizon-europe-dmp-template

Document Context

Note: To work efficiently with the Document Context, you want to use object instead of the JSON-like one. Please read through DocumentContext.md directly (select different version if needed).

Document context is an object that carries all information related to a FAIR Wizard questionnaire in order to produce a document. To investigate it, it is the best to use *Questionnaire Report* template with `JSON` format. The core fields are:

- `config` = object with FAIR Wizard configuration related to documents, e.g., `clientUrl` for referring to the FAIR Wizard
- `createdAt` = timestamp when the document was created
- `createdBy` = object describing author of the document
- **`knowledgeModel` = object describing used KM for the questionnaire**
    - `chapterUuids` = list of UUIDs for chapters
    - `integrationUuids` = list of UUIDs for integrations
    - `tagUuids` = list of UUIDs for tags
    - `entities` = contains `questions`, `answers`, and other maps with UUID-entity pairs
    - `name` = name of the knowledge model
    - `uuid` = UUID of the knowledge model
- `level` = current desirability level selected for the questionnaire
- `levels` = list of desirability levels possible
- `metrics` = list of available metrics
- `organization` = object describing organization that runs the FAIR Wizard
- `package` = object with metadata about the KM package such as `version`, `name`, or `description`
- `questionnaireName` = name of the questionnaire
- `questionnaireReplies` = map of replies with path-reply pairs, each reply has `type` and `value`
- `questionnaireUuid` = UUID of the questionnaire
- `report` = object that contains report for the questionnaire that contains computed information about number of answered questions as well as metric values
- `updatedAt` = timestamp when the document was last updated
- `uuid` = UUID of the document

This structure is provided to a Jinja template in *Step: jinja* and outputed from *Step: json*. We can use the JSON step to observe the actual content of the document context (structure as well as the values). Finally, we can also check *Metamodel Schemas* (the relevant JSON schema for document context).

Objectified Document Context

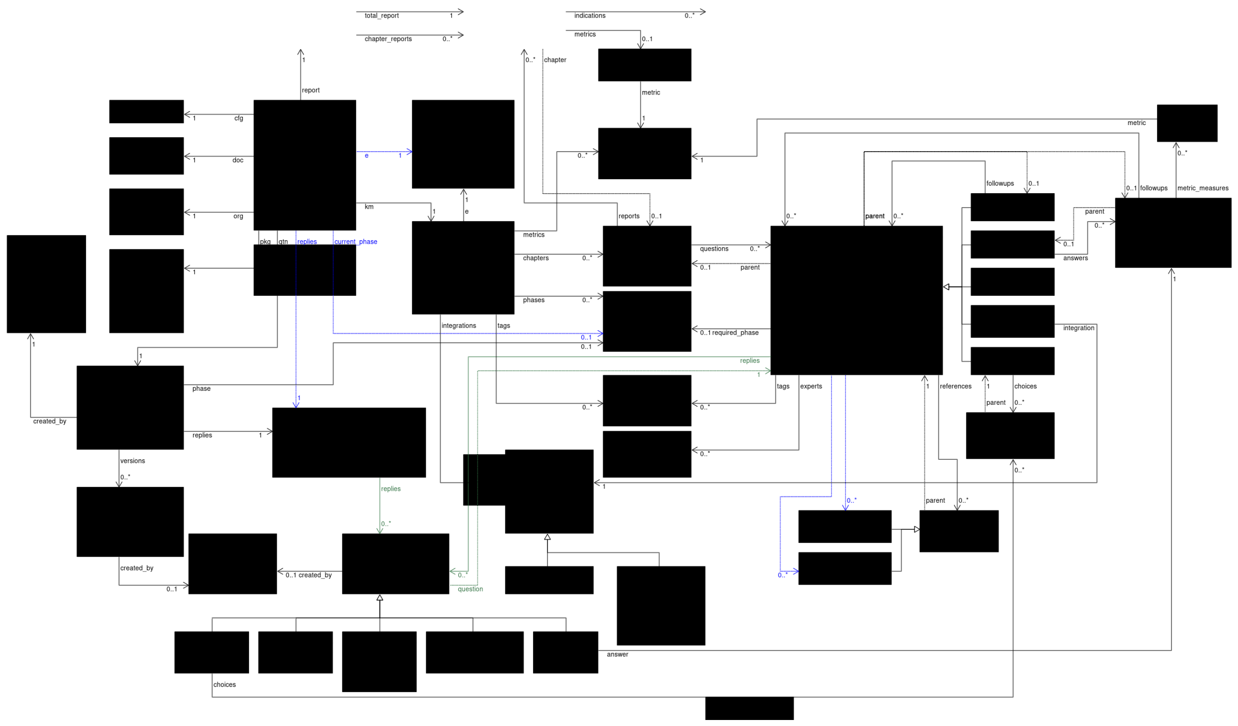
It is possible to easily turn the JSON-like / tree-structured document context into objects with additional helper relations, attributes, methods, and many more to ease up the template development:

```
{%- set dc = ctx|to_context_obj -%}
```

- All data types are using Python, e.g., `str` is textual string, `Optional[str]` is a string or `None`, `list[str]` is a list of strings.
- We use `snake_case` for naming of attributes and variables, `PascalCase` is used for class names.
- `datetime` is the standard datetime module.

Diagram

We provide the structure visualized on a class diagram (right-click and open in to tab to enlarge):



Entities

Here is an interlinked description of each entity and its attributes and links. There are also *aliases* that are convenient shorthands to make template more concise.

DocumentContext

- `config` (*ContextConfig*)
- `current_phase` (`Optional[`*Phase*`]`)
- `document` (*Document*)
- `km` (*KnowledgeModel*)
- `organization` (*Organization*)
- `package` (*Package*)
- `questionnaire` (*Questionnaire*)
- `report` (*Report*)

Aliases:

- `e` (*KnowledgeModelEntities*) - same as `km.entities`
- `doc` (*Document*) - same as `document`
- `org` (*Organization*) - same as `organization`
- `pkg` (*Package*) - same as `package`
- `qtn` (*Questionnaire*) - same as `questionnaire`
- `replies` (*RepliesContainer*) - same as `questionnaire.replies`

ContextConfig

- `client_url` (`str`) - base URL of the FAIR Wizard application (client app)

Document

- `uuid` (`str`)
- `created_at` (`datetime`)
- `updated_at` (`datetime`)

Organization

- `id` (`str`)
- `name` (`str`)
- `description` (`Optional[str]`)
- `affiliations` (`list[str]`)

Package

- id (str) - full ID of KM Package
- organization_id (str)
- km_id (str)
- version (str)
- versions (list[str])
- name (str)
- description (Optional[str])
- created_at (datetime)

Questionnaire

- uuid (str)
- name (str)
- version (Optional[*QuestionnaireVersion*])
- versions (list[*QuestionnaireVersion*])
- phase (Optional[*Phase*])
- replies (*RepliesContainer*)
- created_by (*User*)

QuestionnaireVersion

- uuid (str)
- event_uuid (str)
- name (str)
- description (Optional[str])
- created_by (*SimpleAuthor*)
- created_at (datetime)
- updated_at (datetime)

User

- uuid (str)
- first_name (str)
- last_name (str)
- email (str)
- role (str) - one of: admin, dataSteward, researcher
- image_url (Optional[str])

- `affiliation` (`Optional[str]`)
- `permissions` (`list[str]`)
- `sources` (`list[str]`)
- `created_at` (`datetime`)
- `updated_at` (`datetime`)

SimpleAuthor

- `uuid` (`str`)
- `first_name` (`str`)
- `last_name` (`str`)
- `image_url` (`Optional[str]`)
- `gravatar_hash` (`Optional[str]`)

Report

- `uuid` (`str`)
- `total_report` (*ReportItem*)
- `chapter_reports` (`list[`*ReportItem*`]`)
- `created_at` (`datetime`)
- `updated_at` (`datetime`)

ReportItem

- `indications` (`list[`*ReportIndication*`]`)
- `metrics` (`list[`*ReportMetric*`]`)
- `chapter` (`Optional[`*Chapter*`]`) - set if it is a chapter report

