D5.7 Final Portal Release and System Operation Report

Document Identification			
Status	Final	**Due Date**	30/11/2021
Version	1.1	**Submission Date**	03/12/2021

Related WP		**Document Reference**	D5.7
Related Deliverable(s)	D5.1, D5.2, D5.3, D5.4, D5.6, D6.1, D6.2, D6.5	**Dissemination Level (*)**	Public
Lead Participant	USTUTT	**Lead Author**	Li Zhong (USTUTT)
Contributors	ATOS, USTUTT, PSNC, ICCS, KNOW, ECMWF, PLUS, SZE and BUL	**Reviewers**	Gregor Bankhamer (PLUS)
			Tomaschek Tamás Attila(MK)

Keywords:
HPCaaS, HIDALGO, CoE, Portal, Web, Entrypoint

Document Information

List of Contributors	
Name	Partner
F. Javier Nieto, Raul Santos, Jesús Ramos Rivas, Alexander Bruno	ATOS
Dineshkumar Rajagopal, Li Zhong, Dennis Hoppe, Sergiy Gogolenko, Leyla Kern	USTUTT
Krzesimir Samborski, Piotr Dzierżak, Marcin Lawenda	PSNC
Konstantinos Nikas, Dimitrios Siakavaras	ICCS
Manuela Rauch	KNOW
Milana Vuckovic, Claudio Iacopino	ECMWF

Document History			
Version	Date	Change editors	Changes
0.1	01/07/2021	Dineshkumar Rajagopal	TOC
0.2	30/09/2021	Li Zhong	Integrate the contribution from KNOW, PSNC, ECMWF and PLUS
0.3	15/10/2021	Li Zhong	Update on the tables 3,4
0.4	01.11.2021	Li Zhong	Integrate the changes in Askbot, etc.
0.5	16.11.2021	Li Zhong	Integrate the workflow orchestration
0.6	19.11.2021	Li Zhong	Integrate the contribution from ATOS
0.7	29.11.2021	Li Zhong	Revision based on reviewers' comments and feedback
1.0	30/11/2021	Li Zhong	Final version of the document
1.1	02/12/2021	Marcin Lawenda Francisco Javier Nieto de Santos	Changes incorporated by QM & PC

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Li Zhong (USTUTT)	30.11.2021
Quality manager	Marcin Lawenda (PSNC)	02.12.2021
Project Coordinator	Francisco Javier Nieto de Santos (ATOS)	03.12.2021

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
API	Advanced Programming Interface
A&A	Authentication and Authorization
CI/CD	Continuous Integration / Continuous Deployment
DC	Data Catalogue
DMS	Data Management System
DRF	Django-Rest-Framework
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
FAQ	Frequently Asked Questions
GDPR	General Data Protection Regulation
GSS	Global System Science
GUI	Graphical User Interface
IDAM	Intelligent Digital Asset Management
MooCs	Massive Open Online Courses
MSX	Project Milestone X
MVP	Minimum viable product
OIDC	OpenID Connect
PaaS	Platform as a Service
Q&A	Questions and Answers
REST	Representational State Transfer
SAML	Security Assertion Markup Language
SCM	Source Control Management
SPA	Single-Page Applications
SSH	Secure Shell
SSO	Single Sign On
Tx.y	Task in x work package and y task number
UUID	Universally Unique Identifier
VM	Virtual Machine
WP	Work Package

Table 1: List of acronyms

Executive Summary

This deliverable presents the final version of the HiDALGO Portal and its operations. First of all, it presents the main portal features, the selected architecture, and changes from the second version of this deliverable are pointed out. Noteworthy changes affected the following services and tools:

- Single-Sign-On,
- workflow orchestrator, the changes were done to support the pre-processing of ECMWF weather data in the ECMWF Cloud and the installation of Cloudify with the latest version.
- training, includes now more courses to disseminate the HiDALGO technology within the community and customize GUI with HiDALGO favicon.
- data management,
- visualization,
- Wiki, provide a public page for the *HiDALGO community interaction*.
- notebooks,
- matchmaking, extends its functionality from its previous version to refer the users by using the Machine Algorithm.
- marketplace,
- a new frontend and documentation.

For the *Single-Sign-On* (based on Keycloak), the document describes the changes done in the configuration and the integration with the rest of the components to provide coherent access to all the services in a single place. For the *workflows orchestration* (based on Croupier), this document addresses the changes done to support the pre-processing of ECMWF weather data in the ECMWF Cloud and the installation of Cloudify with the latest version. The *training tool* (based on Moodle) includes now more courses to disseminate the HiDALGO technology within the community. As the *data management and catalogue* (based on CKAN), we address the data-sharing features for CKAN, as well as the deployment of the system in the official HiDALGO sub-domain. In the case of *visualization* (based on Visualizer and COVISE), the document shows new features, including additional types of diagrams and 3D images that can be embedded into websites. This document also extends the information about the *Wiki* to provide a public page for the *HiDALGO community interaction*. It also introduces the new *interactive notebooks feature* (supporting C, C++, Python and R programming languages), so the users can now develop some test code and run it easily with the notebook. *Matchmaking* extends its functionality from its previous version to refer the users by using the Machine Algorithm and user's details with cutting edge technologies. The marketplace is a new service to buy all the HiDALGO products and services in a single place through the portal. The last feature addressed is the new frontend, as the means to centralize all the features and to

provide some enhanced functionality for executing applications. In this version of the Portal, the deliverable describes the system operations of Continuous Integration and Continuous Deployment based on Jenkins, the monitoring system in the infrastructure based on Zabbix, backup the services to improve the security and new HPC, HPDA infrastructure for pilots' execution. Finally, this deliverable presents the accomplishment in the customer support by listing the metrics achieved to meet project goals.

1 Introduction

HiDALGO portal development, infrastructure provisioning (HPC, HPDA and Cloud) and customer support are the key activities to establish a GSS (Global System Science) centre to satisfy the needs of all relevant stakeholders. The final release of the portal extended the functionality by several, crucial value-added services such as i) Single-Sign-On with the self-registration page ii) execution of pilot applications through cloudify iii) publishing training materials in Moodle iv) storing input/output data of simulations in CKAN v) matchmaking to connect with the users vi) Marketplace to buy the products and services vii) customer support from the HiDALGO professionals viii) collaborative editing with wiki and ix) providing a community forum for customer support. Furthermore, the matchmaking service is improved by switching to an Elasticsearch backend and group users by using the Machine Learning technologies to generate the RDF (Resource Description Framework) Graph. Matchmaking groups the users based on their initial answers and their profile details, which is finally integrated with the portal along with other services (Moodle, Marketplace, CKAN, Visualizer, Zammad, Askbot, Interactive notebooks, and Wiki). All services are made securely accessible using best practices by enabling SSO (Single Sign-On) with Keycloak IDM. All the pilot applications are executed through Cloudify and their workflow is defined in the Blueprint with the human-readable YAML file.

Besides the portal development work, Cloud infrastructure is provisioned for hosting the portal and its related services in the different Virtual Machines (VMs) to ensure both security and performance requirements of the service operations. Software deployment is adapted automatically via DevOps tools (Ansible and Jenkins) to improve the quality of the software development by ensuring seamless software integrations with Continuous Integration (CI) and Continuous Deployment (CD) principles. The overall HiDALGO portal development and automatic software deployment are accomplished to provide the services online to satisfy the customer needs.

For the workflow orchestration, an orchestrator based on Croupier [14] is updated, which introduced the possibility to declare scheduled and recurring executions on blueprints, and croupier has been updated to support it. Besides, Croupier has also been updated to allow users to define data sources for their applications. In this way, if an application needs to download data from a URL, users can just add a data node to their application and the required relationships and Croupier will download the necessary data into the needed infrastructure so it can be used by the application. The development regarding the workflow orchestration has been accomplished to meet the requirement of customers.

Customer support is key for establishing a direct and successful relationship with customers and to better understand their needs. Their needs will help HiDALGO to further grow the user community, and to build trust in order to establish a baseline for sustainability, community creation, and commercial service provisioning. The HiDALGO support concept is defined by a two-level support process for HiDALGO end-users. The service was, in comparison to the previous report, further extended to offer internal support within the consortium along with external customer support to address requests and questions in the most efficient way. For this purpose, the Zammad support ticket system is used as an internal bug tracking tool to track all the bugs within the HiDALGO development. The achievement in customer support is summarized at the end of this report by presenting relevant metrics (e.g., the number of Q&As threads closed, number of tickets resolved, and number of wiki pages created, to name but a few).

1.1 Purpose of the document

This document aims at reporting the implementation of the final release of the HiDALGO Portal, the HiDALGO infrastructure for hosting the portal and the customer support for assisting the end-users. The HiDALGO portal gives access to all the services in a simple way by a one-stop solution for the GSS (Global System Science) community, like training, execution of simulations, visualization of results, user support, interactive notebook and data management. The document goes through all the features implemented for the final version, including the implementation of the frontend (and backend) that puts all of them together, reducing the complexity to access the HiDALGO services. The user experience of the HiDALGO Portal is improved by monitoring the usage of the portal by enabling the web analytics service and the collection of user feedback from the end-users to improve the GUI (Graphical User Interface) design.

1.2 Relation to other project work

This document is directly related to D5.1 [1] since it describes the HiDALGO infrastructure and it is the base for executing the pilots' application and providing the CI/CD infrastructure for portal deployment. The features, designs and architecture of the portal are detailed in D5.2 [2], according to the requirements defined in D6.1 [6] and D6.5 [8], which is the base for the HiDALGO portal implementation. D5.3 [3] and D5.6 [5] detailed the implementations of the first and second versions of the HiDALGO portal, which is extended in this deliverable with the final implementation and provide the details of finalized features. D5.4 [4] detailed the process and tools of HiDALGO customer support, which is closely related to this deliverable by providing the metrics of real-time support operations and the changes introduced in the support tools to satisfy the project needs. It is also related to the pilots' applications in WP4

and workflows defined in D6.2 [7] to define the workflow of pilots in the Cloudify-TOSCA blueprint. In a summary, this is the final report from WP5 to provide a holistic view of the tasks - infrastructure operation in T5.1, customer support operations in T5.2, and portal development in T5.3.

1.3 Structure of the document

This document is structured in 5 major chapters:

- **Chapter 2** details the features that have been implemented in the context of the final release of the Portal, in line with the designs done in D5.2.
- **Chapter 3** describes the HiDALGO infrastructure to deploy and operate the portal by following CI/CD principles.
- **Chapter 4** provides a detailed description of the changes have been made on the Hidalgo's orchestrator Croupier
- **Chapter 5** reports the HiDALGO customer support and its operations with the details of accomplishment.
- **Chapter 6** provides a summary and a set of conclusions obtained after the final implementation and release of the HiDALGO portal.

2 HiDALGO Portal

2.1 Features, Architecture and Roadmap

The features and architecture of the Portal have not changed drastically during the last months. In terms of features, the main ones added are the following:

- The Matchmaking tool, so users can find colleagues and interact with other users in the HiDALGO community;
- The Marketplace, a tool that allows the users to publish and find services, applications and datasets.

The Matchmaking is a tool for building the HiDALGO community, where the users can answer a set of questions that will be used to determine the user profile. With such information, the matchmaking can match those users with similar interests, or who may need the knowledge/services of another user. Once a set of questions has been answered, the user will see compatible profiles of other HiDALGO users, so they can start interacting.

On the other hand, the Marketplace is a tool much more focused on the provision of applications and services in the context of HiDALGO. It is a catalogue where stakeholders (especially HiDALGO partners) can publish what they offer (from complete simulation workflows to private datasets). This component acts as a shop in such a way that the purchases can unlock the access to certain assets and resources available. For instance, if a user wants to run a specific simulation, it is necessary to, first of all, 'buy' the application in the marketplace. Once this is done, it will be possible for the user to access the application execution forms from the 'Dashboard' menu.

We updated the architecture to include the new components, showing how they are related to the rest of the Portal components (see Figure 1). In the case of the Matchmaking, it only interacts with the Frontend (that must show its GUI) and with the Authentication and Authorization component (for implementing the SSO mechanism). On the other hand, besides the Frontend and the Authentication and Authorization components, the Marketplace also needs to interact with the Backend, since it will be necessary to check specifically whether a user has access to a concrete application.

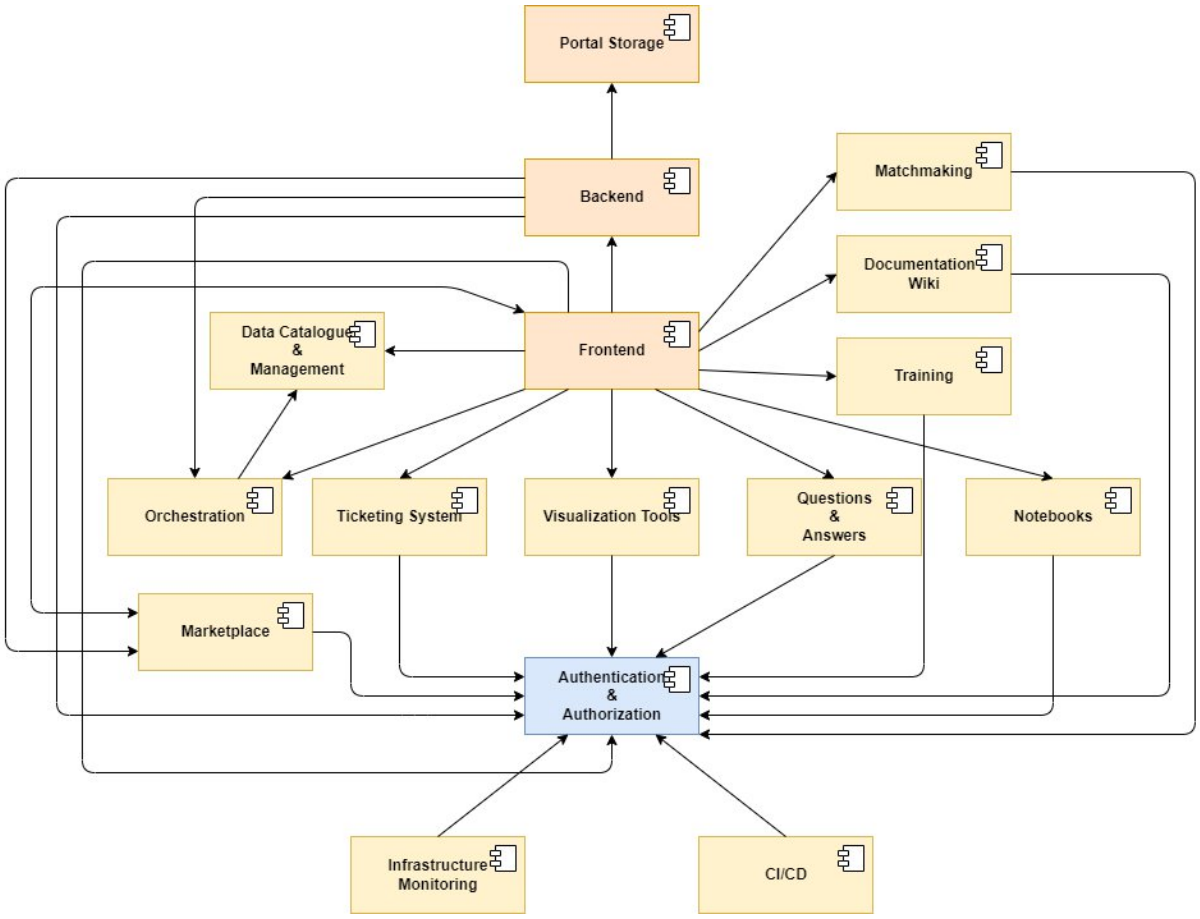

Figure 1: Final architecture of the HiDALGO Portal.

