European
Plant Phenotyping
Network

EPPN 2020

D3.3: Connection and Integration framework and data discovery solutions

Asis Hallab (FZJ), Björn Usadel (FZJ)

This project has received funding from the European Union’s Horizon 2020 research and innovation programme u rant agreement No 731013. This publication reflects only the view of the author, and the European Commission c e held responsible for any use which may be made of the information contained therein.

Document information

EU Project N°	731013	**Acronym**	EPPN^{2020}
Full Title	European Plant Phenotyping Network 2020		
Project website	www.eppn2020.plant-phenotyping.eu		

Deliverable	**N°**	D3.3	**Title**	Connection and Integration framework and data discovery solutions
Work Package	**N°**	3	**Title**	Building a consistent Information System in the different nodes and defining standardisation strategies

Date of delivery	**Contractual**	30/04/2020 (Month 36)	**Actual**	17/06/2020 (Month 38)
Dissemination level	X	**PU Public, fully open, e.g. web**		
		CO Confidential, restricted under conditions set out in Model Grant Agreement		
		CI Classified, information as referred to in Commission Decision 2001/844/EC.		

Authors (Partner)	FZJ			
Responsible Authors	**Names**	Asis Hallab Björn Usadel	**Emails**	**a.hallab@fz-juelich.de** **b.usadel@fz-juelich.de**

Version log

Issue Date	**Revision N°**	**Author**	**Change**
14/05/2020	1	Cloé Paul-Victor	Reviewed by Project Manager
15/06/2020	2	François Tardieu	Reviewed by Coordinator

Executive Summary

Objectives

High throughput phenotyping research institutes produce large amounts of diverse and interdisciplinary datasets that are often hosted on site in data warehouses catering for specific local needs and requirements. However, Integrated access to data hosted in different warehouses at participating research institutes enables meta analysis and increase in statistical power. In order to make integrated data findable, accessible, interoperable, and reusable (FAIR) a standardized unified schema of data formats and standard access functions to this data has been proposed (see e.g. DJRA3.1 and DJRA3.2). A software design and its implementation is needed to integrate data from participating institutes according to the proposed standards.

Rationale

Standard access functions were proposed and analyzed with the goal to first fulfil FAIR data criteria and provide efficient algorithms for an implementation. Based on typical questions researchers have, an exhaustive search interface was proposed that enables any combinations of logical operators, values, and data models. Thus even complex searches can be executed like for example finding all plant height measurements of the model plant Arabidopsis under drought stress conditions. Algorithmic and technical restrictions in existing solutions and infrastructures were identified and analyzed. This analysis provided, as intermediate results, a list of clear concepts that must guide the following software design. Data must remain in the original warehouses, access must be implemented both locally in high level web-services that leave authorization and access control in the hand of the respective institute, and distributed access must be made efficient, i.e. queries must be answered in seconds. Furthermore, both the programmatic and the graphical interfaces must be usable intuitively and be well documented. Finally, the interfaces must be adaptable easily to change, that is new data formats and standards. All these considerations were taken into account and a software design proposed.

Main Results

The proposed software design was evaluated by participating partners and agreed upon. Its access functions fulfil the proposed restrictions and criteria. Furthermore, the interfaces to be developed are not programmed manually but automatically created by code generators, thus ensuring maximum flexibility and adaptability to new specifications. A first case study has been set up and integration of existing data warehouses is being planned for the coming months.

Table of contents

1. INTRODUCTION

One of the major goals in IT for phenotyping is to integrate and standardize high throughput plant phenotyping data generated at participating research institutes (RI). Standardization serves several purposes. Data will be easier to interpret and verified for consistency. This contributes to reproducibility of phenotyping experiments and greatly increases confidence in scientific results from such projects. Additionally, standardized data can be reused in meta-analysis with ease and without the need for complicated transformation and renormalization procedures. Furthermore, analyses software-pipelines once produced for such standards can be reused on future or larger data sets without the need to adjust data parsing methods. Finally, data can be made findable, accessible, interoperable, and reusable (FAIR). For this not only the data formats are to undergo standardization, but also the access functions to these formats should be standardized. This enables the creation of intuitive interfaces to phenotyping data, and most importantly facilitates integration of distributed data warehouses hosted at participating research institutes. Such standard interfaces should comprise exhaustive search mechanisms to enable a scientific user to find just the data needed for the purpose at hand, explore it, and form scientific hypotheses that can be tested downstream.