ReportIndication

- `indication_type` (`str`) - one of: `PhasesAnsweredIndication`, `AnsweredIndication` (use alias)
- `answered` (`int`) - number of answered questions
- `unanswered` (`int`) - number of unanswered questions

Aliases:

- `total` (`int`) - `answered` + `unanswered`
- `percentage` (`float`) - `answered` / `total` (handles zero division, number between `0.0` and `1.0`)
- `is_for_phase` (`bool`) - if it is a phase-related indication
- `is_overall` (`bool`) - if it is an overall indication (not phase-related)

ReportMetric

- `measure` (`float`) - number between `0.0` and `1.0`
- `metric` (*Metric*)

KnowledgeModel

- `uuid` (`str`)
- `annotations` (`dict[str,str]`)
- `entities` (*KnowledgeModelEntities*)
- `chapters` (`list[`*Chapter*`]`)
- `integrations` (`list[`*Integration*`]`)
- `metrics` (`list[`*Metric*`]`)
- `phases` (`list[`*Phase*`]`)
- `tags` (`list[`*Tag*`]`)

Aliases:

- `e` (*KnowledgeModelEntities*) - same as `entities`
- `a` (`dict[str,str]`) - same as `annotations`

Notes:

- Equality of all KM entities is being done using the `uuid` comparison under the hood.
- All KM entities that have `annotations` have also the `a` alias.

KnowledgeModelEntities

Container holding all types of Knowledge Model entities within UUID-key dictionaries:

- `answers` (`dict[str,`*Answer*`]`)
- `chapter` (`dict[str,`*Chapter*`]`)
- `choices` (`dict[str,`*Choice*`]`)
- `experts` (`dict[str,`*Expert*`]`)
- `integrations` (`dict[str,`*Integration*`]`)
- `metrics` (`dict[str,`*Metric*`]`)
- `phases` (`dict[str,`*Phase*`]`)
- `questions` (`dict[str,`*Question*`]`)
- `references` (`dict[str,`*Reference*`]`)
- `tags` (`dict[str,`*Tag*`]`)

Chapter

- `uuid` (`str`)
- `title` (`str`)
- `text` (`Optional[str]`) - possibly Markdown text
- `questions` (`list[`*Question*`]`)
- `reports` (`list[`*ReportItem*`]`)
- `annotations` (`dict[str,str]`)

Question

Superclass with common attributes for all types of questions. You always get a more specific one and never just a `Question`.

- `uuid` (`str`)
- `type` (`str`)
- `title` (`str`)
- `text` (`Optional[str]`)
- `required_phase` (`Optional[`*Phase*`]`)
- `is_required` (`bool`) - if the question is required in the current phase
- `replies` (`dict[str,`*Reply*`]`) - path-key dictionary of replies to the question
- `experts` (`list[`*Expert*`]`)
- `references` (`list[`*Reference*`]`)
- `tags` (`list[`*Tag*`]`)
- `parent` (`Union[`*Chapter*`,`*ListQuestion*`,`*Answer*`]`)
- `annotations` (`dict[str,str]`)

Aliases:

- `url_references` (`list[`*URLReference*`]`)
- `resource_page_references` (`list[`*ResourcePageReference*`]`)

Notes:

- Parent of a question can be of multiple kinds, you may use the `of_type` test to check what it is if needed.

ValueQuestion

- `value_type` (`str`) - type of value, use alias

Aliases:

- `is_string` (`bool`)
- `is_text` (`bool`)
- `is_number` (`bool`)
- `is_date` (`bool`)

IntegrationQuestion

- `integration` (*Integration*)
- `props` (`dict[str,str]`)

OptionsQuestion

- `answers` (`list[`*Answer*`]`)

MultiChoiceQuestion

- `choices` (`list[`*Choice*`]`)

ListQuestion

- `followups` (`list[`*Question*`]`)

Answer

- `uuid` (`str`)
- `label` (`str`)
- `advice` (`Optional[str]`) - possibly Markdown text
- `metric_measures` (`list[`*MetricMeasure*`]`)
- `followups` (`list[`*Question*`]`)
- `parent` (*OptionsQuestion*)
- `annotations` (`dict[str,str]`)

MetricMeasure

Indication of how an answer affects a certain metric.

- `measure` (`float`) - value between `0.0` and `1.0` (inclusive)
- `weight` (`float`) - value between `0.0` and `1.0` (inclusive)
- `metric` (*Metric*)

Choice

- `uuid` (`str`)
- `label` (`str`)
- `parent` (*MultiChoiceQuestion*)
- `annotations` (`dict[str,str]`)

Expert

- `uuid` (`str`)
- `name` (`str`)
- `email` (`str`)
- `annotations` (`dict[str,str]`)

Reference

As for the *Question* class, `Reference` is also a superclass and you will always get an object of its subclass.

- `uuid` (`str`)
- `type` (`str`)
- `annotations` (`dict[str,str]`)

URLReference

- `label` (`str`)
- `url` (`str`)

ResourcePageReference

- short_uuid (str)
- url (str) - URL composed using client_url from *ContextConfig*

Metric

- uuid (str)
- title (str)
- abbreviation (str)
- description (Optional[str]) - possibly Markdown text
- annotations (dict[str,str])

Phase

- uuid (str)
- title (str)
- description (Optional[str]) - possibly Markdown text
- order (int) - order of the phase within the KM
- annotations (dict[str,str])

Integration

- uuid (str)
- id (str)
- name (str)
- item_url (Optional[str])
- logo (Optional[str])
- props (dict[str,str])
- rq_method (str)
- rq_url (str)
- rq_headers (dict[str,str])
- rq_body (str)
- rs_list_field (Optional[str])
- rs_item_id (Optional[str])
- rs_item_template (str)
- annotations (dict[str,str])

Operations:

- item(item_id: str) -> Optional[str] - URL of an item identified by string ID

Tag

- `uuid` (`str`)
- `name` (`str`)
- `description` (`Optional[str]`) - possibly Markdown text
- `color` (`str`)
- `annotations` (`dict[str,str]`)

RepliesContainer

Wrapper around a path-key dictionary of replies.

- `replies` (`dict[str,`*Reply*`]`)

Operations:

- `X[path: str]` (`Optional[`*Reply*`]`) - you can get a reply using square brackets
- `len(X)` (`int`) - number of replies in the container
- `get(path: str) -> Optional[`*Reply*`]`
- `iterate_by_prefix(path_prefix: str) -> Iterable[`*Reply*`]` - *O(n)* iteration with filter
- `iterate_by_suffix(path_suffix: str) -> Iterable[`*Reply*`]` - *O(n)* iteration with filter
- `values() -> Iterable[`*Reply*`]`
- `keys() -> Iterable[str]`
- `items() -> ItemsView[str,`*Reply*`]`

Reply

Superclass with common attributes for all types of replies. You always get a more specific one and never just a `Reply`.

- `path` (`str`)
- `fragments` (`list[str]`) - UUIDs of the path (starting with chapter)
- `type` (`str`)
- `created_at` (`datetime`)
- `created_by` (*SimpleAuthor*)
- `question` (*Question*) - you can assume more specific type of `Question` based on a type of `Reply`

AnswerReply

- `answer` (*Answer*) - selected answer as the option

Aliases:

- `value` (`str`) - UUID of the answer (`answer.uuid`)

Notes:

- `question` is always *OptionsQuestion*

MultiChoiceReply

- `choices` (`list[`*Choice*`]`) - selected answer as the option

Aliases:

- `value` (`list[str]`) - list of UUIDs of the choices

Notes:

- `question` is always *OptionsQuestion*
- You can iterate directly over reply object(`for choice in reply`)

StringReply

- `value` (`str`)

Aliases:

- `as_number` (`Optional[float]`) - tries to cast the value to a number
- `as_datetime` (`Optional[datetime]`) - tries to cast the value to a timestamp

Notes:

- `question` is always *ValueQuestion*

ItemListReply

- `items` (`list[str]`) - list of item UUIDs (used in reply paths)

Aliases:

- `value` (`list[str]`) - same as `items`

Notes:

- `question` is always *ListQuestion*
- You can iterate directly over reply object (`for item in reply`)

IntegrationReply

- `value` (`str`)
- `item_id` (`Optional[str]`) - ID of item if selected using *Integration*

Aliases:

- `id` (`Optional[str]`) - same as `item_id`
- `is_plain` (`bool`) - entered by user ignoring the integration
- `is_integration` (`bool`) - selected by user using the integration
- `url` (`Optional[str]`) - item URL based *Integration* if selected from it

Template Development Kit

Note: Requirements for Local Template Development

- Your favorite text editor or IDE
- Template Development Kit (see below)
- FAIR Wizard application with your admin account
- Python 3.10+ (with pip) or Docker

Our Template Development Kit (TDK) provides a simple way how to work with templates locally. It is a CLI tool written in Python.