It is also interesting to highlight that the Authentication & Authorization component now includes an instance of Vault, as solution for securing credentials, together with the already existing instance of Keycloak [12]. Such addition requires modifications in the Frontend, so users will be allowed to manage the storage of their credentials at the Vault instance.

With all these new features and the modifications in the rest of components (explained in the subsequent sections), we consider that we fulfil those features that were defined in the roadmaps for MS4 and MS5, specified in D5.2 and reviewed in D5.6. Except for the feature "Developers can treat documentation as code, so it is easier to keep it updated" (that will not be implemented in the end, since it does not provide a new feature and we decided to focus on other developments with more added value), the rest of features are already implemented, or they will be ready in the following weeks, once the implementation and integration of the new features are completed.

The Portal has been already presented to the consortium twice internally (the first time in August and the second one in October), collecting valuable feedback that has been incorporated and triggering discussions on how to improve the security management. We are already preparing the beta testing campaign, that will be lunched before the end of the year.

2.2 Portal Frontend

The Portal has undergone several changes focused on increasing its functionality and improving users' experience through a better usability. These main changes are:

- Add a landing page with basic information and access to the main features,
- Add a page explaining how to start and a FAQ,
- Look & feel improved in several areas: messages for the user about actions done, information about the fields to fill in, the way to show lists of applications and instances, navigation, creation of new applications/instances, etc.,
- Improve the creation of dynamic forms (create forms for the input fields according to the blueprint types defined),
- Enable YAML editing in the input fields,
- Add list of events related to an application instance,
- Automatically fill in input parameters when loading from a file,
- Change user information, linked to the credentials stored in Vault (still ongoing).

When the user enters the Portal, the landing page is shown, so the user will login and will have access to the features of the Portal.

Figure 2: Homepage of the Portal

Users can self-register, since the email verification and captcha features have been activated in Keycloak. In case the user has no account, it is possible to register by clicking in the 'Register' link, which will show the corresponding GUI.

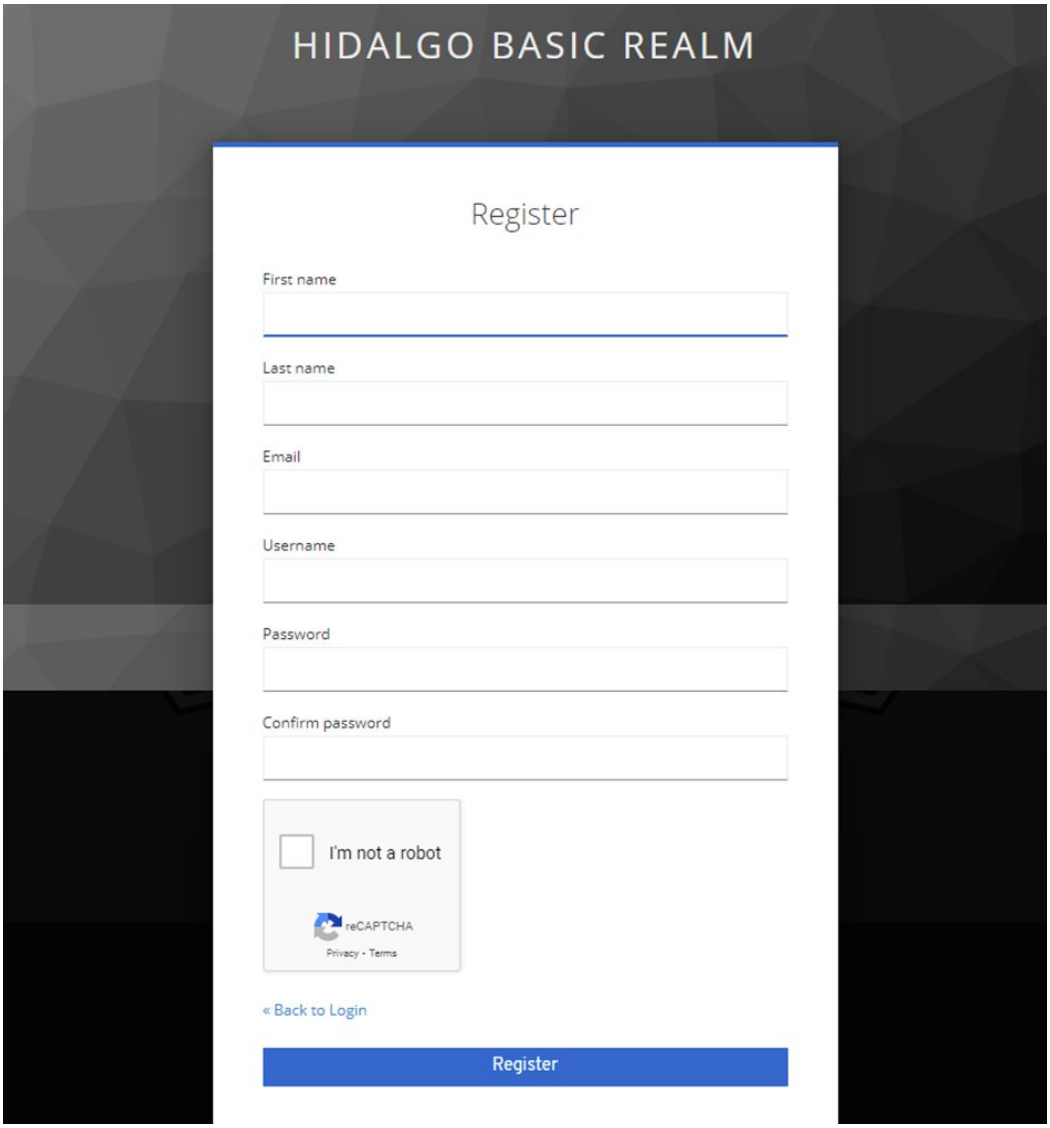

Figure 3: New user registration page with Google captcha

In order to access any service, the user can navigate through the side menu (at the left) and click on the required feature, so the corresponding page will be shown. It is possible to access:

- The interface to list and create applications,
- The dashboard for creating and executing instances,
- The training tool (Moodle),
- The data catalogue (CKAN [11]),
- The ticketing system (Zammad),
- The Jupyter Notebooks,
- The HiDALGO documentation (Wiki),

- The Visualization tools.

If the user needs some guidance, we have already included the 'Getting Started' and FAQ pages, so it is possible to access them once logged in, in order to learn the basics about the Portal and its features. Additionally, it can be also accessed with the question mark icon, in the top right part of the screen.

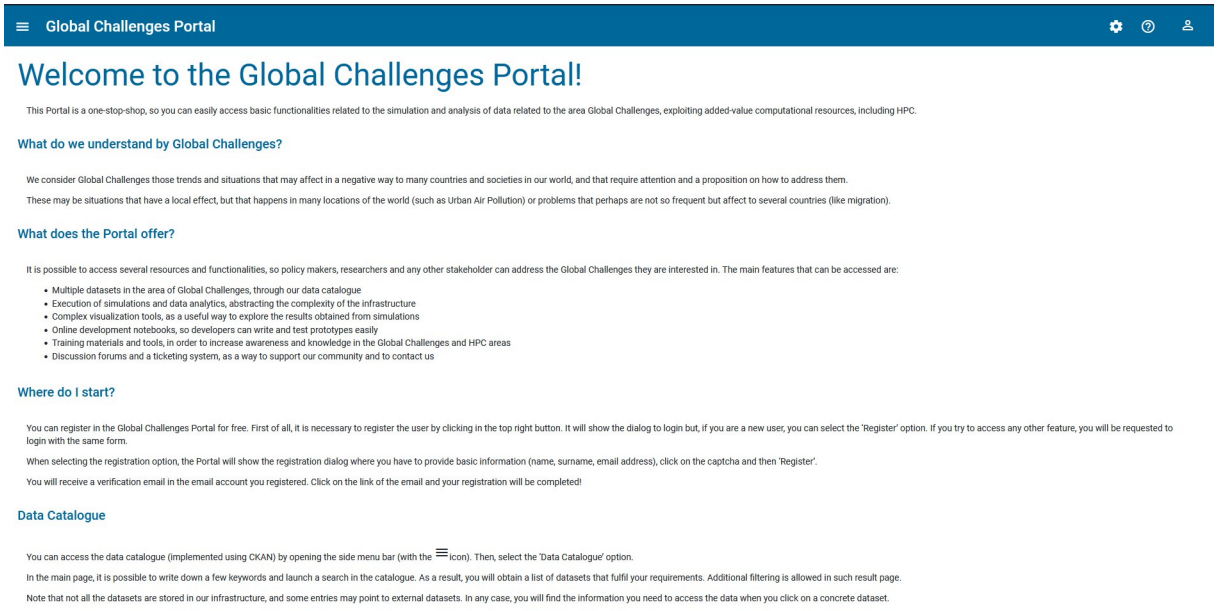

Figure 4: Getting started page of the Portal

In most of the cases, as the content is embedded in the Frontend with iFrames, all the content is navigated in the same area. As Zammad does not support the usage of iFrames as a way to integrate it, the tool is opened in a new tab of the navigator.

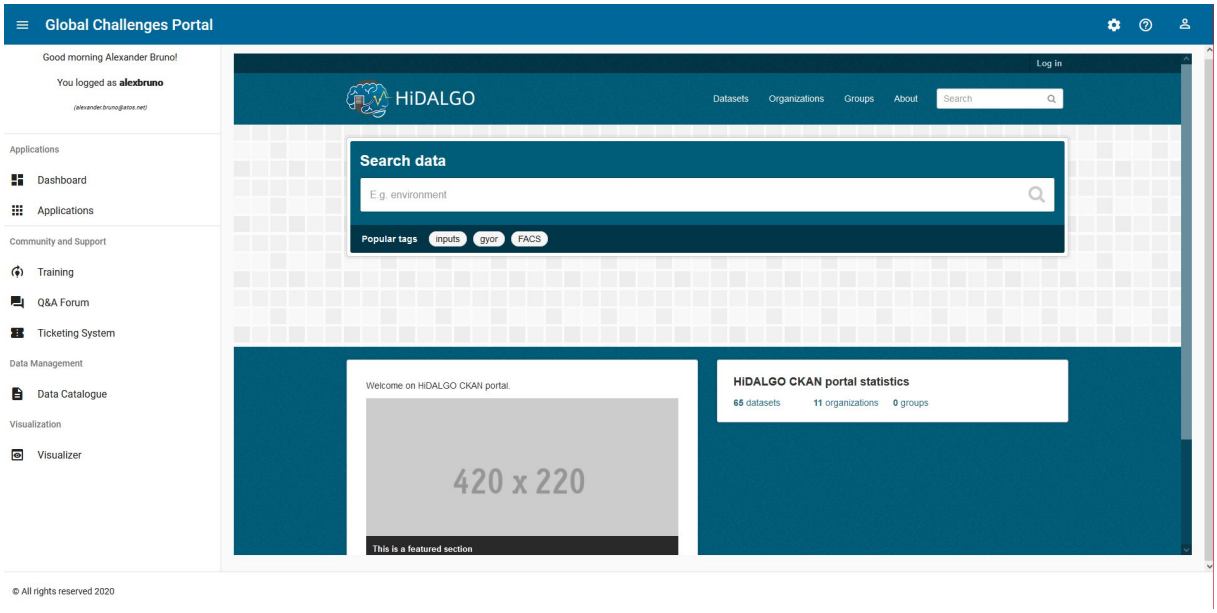

Figure 5: Example of tool integration in the frontend (CKAN)

In order to access the tools, users need to click in the corresponding 'Login' links. Then, the SSO authentication will grant them access. Once finished, the user can logout and close the session.

2.2.1 Execution of Use-case Applications

We facilitate the execution of the use case applications through the Portal by providing a set of dynamic forms that provide a simple way to fill in the required input parameters. We also enable easy execution of the same instance multiple times, in case the user does not need to change the configuration.

First of all, the user must be sure the application is already available in the Portal. It is possible to do so by going to the 'Applications' menu and looking at the list of applications. It is also possible to search by name, as shown in the previous section. In case that the application is not available, if the user has the 'developer' role, the option to add a new application will be enabled. Otherwise, the user should send a ticket to request: a) to be granted the role of developer and/or b) request the application to be added by an administrator.

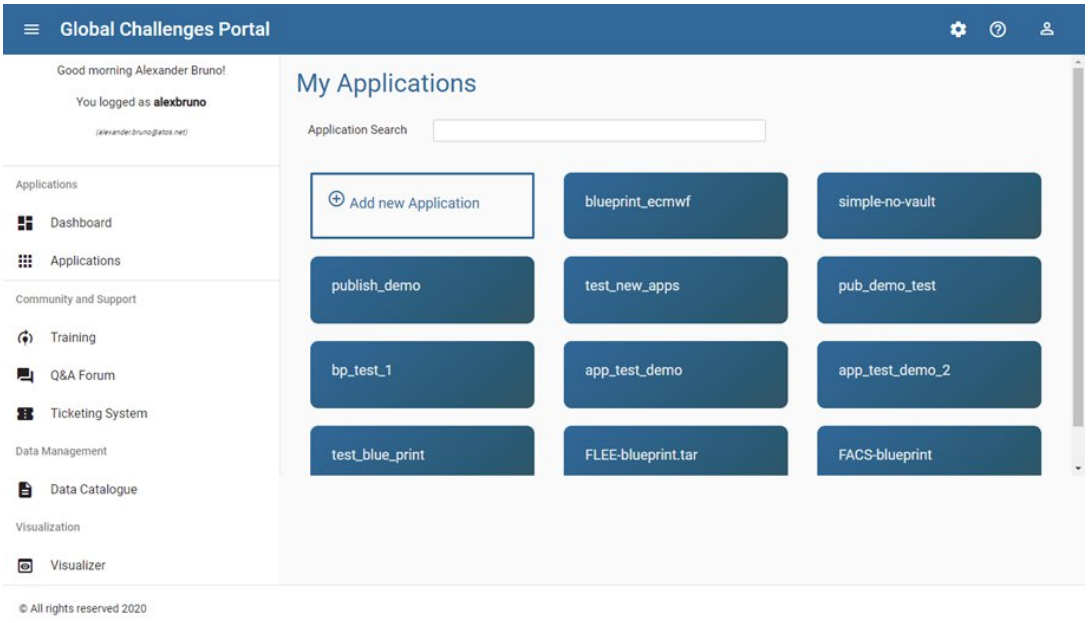

Figure 6: Applications view of the Portal

Once the application is available, the user has just to create a new instance. This is possible by going to the 'Dashboard' menu, so the instances available will be listed and the user will have the possibility to add a new instance for an application.