2. SOFTWARE DESIGN

The need for standardizing high throughput plant phenotyping data and providing standardized access functions to it has been recognized. A unified schema (EPPN schema) of standard formats (data models) has been proposed and agreed upon. The definitions were based upon already published and well established standards that were analyzed and extended upon. These standards were the **Breeding-API (BrAPI)** as a programmatical accession interface and the **Minimal Information about Plant Phenotyping (MIAPPE)** as expert derived standards about necessary metadata. Standardized access functions were defined together with EMPHASIS comprising exhaustive search queries, sorting, and subsetting (pagination) of data. Here too the design leverages an industry standard provided by Facebook's GraphQL. A software design was developed that integrates distributed data warehouses of participating research institutes through the usage of high level web services. Locally each RI installs a server that implements the API and transforms local data formats into a schema standard. These servers then provide secure access to their local data to a central data integration service (EDIS). The EDIS in turn provides the user with two interfaces that offer seamless access to all integrated data in the form of two interfaces. This comprises a graphical browser based interface (GUI) that uses the Material Design standard proposed by Google, thus giving a standard look and feel to the data exploration application. The second programmatic interface (API) enables the user to read data directly into analysis software-pipeline through simple web (HTTP) requests. This enables the use of the interface within any programming language. The programmatic interface encompasses also a browser based graphical support that helps the user to efficiently design, test, and execute queries that then can be included in analyses pipelines. This browser support also provides access to a complete documentation of both the schema and the API.

3. DEVELOPED TOOLS

The software design for the schema and the API is expected to undergo rapid evolution, because the field of high throughput plant phenotyping continually adds new methods and data formats to their methodologies. Thus the definitions forming the standards might change and evolve frequently to the advances made in this field of plant research. Manual programming of the API would thus be cumbersome and most likely ultimately either very costly or even doomed to fail. To prevent the project from suffering from this pitfall the solution of automated programming was selected. In this a set of software tools, so called code generators, actually consume as input the schema and API and automatically generate the graphical browser based and the programmatic interfaces that form the data integration service (EDIS). Furthermore, the high level web-services used at each local research institute to transform local data to the schema and implement the API, the adapters (EAs), are also automatically generated with code comments, alike to bookmarks for the programmer. These comments tell the programmer where to fill in just the bit of code that is needed to read data out of the respective local data warehouse. Because these warehouses all have different non standard internal structures this part of the EAs cannot be automatically generated. However, programming work is greatly minimized (see figure 2).

The code generators provide efficient and secure access to big data integrated from distributed data warehouses. User authentication and access control through role based authorization is performed using internet standards. In this, the security checks are executed both locally at each research institute at the level of the EA web-services, and in the central integrative EDIS. Thus each research institute retains full control over which data is made available to whom. Because of the nature of high throughput plant phenotyping the respective data is truly in the realm of so called “big data”. This requires access functions to apply efficiency algorithms so that the user has not to wait long times to receive responses to data queries. All necessary algorithmic adaptations have been made and efficient data access is guaranteed, even if some participating nodes, i.e. research institutes, might be unavailable during a data request.

The functionality and design of the code generators and how they produce the EDIS interfaces is depicted in figure one.

