D2.7 HiDALGO2 Dashboard and Services

Date: April 04, 2024

Document Identification			
Status	Final	**Due Date**	28/02/2024
Version	1.0	**Submission Date**	04/04/2024

Related WP	WP2	**Document Reference**	D2.7
Related Deliverable(s)	D2.1, D2.4	**Dissemination Level (*)**	PU
Lead Participant	SZE	**Lead Author**	Ákos Kovács (SZE)
Contributors	ATOS, PSNC, SZE, USTUTT	**Reviewers**	Vincent Chabannes (UNISTRA)
			Nikolaos Chalvantzis (ICCS)

Keywords:
Operational Environment, Infrastructure Provisioning, Dashboard Architecture, Dashboard components, component interfaces

Document Information

List of Contributors	
Name	**Partner**
Ákos Kovács	SZE
Sameer Haroon	USTUTT
Jesus Gorronogoitia	ATOS
Bartosz Bosak	PSNC

Document History			
Version	**Date**	**Change editors**	**Changes**
0.0.2	12/01/2024	SZE	Table of Contents
0.1	08/02/2024	SZE	Dashboard technical preview (section 3)
0.1	11/02/2024	SZE	Dashboard integration and roadmap (section 3)
0.2	12/02/2024	ATOS	Data Management (Section 5)
0.3	13/02/2024	USTUTT	Services integration (Section 5)
0.4	23/02/2024	PSNC	QCG integration (section 4)
0.5	24/02/2024	SZE	Conclusion and adding requirements table
0.6	08/03/2024	PSNC	Comments and proofreading
0.8	13/03/2024	SZE	Rewriting some chapters, new images
0.9	19/03/2024	SZE	Internal review comments
1.0	04/04/2024	SZE, PSNC	Final version

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Deliverable leader	Ákos Kovács (SZE)	29/03/2024
Quality manager	Harald Koestler (FAU)	03/04/2024
Project Coordinator	Marcin Lawenda (PSNC)	04/04/2024

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
CoE	Centre of Excellence
DMS	Data Management System
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
JWT	JSON Web Token
MathSO	Workflow orchestrator developed by SZE
MFA	Multi Factor Authentication
QCG	Workflow orchestrator developed by PSNC
SSO	Single Sign On
UI	User Interface
WFO	Workflow Orchestrator (QCG, MathSO)
WP	Work Package

Executive Summary

This deliverable document provides an overview of the dashboard component and the connected services of the HiDALGO2 ecosystem and supports the project's users and stakeholders for a one-gate architecture to reach HiDALGO2 services. It builds upon the work of previous deliverables such as D2.4 "Infrastructure Provisioning, Workflow Orchestration and Component Integration" and D4.1, "Data Management and Coupling Technologies".

In this document, HiDALGO2 outlines the vision of using one single dashboard to reach all the provided services such as Data Management, as well as running HPC jobs, exchanging information through a Forum, reading documentation on a self-hosted Wiki, and requesting support through a ticketing system. The biggest part of this document outlines the basic concepts of the dashboard and it's functionalities which will be delivered to production in M24 and will be documented in upcoming deliverable, such as D2.8 "HiDALGO2 Dashboard and Services (M28)".

An in-depth description of the central components supporting the pilot use cases, reaching the Workflow Orchestrators, MathSO and QCG, is given. A careful analysis of the general and specific requirements of the individual pilot simulations has been gathered from the partners, and a detailed development roadmap attempts to address these requirements, as well as introduce future features. The deliverable rounds out the system design of the HiDALGO2 environment with a strategy and vision for integrating all of its constituent components.

Finally, the roadmaps drawn out in the individual chapters give the direction the HiDALGO2 technical infrastructure and supporting development will follow, which will then be documented in future iterations of this deliverable, D2.6 and D2.8.

1. Introduction

1.1. Purpose of the document

The deliverable “D2.7 HiDALGO2 Dashboard and Services (M14)” aims to outline the dashboard as a landing page for HiDALGO2, which will be used for reaching all the provided services of the project’s including a helpdesk, online encyclopaedia, e-Learning platform and of course the Workflow Orchestrators (WFO). This work will be continuously reviewed throughout the runtime of the project and documented in future iterations of this document (D2.8 in M28 and D2.9 in M42).