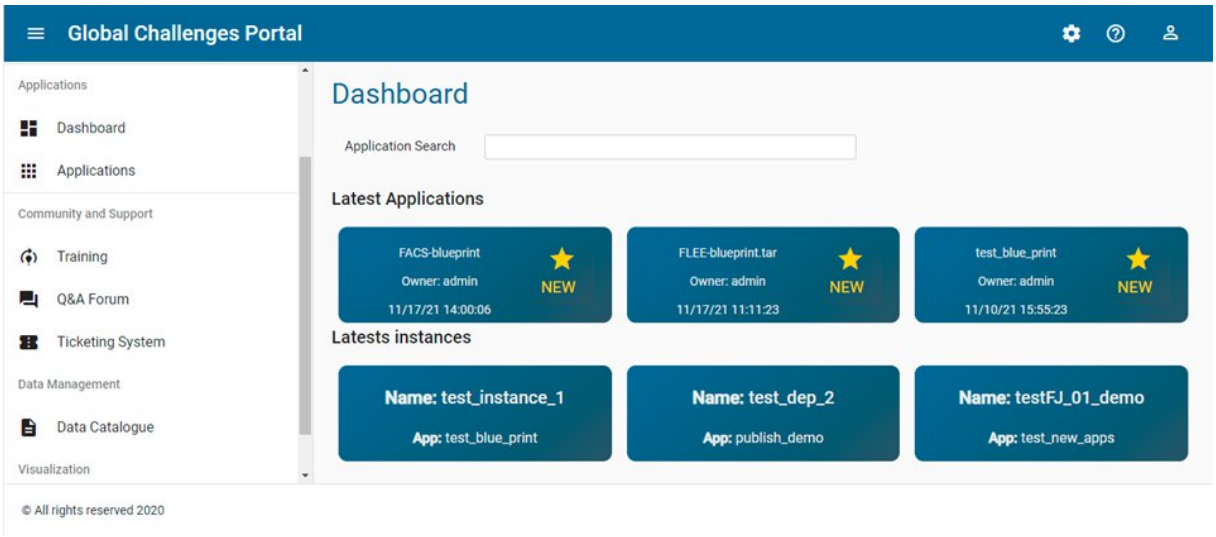

Figure 7: Dashboard view of the Portal

When the user selects the application, a page with three tabs is shown. The first tab is a form showing the basic inputs that must be provided (these are input parameters without a default value). The second tab shows a form with all the input parameters available for the application. The third tab is a simpler form that allows to upload a YAML file with the inputs.

In the first two tabs, the user can fill in the input parameters, according to the types definition. The Portal will support the users when defining the input parameters of the workflow (including complex structures) if the input data types were correctly defined in the blueprint. In those cases in which this is not available, it is possible to edit YAML code in a free ways (as it embeds a simple YAML editor).

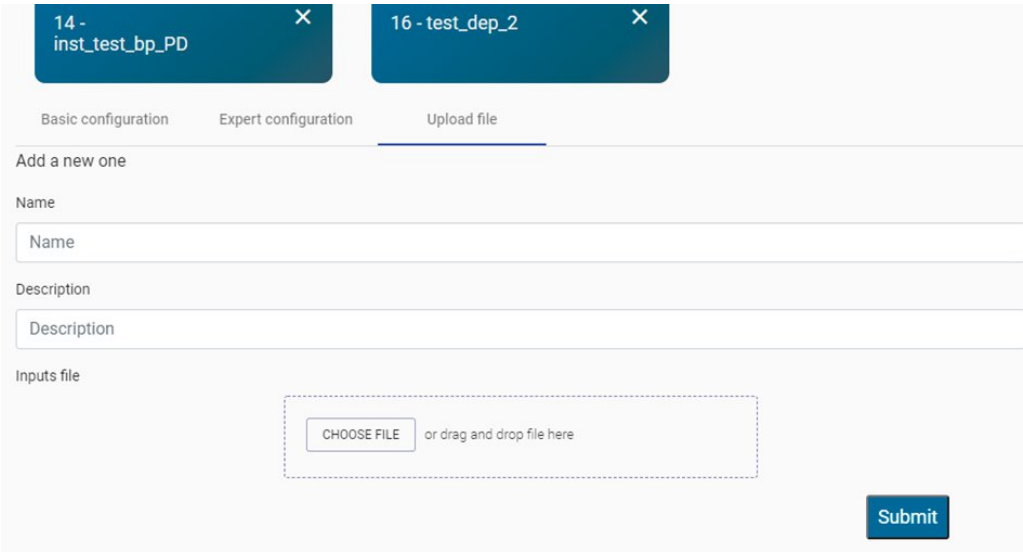

Figure 8: Forms for creating new instances of an application

When using an existing file, the Portal can load the file, extract the input parameters and fill in the corresponding fields in the forms (first and second tabs).

Once the configuration is ready, the user can create the instance with a single click and this one will be available for execution.

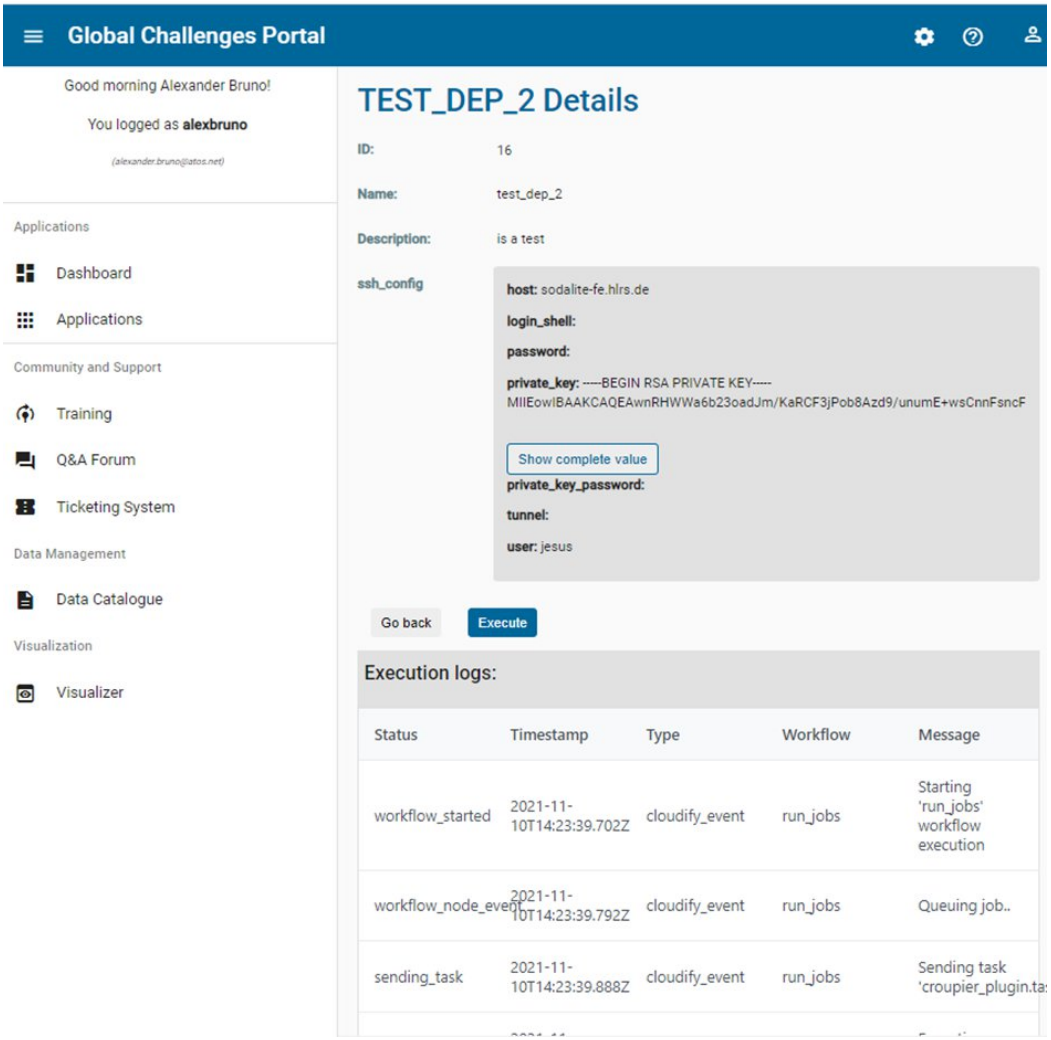

Figure 9: Instance execution view, listing the events of the instance

When executing an existing instance, the Portal will indicate that the execution is ongoing and it will show all the events that are generated while the application is running. The user is allowed to execute instances as many times as required.

2.2.2 Accessing the Value-added Services

As explained in D5.6, the Portal is a one-stop-shop for HiDALGO, collecting together all the added value services that we provide as a CoE. All these services can be accessed from the menu at the left. Such menu highlights the different categories and groups services by such categories, so it is easier for the user to find the required services. This menu can be accessed any time, and it can be hidden if necessary.

On the other hand, the new land page that the Portal has included provides direct access to the main features of the Portal. It is possible to access to the following services:

- Applications (access to the applications available);
- Data (access to the CKAN);
- Support (access to the Askbot);
- Deployment (access to the 'Dashboard');
- Documentation ('Wiki').

Since they are integrated with the Keycloak (as the implemented SSO mechanism), it is only necessary to login once in the Frontend. For accessing the rest of tools, their 'login' button will redirect them to the Keycloak instance, activating he SSO feature.

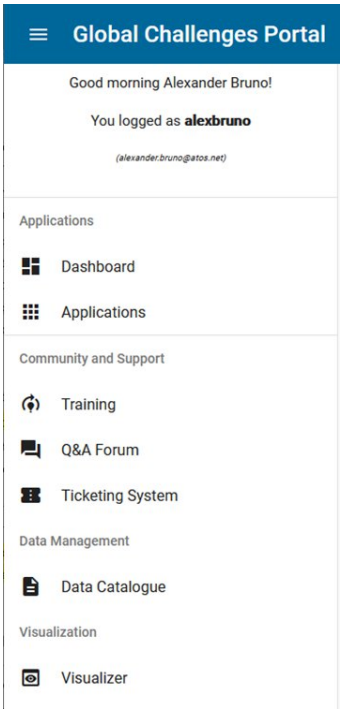

Figure 10: Portal menu

Thanks to the modular design, in case that additional value-added services are implemented, it will be possible to integrate them easily in the Portal.

2.3 Portal Integration and its Backend

The Portal separates the Frontend that shows the GUIs (implemented in Angular) from a Backend that manages connections with the Orchestrator (implemented in Python). The Portal already included a feature that allowed the upload of new applications, as well as the execution of new instances.

The Backend already included endpoints for:

- Listing, creating and accessing information of applications;
- Listing, creating and accessing information of instances;
- Execution of instances.

The background model of applications and instances has been improved, in order to include additional features. Now it is possible to:

- Filter the list of application and instances, by instance and application name,
- Filter the access to applications and features by user identifier, so it is possible to control the access per user (instead of having all the applications 'public' as created by an admin user),
- The backend is fully synchronized with the orchestrator, creating and removing applications and instances as they change in the orchestrator,
- Now the backend differentiates when an application/instance is new and when it has received some update (so the Frontend can highlight it). Instances can also provide more information about the input parameters,
- A new model has been created for executions, in order to enable a more detailed monitoring of the execution of instances,
- Retrieve the list of tasks/operations, in order to maintain information about the current execution of an instance,
- The full list of events of an instance can be retrieved.

As a consequence, the definition of the REST API and its usage has changed with respect to the previous version of the Portal[5]. The following table shows the usage of those methods that have been updated and of the new methods:

REST API endpoints	API Description
`apps?name={app_name}`	Used to collect the list of applications stored in the database (GET method). The parameter 'name' is a string for filtering applications that contain such string in their name.
`instances?name={i_name) &app={a_name}`	Used to collect the list of instances stored in the database (GET method). It filters by instance name (i_name) and application name (a_name)
`instances/{id}/events/`	Used to retrieve information about all the events associated to one instance, using the instance identifier (GET method).
`executions?inst={i_name} &app={a_name}`	It lists all the executions that are available in the backend. It can be filtered by instance name (inst

REST API endpoints	API Description
	parameter) and application name (app parameter).
`executions/{id}/`	It provides the complete information available about an execution.

Table 2: New Methods of the Backend REST API

According to the new implementation, the Backend adds additional information to the response when accessing the information of an application:


```
[
 {
  "id": 8,
  "name": "blueprint_ecmwf",
  "description": null,
  "owner": "admin",
  "main_blueprint_file": "blueprint_ecmwf.yaml",
  "created": "2021-07-29T08:16:49.165000Z",
  "included": "2021-08-17T06:46:49.067943Z",
  "updated": "2021-10-14T15:11:26.032000Z",
  "is_new": false,
  "is_updated": true
 }
]
```


When accessing instances, additional information is available as well (details of input parameters, information about update date and if it is new). The new JSON responses look like the following one:


```
{
 "id": 18,
 "name": "publish_01_demo",
 "description": null,
 "owner": "admin",
 "created": "2021-08-19T10:51:01.052000Z",
 "updated": "2021-08-19T10:51:01.052000Z",
 "app": 11,
 "last_execution": "3ca3f4f5-e8c1-40ef-a611-291038fb393b",
 "is_new": false,
 "inputs": "[[{\"name\": \"hpc_base_dir\", \"value\": \"$HOME\"}, {\"name\":
\"hpc_interface_config\", \"value\": {\"country_tz\": \"Europe/Madrid\",
\"infrastructure_interface\": \"SHELL\"}}, {\"name\": \"hpc_interface_credentials\",
\"value\": {\"host\": \"sodalite-fe.hlrs.de\", \"user\": \"xxxx\", \"private_key\": \"-----BEGIN
RSA PRIVATE KEY-----xxxxxxxxxxx-----END RSA PRIVATE KEY-----\\n\"}}, {\"name\":
\"ecmwf_ssh_credentials\", \"value\": {\"host\": \"136.156.90.143\", \"user\": \"xxxxxxxx\",
\"private_key\": \"-----BEGIN RSA PRIVATE KEY-----\\xxxxxxxxxxx-----END RSA PRIVATE KEY-----
\\n\"}}, {\"name\": \"keycloak_credentials\", \"value\": {\"user\": \"xxxx\", \"pw\": \"xxx\"}},
```

{\"name\": \"ecmwf_query\", \"value\": {\"params\": \"u/v\", \"date\": 20210503, \"area\": \"45/18/46/19\", \"time\": 0, \"collection\": \"hidalgo-test\"}}, {\"name\": \"ckan_outputs_dataset\", \"value\": {\"config\": {\"entrypoint\": \"https://hidalgo1.man.poznan.pl\", \"key\": \"xxxxxxx\"}, \"dataset\": {\"id\": \"tests_croupier\"}, \"type\": \"ckan\"}}, {\"name\": \"mpi_load_command\", \"value\": \"\"}, {\"name\": \"singularity_load_command\", \"value\": \"\"}, {\"name\": \"singularity_image_uri\", \"value\": \"\"}, {\"name\": \"singularity_image_filename\", \"value\": \"\"}, {\"name\": \"partition_name\", \"value\": \"default\"}, {\"name\": \"singularity_image_storage\", \"value\": \"$HOME\"}, {\"name\": \"singularity_mount_point\", \"value\": \"/mnt\"}, {\"name\": \"scratch_voulume_mount_point\", \"value\": \"/scratch\"}, {\"name\": \"job_config_content\", \"value\": \"\"}, {\"name\": \"job_config_pathname\", \"value\": \"\"}, {\"name\": \"monitoring_options\", \"value\": {}}, {\"name\": \"accounting_options\", \"value\": {}}, {\"name\": \"input_url\", \"value\": \"\"}, {\"name\": \"data_dest\", \"value\": \"$HOME\"}, {\"name\": \"test_file\", \"value\": \"Test file\"}, {\"name\": \"test_bool\", \"value\": false}, {\"name\": \"test_int\", \"value\": 0}, {\"name\": \"test_float\", \"value\": 0.0}, {\"name\": \"test_complex_input\", \"value\": {\"first_tag\": {\"first_first_tag\": \"test1.1\", \"first_second_tag\": {\"foo\": \"bar\"}}, \"second_tag\": \"test2.1\"}}, {\"name\": \"job_prefix\", \"value\": \"atos\"}, {\"name\": \"monitor_entrypoint\", \"value\": \"\"}], null]"
}

When listing events associated to an instance, the JSON generated is the following:

{
  "logs": [
    {
      "_storage_id": 613,
      "timestamp": "2021-11-05T12:28:58.293Z",
      "reported_timestamp": "2021-11-05T12:28:58.293Z",
      "blueprint_id": "test_new_apps",
      "deployment_id": "testFJ_01_demo",
      "deployment_display_name": "testFJ_01_demo",
      "execution_id": "b50a67c2-1f46-465b-8355-7ac4e76591ad",
      "execution_group_id": null,
      "workflow_id": "run_jobs",
      "message": "Task succeeded 'croupier_plugin.tasks.cleanup_job'",
      "error_causes": null,
      "event_type": "task_succeeded",
      "operation": null,
      "source_id": null,
      "target_id": null,
      "node_instance_id": "job_7ahkm8",
      "node_name": "job",
      "type": "cloudify_event"
    },
    {
      "_storage_id": 614,

```
    "timestamp": "2021-11-05T12:28:58.501Z",
    "reported_timestamp": "2021-11-05T12:28:58.501Z",
    "blueprint_id": "test_new_apps",
    "deployment_id": "testFJ_01_demo",
    "deployment_display_name": "testFJ_01_demo",
    "execution_id": "b50a67c2-1f46-465b-8355-7ac4e76591ad",
    "execution_group_id": null,
    "workflow_id": "run_jobs",
    "message": "'run_jobs' workflow execution succeeded",
    "error_causes": null,
    "event_type": "workflow_succeeded",
    "operation": null,
    "source_id": null,
    "target_id": null,
    "node_instance_id": null,
    "node_name": null,
    "type": "cloudify_event"
   }
 ],
 "last": 21,
 "status": "terminated"
}
```


In the case of executions, the following information is provided


```
[
 {
  "id": 13f79755-ebc2-4ec3-807f-8cce21c78a7c,
  "instance": "publish_01_demo",
  "created": "2021-08-19T10:54:27.023020Z ",
  "finished": "2021-08-19T10:55:42.130201Z ",
  "status": “Terminated”,
  "has_errors": false,
  "num_errors": 0
 }
]
```


2.3.1 Integration of Use-case applications

Each pilot application has its own blueprint which is uploaded to the orchestrator. Each deployment of the application is created based on its blueprint, therefore, the blueprint defines how the application will be launched. Blueprints for the different pilot applications have been updated to reflect and take advantage of the new functionalities included in the new version of the orchestrator. These new features are explained in section 5. Every blueprint has been modified so that no HPC credentials are passed through a deployment’s inputs, instead, the credentials used to connect to the HPC are stored on Vault, a much more

secure location, and only a temporary token is used as a deployment input. This token is used to retrieve all the credentials from Vault.

All the blueprints have also been modified to comply with the new datatypes defined in croupier. This has the benefit that if a user doesn't give the correct input fields when running an application, the deployment will fail at creation time, not at run time, which improves efficiency.

The UAP pilot has also been updated to utilize the new ECMWF integration with croupier. The blueprint now includes a new node representing the weather data needed to run the simulations. The orchestrator requests this data from ECMWF and downloads it to the working directory, as described in section 5.

2.3.1.1 Enhancing Data Management

All the pilot applications use CKAN whether to download data, upload data or both, therefore as part of their inputs they have the user's CKAN API token. As mentioned previously, all the inputs given to cloudify are not secure, so to manage secrets such as API tokens Vault should be used. In order to be able to use Vault to store the CKAN API token, all the calls to CKAN must be made from the orchestrator itself instead of from the application's scripts. Therefore, to support this feature, a new connector is being developed in the orchestrator to manage all the communications with CKAN. The new connector will allow a user to define which file from which dataset they need to download into their working directory directly on the blueprint by directly providing the URL or by giving keywords that will allow the orchestrator to find the desired dataset. The connector will also allow users to define destination datasets where they want their job results to be published.

As soon as this connector is completed all the pilot blueprints will be updated again to make use of this new feature, which will complete the securing of all the inputs.

2.3.2 Integration of Value-added services

As described in previous deliverables, the Keycloak is the component that takes care of the Single-Sign-On solution (SSO). It has been configured in such a way that we have a specific realm for HiDALGO, in which we configured all the components as clients enabled for SSO. Each component needs to configure its corresponding Keycloak client key and they have to configure the URL for the OAuth endpoints.

The Keycloak configuration is similar as before, but includes two changes:

- We modified the registration workflow, so now email verification is enabled and Google Captcha is integrated;

- We have defined two users' groups: Developers and Guests. We are using such groups to differentiate roles and disable the access to certain features.

Service Name	Application Name	SSO Support
Data Catalogue	CKAN	CKAN has SSO support. PSNC has installed and updated plugin to the CKAN. Users can log in using HiDALGO Keycloak.
Training	Moodle [10]	OAuth2
Users Matchmaking	Matchmaking	OpenIDC [17]
Interactive Notebook	Jupyter Hub	OAuth2
Visualization	Visualizer	Currently, Visualizer has no SSO support; however, it will be investigated whether the HiDALGO Keycloak instance can be used to verify user authentication.
	COVISE	No SSO support
Customer Support Tools	Zammad [16]	OAuth2
	Askbot	OAuth2
	Wiki [15]	OpenIDC
Customer Marketplace	WooCommerce	OAuth2

Table 3: Value-added services with the SSO support for the portal Integration.

2.4 Value-added Services in the Portal

The portal is not only a platform for executing pilot applications, but it provides different value-added services to satisfy the customer needs by offering various services as shown in Table 4. The implementation of value-added services in the final version of the portal and changes introduced from its predecessor are detailed in the rest of the chapter, which is highlighted below.

1. Training, data catalogue, visualization and support tools are stable and have been used in the operations since the first versions of the portal, so it is reported with the details of changes in the configuration and actual usage.
2. Matchmaking is enhanced to use the algorithms of Machine Learning to match users effectively by using the details of personal information (profession, location).
3. The Marketplace is a new service introduced in this version for buying the different HiDALGO products and services by customers securely in a single place.

Service Name	Application Name	Application Version
Data Catalogue	CKAN	V2.9.2
Training	Moodle	V3.7
Users Matchmaking	Matchmaking	V2.0
Interactive Notebook	Jupyter Hub	V0.9.0
Visualization	Visualizer	V2.0
	COVISE	v2020.9
Customer Support Tools	Zammad	V3.2
	Askbot	V0.11
	Wiki	V2.4.107
Customer Marketplace	Marketplace	V1.0

Table 4: List of services, applications and their version of Installation in the HLRS production Cloud.

2.4.1 Data Catalogue

PSNC has upgraded the CKAN software and migrated it to the new machines with Ubuntu Server 20.04 LTS. Latest version of the CKAN is 2.9.4. In the lists of changes are security improvements, performance improvements and support for full Python 3.

Ubuntu Server 20.04 offers latest versions of PostgreSQL and Nginx servers used by the CKAN. The new CKAN virtual machines have mounted new 2TB volumes.

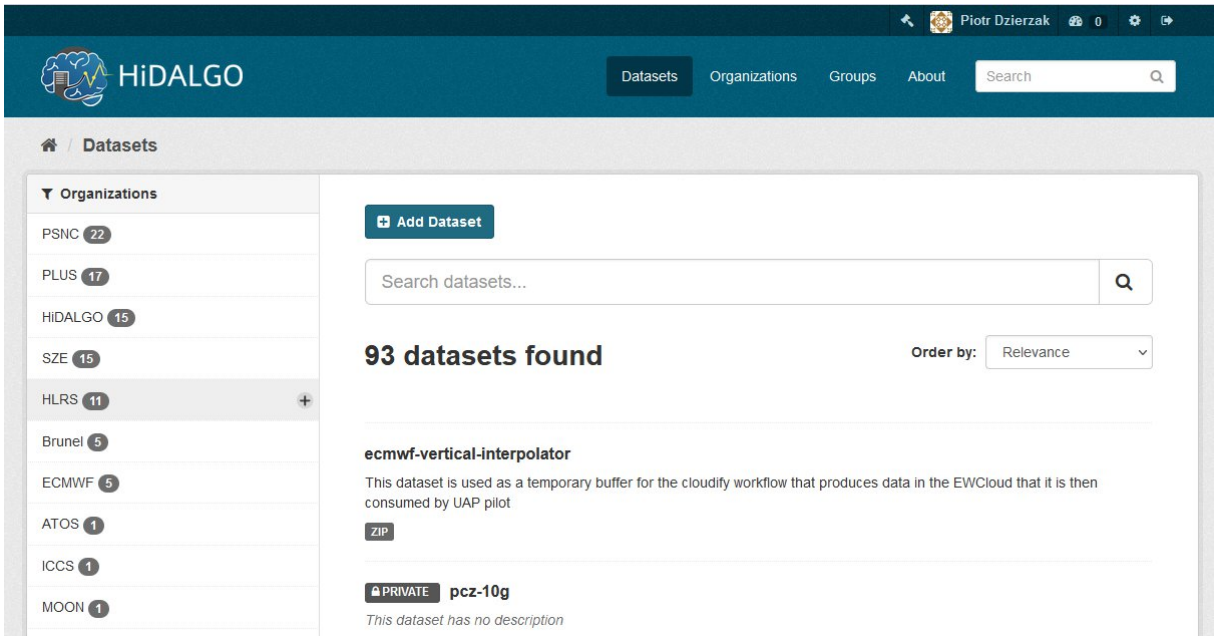

Figure 11: The main view of the CKAN web interface

PSNC has also implemented scripts and configurations, that allows to upload data to the CKAN with other transfer protocols like GridFTP and SCP.

2.4.2 Training

Training is one of the key services of the HiDALGO portal. More details of it can be found in deliverable D5.3 and D5.6, which described the architecture, installation and configurations of the Moodle application on cloud VMs. HiDALGO favicon image support, disable guest access and enable the data retention policy are the few changes introduced in the configuration of Moodle application to provide an authentic training service from the HiDALGO portal. The analytics module is enabled for getting the prediction of the possible students drops out from each course as shown in Figure 12, which is useful for the course instructors to retain the student by adapting the course content based on the students overall needs. New HiDALGO courses are created by the course instructors for disseminating the HiDALGO technologies from the second version of the portal, which is for student access and listed in Table 5.

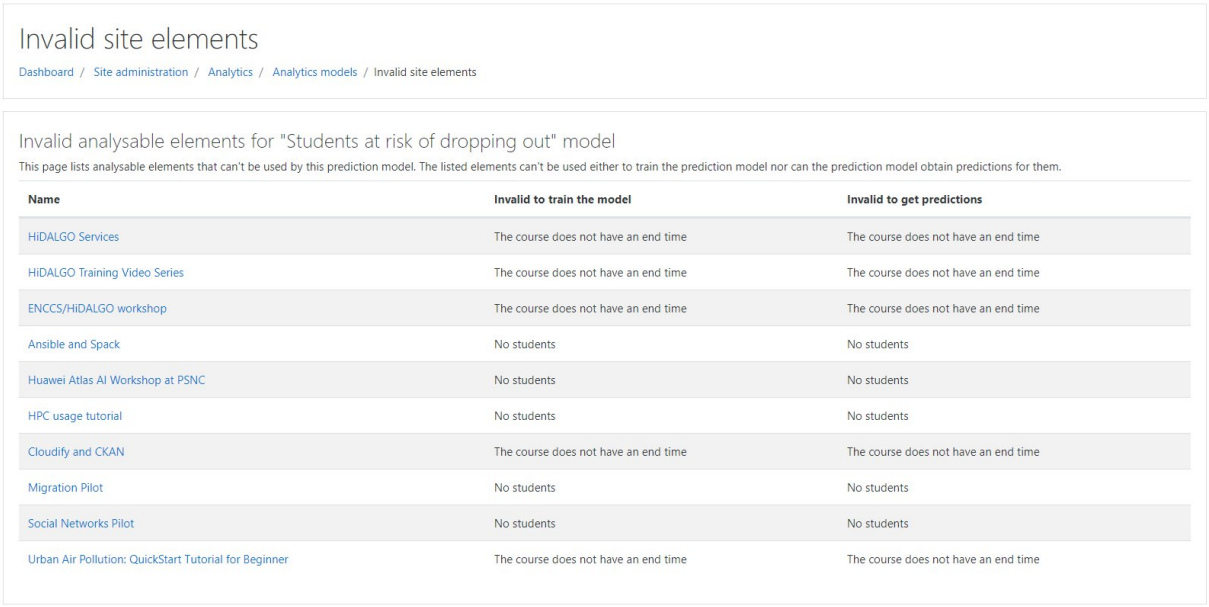

Figure 12: Moodle analytics module to predict the risk of dropping out the courses.

Course Name	Purpose & Description	Course Instructor
ENCCS/HiDALGO workshop on High Performance Data Analytics	Course content of ENCCS/HiDALGO Workshop.	USTUTT, PSNC and SZE
Reproducible Software Environments & Benchmarks with Ansible and Spack	Course slide for creating reproducible software environments and benchmarks with Spack and Ansible.	USTUTT
Huawei Atlas AI Workshop at PSNC	Material from an AI workshop with Huawei engineers organised by PSNC.	PSNC

Table 5: List of courses, course instructors and the objective of the course is detailed here.

The application is a standalone application, which is running in the HiDALGO sub-domain - https://moodle.hidalgo-project.eu/. The service is successfully integrated with the portal by using SSO authentication, so it can be accessed directly from - https://portal.hidalgo-project.eu/. If the user self-registers in Keycloak and tries to access this service through HIDALGO portal, then the new user account is created in Moodle's user management with the 'Student' role. If the user logged in HiDALGO portal through Keycloak, then they can access the service with SSO by using the URL redirection as shown in Figure 13. 'Course Creator' and 'Manager' roles are only assigned by the system administrator of Moodle, so they are allowed to access the system only through Moodle default login to create and change the course

contents securely. The service is backed up regularly and monitored automatically with Zabbix to ensure secure administrations. There is a possibility for bot attack through Keycloak registration and it can be protected by enabling E-mail verification for user access. It is completely resolved by disabling the "self-register" feature in Moodle and enabling "Google reCAPTCHA" on Keycloak to avoid any brute-force attacks.

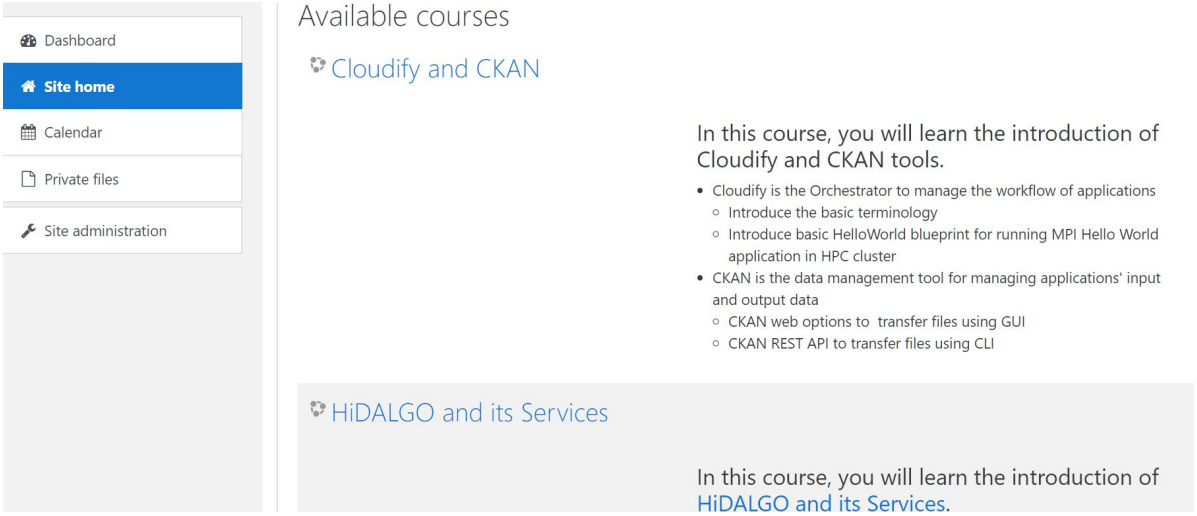

Figure 13: The graphical interface of Moodle while accessing it in the final HiDALGO portal

2.4.3 Matchmaking of Users

Matchmaking is the service for finding similarly interested people within the GSS domain. The functionality and API are not changed from D5.3 [3] but It is improved to use Django-rest classes and python3.6 for easy maintenance and support the SSO login. It was earlier based on the Django authentication, which is replaced by using Keycloak authentication and authorization by using **mozilla-django-oidc**[1] library. The Keycloak IDM provides a "matchmaking" client for SSO login and those details are stored as the configuration in Matchmaking settings.py file for connection between Keycloak and Matchmaking. The mozilla-django-oidc library is the wrapper for the authentication, so each access to the REST API is redirected to Keycloak instead of the default Django authentication.