Video Tutorial

This is a comprehensive video tutorial on how to use the Template Development Kit.

https://youtu.be/FFElv-e24NE

Installation

You can install it easily using pip from Python Package Index (PyPI). Optionally, you can use virtual environment or other installation option described in the TDK repository.

```
pip install dsw-tdk
dsw-tdk --help
```

It is also possible to use datastewardshipwizard/dsw-tdk Docker image when you don't have Python locally:

```
docker run datastewardshipwizard/dsw-tdk --help
```

Commands

There are these basic commands:

- `new` = create a new template project, it launches a simple interactive wizard for template metadata
- `list` = list all templates (latest versions) from configured FAIR Wizard
- `get` = download a template project with specified template ID from FAIR Wizard
- `put` = upload the local template project to FAIR Wizard (once or continually on-change when `--watch` flag is used)
- `verify` = check the metadata of the local template project
- `package` = create a ZIP distribution package from the local template project (ZIP is importable to FAIR Wizard via its web interface)

Default template directory is current one for `put`, `verify`, and `package`. But `new` and `get` will create a new folder according to the template ID if not explicitly set in other way.

You can use `--help` to find out details:

```
dsw-tdk new --help
```

Environment variables and .env file

You can use environment variables to authenticate:

- `DSW_API_URL` = URL of FAIR Wizard API with which you want to communicate. Hover mouse over your profile name to find the About section where URL is specified.
- `DSW_API_KEY` = your API Key. Hover mouse over your profile name, click on *Edit Profile* and then navigate to *API Keys* From there, you can generate a new API Key for the authentication.

To make this even easier, you can store those in `.env` file in the project root and it will be loaded automatically. Or you can specify the path to a `.env` file:

```
dsw-tdk --dot-env /path/to/.env list
```

Document Template Specification

Each document template in FAIR Wizard has metadata stored. If developing locally with *Template Development Kit*, you can find and manage them in `template.json` file. In case of using *Document Template Editors*, you can manage them on **Settings** tab.

Specification Structure

- `id` = composed full ID of the template (`organizationId:templateId:version`)
- `organizationId` = identifier of organization developing the template (lowercase, numerics, dot)
- `templateId` = identifier of template (lowercase, numerics, dash)
- `version` = version (semver) in X.Y.Z format where X, Y, and Z are non-negative numbers
- `name` = name of the template

- `description` = short description of the template
- `license` = name of the used license
- `readme` = longer description usually containing changelog
- `metamodelVersion` = supported version of template metamodel, it affects with which FAIR Wizard version is can be used
- `allowedPackages` = list of package filters (see *Package Filters*) to specify supported packages
- `formats` = list of available formats (see below *Formats*) with specified steps for generation
- `_tdk` = TDK configuration for local development (not stored in FAIR Wizard, see *TDK Config*)

Note: TDK handles `id` and `readme` for you, so you can skip them and naturally use `README.md` file separately.

Package Filters

For filtering, the `null` value serves as wildcard, i.e., filter with all `null` values means that all packages are allowed.

- `orgId`: identifier of organization (e.g. `myorg`)
- `kmId`: identifier of knowledge model (e.g. `root`)
- `minVersion`: minimal package version (in format X.Y.Z, inclusive)
- `maxVersion`: maximal package version (in format X.Y.Z, inclusive)

Formats

A template can describe how to produce several formats, each with these metadata:

- `uuid`: UUID of the format (within template)
- `name`: display name of the format
- `icon`: icon style (CSS classes), preferably Font Awesome, e.g. `fas fa-file-word`
- `steps`: list of steps for document worker to produce the document with this format, each step has `name` and `options` (see *Steps*)

Steps

Each step of template produces output based on its (optional) input and options. Steps can be chained in order to generate the document and eventually transform it. All steps have always `name` and `options` based on one of the desired step. There are the details for steps supported by the *document worker* component:

Step: `archive`

status stable metamodel version ≥ 11

Step that puts file from previous step to an archive file (ZIP or TAR).

Input

Any input file provided from the previous step.

Output

ZIP or TAR archive (based on `options`) containing the file from the previous step.

Options

- `inputFileDst` = destination of the file inside the archive (POSIX-like path including filename)
- (optional) `type` = whether to produce `zip` or `tar` (defaults to `zip`)
- (optional) `compression` = compression method to be used (`none`, `gzip`, `bzip2`, `lzma`; defaults to `none`)
- (optional) `compressionLevel` = value specifying level of compression (`0` to `9`; defaults to `9`)
- (optional) `format` = only for `tar` it allows to specify format (`ustar`, `gnu`, `pax`; defaults to `pax`)

Notes

- Currently, only a single file can be put into the produced archive.
- Value of `compressionLevel` must be provided as a string (even though it is a numeric value).
- For `zip`, zipfile standard library from Python is used.
- For `tar`, tarfile standard library from Python is used.
- For `bzip2`, if `compressionLevel` is set to `0`, it is automatically fixed to value `1`.

Example

```
{
  "name": "archive",
  "options": {
    "type": "tar",
    "compression": "bzip2",
    "compressionLevel": "5",
    "format": "gnu",
    "inputFileDst": "example/file.html"
  }
}
```

Step: `enrich-docx`

status stable metamodel version ≥ 1.1

Enrichment step for MS Word (`docx`) documents.

Input

Gets a `docx` document as input.

Output

Results in a `docx` document as input.

Options

Options are used as a dictionary for rewrites with the following syntax:

- Keys can be prefixed with:
  - `rewrite:` and followed by path of file to be rewritten
  - *(currently there are no other prefixes then ``rewrite``)*
- Values can be prefixed with:
  - `static:` and followed by path to a file in a template; then it is used as-is to rewrite the original file in `docx`
  - `render:` and followed by path to a file in a template; then it is rendered first as `jinja` template with document context provided and result is used to rewrite the original file in `docx`

Notes

- Internally, the step unpacks the provided `docx` file, makes adjustments on the level of internal `XML` (and other) files, and packs it back to `docx`.
- To figure out what to rewrite, you should first generate the `docx` later used as input, unzip it and go through the contents.
- A good way to adjust things is to put there some placeholder first (e.g. via `reference.docx` passed to `pandoc`) and then just adjust the placeholder with other / dynamic content.
- Paths to files in a template are relative to template root, i.e. directory with `template.json`.
- It does not matter if the file to be rewritten is missing in the `docx`, then the desired file is simply added.
- The document context is provided in `ctx` variable, other variables, filters, and tests are documented in other documents (same as for `jinja` step).

Example

```
{
  "name": "enrich-docx",
  "options": {
    "rewrite:word/footer1.xml": "static:src/docx/footer1.xml",
    "rewrite:word/header1.xml": "render:src/docx/header1.xml.j2"
  }
}
```

Step: `excel`

status stable metamodel version ≥ 11

Step producing Excel spreadsheets from JSON file with instructions.

Input

JSON file containing instructions how to construct the desired Excel spreadsheet as described further in this section.