1.2. Relation to other project work

This document is based on the rationale for integrating diverse services and strategies from deliverables within HiDALGO2, in addition to the requirements gathered in "D2.1 Requirements Analysis and Scenario Definition (M6)." It further aligns closely with "D4.1 Data Management and Coupling Technologies (M11)" and "D3.1 Scalability, Optimization and Co-Design Activities (M12)." D2.7 also provides platform for D2.5 “Infrastructure Provisioning, Workflow Orchestration and Component Integration.”

1.3. Structure of the document

Chapter 3 describes the method of collecting requirements from partners, service providers and pilot owners for the upcoming development of HiDALGO2 dashboard.

Chapter 4 summarizes the proposed features of the HiDALGO2 Dashboard and its planned development roadmap. And the interfacing method of WFOs.

Chapter 5 discusses HiDALGO2 project’s already deployed and upcoming services, including Data Management service development, describes the way project delivers its services and what kind of pilots we offer as service.

Chapter 6 summarises and concludes this deliverable while giving an outlook about future objectives and challenges.

2. Requirements analysis and expectations

For the collection of requirements for the Dashboard, we are planning two main campaigns.

- The first will be focusing on collecting feedback from internal partners, including their opinions and requirements.
- The second campaign will be built around a questionnaire addressed to wider participants form other CoEs and potential stakeholders.

The internal collection is performed by a shared datasheet with predefined examples to facilitate completion.

Creating and managing a questionnaire workflow is a complex process that involves multiple stages, from initial conception to data analysis. This document will explore the key steps involved in designing, distributing, conducting, and analysing a questionnaire, ensuring that the process is both efficient and effective.

2.1. External stakeholders

The questionnaire will be sent to potential stakeholders recruited by the WP leadership team and will be integrated and analysed by future iterations of this deliverable, namely D2.8 and D2.9. The questionnaire will be based on the internal feedback of the first release of the HiDALGO2 Dashboard. We will collect this feedback from pilot owners and other users of the project. With this knowledge we can create presentations and demos for external stakeholders and even provide demonstration accounts to different pilots. After a trial period of 2–4 weeks of usage, we plan to collect feedback through the questionnaire.

2.2. Functional requirements

The list of functional requirements was compiled by contacting internal partners. It is based on feedback coming not just from pilot owners, documenting their opinions about the Dashboard service, but also from WFO developers and other service providers like DMS service developers.

The requirements are marked as DASH-REQ-XX, and in this chapter we will address this gathered information. The full list of Requirements can be found in Annexes.

2.3. Overview of the requirements

In general, the requirements focus mainly on data movement and the management of input and output data. Many of the requirements build on the operation of Workflow Orchestrators that interface between HPC and users. At the same time, there are also requirements from WFO developers. These are mainly focused on the provision of API endpoints for the services they offer and on unified user management.

The most restrictive requirement (based form the requirements table) is that the dashboard must be able to use most of the features of the CKAN data management system. (DASH-REQ-002, DASH-REQ-006, DASH-REQ-012). In this section we clarify these requirements by partners:

- **SZE Requirements**
  As one of the WFO, Pilot and Dashboard developer SZE highlighted on the importance of SSO authentication, the accessibility of the Pilots and the data management integration.
- **ATOS Requirements**
  As ATOS is responsible for Data management and movement subsystem it crucial to seamlessly move data from data source to the HPCs.
- **UNISTRA Requirements**
  UNISTRA requires to implement the tools already used by its pilot. These tools are Ktirio (https://ktirio.fr/) and various web-based document management systems. Some of these are likely to be integrated into Wiki.js in some automated way.
- **PSNC Requirements**

PSNC's needs derive from its pilot on Renewable Energy. Additional requirements include the support of data transfer through the CKAN API and integrated visualization, which is developed by SZE.

- **MTG Requirements**
  The requirements for MeteoGrid include the integration of map preview functionality to facilitate the selection of boundaries for the Wildfires pilot.

Additional functionalities addressing secondary and non-priority requirements will be classified, examined by the developer team, and will be integrated into the roadmap and the future releases of the dashboard.

3. HiDALGO2 dashboard

In this section we will go through the Dashboard in more detail. The particular topics are explained in the following separate subsections:

- the Overview of the Dashboard,
- authentication and planned functions,
- the technical structure of the Dashboard,
- the development roadmap for these functions.