The algorithm for matchmaking was earlier based on the Geometric Mean algorithm to calculate the user's relation, and it is enhanced with the machine learning algorithms (RDF Graph) to group users based on the users' profession, location and preference information. To support the Machine learning algorithm to refer users, the details are stored in the Elasticsearch data storage, so it can be accessed as a JSON later in the node2vec algorithm and generate the RDF graph for grouping the users. Matchmaking uses PostgresSQL as a primary database for storing all the details and synchronizes with Elasticsearch when the operations of the POST and PUT calls on the REST endpoints. The new architecture of

[1] Django Keycloak library from Mozilla - https://mozilla-django-oidc.readthedocs.io/en/stable/installation.html

Matchmaking is depicted in Figure 14 by including the details of Elasticsearch and RDF-graph-generator. Matchmaking follows the GDPR policy by getting the consent of users before using their data in the service when they access it through the portal frontend.

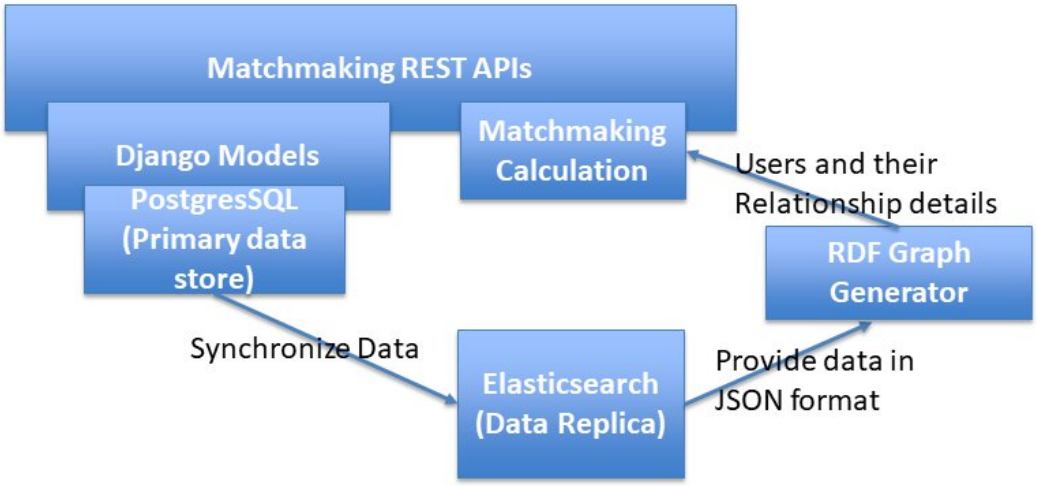

Figure 14: Matchmaking architecture in the final HiDALGO portal.

2.4.4 Interactive Notebook

As described in D5.6, HiDALGO has integrated Jupyter Notebooks to its portal. More specifically, in order to provide the desired scalability and allow numerous users, HiDALGO has set up a Kubernetes cluster and installed JupyterHub on top of it. Jupyter Notebooks have also integrated with the HiDALGO CKAN repository to allow the portal user to access and manipulate their data stored in the repository.

The initial installation of Jupyter Notebooks has been extended in the following ways:

- Support has been added for more programming languages. Besides Python, HiDALGO Jupyter Notebooks support now R and C/C++.
- JupyterHub has been integrated with the Single-Sign-On (SSO) infrastructure of the HiDALGO portal. This means that when a user wants to use the HiDALGO Jupyter Notebooks, she can login through the portal using the same credentials used for any other services provided by the HiDALGO portal.
- JupyterHub has been integrated with the project's CI/CD infrastructure. This has enabled us to monitor the JupyterHub installation through the HiDALGO's Jenkins [9] installation together with all the other monitored services.

2.4.5 Visualization

Visualization tools integrated into the HiDALGO can be divided into two categories, based on the respective software that is employed.

2.4.5.1 COVISE

COVISE is a software for (collaborative) visualization, developed for application in Virtual Reality Environments, such as CAVEs (cave automatic virtual environment) and HMDs (head-mounted displays), and Desktops. This task aims at enabling usage also through web interfaces i.e. the HiDALGO Portal. The implementation is thus realized with respect to the framework of HiDALGO and the given demands. In general, the goal is to enable analysis of simulation results, such as for the Urban Air Pollution use case, through visualization with COVISE.

Detailed analysis in an interactive manner can be conducted in CAVEs. However, this requires certain efforts to assemble and adjust the processing pipeline according to the given dataset.

Thus, a one-click solution is created which allows users to create visualizations of their simulation data without additional adjustments or knowledge on the software. To this goal, a visualization workflow is provided which automatically acquires the needed data, runs the visualization software and extracts snapshots of the rendering. Several images are produced for different perspectives, data fields or time steps, allowing a quick first visualization of the simulation results.

The workflow is based on the JavaScript environment Node.js, which implements the front-end and calls a script to retrieve the dataset. One concluded, COVISE is started and a predefined COVISE module map loaded. The COVISE module map is a collection of processing modules to import, process and render the data. Among this, OpenCOVER, the rendering component integrated into COVISE is executed. OpenCOVER comes with a plug-in to extract snapshot of a rendered scene. This is employed to generate the images.

The visualization workflow can be started through the portal, where the user has to specify a dataset available through the HiDALGO CKAN. Once the workflow is finished, the user will see a preview of the extracted snapshots of the rendered scene.

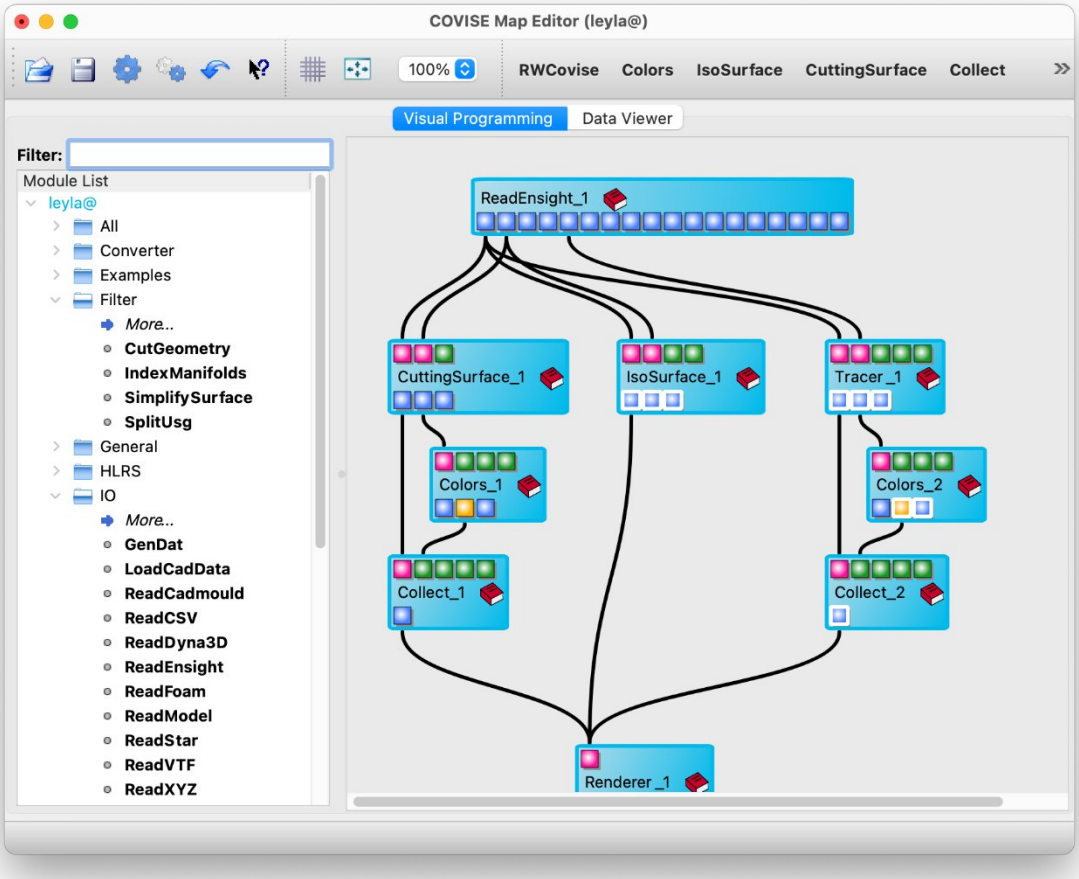

Figure 15: COVISE GUI which demonstrates a module map for the HiDALGO use case UAP

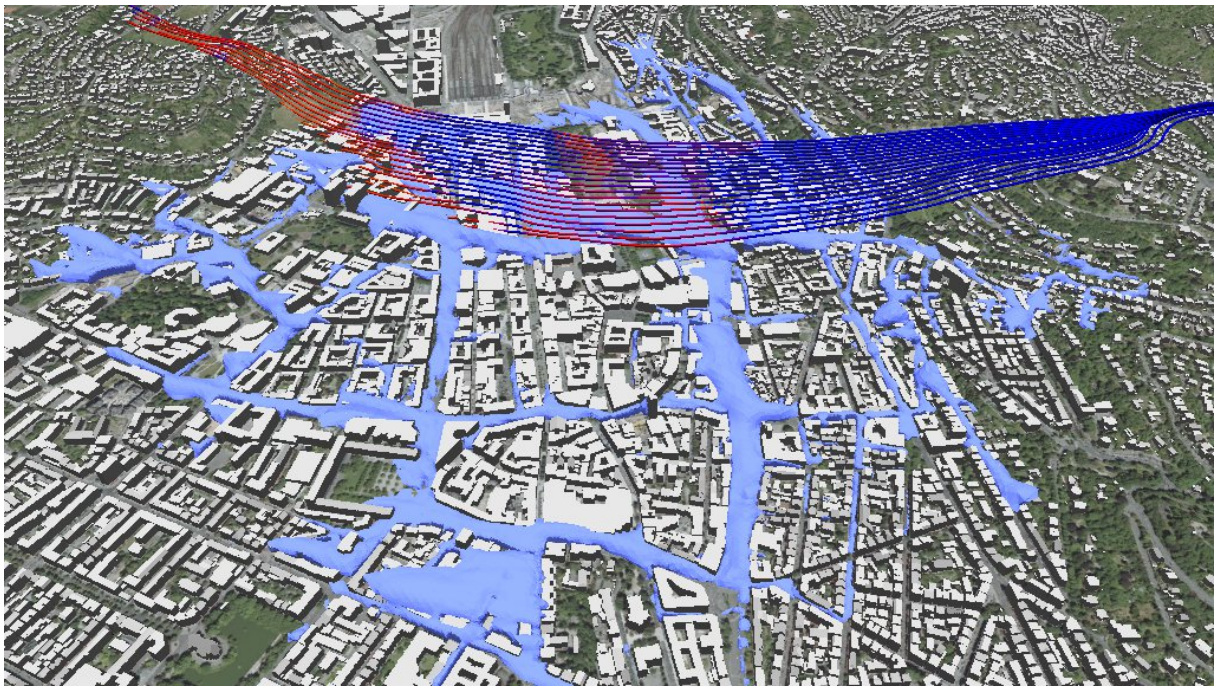

Figure 16: Rendering of UAP data from Desktops

Figure 17: Rendering of UAP data within CAVE

2.4.5.2 Visualizer

Visualizer is a web-based visualization tool for CSV data using coordinated multiple views. Details about Visualizer have been provided in Deliverable 3.2, 3.3, 3.4 and 5.6.

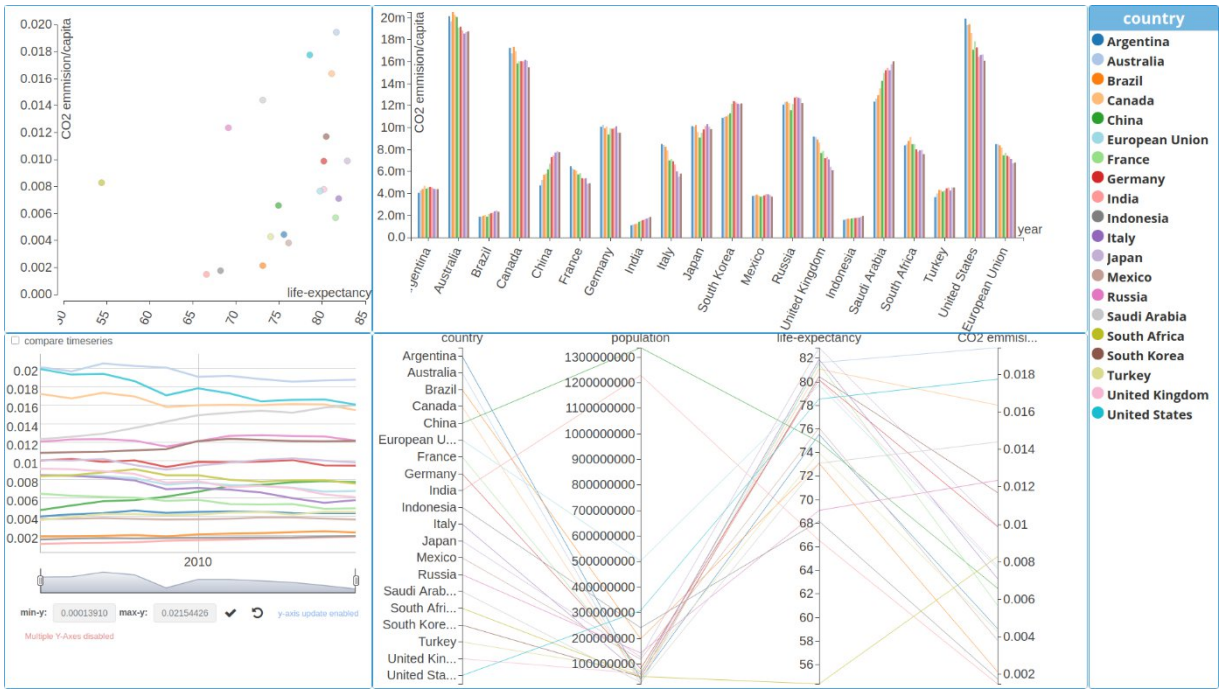

Figure 18: Sample Visualizer dashboard in integration mode

As already described in Deliverable 5.6, Visualizer can be easily integrated to any other website using iFrames. Minor extensions have been made to the API. Besides passing the CSV data, additional information can be passed to Visualizer, including colour schemes.