Properties

It allows to set both basic and custom properties of the workbook / spreadsheet.

```
{
  "properties": {
    "document": {
      "title": "My example workbook",
      "subject": "",
      "author": "Albert Einstein",
      "manager": "",
      "company": "ACME",
      "category": "",
      "keywords": "test,example,foo,bar",
      "created": "2018-01-01",
      "comments": ""
    },
    "custom": [
      {
        "projectUuid": "...",
      }
    ]
  }
}
```

Options

It allows to specify various workbook options.

```
{
  "options": {
    "strings_to_numbers": true,
    "strings_to_urls": true,
    "use_future_functions": true,
    "max_url_length": 255,
    "nan_inf_to_errors": true,
    "default_date_format": null,
    "remove_timezone": true,
    "use_zip64": false,
    "date_1904": false,
    "calc_mode": "auto",
    "read_only_recommended": false,
    "active_sheet": 0,
    "vba_name": "foo",
    "size": {
      "width": 0,
      "height": 0
    },
    "tab_ratio": 50
  }
}
```

Notes:

- `active_sheet` is an index of sheet to be active when document is opened.
- `size` sets the default window size.
- `tab_ratio` sets ratio between the worksheet tabs and the horizontal slider.

Definitions

It allows to define a name to be then used as a variable (see define_name).

```
{
  "definitions": {
    "Exchange_rate": "=0.96"
  }
}
```

Formats

It specifies formats in the spreadsheets that can be then used for cells in sheets. Possible options can be found in the documentation.

```
{
  "formats": {
    "myBoldFormat": {
      "bold": true
    }
  }
}
```

The example above creates a format named `myBoldFormat` that has bold text.

Charts

It specifies charts that can be then inserted inside sheets or used as chartsheets.

The options are documented here. Basically, each chart must have a unique `name` and then can have some `options`, series, and `axis` (e.g. x axis).

Finally, there are some basic and advanced settings:

- Basic: size, title, legend, chartarea, plotarea, style, table
- Advanced: combine, up_down_bars, drop_lines, high_low_lines, show_blanks_as, show_hidden_data

```
{
  "name": "myChartA",
  "combine": "myChartB",
  "options": {
    "type": "bar",
    "subtype": "percent_stacked"
  },
  "series": [
    {
      "name": "=Sheet1!$B$1",
      "categories": "=Sheet1!$A$2:$A$7",
      "values": "=Sheet1!$B$2:$B$7"
    }
  ],
  "axis": {
    "x": {"name": "Test number"},
    "y": {"name": "Sample length (mm)"}
  }
}
```

sheets

It is the main part specifying a list of sheets in the workbook, where each sheet has name (optional), type (optional, work or chart), options and then based on the type it has either chart (for chartsheet) or data (for worksheet). Some of the options are common for both chartsheet and datasheet. The order of sheets in the list corresponds to the order in the Excel spreadsheet.

Chartsheet

A chartsheet simply refers to a chart (by its name) that should be placed in this chartsheet.

The possible options are:

- Basic: first_sheet, protect, zoom, tab_color, page_view, select, hide
- Print: orientation (landspace or portrait), paper, margins, header, footer, center_horizontally, center_vertically

```
{
  "name": "Nice chart",
  "type": "chart",
  "chart": "myChartA",
  "options": {
    "tab_color": "red"
  }
}
```

Worksheet

Traditional worksheet with many options and data placed into cells. There are more options when compared to chartsheets.

The possible options are:

- Basic (common): first_sheet, protect, zoom, tab_color, page_view, select, hide
- Print (common): orientation (landspace or portrait), paper, margins, header, footer, center_horizontally, center_vertically
- Basic: comments_author, hide_zero, hide_row_col_headers, right_to_left, hide_gridlines, ignore_errors, vba_name
- Print (advanced): print_row_col_headers, print_area, print_across, fit_to_pages, start_page, print_scale, print_black_and_white
- Special ranges: unprotect_ranges, top_left_cell, selection
- Repeats: repeat_rows, repeat_columns, default_row
- Paging: h_pagebreaks, v_pagebreaks, outline_settings
- Panes: split_panes, freeze_panes
- Filters: filter_column_lists each with col and filters, filter_columns each with col and criteria, autofilter with range directly or first_row, first_col, last_row, last_col attributes (see docs)

- Merge ranges: merge_ranges list can be used to merge cells (range or combination of first_row, first_col, last_row, last_col attributes) together and apply a format (via format reference attribute), also contents can be set via data attribute
- Data validations: data_validations list can be used to validate data in cell ranges (see docs)
- Conditional formats: conditional_formats list can be used to define conditional formats on cell ranges (see docs)
- Tables: tables list can be used to define formatted tables (see docs)
- Sparklines: sparklines list (see docs)
- Row/col sizing: columns, column_pixels, rows, and row_pixels lists can be used to adjust cell sizing (unfortunately automatic sizing is not possible)
- Background: background can be used to set worksheet background via filename or b64bytes attributes

Inserting data

In JSON as part of worksheet's attribute data, you can in the list specify data to be inserted to cells in four ways (type):

- cell writes cell according to the possibly specifiedsubtype (see below)
- row writes row using provided data as a list
- column writes column using provided data as a list
- grid writes rows using provided data as a list of lists (list of rows)

For data, there are the following subtypes possible (for type set to cell):

- (unspecified) tries to directly all write with provided arguments, type should be then decided based on provided values and attributes
- string writes string from value
- number writes number from numeric value
- datetime writes datetime; it tries to parse date/datetime value from string as JSON does not have a format for datetime, standard ISO formats are recommended (e.g. 2022-12-24 or 2022-12-24T12:00:00Z)
- formula writes formula with formula in value and optional result value
- blank writes blank value (no attributes except format)
- boolean writes boolean value with boolean value (i.e. true or false)
- url writes URL value with url, value, and tip attributes
- rich_string writes rich string that allows formatting; for using format in the string_parts use prefix !fmt:: before name of the desired format

All options above may specify format (refer to defined format via its name).

```json
{
  "type": "cell",
  "subtype": "string",
  "cell": "A1",
  "value": "X"
}
```

```
{
  "type": "column",
  "subtype": "string",
  "cell": "A3",
  "data": [
    "ID",
    "Name",
    "Created at",
    "Author"
  ],
  "format": "myBoldFormat"
}
```

```
{
  "type": "grid",
  "row": 5,
  "col": 5,
  "data": [
    ["A", "B", "C"],
    ["D", "E", "F"]
  ]
}
```

Inserting other elements

Aside from data in cell, there is also possibility to insert other elements to the worksheet (`type` vales):

- `button` with options such as `macro` or `caption`
- `textbox` with `text` and options such as styling or offset in pixels
- `comment` with `comment` text and options such as `color` or `author`
- `chart` with `chart` (name) and options
- `image` with `filename`, `b64bytes`, and options

All of these are used with corresponding `type` and are placed to desired `cell` (or `col`/`row` indices).

```
{
  "type": "button",
  "cell": "B5",
  "options": {
    "caption": "Press Me"
  }
}
```

Header, Footer, Images

To allow easily add figures, those can be supplied as BASE64 encoded data directly via JSON as shown in the examples below.

For header and footer, the syntax of `content` is according to the documentation.

```
{
  "header": {
    "content": "&L&G &CExample Excel Document",
    "options": {
      "image_left": "logo.png",
      "image_data_left": "data:image/png;base64,iVBORw0KGgoAA..."
    }
  }
}
```

```
{
  "name": "bg-test",
  "options": {
    "background": { "b64bytes": "data:image/png;base64,iVBORw0KGgoAA..." }
  }
}
```

```
{
  "type": "image",
  "cell": "D2",
  "filename": "logo.png",
  "b64bytes": "data:image/png;base64,iVBORw0KGgoAA...",
  "options": {
    "x_scale": 0.6,
    "y_scale": 0.6,
    "url": "https://ds-wizard.org"
  }
}
```

`vba_projects`

List of VBA projects (with macros) to be embedded in the spreadsheet.

```
{
  "vba_projects": [
    {
      "project": "./vbaProject.bin",
      "is_stream": false
    }
  ]
}
```

Output

Desired Excel spreadsheet based on instructions from input JSON, it can be one the following formats (whether it uses macros or not):

- `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet` (extension `.xlsx`)
- `application/vnd.ms-excel.sheet.macroEnabled.12` (extension `.xlsm`)

Options

No options, everything comes from the input JSON file

Notes

- XlxsWriter library is used to construct Excel spreadsheet.
- Most likely this step will follow `jinja` step that constructs the JSON file.