3.1. Overview of the dashboard

The Dashboard aims to provide a central interface from which a user can:

- get a comprehensive overview of the HiDALGO2 project's goals and actual status,
- register using a new profile and log in with it,
- when logged-in, receive information about each plug-in that can be navigated to from the Dashboard,
- arrive at any snap-in web interface already logged in after dashboard logging, avoiding the need to log in again, thanks to SSO login.

Dashboard page url: https://dashboard.hidalgo2.eu

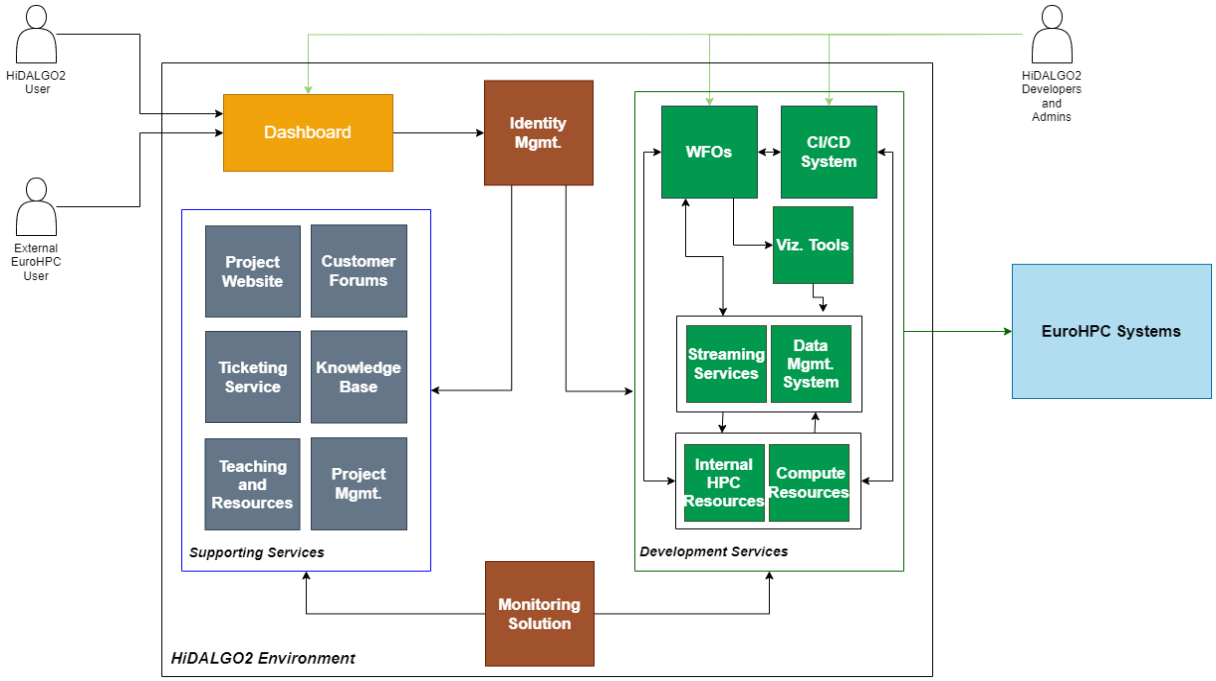

Figure 1: Overall System Diagram of HiDALGO2 and Dashboard

The HiDALGO2 Dashboard is designed to ensure the integrity of the various plug-in web applications, which, although separate systems, provide the logged-in user with the feel of a complete application through SSO login.

As in Figure 1. shown, one of the most crucial component of the system is the Identity management system. All other components are relied on it.

The following applications will be included in the Dashboard:

- **Project website:** (link: https://hidalgo2.eu)

The project website provides the main source of news for Users and interested parties, to stay updated on the progress of the project. Publications and Blog posts etc, are all provided through the project website.

- **MathSO WFO:** (link: https://portal.hidalgo2.eu)

A web application developed by Széchenyi University, which provides the possibility to upload, manage and run various simulations and calculations on different HPCs, requiring high computational capacity. The user has the possibility to set the parameters and input files required for the deployment run. Also, monitor the simulations and view the results (even its successful or unsuccessful).