The following example shows the updated API for passing data and colour schemes as well as reloading the page after passing new data to the iFrames:

```
<html>
 <head>
  <style>
   iframe {
    width: 800px;
    height: 800px;
    border: 1px solid black;
   }
  </style>
 </head>
 <body>
  <iframe src="URL" frameborder="0" id="visualizer" name="visualizer"></iframe>
  <script>
   var data = ""; // CSV string
   window.addEventListener("message", function (event) {
    if (event.data === "visualizer-ready") {
     setColors()
     show();
     reload();
```

```
    }
  });

  function show() {
   //providing a dataset name is optional
   window.frames["visualizer"].postMessage({ key: "data", value: data, name:
    datasetName}, "*");
  }
  function setColors() {
    //set color scheme
    colorScheme = "#980041,#dd1c71,#df65b1,#c994c1,#d4b9d5,#f1eef4]"
   window.frames["visualizer"].postMessage({ key: "colorArray", value: colorScheme},
    "*");
   //set colour map
   colorMap=
    "Europe:#980041,Africa:#dd1c71,America:#df65b1,Asia:#c994c1,Austrialia#d4b9d5";
   window.frames["visualizer"].postMessage({ key: "colorMap", value: colorMap},
    "*");
  }
  function reload() {
   //call this after passing new data to Visualizer
   window.frames["visualizer"].postMessage({ key: "reload" }, "*");
  }
 </script>
</body>
</html>
```

Besides the dashboard view, allowing users to configure their own visualizations, and the presentation mode, allowing modifications of dashboards on demand, a new mode has been implemented in Visualizer, called integration mode. The integration mode allows to specify a dashboard, which cannot be modified after configuration. The integration view only contains the created visualizations and the legend on demand and restricts user interaction possibilities.

Visualizer has been extended to support multiple AI methods within analytical workflows. These include selected clustering and outlier detection algorithms. Therefore, a python server has been implemented in a student project, to execute AI algorithms, and Visualizer has been extended to support configuration and execution of these workflows. Detailed information about analytical workflows and sample dashboards will be provided in Deliverable 3.5.

2.4.6 Customer Support Tools

Customer support tools are installed in the production infrastructure for provisioning services from the second version of the portal, which is detailed in the deliverables D5.4 and D5.6.

Zammad is the support ticketing tool, which is used by customers for raising private queries by E-mail or Zammad web GUI. Askbot is a community forum, which is used by the community to discuss problems and solutions on a public platform. Wiki is an information-sharing tool to share the information within the consortium and external customers for fostering communication. New features and changes in the configurations from the previous version are detailed in the rest of the chapter for each supporting tool.

2.4.6.1 Zammad

Zammad is configured with an automatic E-mail response, SSO login and agent assignment functionalities to satisfy the needs of the HiDALGO support operations and the integration with the portal. The support agents and admin are allowed to log in with the built-in Zammad authentication to avoid any security issues. The self-registration option is disabled in Zammad authentication for avoiding any bot attacks. The self-registered users from Keycloak can log in with the Oauth2 protocol and it is an entry point for the portal integration with the frontend. Zammad is successfully integrated with the portal for customer access as shown in Figure 19.

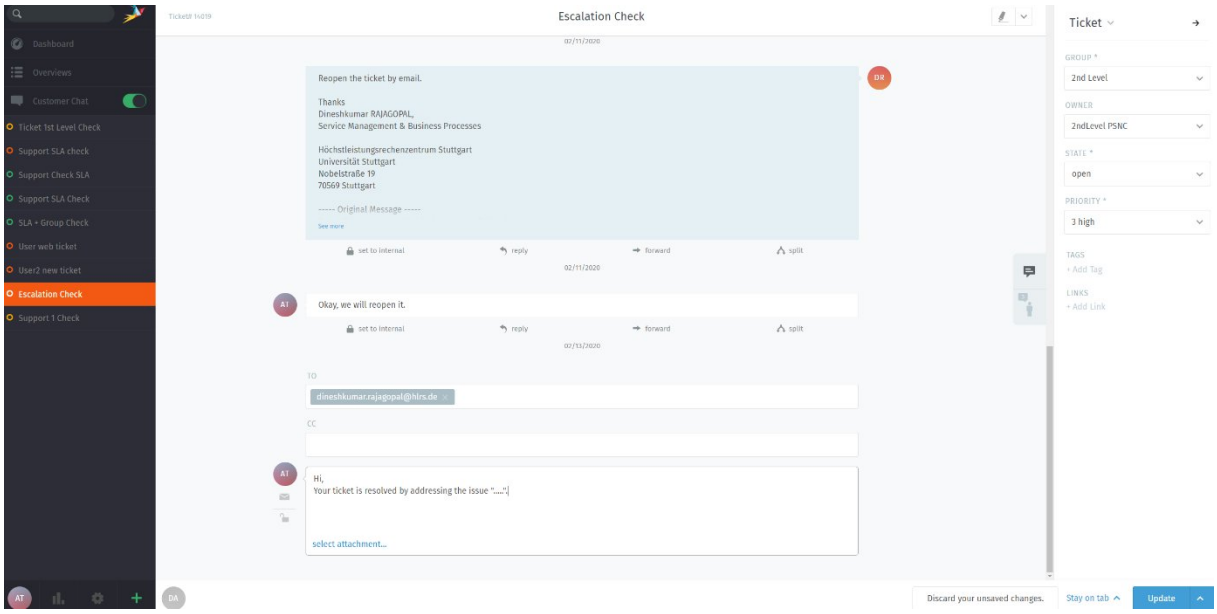

Figure 19: The graphical interface of Zammad while accessing it in the final HiDALGO portal

2.4.6.2 Askbot

Askbot is community support forum, where HiDALGO users can ask and answer questions. Several changes have been done, including updating to latest 0.11 version, implementing user and application data backup solution, integrating with Keycloak SSO.

This version 0.11 of Askbot supports only Django 2 and Python 3. Versions 0.10.x and earlier supported only deprecated Python 2.

New version has been built in a docker container with volumes for application and user data, so it can be easily and safely stored outside the virtualized environment.

Apart from that users now can log in using integrated Keycloak SSO option, which eliminates the need to register a separate account.

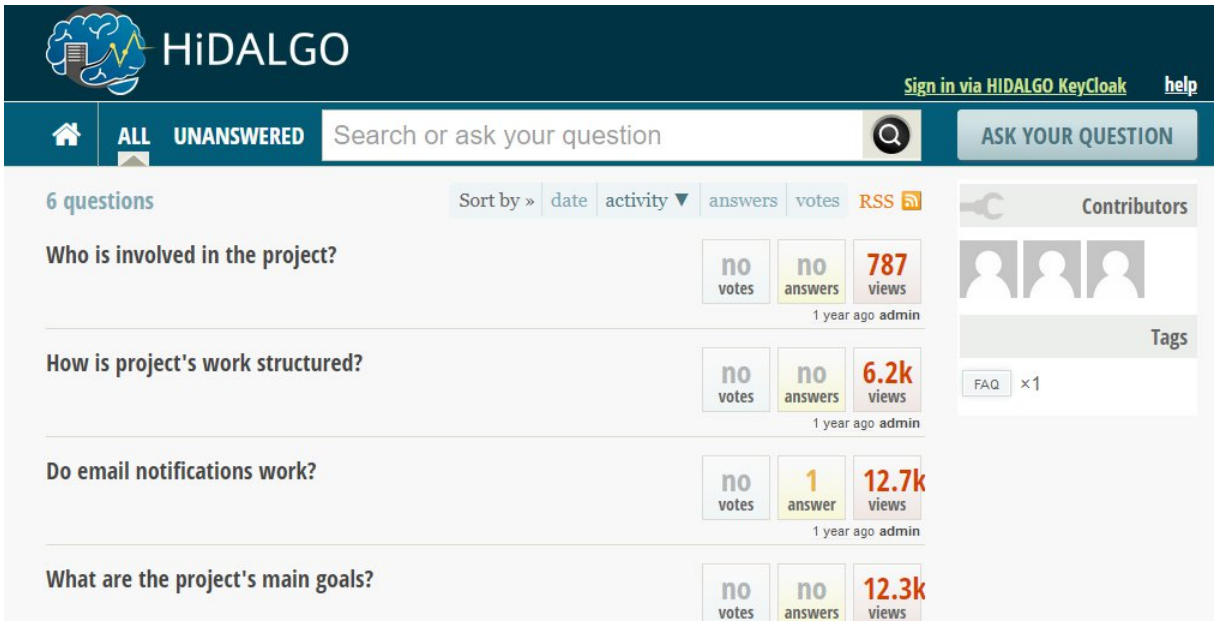

Figure 20: New front page of Askbot service with option to sign in via Keycloak SSO.

2.4.6.3 Wiki

Wiki.js is a collaborative editing platform for sharing information with HiDALGO end-users and internal consortiums. Wiki.js is upgraded to support the latest bug fixes of automatic E-mail verification and forgot passwords to improve the service. The public page is created in wiki - https://wiki.hidalgo-project.eu/public, which is accessible openly in the internet without any authentication. If the user is authenticated through Keycloak SSO, then they can edit the public page with the write permission. The internal wiki pages are only accessible by the HiDALGO consortium with the Wiki.js built-in authentication mechanism to protect from the security vulnerability. Wiki.js is successfully integrated with the portal for end-user to access the public page as shown in Figure 21.

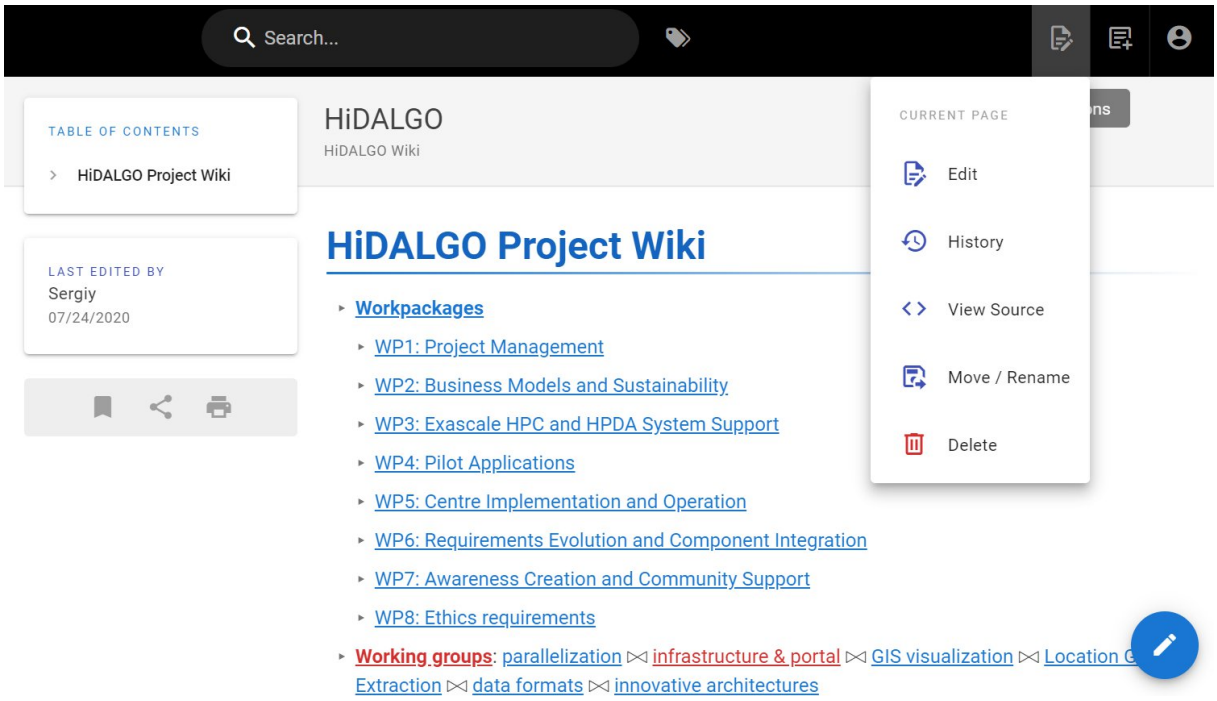

Figure 21: The graphical interface of Wiki while accessing it in the final HiDALGO portal through its front-end.

2.4.7 Customer Marketplace

The Customer Marketplace is the tool that provides the mean to commercialize certain services in the HiDALGO CoE. After evaluating several solutions, the preferred solution was WooCommerce, an open source Wordpress based marketplace that provides the required features:

- It allows to enable SSO with Keycloak (through the corresponding Wordpress plugin for SSO);
- It is able to generate a catalogue of products that can be customized by HiDALGO (so we define our own categories, like tools, applications and datasets);
- Products can be commercialized, including the capability to pay from the marketplace (there are multiple extensions for paying with credit card, PayPal and more);
- It provides APIs that can be used for integrating the solution with the Portal backend (it provides a REST API).

This component is being customized and integrated, so it will be possible to have the backend and the Marketplace synchronized. Such synchronization is focused on the collection of information from the Marketplace, so the backend will know which applications can be accessed by a user and will filter the results that such user can see.

The Marketplace will be populated initially by the consortium, and it will be possible to increase the catalogue later. In principle, we will include free elements in the Marketplace and we will analyse which services to include with some fee.

2.4.8 Secret Management

In order to improve security throughout the platform, especially with regards to HPC SSH credentials, an instance of Hashicorp's Vault has been deployed and integrated with Croupier. Vault needs to be configured so that users can save their credentials for the different infrastructures securely and make sure no other users or the platform's administrators have access to them. Vault, on its own, does not allow the creation of user-specific permissions for secrets with a single configuration, instead a new role and policy needs to be created on Vault for each user.

In order to allow automatic creation of policies and roles for each user, and to ease general secret management on Vault, an API component has also been deployed called vault-secret-uploader, which is the component the Portal will use to let users interact with Vault. Users can upload, list, view, modify and delete only their own secrets thanks to this component.

In that sense, the team is implementing a GUI, as part of the Frontend, that will collect users' information from the Keycloak and Vault, allowing them to manage some basic information (name, email address) and the credentials they want to use. The Frontend and Backend will not store any of this information, relying its management on the Keycloak and Vault. These are the solutions that can store this personal information securely, fulfilling GDPR requirements.

2.5 Quality of the Portal

In order to guarantee a good user experience, the consortium has been working in two main directions: internal demos and a beta testing campaign.

In the case of internal demos, the objective was to show the different features of the Portal, so the partners were providing their feedback about the look & feel and about the features available.

Two demos were organized (one in August 2021 and another one in October 2021), in which most of the partners were participating. In both cases, they provided valuable feedback to be taken into account (such as the way to show certain information, the inclusion of a simple YAML editor for certain input parameters, see historical execution information and create new executions from those). They also triggered discussions around the way to deal with consultancy and the integration of Vault as solution for credentials storage.

Most of such requests have been already implemented and others are ongoing (and will be finished in the coming weeks). Once they are finished, we will proceed with the beta testing campaign.

The ReachOut project (https://www.reachout-project.eu/) supports the preparation and execution of beta testing campaigns, helping other H2020 projects with several tools and a list of available beta testers. The objectives of the beta testing campaign are the following:

- Check if the look & feel of the Portal is adequate for the users;
- Check if users can access the main features easily or if they find issues when trying to find features;
- Test that all the components work correctly and, in case that something fails, identify the problem source.

In order to do so, the consortium will prepare the production environment of the Portal with the following configuration:

- Several datasets will be available in the CKAN;
- Some threads will be already available in the Askbot;
- Partners will monitor constantly the tickets created in Zammad;
- Self-registration will be active;
- A pre-set of simple applications will be available, with workflows that do not require a high computation;
- Example instances will be available for all users (so they can run an existing one or create their own);
- Users will not be allowed to create new applications (in principle, they will not be granted the 'developer' role).

We will also analyse if it is convenient to limit the number of executions allowed for each user. Ideally, our first option would be to use the PSNC training infrastructure, with HPC accounts created for the beta testers.

Users will be requested to:

- Create an instance and execute it (with and without support);
- Navigate through the available datasets;
- Participate in the community forums;
- Visualize the outcome of a simulation;
- Send requests through the ticketing system.