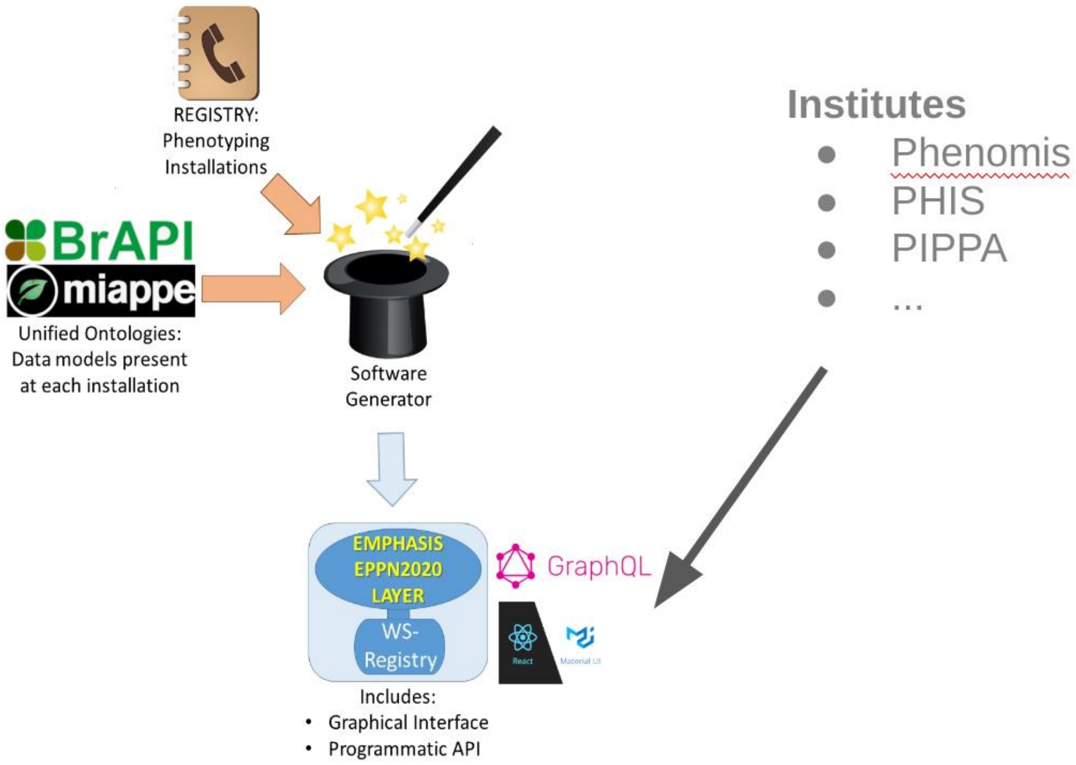


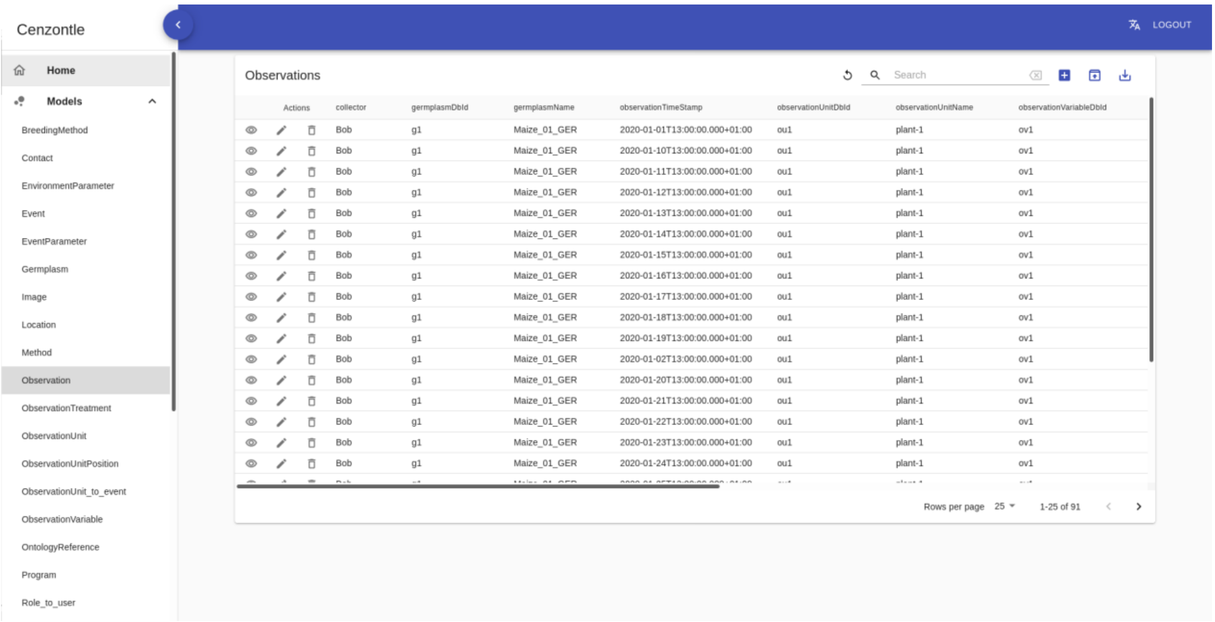
Figure 1: Using the code generators (magic hat) to create two interfaces to standardized data integrated from participating research institutes. The input for the code generators is a unified schema comprising data format definitions and access functions to it. Along with this a registry is provided which informs about participating research institutes and how to invoke their respective data delivery adapters. These adapters, in turn can be produced largely by the code generators and thus minimize programming effort. The result (output) of the process are two interfaces to the integrated data. A graphical browser based interface implementing Google’s material design standard and a programmatic data access interface leveraging Facebook’s GraphQL framework.

```
static readById(id) {
    /* YOUR CODE GOES HERE */
    throw new Error('ReadOne function is not implemented');
}

static countRecords(search) {
    /* YOUR CODE GOES HERE */
    throw new Error('Count function is not implemented');
}

static readAllCursor(search, order, pagination) {
    /* YOUR CODE GOES HERE */
    throw new Error('Cursor based pagination is not implemented');
}
```