- **QCG WFO:** (link: https://qcg.hidalgo2.eu)

QCG-Portal, being a part of QCG software family developed by Poznan Supercomputing and Networking Center, is a modern web-based system for remote execution of computational experiments on large-scale resources, HPC in particular. In addition to the core functionality, that includes secure submission, management and monitoring of tasks as well integrated data management capabilities, QCG-Portal provides various means for the customisation of the interface and adjusting it to specific applications and use cases. The first of them is a built-in templating mechanism, which allows for the easy creation of custom application forms, while the other is a dedicated API, which enables seamless integration of custom-made java script application interfaces for both input parameters specification and results visualisation.

- **Askbot:** (link: https://askbot.hidalgo2.eu)

Askbot is an open-source question and answer (Q&A) forum software, inspired by platforms like Stack Overflow and Yahoo Answers. It's designed to help create community-driven Q&A sites where users can ask questions, receive answers, and vote on both questions and answers for relevance and accuracy. This democratic approach to information sharing and problem-solving leverages the collective knowledge and expertise of the community, fostering a space where users can contribute to and benefit from shared learning experiences.

- **CKAN:** (link: https://ckan.hidalgo2.eu)

CKAN (Comprehensive Knowledge Archive Network) is an open-source data management system that makes it easy for users to access, share, and discover data. Primarily used by governments, organizations, and research institutions to publish datasets, CKAN is designed to promote data transparency and facilitate open data initiatives. Its user-friendly web interface allows for the efficient search and discovery of datasets, while its powerful API enables developers to build creative applications that utilize public data. CKAN supports a variety of data formats and provides tools for data visualization, making it an invaluable resource for data-driven decision making and research. By fostering an environment where data is readily accessible and reusable, CKAN plays a crucial role in advancing knowledge, innovation, and transparency across numerous fields [1].

- **ZAMMAD:** (link: https://ticket.hidalgo2.eu)

Zammad is a versatile, open-source support and ticketing software designed to streamline customer service operations. Its user-friendly interface and flexible configuration make it suitable for businesses of all sizes, enabling efficient management of customer inquiries, communications, and support tickets. Zammad supports multi-channel integration, allowing teams to handle emails, social media messages, and live chats from a single platform. Its powerful features include automation rules, reporting tools, and a knowledge base to reduce response times and improve service quality. With its focus on usability and collaboration, Zammad enhances transparency and teamwork among support staff, fostering a more organized and responsive customer service experience. [2]

- **WIKI:** (link: https://wiki.hidalgo2.eu)

Wiki.js is a modern, open-source wiki software that redefines the way knowledge bases and documentation are created and maintained. Built on the Node.js framework, it offers a sleek and intuitive user interface that makes content creation and collaboration seamless for users. Wiki.js supports various content types, including rich text, markdown, and HTML, catering to diverse content creation needs. Its robust features include version control, real-time editing, and a powerful search engine, ensuring that information is easily accessible and up to date. With its emphasis on security and customization, Wiki.js is an ideal platform for organizations looking to build a comprehensive and secure knowledge base or documentation portal. [3]

- **Moodle:** link(https://moodle.hidalgo2.eu)

Moodle is a free, open-source learning management system (LMS) widely used by educational institutions and organizations worldwide to create personalized learning environments. It provides educators with a flexible platform for delivering online courses, training programs, and collaborative learning experiences. Moodle's extensive range of features includes course management, quizzes, grading, forums, and multimedia integration, supporting a diverse array of teaching and learning methodologies. Its modular design and plugin system allow for extensive customization and scalability, meeting the needs of both small classes and large university operations. Moodle's emphasis on accessibility, security, and community collaboration has made it a cornerstone tool in e-learning, facilitating engaging and effective education online. [4]

- **OpenProject:** (link: https://project.hidalgo2.eu)

OpenProject is a powerful open-source project management software that facilitates team collaboration and project planning. Designed to support the entire project lifecycle, it includes features for task management, timelines, cost reporting, and document management. OpenProject excels in enabling agile methodologies, offering boards for sprint planning and task tracking, which are essential for iterative development and project management. Its user-friendly interface allows teams to efficiently organize work, track progress, and achieve project goals. With robust security measures and a flexible API for customization and integration, OpenProject is highly regarded by businesses, NGOs, and educational institutions for fostering effective teamwork and project transparency. [5]