Their feedback will be collected through the ReachOut tools, so it will be possible to understand if the users succeeded and where the main problems are.

3 HiDALGO Infrastructures and Operations

The HiDALGO infrastructure is provided by the infrastructure partners (PSNC, HLRS and ECMWF) for both the execution of the pilots' applications and for hosting the HiDALGO Portal. HPC and HPDA infrastructures are provided by PSNC and HLRS, whereas EWC (European Weather Cloud) is a cloud service offered by ECMWF to support the execution of the pilots' application for different steps – pre-processing, simulation and postprocessing. PSNC Cloud and HLRS Cloud are provided for integrating and deploying the HiDALGO Portal by following a CI/CD methodology. The changes in the infrastructure and the current status of the operation are detailed in the rest of this chapter to provide a holistic view of the HiDALGO Infrastructure.

3.1 HPC and HPDA infrastructures

HLRS and PSNC are the two infrastructure providers for HPC and HPDA infrastructures within the consortium and allocated the computing budget (core-hours) for different pilots and benchmarks, which is detailed in the deliverable D5.1. The pilots and others used the computing budget effectively to accomplish their tasks, which is detailed in Table 6 and Table 7 for PSNC and HLRS centres respectively. The social networking pilot required more than the allocated computing budget in HLRS Hawk, so the PRACE resource is provided for satisfying the needs. The PRACE resource is allocated to different pilots, which is detailed in Table 8.

Purpose	Allocated Budget (in core-hours)	Consumed Budget (in core-hours)
Social Networks Pilot	500,000	161,951
Urban Air Pollution Pilot	500,000	313,376
Migration Pilot	1,000,000	643,477

Table 6: Compute-budget allocated and consumed in PSNC HPC Cluster.

Purpose	Allocated Budget (in core-hours)	Consumed Budget (in core-hours)
Social Networks Pilot	200,000	858,584 (Hazelhen) + 190,660 (Hawk) = 1,049,244 (Hazelhen /Hawk)
Urban Air Pollution Pilot	200,000	3,504 (Hazelhen) + 302,830 (Hawk) = 306,334 (Hazelhen /Hawk)
Migration Pilot	200,000	0
Preparation	66,667	15,485 (Hazelhen) + 933,559 (Hawk)

		= 949,044 (Hazelhen /Hawk)
CoE Benchmark	10,000	14,002(Hawk)

Table 7: Compute-budget allocated and consumed in HLRS HPC Cluster.

HPC Centre	Use case application	Allocated Budget (in core-hours)	Consumed Budget (in core-hours)
MareNostrum4	Migration	155,000	62,470
MARCONI 100	Migration	200,000	0
Piz Daint	Urban Air Pollution	250,000	0
Joliot-Curie Rome	Urban Air Pollution	350,000	11,788
Hawk	Social Networks	600,000	600,000
SuperMUC-NG	Social Networks	86,500	86,500

Table 8: Compute-budget allocated and consumed in the PRACE resource.

HLRS Vulcan, PSNC Altair and ECMWF EWC are the newly introduced infrastructures in HiDALGO, which are detailed one by one with their implementation configuration and the integration through Cloudify. Cray Urika GX was the HPDA infrastructure in HLRS, which was previously used for executing Apache Spark applications (cf. Deliverable D5.1). The Cray Urika GX system is decommissioned from operations in 2021. Instead, HLRS installed and offered access to a new AI-focused cluster, a Cray CS-Storm, to support not only Apache Spark workflows, but also enables users to perform deep learning via TensorFlow and/or PyTorch.

It should be noted that the Cloudify Croupier plugin, which was previously extended in the project to support the Cray Urika GX system through the well-established Mesos scheduler (cf. Deliverable D5.3), is no longer required for the CS-Storm. In contrast to the Cray Urika GX, the new system by Cray uses PBS Pro as the resource manager instead of Mesos, and thus we can rely on the already implemented PBS Pro plugin of Croupier. HiDALGO pilots can run the HPDA application on the Cray CS-Storm through Croupier by customizing their blueprint to specify the PBS Pro as a workload manager, while they define their workflow.

3.2 Cloudify and EWCloud Integration

In December 2018, ECMWF and EUMETSAT joined forces to set up a federated Cloud Computing infrastructure focused on meteorological data, called EWCloud. The vision is to establish a “European Weather Cloud” to serve the European Meteorological Infrastructure and its users. EWCloud revolves around the concept of “users-to-the data”, by providing to users’ transparent access to services, infrastructure and data holdings based on agreed federation principles. More details on the EWCloud were provided in deliverable D5.6.

As part of the coupling of ECWMF weather data for the UAP pilot, additional functionality has been added to Croupier, a plugin for Cloudify, the HiDALGO orchestrator Cloudify has been developed with the aim of accessing cloud resources hosted by the EWCloud to perform data retrieval and post-processing. A virtual machine (VM) on EWCloud has been provisioned for HiDALGO and is dedicated to run a custom post-processing algorithm, developed for the UAP workflow, called vertical interpolator. The aim of this integration is to dynamically serve vertical interpolated data rather than the raw meteorological data to UAP simulations as part of the orchestrated workflow. The benefits of this approach are the following:

- Reduce request time - Requests submitted to ECMWF data retrieval API (WCDA) from EWCloud have higher priority
- Reduce transfer time – The vertical interpolated data are 10 times smaller in size than the raw data
- Edge Computing - Take advantage of PaaS closer to the data such as EWCloud
- Ensemble-based simulation – Simulations based on ensemble data could be very valuable if completed just few hours after the latest forecast available.

Figure 22 shows the overall UAP workflow that thanks to the integration between Croupier and the EWCloud is able to request vertical interpolated data. Here is a description of the workflow:

1. User interfaces with the HiDALGO portal to set the various parameters needed to define the simulation like geographical location, time period, simulation type, etc.
2. The portal interfaces with Cloudify to spawn the blueprint associated to this workflow
3. Cloudify's plugin Croupier uses SSH to connect to the EWCloud VM and to run the vertical interpolator as a command line application.
4. The vertical interpolator authenticates the request against HiDALGO Keycloak. It then requests the meteorological data to the ECMWF archive and executes the interpolation.
5. Once the interpolation is completed the files produced are uploaded to the ecmwf-vertical-interpolator dataset on CKAN. Each file uses a UUID as name to avoid collisions with older files. Moreover, before terminating, any files older than 7 days is deleted as obsolete. This avoids the dataset to increase in size over time.
6. The interpolator workflow ends by sending a notification as a POST message back to Croupier with the URLs of the file produced.
7. Croupier downloads the data directly from CKAN to the HPC and spawns the UAP simulation.

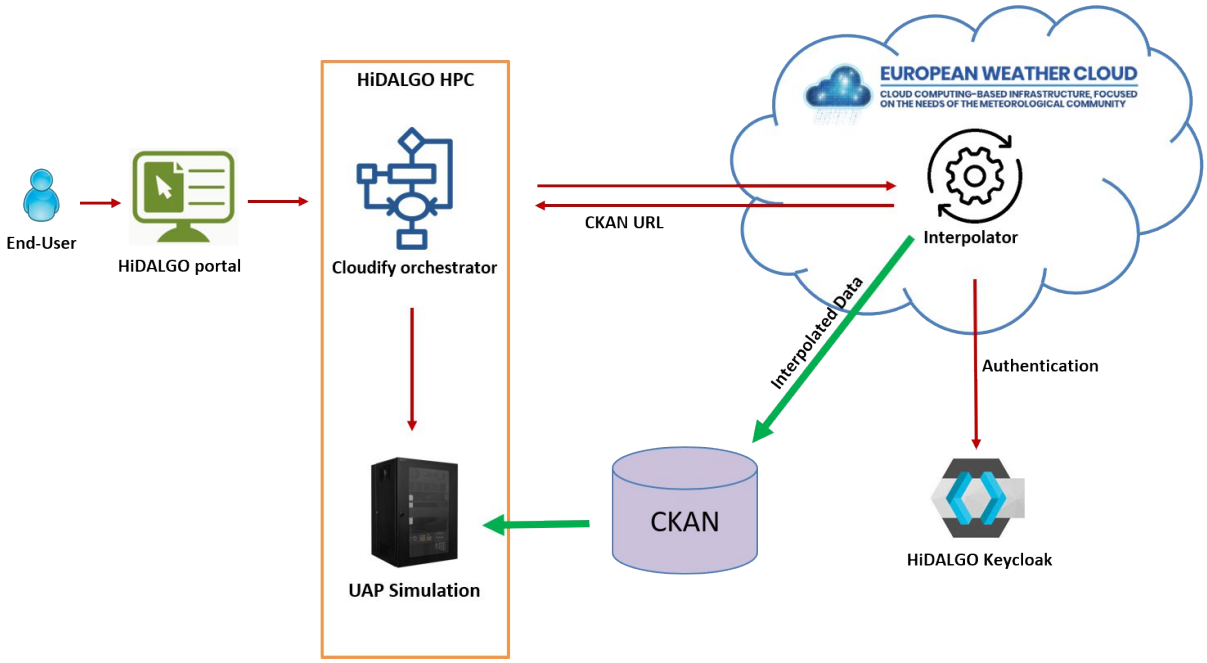

Figure 22: UAP workflow using EWCloud vertical interpolator

This workflow has been successfully implemented and verified but further tests are required to demonstrate the benefits of this approach. More details on the vertical interpolator functionalities and a benchmark of its runtime will be provided in deliverable D4.4.

3.3 Cloud Infrastructures

The HiDALGO Portal is deployed in the HLRS Cloud infrastructure to provide its service to the end-users; it is accessible through the official HiDALGO sub-domain. PSNC offers the integration infrastructure for developers to integrate and test the portal before deploying the stable version in the production Cloud. The list of virtual machines provided for each service in the integration and production cloud are detailed in Table 9 and Table 10 respectively. All the VMs and the services running on it are automatically monitored in the Zabbix monitoring system to ensure the proper operation in the Cloud, which is detailed Section 3.5. There is a minor change in the Cloudify infrastructure to run the production instance in the PSNC cloud to provide seamless interconnection between the HLRS HPC, HPDA infrastructure and Cloudify, due to the restriction in the network policy of HLRS cloud to access the HLRS HPC and HPDA infrastructures.

Application Name	VM Public IP	Domain Name
Streaming	62.3.171.145	sophora-145.man.poznan.pl
Visualization	62.3.170.209	prunus-209.man.poznan.pl
Support ticket	62.3.171.210	sophora-210.man.poznan.pl
Askbot	62.3.171.76	sophora-76.man.poznan.pl
Cloudify	141.58.0.227	https://hidalgo-cfy.hlrs.de/
Matchmaking	62.3.171.89	sophora-89.man.poznan.pl
Zabbix	62.3.171.109	sophora-109.man.poznan.pl
Moodle	62.3.171.102	sophora-102.man.poznan.pl
Jenkins	62.3.171.42	sophora-42.man.poznan.pl
FrontEnd	62.3.171.105	sophora-105.man.poznan.pl
IDM	62.3.170.212	prunus-212.man.poznan.pl
Wiki	62.3.171.187	sophora-187.man.poznan.pl
Spark	150.254.165.237	ribes-237.man.poznan.pl
Notebook	62.3.171.147	sophora-147.man.poznan.pl

Table 9: Virtual Machines provided for portal integration in PSNC Cloud[2].

Application Name	VM Public IP	Domain Name
Moodle	141.58.0.224	hidalgo-fe.hlrs.de/, moodle.hidalgo-project.eu
Cloudify	141.58.0.227	hidalgo-cfy.hlrs.de/
Zammad	141.58.0.231	support.hidalgo-project.eu/
Matchmaking	No public IP	Internal service accessible with LAN, so the domain name is not assigned for this service.
Jenkins	141.58.0.225	hidalgo-jenkins.hlrs.de
Wiki	141.58.0.223	wiki.hidalgo-project.eu
CKAN	141.58.0.226	hidalgo-ckan.hlrs.de
Askbot	141.58.0.228	askbot.hidalgo-project.eu
Zabbix Monitoring	141.58.0.222	hidalgo-monitor.hlrs.de
Keycloak IDM	141.58.0.229	hidalgo-idm.hlrs.de
Portal Frontend & Backend	141.58.0.230	portal.hidalgo-project.eu
Notebook	141.58.0.232	notebook.hidalgo-project.eu

[2] Some of the urls might not be accessible at the time because some of the services are not running always

Application Name	VM Public IP	Domain Name
Notebook compute node-1	No public IP	
Notebook compute node-2	No public IP	
Visualizer / COVISE	141.58.0.233	visualization.hidalgo-project.eu
Vulnerabilities scanner	No public IP	

Table 10: Virtual Machines provided for portal deployment in HLRS Cloud. Cloudify is deployed in PSNC Cloud for providing access to HLRS infrastructures, due to the restriction of network policy in HLRS Cloud[3].

3.4 Continuous Integration and Continuous Deployment

The software development process follows the various steps from the requirement gathering to the deployment of the application on the infrastructure, and it can be optimized with the Continuous Integration (CI) and Continuous Deployment (CD) methodology to improve the quality of the product development. Jenkins, Git and Ansible are the well-known tools used for supporting CI/CD methodology in the HiDALGO portal, which is installed as mentioned in D5.3 and D5.6 for providing the experience of seamless integration and deployment within the project. Jenkins is successfully adopted within the portal development to automate the integration, deployment and testing of its services by using the Ansible scripts. Ansible is an automatic application deployment tool, which is used with Jenkins for automating the application deployment in the cloud infrastructures. Portal and its services were already deployed using Ansible script and defined the Jenkins pipeline for the different steps like integration, testing and deployment. The list of services following CI/CD with Jenkins and Ansible are detailed in Table 11.

Application Name	Ansible Script	Jenkins Pipeline	Zabbix Monitoring	Backup & Restore
Moodle	Yes	Yes	Yes	Yes
Cloudify	Yes	Yes	Yes	Yes
Zammad	Yes	Yes	Yes	Yes
Matchmaking	Yes	Yes	Yes	Yes
Wiki	Yes	Yes	Yes	Yes
CKAN	No	No	Yes	Yes

[3] Some of the urls might not be accessible at the time because some of the services are not running always

Application Name	Ansible Script	Jenkins Pipeline	Zabbix Monitoring	Backup & Restore
Askbot	Yes	Yes	Yes	Yes
Keycloak IDM	No	No	No	Yes
Portal Frontend & Backend	No	No	No	Yes
Notebook	Yes	Yes	Yes	Yes
Visualizer	Yes	Not yet done	Yes	Yes
COVISE	Yes	Not yet done	No	No

Table 11: List of services following backup procedures, Zabbix monitoring, CI/CD with Jenkins and Ansible.

Jenkins and its current setup satisfied with the project goals of Portal CI/CD operations so that the changes are only limited to the Ansible script of the portal services (Matchmaking, Visualizer, COVISE and JupyterHub) for supporting the new deployment procedures for the respective services. The Ansible script of Matchmaking is updated to include the details of Elasticsearch installation and Keycloak authentication to support the new features in Matchmaking. The Ansible script of COVISE is updated to include the ckan-client Perl script installation for copying the visualization outcome back to the CKAN and validate the installation after the successful installation to ensure the correctness of operations.

Gitlab CI/CD, Spack and Ansible script is used for satisfying the needs of pilots CI/CD operations.

3.5 Monitoring the Portal and its Infrastructure

Zabbix is an enterprise-level software designed for real-time monitoring of various metrics collected from servers, virtual machines and network devices.

In HiDALGO it is used to keep track of availability and status of services as well as notify by email if any issues are occurring.