Example

```
{
  "name": "excel",
  "options": {}
}
```

Step: `jinja`

status stable metamodel version ≥ 1

Renders requested Jinja2 template with document context and optionally other data.

Input

If not used as a first step, then the previous document is available from `document` variable.

Output

Results to a file of specified type (via `content-type` option) and file extension (via `extension` option).

Options

- `template` = path to template file to be rendered
- `content-type` = MIME type of resulting file
- `extension` = file extension for the produced file (without leading dot)

Optional:

- `jinja-ext` = comma-separated list of Jinja2 extensions to be enabled (supported values: `debug`)
- `i18n-dir` = location (relative to template root) of translations
- `i18n-domain` = domain string of translations
- `i18n-lang` = language code used in the template
- `extras` = comma-separated list of related entities to query in addition to *Document Context* (possible values: `submissions`, `questionnaire`); values will be added to `extras` attribute of the document context

Template (Jinja2)

Variables

The following variables are set:

- `ctx` = contains JSON-like plain *Document Context* (possibly with `extras` attribute, if configured)
- `secrets` = dictionary of secret values, only if enabled by configuration file
- `requests` = wrapper of requests module only if enabled by configuration file

Filters

Within Jinja templates, you can use so-called filters.Basically, those are functions applied to a first argument using pipe `|` symbol.

Bultin Filters

There are several widely used builtin filters directly in Jinja.

Value Conversion

We provide several filters that can be used for conversion of values:

- `datetime_format` = *Formats timestamp*
  - Example: `x.created_at|datetime_format("%d/%m/%y")`
  - Arguments:
    * `iso_timestamp` - `datetime` or ISO 8601 `str`
    * `fmt` - datetime format passed to strftime
- `of_alphabet` = *Converts integer to characters*

    - Example: `x|of_alphabet`
    - It prints `a` (for 0) to `z` and then continues with `aa`, `ab`, etc.
    - Arguments:
        * `n` - integer >= 0, usually some index

- `roman` = *Converts integer to Roman numeral*
    - Example: `x|roman`
    - Arguments:
        * `n` - integer >= 0, usually some index
- `markdown` = *Converts markdown to HTML*
    - Example: `x|roman`
    - Arguments:
        * `md_text` - string containing Markdown syntax
- `dot` = *Ends sentence if not already ended*
    - Example: `"This sentence has no end"|dot`
    - Arguments:
        * `text`
- `extract` = *Extracts values from object by having keys*
    - Example: `entities.questions|extract([uuid1, uuid2, uuid3])`
    - Arguments:
        * `obj` - object for getting values (typically `dict`)
        * `keys` - list of keys to retrieve

Reply Helpers

These filters are handy when you need to work with `repliesMap` from the plain JSON-like context.

- `reply_path` = *Joins list of UUIDs into a path*
    - Example: `[uuid1, uuid2, uuid3]|reply_path`
    - Arguments:
        * `uuids` - list of UUIDs
- `find_reply` = *Tries to find a reply value using a path*
    - Example: `replies|find_reply(path, "list")`
    - Arguments: - `replies` - dict with replies - `path` - list of UUIDs or path-string - `xtype` (optional) - desired type of return value (`"string"`, `"int"`, `"float"`, `"list"`)
- `reply_str_value` = *Extracts string value from a reply if possible*
    - Returns an empty string if not possible to extract it from the reply. Suitable for `AnswerReply`, `StringReply` and `IntegrationReply`.
    - Example: `reply|reply_str_value`

  - Arguments: - `reply` - object that might a reply

- `reply_int_value` = *Extracts integer value from a reply if possible*
  - Returns zero if not possible to extract it from the reply. Suitable for `StringReply` with numeric value type.
  - Example: `reply|reply_int_value`
  - Arguments:
    * `reply` - object that might a reply
- `reply_float_value` = *Extracts float value from a reply if possible*
  - Returns zero if not possible to extract it from the reply. Suitable for `StringReply` with numeric value type.
  - Example: `reply|reply_float_value`
  - Arguments:
    * `reply` - object that might a reply
- `reply_items` = *Extracts list of strings from a reply if possible*
  - Returns empty list if not possible to extract it from the reply. Suitable for `MultiChoiceReply` and `ItemListReply`.
  - Example: `reply|reply_items`
  - Arguments:
    * `reply` - object that might a reply

Special

These filters are more complex and add various support to template development.

- `to_context_obj` = *Converts plain context to well-defined objects*
  - This filter is used for easier transition and might be removed in the future.
  - Arguments:
    * `ctx` - plain JSON-like document context

Tests

Within Jinja templates, you can use so-called tests. Basically, those are helpers usable in conditions after `is` keyword:

```
{% if loop.index is divisibleby 3 %}
    {# ... #}
{% endif %}
```

Bultin Tests

There are several widely used builtin tests directly in Jinja.

Custom Tests

- `not_empty` = *Checks if size of a collection is higher than 0*
  - Example: `items is not_empty`
- `of_type` = *Checks if an object is instance of a certain type / class*
  - The name must be a string; however, it is case-insensitive. It also checks all superclasses.
  - Example: `parent is of_type "ListQuestion"`

Notes

- All paths (e.g. for `import` or `extends` in Jinja2 templates are relative from the template root, i.e. directory with `template.json`).
- The do Jinja2 extension is enabled.
- Using file extension `.j2` or `.jinja2` for templates is just a convention.
- The *document context* is provided in `ctx` variable, other variables, filters, and tests are documented in other documents.

Example

```
{
  "name" : "jinja",
  "options" : {
    "template" : "src/default.html.j2",
    "content-type" : "text/html",
    "extension" : "html"
  }
}
```

Step: `json`

status stable | metamodel version ≥ 1

Trivial step that dumps document context to JSON file.

Input

No input from previous step (should be first step)

Output

Always results in a JSON file (`application/json`) with file extension `.json`.

Options

No options

Example

```
{
  "name" : "json",
  "options" : {}
}
```

Step: `pandoc`

status stable | metamodel version ≥ 1

Transformation step that converts Pandoc-compatible document formats.

Input

Gets a file from the previous step (otherwise it fails), format needs to be specified using `from` option.

Output

Results in a document in desired format specified using `to` option.

Options

- `from` = specification of the input format (passed to Pandoc via `--from`, see docs)
- `to` = specification of the output format (passed to Pandoc via `--to`, see docs)
- (optional) `args` = additional command line arguments passed to pandoc
- (optional, experimental) `filters` = additional Pandoc filters to be used, need to be located under `/pandoc/filters` directory (or other set by `PANDOC_FILTERS` environment variable), comma separated
- (optional, experimental) `template` = Pandoc template to be used, need to be located under `/pandoc/templates` directory (or other set by `PANDOC_TEMPLATES` environment variable)

Notes

- Pandoc filter `pandoc-docx-pagebreakpy` can be found in addons directory.
- Pandoc filter `pandoc-docx-pagebreakpy` will be removed with the next template metamodel version, use `` for the `filters` option instead.

Example

```
{
  "name" : "pandoc",
  "options" : {
    "from" : "html",
    "to" : "docx",
    "args": "--filter=pandoc-docx-pagebreakpy --reference-doc=src/reference.docx",
    "filters": "docx-pagebreak.lua, docx-toc.lua"
  }
}
```

Step: `rdflib-convert`

status stable | metamodel version ≥ 1

Transformation step that converts between RDF formats.

Input

Gets an RDF file from the previous step of one of the following formats: `rdf` (XML), `nt`, `n3`, `ttl`, `trig`, or `json-ld` (specified using `from` option).