- **Bitbucket:** (link: https://git.man.poznan.pl/stash/projects/HIDALGO2)

Bitbucket is a web-based version control repository hosting service owned by Atlassian, primarily used for source code and development projects that use either Mercurial (since deprecated) or Git version control systems. It offers both free accounts and subscription-based accounts with additional features, making it a versatile tool for individual developers and teams alike. Bitbucket integrates closely with other Atlassian products like Jira, Bamboo, and Confluence, facilitating a seamless workflow for planning, coding, testing, and deploying applications. Its key features include pull requests for code review, branch comparison and commit history, and a powerful CI/CD pipeline for automated building and deployment processes. Bitbucket's collaboration features, along with its emphasis on security, make it a popular choice for software development teams looking to manage their codebase efficiently.[6]

Frontend, backend, and infrastructure form the foundational components of modern web development. The frontend is the user-facing side, crafted with HTML, CSS, and JavaScript to create engaging and interactive user experiences. It's what users see and interact with directly in their web browsers. The backend, on the other hand, operates server-side, handling database interactions, user authentication, and server logic. It's built using languages like Python, Ruby, or Java and frameworks that support application architecture. Infrastructure encompasses the hardware and software environments that host and run applications, including servers, databases, and cloud

services. Together, these elements enable the development and operation of dynamic, scalable web applications. The fronted is shown in Figure 2.

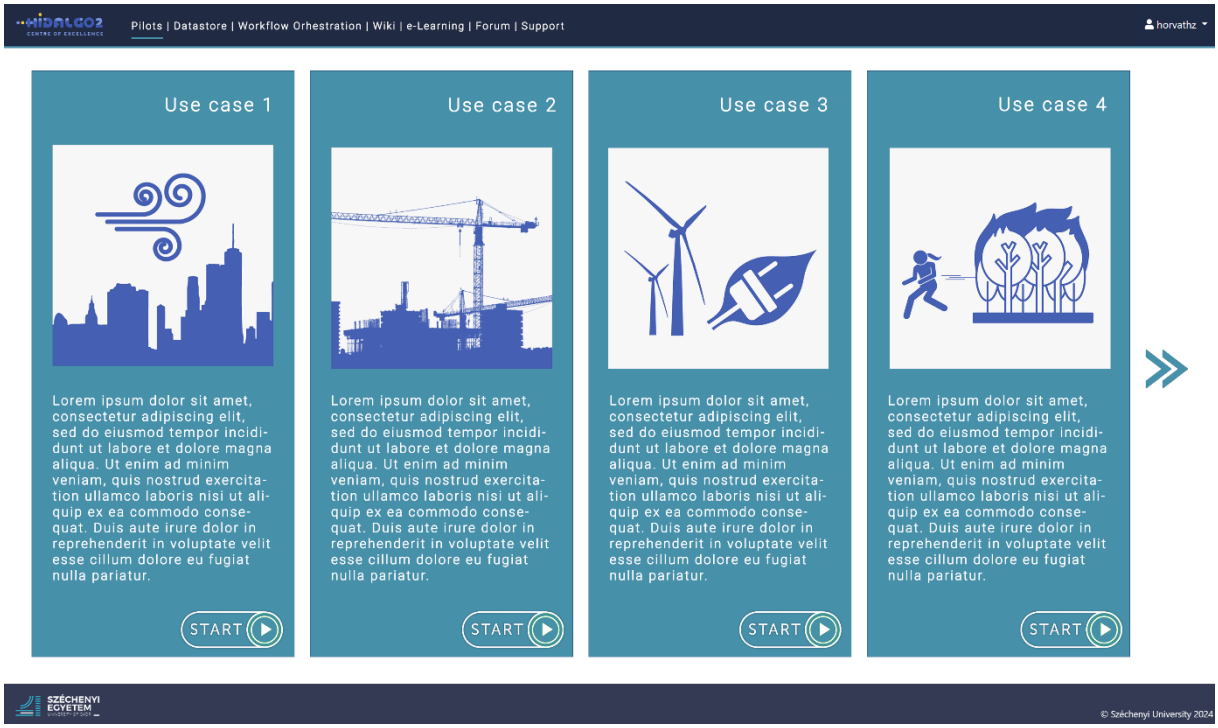

Figure 2: First plan of UI

1. **Frontend:** visual interface presented to the user (UI). The user can interact with this interface to create their own profile or log in with it to navigate to specific web applications. At the top of the dashboard is the menu bar, which provides navigation to each page of the dashboard:
    - Datastore (CKAN): CKAN integration
    - Pilots: main starting page for selecting a pilot and its proper WFO.
    - Forum: Askbot integration
    - Support: Zammad integration
    - Wiki: Wiki.js integration
    - e-Learning: Moodle integration
    - Workflow Orchestration: MathSO and QCG integration
    - Login: SSO Login trough the Keycloak