To ensure each service is monitored in analogous way, we have implemented Zabbix templates. They help provide uniform behaviour and metrics for each of supervised resources, e.g. Linux template checks for CPU, memory and disk usage, OS integrity and network performance.

The particular items tracked on each service are presented in Table 12.

Service	Procedure
Askbot	Monitoring agent, Webpage monitoring, Linux monitoring
Moodle	Monitoring agent, Webpage monitoring, Linux monitoring
Cloudify	Monitoring agent, Webpage monitoring, Linux monitoring
Zammad	Monitoring agent, Webpage monitoring, Linux monitoring
Matchmaking	Monitoring agent, Linux monitoring
CKAN	Monitoring agent, Webpage monitoring, Backend monitoring, Linux monitoring
Streaming	Monitoring agent, Kafka server, Linux monitoring
Jenkins	Monitoring agent, Linux monitoring
Keycloak	Monitoring agent, Linux monitoring
Moodle	Monitoring agent, Webpage monitoring, Linux monitoring
Visualizer	Monitoring agent, Webpage monitoring, Web server monitoring, Linux monitoring
Wiki	Monitoring agent, Webpage monitoring, Linux monitoring
Zabbix	Monitoring agent, Linux monitoring

Table 12: Monitoring of HiDALGO services in Zabbix.

3.6 Backup and Restore

HiDALGO portal is the single entry point for accessing all the services by the end-users so that its operations and data are important to protect from any vulnerabilities. The portal and its services are deployed securely by following the security principles (Eg. Self-register with captcha, etc) to avoid any vulnerabilities. The data of the application is crucial to protect from any threats (e.g. accidental actions by end-users and service crashes), so the data is regularly backed up to manage the risk of data loss. HLRS provided one TB (TeraByte) of disk storage in the production VMs for backup operations and all the services are successfully backed up their lively to protect from any cause of data losses. Each service provider stores the recent three backups in the disk storage and older backup are copied back by the HLRS administrator to the tape storage by reducing the cost of backup and increasing reliability in the backup procedures. The backup data of the services are restored in the respective integration VM to test their integrity before completing the backup procedure and update the status in Wiki. The list of services that adopted the backup procedures successfully is detailed in Table 11.

4 Orchestrator

HiDALGO's orchestrator, Croupier, which is a plugin of Cloudify[13], has been updated with several new features since the version presented in D5.6, these updates are shown here.

4.1 Cloudify version

Croupier now supports Cloudify's version 6.2.0, which requires Python 3.6, therefore, all the Croupier code has been ported to Python3. Cloudify 6.0.0 also introduced the possibility to declare scheduled and recurring executions on blueprints, and croupier has been updated to support it.

4.2 Vault integration

As explained in section 2.4.8, a new Vault instance has been deployed to hold the user's secrets. This will be the preferred way to store the user's credentials used by the orchestrator to connect to HPC frontends and other VMs included in their applications as well as API tokens, etc. Croupier has been updated to be able to retrieve these credentials and use them without exposing them to other users and administrators of the platform.

Before, users needed to include all the credentials their applications needed in plain text as part of the inputs sent to Cloudify, which were visible by anyone with access to Cloudify. Now, users only need to provide a Vault token that allows croupier access to the user's secrets. This token is revoked when all the credentials needed have been downloaded, and in any case, it has a maximum lifespan of 5 minutes. Generation of this token is done through the portal, which uses Keycloak's SSO token to authenticate the user.

Once the credentials have been downloaded into croupier, they are saved in memory and are not accessible through Cloudify's interface or API.

In order to make use of this new feature, the user must declare a Vault node on the application blueprint, and any component that needs to retrieve credentials from it only needs to include a relationship that points to this node.

4.3 ECMWF EWC integration

Croupier has been updated to allow users to define data sources for their applications. In this way, if an application needs to download data from a URL, users can just add a Data node to their application and the required relationships and Croupier will download the necessary data into the needed infrastructure so it can be used by the application.

In the case of ECMWF European Weather Cloud's data, the data is generated on demand as part of the application's workflow. To cover this case, in which the data URL is not known at application deployment time, a new node type has been created in Croupier, called ECMWFVerticalInterpolation. The users just need to provide the query details for their application (area, parameters, date and time, etc) and Croupier will make sure the data requested is available in the Infrastructure when the application runs.

The steps Croupier takes to achieve it are as follows:

- The user's Keycloak credentials as well as the credentials needed to access ECMWF's VM are retrieved from Vault,
- An SSH connection to ECMWF's VM is established,
- The interpolation.py script is executed, using as arguments the query details defined by the user on the blueprint as well as the user's Keycloak credentials and the address the script will have to notify once the data is ready,
- The SSH session is closed,
- A Flask server is started to listen for the response from ECMWF's VM,
- Once a response is received, the server is stopped. If the request has been successful, the CKAN URL is included in the response,
- An SSH connection is made to the infrastructure,
- A script is run to download the data from CKAN directly to the infrastructure,
- The data is available, and the application can be started.

The details of how the interpolator.py script works and how the data is generated and uploaded to CKAN are explained in section 3.2.

4.4 Scheduled and recurring executions

Croupier has been updated to support Cloudify's new feature that allows users to define scheduling of executions on their blueprints, including recurring executions. This way, a user can create a blueprint for an application that is executed every day at a certain hour, for example.

The changes made to croupier are centred around the fact that, for applications with recurring executions, installation needs to happen only once, while configuration needs to happen at every execution of the application, therefore, these tasks have been segregated in two different workflows: `croupier_install` and `croupier_configure`. During installation, credentials are downloaded from Vault and infrastructure connection is checked. During configuration, workdirs for the current execution are generated, data is downloaded onto the workdirs and job bootstrap scripts are run.

Another change made to croupier to support recurring workflows is the support for recurring reservations. If the jobs make use of the reservation functionality some HPCs offer and it is a

recurring workflow, and if the `reservation_id` changes depending on the date and time, the users can now declare the format the reservation id uses and if the `recurring_reservation` flag is activated, croupier will generate the `reservation_id` on every execution depending on the execution date and time.

4.5 Support for SHELL jobs

Sometimes users might want to run small jobs directly on the HPC infrastructure without submitting it to a scheduler or may want to include a non-HPC infrastructure (generic VMs) in their blueprint to carry out non computing intensive tasks. Croupier has been updated to fully support these tasks.

In order to include this functionality, a user just needs to include in their blueprint a node of type "`croupier.nodes.InfrastructureInterface`" with SHELL as its "`config.infrastructure_interface`" property, then any job that is to be executed through the SHELL interface (not through a scheduler) just needs to point to this node in its "`task_managed_by_interface`" `relationship`.

The job's commands/script will be executed through a shell directly, instead of being sent to a scheduler.

4.6 Support for scripts upload

Croupier has been updated so that users can define the commands to be executed in their jobs in additional ways. Now, the commands can be defined in these 3 ways:

- **commands**: Just like before, the user just gives a list of commands to be executed on the HPC, croupier generates a script containing these commands and submits it to the scheduler.
- **remote_script**: previously 'script', croupier executes a script that already exists in the workdir. This is especially useful when used with the bootstrap method, in which the bootstrap script generates a script and then the generated script is what is submitted to the infrastructure interface (generally the scheduler).
- **local_script**: A path to a script bundled with the blueprint is provided, croupier takes the contents of the script, adds the selected scheduler settings, if needed, and sends it to the interface. This new feature is useful when the script that needs to be executed on the HPC is long and adding it in the form of commands to the blueprint is not practical.

4.7 Blueprint example

What follows is an example of a test blueprint that makes use of the new features introduced in croupier and supported by the deployed services in Hidalgo. The blueprint defines an application that is run every day at 12:00 in the Europe/Madrid time zone. The application downloads the current wind velocity data in the area 45/18/44/19 from ECMWF's EWC and then executes the script "execute_job.sh". The job is sent to the reservation TEST_%d-%m on the HPC, where %d and %m are substituted for the current day and month in the HPC. All the credentials need to connect to the HPC and get the data from ECMWF are downloaded from Vault, and the only 2 inputs needed are the user's Keycloak user and a valid Vault token for this user.

```
tosca_definitions_version: cloudify_dsl_1_3

imports:
  - http://raw.githubusercontent.com/ari-apc-
lab/croupier/hidalgo/resources/types/cfy_types.yaml
  - plugin:croupier

inputs:
  keycloak_user:
    type: string
  vault_token:
    type: string

node_templates:
  hpc:
    type: croupier.nodes.InfrastructureInterface
    properties:
      config:
        infrastructure_interface: TORQUE
      ssh_config:
        host: hawk.hlrs.de
      job_prefix: test
      base_dir: "$HOME"
      monitoring_options:
        monitor_period: 15
      skip_cleanup: true
      workdir_prefix: "test"
    relationships:
      - type: retrieve_credentials_from_vault
        target: vault

  vault:
    type: croupier.nodes.Vault
    properties:
```

```
        user: { get_input: keycloak_user }
        token: { get_input: vault_token}

  job:
    type: croupier.nodes.Job
    properties:
      job_options:
        script: 'execute_job.sh'
        max_time: "03:00:00"
        reservation: 'TEST_%d-%m'
        recurring_reservation: True
      skip_cleanup: True
    relationships:
      - type: task_managed_by_interface
        target: hpc
      - type: job_needs_data
        target: weather_data

  weather_data:
    type: croupier.nodes.ECMWFVerticalInterpolation
    properties:
      ssh_config:
        host: 136.156.90.143
      keycloak_credentials:
        user: { get_input: keycloak_user }
      query:
        area: 45/18/44/19
        params: u/v
    relationships:
      - type: task_managed_by_interface
        target: hpc
      - type: retrieve_credentials_from_vault
        target: vault

deployment_settings:
  default_schedule:
    config:
      workflow: croupier_configure
      since: '2021-10-09 11:55
      timezone: 'Europe/Madrid'
      recurrence: '1d'
    run:
      workflow: run_jobs
      since: '2021-10-09 12:00
      timezone: 'Europe/Madrid'
      recurrence: '1d'
```

5 HiDALGO Customer Support

Customer support is a key service for establishing a connection with the customers to better understand their needs and build a brand by establishing trust with them. This support is a free service to fix the issues within the HiDALGO toolbox, while the HiDALGO end-users are facing an issue with it. This service collects customer feedback in the form of queries and addresses the queries professionally to improve the quality and usability of the toolbox continuously. A customer support system is defined in the deliverable D5.4 to establish a two-level support process and use the support tools for collecting queries from the customers. In this deliverable, support operation is detailed with the metrics of achievement in the different support systems and tools – Askbot, Wiki and Zammad. If the customer is interested in commercial support or new developments in the HiDALGO toolbox, then the business development team from WP2 will evaluate the request by providing further assistance in the commercial aspects. The commercial support is not part of this deliverable, so it is not covered in this deliverable.

5.1 HiDALGO Support Forum

Askbot is a support forum for public exchange of common issues and solutions. It is used to collectively create an open knowledge base. This service is used by both HiDALGO members and unregistered guests.

The metrics of Askbot usage and engagement are presented in Table 13.

Metrics	Askbot
Number of Q&A threads	6
Number of responses	2
Number of Users (Active Contributors)	3
Number of Users (Passive Readers)	9

Table 13: The metrics of Askbot usage.

5.2 HiDALGO Wiki

HiDALGO wiki is a collaborative editing platform for exchanging information. The service is used by both internal consortium and external customers to foster both internal and external communication. It is one of the actively used services within the portal and it is successfully

adopted for both internal and external communication. The metrics of wiki usage is reported in Table 14 at the time of writing this report. The metrics reflect the active usage of the services by highlighting the key achievements as below.

1. 235 wiki pages for the wiki.
2. 67 users registered from the consortium.
3. 55 users actively contributing to the wiki.

Metrics	Wiki
Number of Wiki Pages	235
Number of Users registered	62
Number of Users (Active Contributors)	55
Number of Users (Passive Readers)	62

Table 14: The metrics of wiki usage for the consortium.

5.3 HiDALGO Support Ticket

Zammad is a ticketing tool to manage customer queries confidentially and securely to respect the privacy of end-users. If the user wants to share some confidential details and prefers their data to be processed securely, then the public support forum and wiki are not the correct platforms to handle it. Zammad is used for both internal bug tracking and external customer support ticket processing to track all the internal quality monitoring and customer feedback in the tool properly. If the user raised a query in the public forum and it is a bug in the toolbox, then a new bug is created to assign the corresponding responsible person for addressing it properly.

6 Conclusions

This document describes the progress done in the final stage of the HiDALGO Portal development and its operations, including the main portal features, the selected architectures, the utilization of the infrastructures, the orchestrator and the customer support. The changes done since the last deliverable are pointed out in various aspects. For the *Single-Sign-On* (based on Keycloak), we have updated the configuration and the integration with the rest of the components to provide coherent access to all the services in a single place. For the *workflows orchestration* (based on Croupier), we have updated it to support the new version of Cloudify as well as to support the pre-processing of ECMWF weather data. We also updated the tools for users support (ticketing system, Askbot, Wiki, etc.) and we also included the Notebooks, which supports C, C++, Python and R. The visualization (Visualizer and COVISE) are also integrated, with more features. We also addressed the deployment based on Jenkins, the monitoring system in the infrastructure based on Zabbix, backup the services to improve the security and new HPC, HPDA infrastructure for pilots' execution. Finally, this deliverable presents the accomplishment in the customer support by listing the metrics achieved to meet project goals.

All the implemented and updated features of the portal are targeted for providing more and better services which facilitate the user experience with Hidalgo. With different services integrated into the Portal, they are now tied up tight together to serve as a whole and to have a one-stop-shop.

Comparing the current status to the original roadmap, it can be seen that most of the features have been successfully implemented and meet the expectation. For the tasks that have not been finished yet, e.g., the integration of the different modules in portal, the Jenkins for Visualizer, etc. are also on a good path and can be finished before end of the project as planned.

References

[1] HiDALGO, D5.1 – HiDALGO System Environment. Abhishek. 2019.

[2] HiDALGO, D5.2 – Portal Architecture and Roadmap. Carnero, Javier. 2019.

[3] HiDALGO, D5.3 – First HIDALGO Portal Release and System Operation Report. F. Javier Nieto. 2019.

[4] HiDALGO, D5.4 – HiDALGO Support Concept. Rajagopal, Dineshkumar. 2020.

[5] HiDALGO, D5.6 – Second HIDALGO Portal Release and System Operation Report. F. Javier Nieto. 2021.

[6] HiDALGO, D6.1 – Requirements Process and Results Definition. Maritsch, Martin. 2019.

[7] HiDALGO, D6.2 – Workflow and Services Definition. Maritsch, Martin. 2019.

[8] HiDALGO, D6.5 – Intermediate Report on Requirements, Components and Workflow Integration. Gogolenko, Sergiy. 2020.

[9] Jenkins. URL: https://jenkins.io/. Last visited in December 2019.

[10] Moodle Training Service. *URL:* *https://www.moodle.org*. Last visited in December 2019.

[11] CKAN: https://ckan.org/, last visited in December 2019

[12] Keycloak Server Administration Guide. URL: https://www.keycloak.org/docs/latest/server_admin/hlast visited in December 2019

[13] About Cloudify. URL: https://docs.cloudify.co/5.0.0/about/, last visited December 2019

[14] Croupier GitLab Repository. URL: https://github.com/ari-apc-lab/croupier

[15] Wiki.js for HiDALGI wiki. *URL:* *https://js.wiki*. Last visited in December 2019.

[16] Zammad for a customer support ticket. *URL:* *https://zammad.org*. Last visited in December 2019.

[17]OpenID. OpenID Connect Core 1.0. 8th November 2014. https://openid.net/specs/openid-connect-core-1_0.html. Last visited December 2019.