Output

Results in an RDF document of one of the following formats: `rdf` (XML), `nt`, `n3`, `ttl`, `trig`, or `json-ld` (specified using `from` option).

Options

- `from` = specification of the input format (`rdf`, `nt`, `n3`, `ttl`, `trig`, `json-ld`)
- `to` = specification of the output format (`rdf`, `nt`, `n3`, `ttl`, `trig`, `json-ld`)

Example

```
{
  "name" : "rdflib-convert",
  "options" : {
    "from" : "ttl",
    "to" : "rdf"
  }
}
```

Step: `weasyprint`

status stable | metamodel version ≥ 1.2

Transformation step that converts HTML file from previous step to PDF using WeasyPrint.

Input

Gets HTML file from the previous step (otherwise it fails).

Output

Always results in a PDF file (`application/pdf`) with file extension `.pdf`.

Options

- (optional) `render.presentational_hints` = whether HTML presentational hints are followed (default: `False`)
- (optional) `render.optimize_size` = specify what should be optimized (`''`, `'fonts'`, `'images'`, `'fonts, images'`, default: `'fonts'`)
- (optional) `render.forms` = whether PDF forms have to be included (default: `False`)
- (optional) `pdf.zoom` = zoom value as a floating number (default: `'1'`)
- (optional) `pdf.variant` = a PDF variant name
- (optional) `pdf.version` = a PDF version number
- (optional) `pdf.custom_metadata` = whether custom HTML metadata should be stored in the generated PDF

Notes

- Check the official WeasyPrint documentation and examples for more information.

Example

```
{
  "name" : "weasyprint",
  "options" : {
    "render.optimize_size": "fonts,images",
    "render.forms": "True",
    "pdf.zoom": "1.2"
  }
}
```

TDK Config

Those are local-only metadata used for development of the template. You can use them in versioned `template.json` but those are never stored directly in FAIR Wizard.

- `version`: metadata version for needs of migrations
- `readmeFile`: files used to get content for `readme` of the template, usually `README.md`
- `files`: list of patterns to specify files that are part of the document template (it uses Git wildcard-match patterns, so you can also exclude files or directories)

Template Metamodels

Here are described the changes in metamodel for template specification as well as *document context* so developers can easily update their templates to a newer metamodel version when needed. It is also possible to check JSON schemas in higher detail, see *Metamodel Schemas*.

Version 12 (since 4.1.0)

- Dropped support of deprecated `wkhtmltopdf` (for PDF, `weasyprint` is used instead).
- Changed several properties of `Integration` and `IntegrationReply` to optional (see *Document Context*).

Version 11 (since 3.20.0)

- Removed `recommendedPackageId` from template metadata and `shortName` together with `color` from formats.

Version 10 (since 3.12.0)

- New possible value types for value questions: `DateTimeQuestionValueType`, `TimeQuestionValueType`, `EmailQuestionValueType`, `UrlQuestionValueType`, and `ColorQuestionValueType` (no changes needed in existing KM-specific templates).

Version 9 (since 3.10.0)

- If you are using integration object, the `requestItemUrl` is changed to `itemUrl`.
- Integrations now have type, where the new Widget Integration has a different fields than API Integration (see schema).

Version 8 (since 3.8.0)

- Annotations and integration HTTP headers are changed from dict-like object with string-string key and value to a list of string-string tuples. Be aware that now there can be more values with the same "key" but that is usually unlikely.

Version 7 (since 3.7.0)

- Added description and project tags to the questionnaire object (if you do not need them, nothing has to be changed in the template).

Version 6 (since 3.6.0)

- Integration item template replaced item name. In templates you probably need to rename for integrations the property `itemUrl` to `responseItemUrl`.

Version 5 (since 3.5.0)

- All KM entities has now annotations (key-value dictionary). If you do not want to use those in your template, no changes are required.

Version 4 (since 3.2.0)

- Levels are renamed into phases and are using UUIDs. Phases are as part of the KM in `knowledgeModel.entities` of the context.
- Metrics are now also identified by UUID and part of the KM.

Version 3 (since 2.12.0)

- Additional metadata about each replies has been added and structure of reply is changed (extra `.value` needed). In case you are using filters such as `reply_str_value` no changes are needed.
- For integration reply, the type values are renamed `IntegrationValue` -> `IntegrationType` and `PlainValue` -> `PlainType` for consistency.

Version 2 (since 2.6.0)

- Changed `questionnaireReplies` to use path-reply map and removed then redundant `questionnaireRepliesMap` from document context.
- Replies for list question represented as list of UUIDs instead of size used for numeric indexing.

Version 1 (since 2.5.0)

- Initial version of metamodel, introduced in FAIR Wizard 2.5.0 as start of versioning.

1.6.3 Integration Questions

FAIR Wizard can be integrated with other services using so called *integration question*. The answer to that type of question does not contain only the answer itself but also a link to that external resource (which can be done, for example, using a persistent identifier). Therefore, these answers help clearly understand what the researchers use and promotes interoperability.

Examples of such integrations that are used within Common DSW Knowledge Model is FAIRsharing or ROR.

There are two ways of how we can connect FAIR Wizard to these services:

- **API** - using an API provided by the external service to search for the results
- **Widget** - using a specialized widget implemented for the connection with the FAIR Wizard