2. **Backend:** a system running in the background, invisible to the user, containing the necessary system logic:
    - Display user data stored in Keycloak
    - To navigate to the selected system

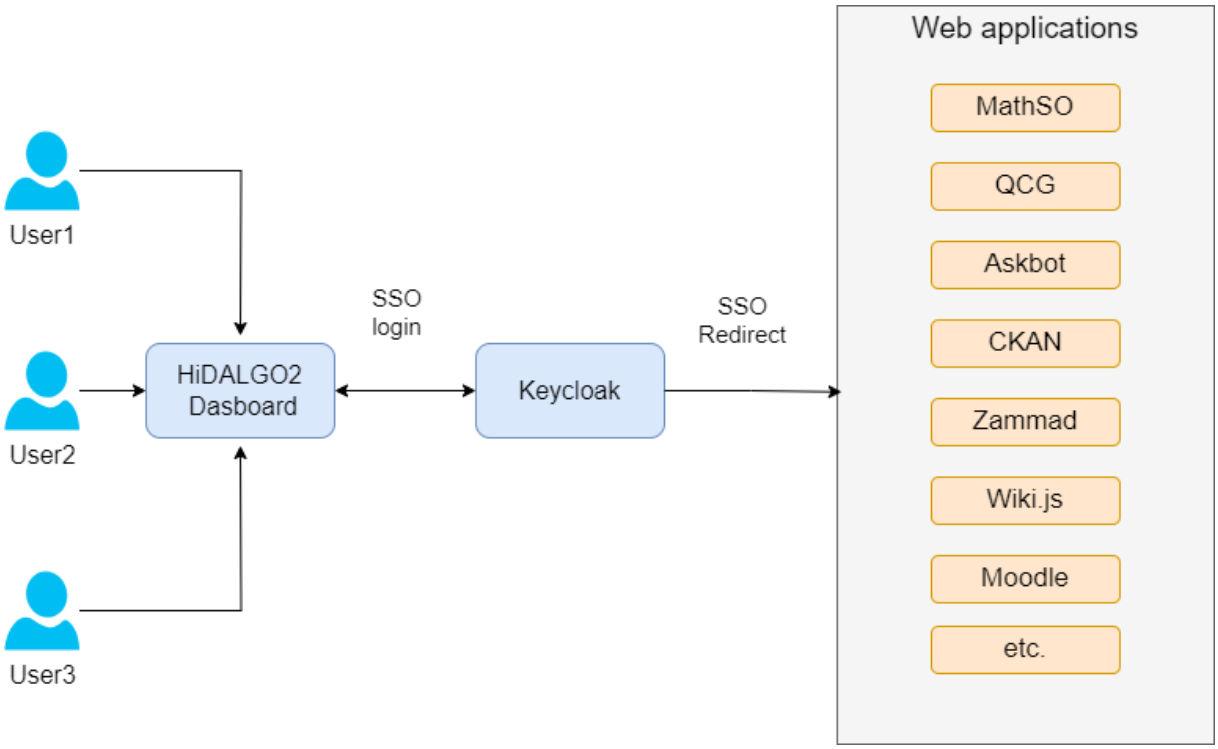

Figure 3: Use-case diagram of the connection between elements

3.2. Authentication, planned functions

The need for Single Sign-On (SSO) in modern systems is increasingly becoming a crucial element of user management and security in the digital age. As organizations grow and adopt a multitude of software solutions, the challenge of managing multiple user credentials has become not just a nuisance for users, but a significant security risk. SSO addresses these challenges by allowing users to access multiple applications with a single set of credentials, streamlining the login process and enhancing user experience while bolstering security measures. The overview of the SSO procedure is shown in Figure 3.