Figure 2: Shown here is an example of an EPPN adapter automatically generated for each data model in the unified EPPN schema. The adapter implements the standardized access functions defined in the EPPN API. The workload and implementation difficulty is minimized for local programmers at a research institute integrating their data into the EPPN data integration service (EDIS) as they only have to fill in their code at the indicated lines (highlighted in light green).

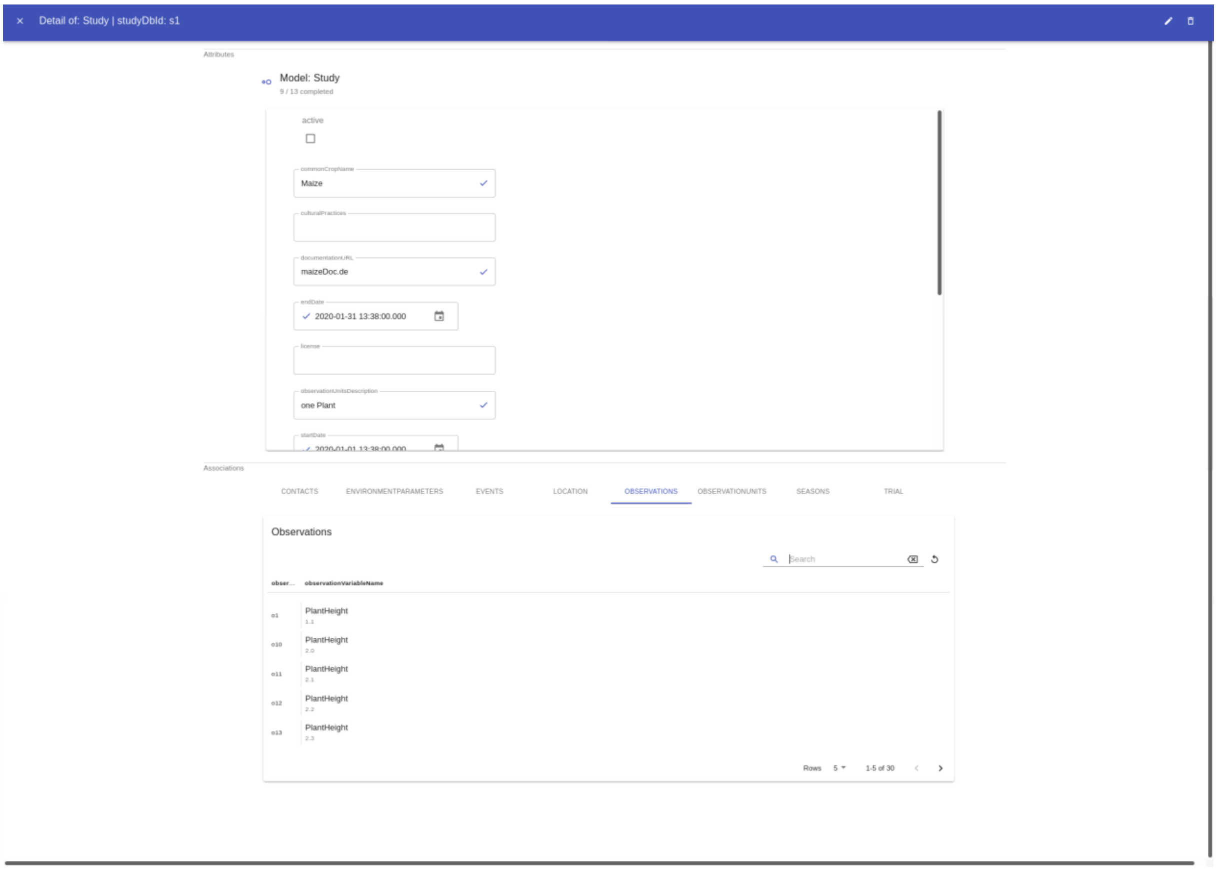
4. PROTOTYPE

A first prototype has been created using the unified schema and API definitions as input to the code generators. The graphical browser based interface implementing Google's standard material design is shown and explained in figure three (3.1 and 3.2). The programmatic interface enabling direct integration of data in analysis pipelines is explained in figure 4.