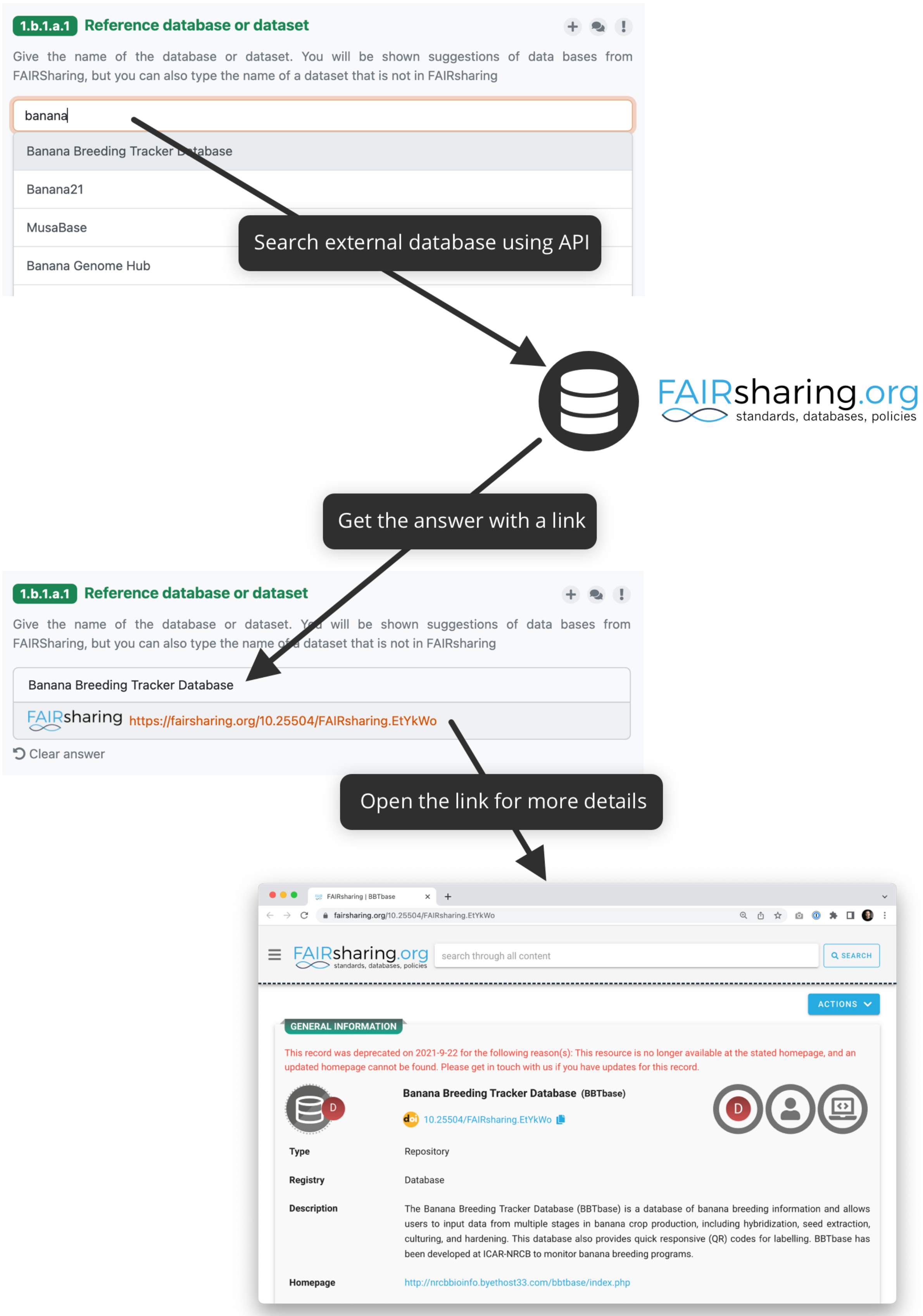
Integration Question - API

Integration question can be connected to an external resource using its API. We can then search for the results from the external service using the FAIR Wizard questionnaire interface. When we select an answer it is not only the text (such as a name of the database), but also a link to the external service to the selected item. The whole flow is denoted in the following diagram.

External Service Requirements

If we want to connect an external service using the API there are certain requirements for it to make the connection to FAIR Wizard possible.

- **Allows search using free text**
  - There must be a way to send a search phrase to the API so that it can filter the results based on it
- **Returns a JSON response with a list of results**
  - The response must be JSON so FAIR Wizard can parse it
  - There needs to be a JSON list where all the items matching the search query are


Search external database using API
Get the answer with a link
Open the link for more details
FAIRsharing.org
standards, databases, policies
+
ACTIONS
GENERAL INFORMATION
SEARCH
search through all content

1.b.1.a.1 Reference database or dataset

Give the name of the database or dataset. You will be shown suggestions of data bases from FAIRSharing, but you can also type the name of a dataset that is not in FAIRsharing

banana

Banana Breeding Tracker Database

Banana21

MusaBase

Banana Genome Hub

1.b.1.a.1 Reference database or dataset

Give the name of the database or dataset. You will be shown suggestions of data bases from FAIRSharing, but you can also type the name of a dataset that is not in FAIRsharing

Banana Breeding Tracker Database

FAIRsharing https://fairsharing.org/10.25504/FAIRsharing.EtYkWo

Clear answer

FAIRsharing | BBTbase

fairsharing.org/10.25504/FAIRsharing.EtYkWo

This record was deprecated on 2021-9-22 for the following reason(s): This resource is no longer available at the stated homepage, and an updated homepage cannot be found. Please get in touch with us if you have updates for this record.

Banana Breeding Tracker Database (BBTbase)

10.25504/FAIRsharing.EtYkWo

Type	Repository
Registry	Database
Description	The Banana Breeding Tracker Database (BBTbase) is a database of banana breeding information and allows users to input data from multiple stages in banana crop production, including hybridization, seed extraction, culturing, and hardening. This database also provides quick responsive (QR) codes for labelling. BBTbase has been developed at ICAR-NRCB to monitor banana breeding programs.
Homepage	http://nrcbbioinfo.byethost33.com/bbtbase/index.php

Fig. 115: How integration question connected to, for example, FAIRsharing API works.

- **It is possible to construct a link to the selected item**
  - We also need to be able to construct the link to the item from the data we get in the response so we can provide it with the answer

Configuration

The configuration is done in the *knowledge model editor*. First of all, we need to create a new integration and choose its **Type** to be **API**. Then, there are some metadata, such as **ID**, **Name**, or **Logo URL**.

Request Configuration

In the **Request** section, we configure how to make an HTTP requrests to the external service's API. For that, we need to configure the following (the specific values depends on how the API works):

- **Request URL** - what is the URL where we want to send search requests
- **Request HTTP Method** - what HTTP method should be used
- **Request HTTP Headers** - some headers might be needed, such as `Accept:  application/json` to have a correct response type
- **Request HTTP Body** - if we need to send some HTTP body
- **Allow Empty Search** - some APIs don't work if we try to search with an empty string, turn this of it's the case

There is a special property `${q}` that we can use within those fields. The property represents the string that users type to the questionnaire. So for example, we can write **Request URL** as:

```
http://example.com/api/search?q=${q}
```

Response Configuration

In the **Response** section, we configure how to process the JSON respnonse from the external service. For that, we need to configure the following:

- **Response List Field** - where in the JSON response is the list of items corresponding to the search query
- **Respone Item ID** - what field represents an item ID in the returned JSON
- **Response Item Template** - how we want to present the result to the user

We can use Jinja2 templates (Ginger implementation) in Response Item ID and especially in Response Item Template to make the response item look better.

Secrets and Other Properties

Sometimes, we might need to use some secrets (for example for authentication token), additional properties (such as API URL if we want to use different one for testing and production), or basically any information that we do not want to include in the knowledge model. In that case, we can define some properties in the FAIR Wizard settings.

We need to navigate to *Administration* → *Settings* → *Knowledge Models* and there is a field called **Integration Config**. It is a YAML organized by the **Integration ID** at the top level and key value pairs for each property.

We can fill some propertes in. So, for example, if the **INtegration ID** of our integration is *ourIntegration* we can write:

```
ourIntegration:
    authorizationToken: "abcd"
    apiUrl: "http://example.com/api"
```

Then, in the configuration of our integration, we can use these properties in the request configuration, so for example the **Request URL** can be:

```
${apiUrl}/search?q=${q}
```

And we can add a header such as:

```
Authorization: Bearer ${authorizationToken}
```

Note: These properties can be accessed only from the integration with matching ID.

Video Tutorial

We have the following video tutorial showing how to set up the integration question using API.

https://youtu.be/x-kx6ppVBo0

External Resources

- How to Configure Integration Question in FAIR Wizard
- How to Improve Integration Question Item Template in FAIR Wizard
- Ginger Documentation

Integration Question - Widget

Integration question can be connected to an external resource using a widget integration. When there is this type of question, instead of writing an answer, reserachers click on *Select* button. It will open the widget where they can pick their answer and it is then sent back to the FAIR Wizard. The whole flow is denoted in the following diagram.

Configuration

The configuration is done in the *knowledge model editor*. First of all, we need to create a new integration and choose its **Type** to be **Widget**. Then, there are some metadata, such as **ID**, **Name**, or **Logo URL**, as well as the **Widget URL** which is the URL where the widget is deployed. The URL will be open in popup window when researchers click on the *Select* button when filling in the questionnaire.