Firstly, SSO significantly improves user experience. In an era where time is of the essence, users often find it frustrating to remember multiple usernames and passwords. This frustration can lead to poor security practices, such as the reuse of passwords across multiple platforms. SSO eliminates this issue by enabling a single authentication process that grants access to all required systems. This not only speeds up the access process but also reduces the cognitive load on users, leading to a more seamless interaction with digital platforms.

From a security standpoint, SSO minimizes the risks associated with password management. By reducing the number of credentials required, SSO decreases the likelihood of password reuse and simplifies the process of implementing strong, unique passwords. Furthermore, it allows for more efficient management of user access rights. In the event of an employee leaving an organization, for example, SSO facilitates the immediate revocation of access to all associated systems, thereby mitigating potential security breaches.

Moreover, SSO plays a critical role in compliance and auditing. Modern regulations often require stringent control over access to information, and SSO provides a centralized point of authentication where access can be monitored and logged. This centralized approach simplifies compliance with various regulations, such as GDPR or HIPAA, by providing clear audit trails of user activities across all integrated systems.

However, the implementation of SSO is not without challenges. It requires a robust infrastructure and careful planning to ensure security and reliability. The centralized nature of SSO creates a single point of failure; if the SSO system is compromised, it potentially opens up access to all connected applications and services. Therefore, implementing strong security measures, including multi-factor authentication and regular security assessments, is essential to mitigate these risks.

In addition, the integration of SSO with existing systems can be complex, particularly for legacy systems that may not support modern authentication protocols. This requires careful planning and often customization to ensure seamless integration without compromising the security or functionality of the connected systems.

Despite these challenges, the benefits of SSO far outweigh the drawbacks, especially as organizations continue to adopt cloud services and remote working becomes more

prevalent. The convenience of SSO supports the mobility and flexibility that modern workforces require, enabling secure access from any location and on any device.

This accessibility is crucial for maintaining productivity in today's fast-paced, digital work environment. Furthermore, SSO solutions often come with advanced features, such as adaptive authentication, which assesses the risk level of each login attempt and adjusts the authentication requirements accordingly. This ensures a balance between security and usability, providing an extra layer of protection without unduly inconveniencing users.

The adoption of SSO also facilitates greater collaboration across platforms and organizations. By simplifying the login process, SSO removes barriers to accessing shared resources, encouraging collaboration and the free flow of information. This is particularly beneficial in environments that rely on a variety of software tools and platforms to support their operations.

In conclusion, the need for Single Sign-On in modern systems is driven by a combination of factors, including the need to improve user experience, enhance security, and comply with regulatory requirements. While the implementation of SSO presents certain challenges, the advantages it offers in terms of convenience, security, and operational efficiency make it an essential component of modern IT infrastructures. As digital transformation continues to evolve, SSO will remain a key enabler of secure, efficient access to the multitude of applications and services that underpin today's organizations.

About Keycloak

Keycloak is an open-source identity and access management solution developed by Red Hat. It provides functionalities such as Single Sign-On (SSO), user authentication, authorization, and user federation for web applications and services.

At its core, Keycloak acts as a centralized authentication and authorization server that manages user identities and their permissions. It supports various authentication mechanisms including username/password, social login (OAuth, OpenID Connect), and multi-factor authentication (MFA).

Keycloak provides an administration console and RESTful APIs for managing users, roles, permissions, client applications, and other aspects of identity and access

management. Additionally, it offers features for securing APIs and microservices through token-based authentication and authorization.

OAuth

As the part of the authentication, we use Oauth2. The OAuth is a protocol designed for delegated authorization, used in distributed systems where a client application needs to access resources on behalf of a user from a resource server.

Keycloak configuration

In order for SSO login from Dashboard to work properly, each web applications (CKAN, Moodle, etc..) must have the correct login settings. Some setup must make only once for the realm (that contains all the clients), and others has to make for each client that represent 1-1 web application.

Realm related setup

1. **Update default configurations:** For proper user experience (like registration, and forgotten password restoration), some default configuration has to be changed:
    a. Enable user registration
    b. Enable forgotten password restoration
    c. Verify email address
    d. Increase SSO Session Idle, and SSO Session Max to ~60days
2. **Set realm email address:** For proper work of the user registration and forgotten password restoration we had to set and email sending parameters (host, port, sender credentials) to send emails to users.