Actions	collector	germplasmDbId	germplasmName	observationTimeStamp	observationUnitDbId	observationUnitName	observationVariableDbId
	Bob	g1	Maize_01_GER	2020-01-01T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-10T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-11T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-12T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-13T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-14T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-15T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-16T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-17T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-18T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-19T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-02T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-20T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-21T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-22T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-23T13:00:00.000+01:00	ou1	plant-1	ov1
	Bob	g1	Maize_01_GER	2020-01-24T13:00:00.000+01:00	ou1	plant-1	ov1

Figure 3.1: Graphical browser based interface to standardized data integrated from participating research institutes. Shown here is a tabular display of "Observations", the result of a measurement e.g. plant height. On the left the user can select different data models for each of which a similar table is provided. The table can be sorted by each column. The user can search for records and paginate through results forward and backward. Here, page size can be set to different values.



Model: Study
9 / 13 completed

active
☐

commonCropName: Maize ✓

culturalPractices:

documentationURL: maizeDoc.de ✓

endDate: ✓ 2020-01-31 13:38:00.000

license:

observationUnitsDescription: one Plant ✓

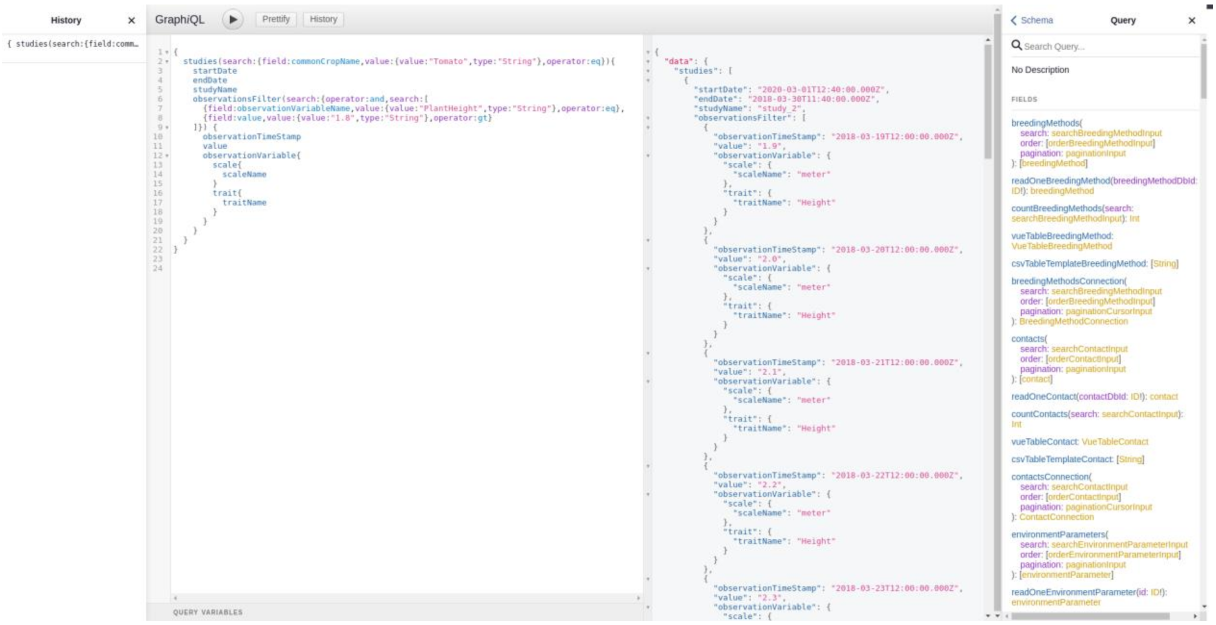
startDate: ✓ 2020-01-01 13:38:00.000

Observations

obser…	observationVariableName
o1	PlantHeight 1.1
o10	PlantHeight 2.0
o11	PlantHeight 2.1
o12	PlantHeight 2.2
o13	PlantHeight 2.3