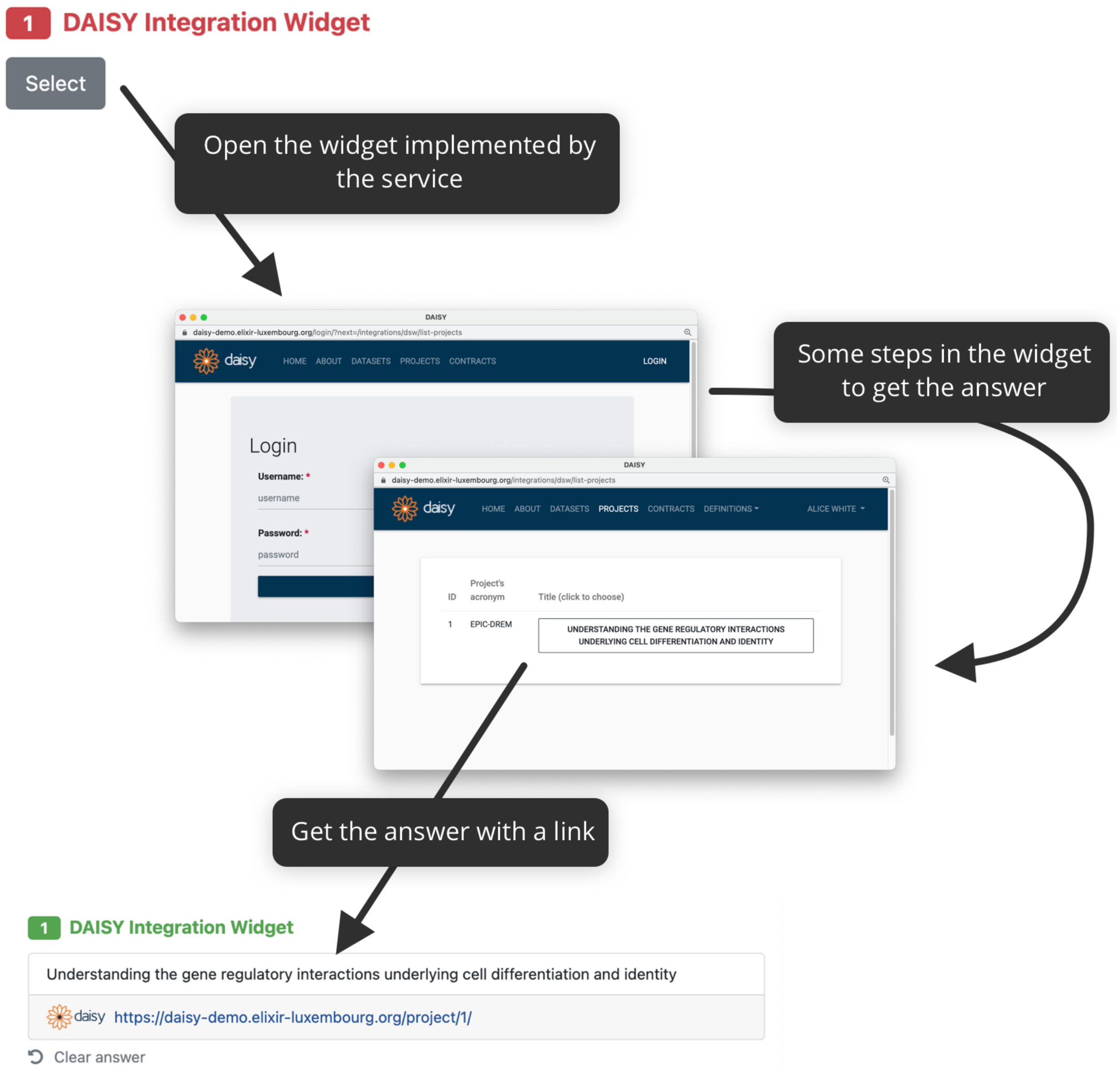

1 DAISY Integration Widget
Select
Open the widget implemented by the service
Some steps in the widget to get the answer
daisy HOME ABOUT DATASETS PROJECTS CONTRACTS LOGIN
Login
Username: *
username
Password: *
password
daisy HOME ABOUT DATASETS PROJECTS CONTRACTS DEFINITIONS ALICE WHITE
ID
Project's acronym
Title (click to choose)
1
EPIC-DREM
UNDERSTANDING THE GENE REGULATORY INTERACTIONS UNDERLYING CELL DIFFERENTIATION AND IDENTITY
Get the answer with a link
1 DAISY Integration Widget
Understanding the gene regulatory interactions underlying cell differentiation and identity
daisy https://daisy-demo.elixir-luxembourg.org/project/1/
Clear answer

Fig. 116: How integration widget connected to, for example, DAISY works.

Implementation

The actual implementation is done using DSW Integration SDK. We recommend reading the readme and explore the examples to understand how it works.

1.6.4 Project Importers Development

Warning: Project importers are an experimental feature.

Project importers can be used to import the data from an external resource to FAIR Wizard questionnaire. The importer creates the replies based on the data, therefore it needs to know the structure of the knowledge model it is compatible with.

FAIR Wizard comes with two importers already implemented.

- Replies Importer
- maDMP Importer

Note: More importers can be added in the future. Please contact the development team if you are interested in adding a new importer.

1.6.5 Submission Service

As administrators, we can configure submission services using *Document Submission Settings*. The configured HTTP request is then used when a user clicks *Submit* for an allowed document for submission and selected the desired submission service. The document is sent as a body of the request (or as multipart, based on the configuration) to the external service that should process it and return HTTP response with status code, and possibly also the *Location* header and some textual message.

Usually, we will need a simple proxy service to be developed that will accommodate this to API of some information system, database, storage, or other service. For example, such a proxy service will be able to receive the JSON documents from FAIR Wizard, retrieve additional information through FAIR Wizard API as needed, transform it to some other resulting artifact and store it in some local database that is used by other systems.

Example Submission Services

There are some submission services already implemented and can be used to check the implementation possibilities:

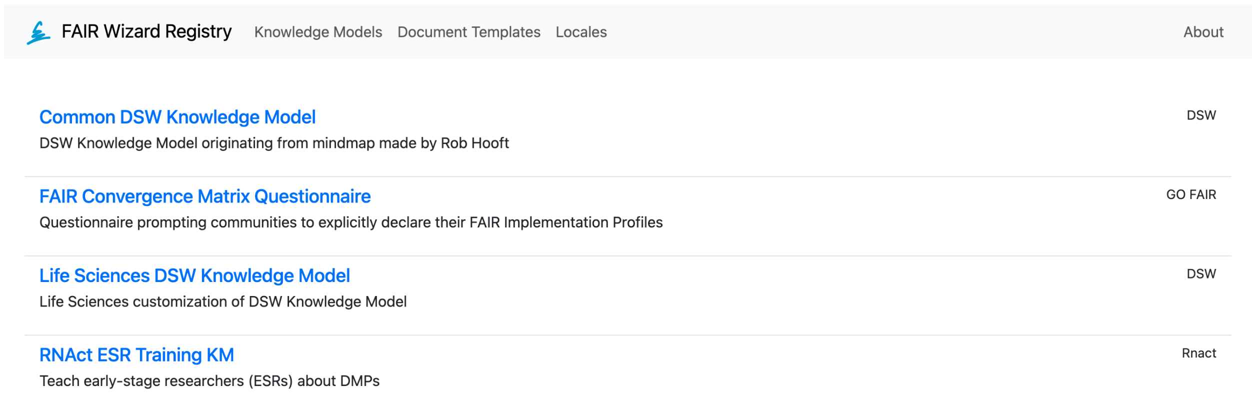
- Dummy Submission Service which just based on the headers returns example result or error.
- Email Submission Service sends an email through SMTP connection with the submitted document attached (or processed in case of JSON).
- Nanopub Submission Service allows to store a nanopublication (in RDF TRiG format) in the distributed network of nanopublication servers.

1.7 Miscellaneous

Additional information related to FAIR Wizard that might be useful.

1.7.1 FAIR Wizard Registry

FAIR Wizard Registry is a place where we publish knowledge models, document templates and locales. It is very easy to get those into a FAIR Wizard and use.



Common DSW Knowledge Model — DSW
DSW Knowledge Model originating from mindmap made by Rob Hooft

FAIR Convergence Matrix Questionnaire — GO FAIR
Questionnaire prompting communities to explicitly declare their FAIR Implementation Profiles

Life Sciences DSW Knowledge Model — DSW
Life Sciences customization of DSW Knowledge Model

RNAct ESR Training KM — Rnact
Teach early-stage researchers (ESRs) about DMPs

Fig. 117: FAIR Wizard Registry with a list of knowledge models.

Registry allows us to:

- *Import knowledge models from FAIR Wizard Registry*
- *Import document templates from FAIR Wizard Registry*
- *Import locales from FAIR Wizard Registry*

1.7.2 Markdown Cheatsheet

Various text fields in FAIR Wizard can be formatted by using Markdown formatting language. Here you can get a basic overview of what can be achieved with Markdown.

Basic Syntax

These are the basic Markdown elements supported by all applications.

Extended Syntax

These elements extending the basic syntax are supported in FAIR Wizard.