Rows 5 1-5 of 30

Figure 3.2: Another part of the graphical browser based interface. Here a record is shown in detail including its associations (relations) to records of other data models. For example a study and the observations (see figure 3.1) made during the cause of the study. Note that all fields and attributes of the study are shown and that the associated observations are presented in a separate pane. Here, pagination and search is supported, so that the user can home in on the observation of interest. Tabs to other associations reside next to the observation tab.



```
{ studies(search:{field:comm…
```

```
{
  studies(search:{field:commonCropName,value:{value:"Tomato",type:"String"},operator:eq}){
    startDate
    endDate
    studyName
    observationsFilter(search:{operator:and,search:[
      {field:observationVariableName,value:{value:"PlantHeight",type:"String"},operator:eq},
      {field:value,value:{value:"1.8",type:"String"},operator:gt}
    ]}) {
      observationTimeStamp
      value
      observationVariable{
        scale{
          scaleName
        }
        trait{
          traitName
        }
      }
    }
  }
}
```

```
{
  "data": {
    "studies": [
      {
        "startDate": "2020-03-01T12:40:00.000Z",
        "endDate": "2018-03-30T11:40:00.000Z",
        "studyName": "study_2",
        "observationsFilter": [
          {
            "observationTimeStamp": "2018-03-19T12:00:00.000Z",
            "value": "1.9",
            "observationVariable": {
              "scale": {
                "scaleName": "meter"
              },
              "trait": {
                "traitName": "Height"
              }
            }
          },
          {
            "observationTimeStamp": "2018-03-20T12:00:00.000Z",
            "value": "2.0",
            "observationVariable": {
              "scale": {
                "scaleName": "meter"
              },
              "trait": {
                "traitName": "Height"
              }
            }
          },
          {
            "observationTimeStamp": "2018-03-21T12:00:00.000Z",
            "value": "2.1",
            "observationVariable": {
              "scale": {
                "scaleName": "meter"
              },
              "trait": {
                "traitName": "Height"
              }
            }
          },
          {
            "observationTimeStamp": "2018-03-22T12:00:00.000Z",
            "value": "2.2",
            "observationVariable": {
              "scale": {
                "scaleName": "meter"
              },
              "trait": {
                "traitName": "Height"
              }
            }
          },
          {
            "observationTimeStamp": "2018-03-23T12:00:00.000Z",
            "value": "2.3",
            "observationVariable": {
              "scale": {
```

No Description

FIELDS

```
breedingMethods(
  search: searchBreedingMethodInput
  order: [orderBreedingMethodInput]
  pagination: paginationInput
): [breedingMethod]

readOneBreedingMethod(breedingMethodDbId: ID!): breedingMethod

countBreedingMethods(search: searchBreedingMethodInput): Int

vueTableBreedingMethod: VueTableBreedingMethod

csvTableTemplateBreedingMethod: [String]

breedingMethodsConnection(
  search: searchBreedingMethodInput
  order: [orderBreedingMethodInput]
  pagination: paginationCursorInput
): BreedingMethodConnection

contacts(
  search: searchContactInput
  order: [orderContactInput]
  pagination: paginationInput
): [contact]

readOneContact(contactDbId: ID!): contact

countContacts(search: searchContactInput): Int

vueTableContact: VueTableContact

csvTableTemplateContact: [String]

contactsConnection(
  search: searchContactInput
  order: [orderContactInput]
  pagination: paginationCursorInput
): ContactConnection

environmentParameters(
  search: searchEnvironmentParameterInput
  order: [orderEnvironmentParameterInput]
  pagination: paginationInput
): [environmentParameter]

readOneEnvironmentParameter(id: ID!): environmentParameter
```

Figure 4: Programmatic interface to the standardized data obtained from integrated participating research institutes. Shown here is the included browser based “query builder” interface that helps the user to construct the ideal query to obtain exactly the desired data. The interface offers a

complete documentation of the data formats and access functions (seen in the column on the right), a query history (seen in the column on the left), automated formatting of the query (large pane and “Prettify” button), and automated execution of the query. The resulting data is shown in standard JSON format in the central right pane.

5. ONGOING DEVELOPMENT

Due to the Corona crisis already planned and scheduled meetings with the developers of each participating research institute had to be cancelled. In these meetings the adapters required to integrate the respective institutes’ data warehouse into the EDIS were to be adapted in the manner of a “hackathon” in the matter of three to five work days each. The development of the adapters is being re-planned in distributed remote work setups and will be executed in the coming months.