Client related configuration

1. **Client Registration:** Each web-application has to be registered in Keycloak as client. This involves providing details about your application such as the client ID, client secret, and redirect URIs. This information will be used to authenticate and authorize to applications with SSO. During the creation, some parameters have to be set properly:

a. Client type: OpenID Connect (Shown inf Figure 4.)
b. Client authentication: is enabled (Shown inf Figure 5.)

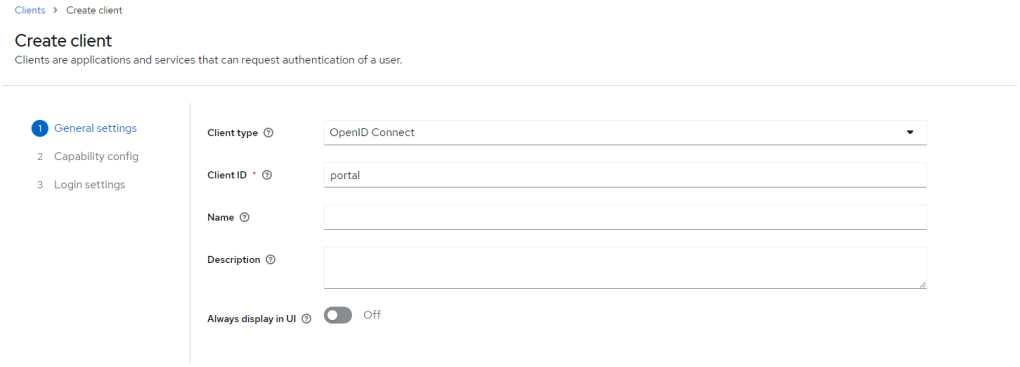

Figure 4: Client must be of type "OpenID Connect" during its generation

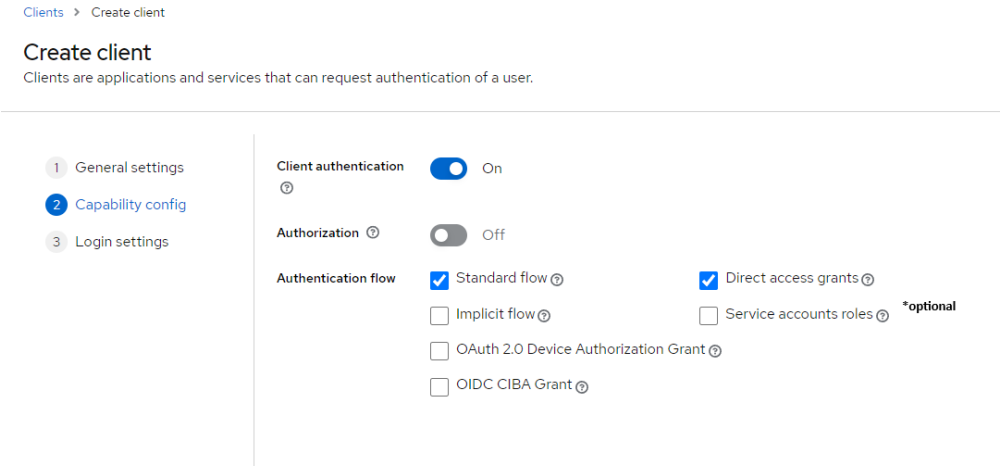

Figure 5: Figure Client authentication must be enabled

2. **Define Roles and Permissions (optional):** Keycloak allows us to define roles and permissions that can be assigned to users or clients. Roles represent the actions or access levels that users can have, while permissions define specific resources or operations that users can perform. We can create roles and permissions in the Keycloak "Roles" section.
3. **Configure Token Settings:** Keycloak provides various token settings that can be configured based on requirements. These settings include token lifespan, token format (JWT or opaque), and token validation settings. We can customize these settings under the "Realm Settings" or "Client Settings" sections.

Dashboard planned user actions

In order to properly redirect to each web application as a logged in user, the Dashboard will not navigate directly to the web interface, but to the Keycloak SSO endpoint associated with the web application, and Keycloak will navigate the user to the target interface after logging in. The key functions for user regarding the authentication are the following:

- **Account registration, and email confirmation:** User related account generation for access of web-applications features (Figure 6).

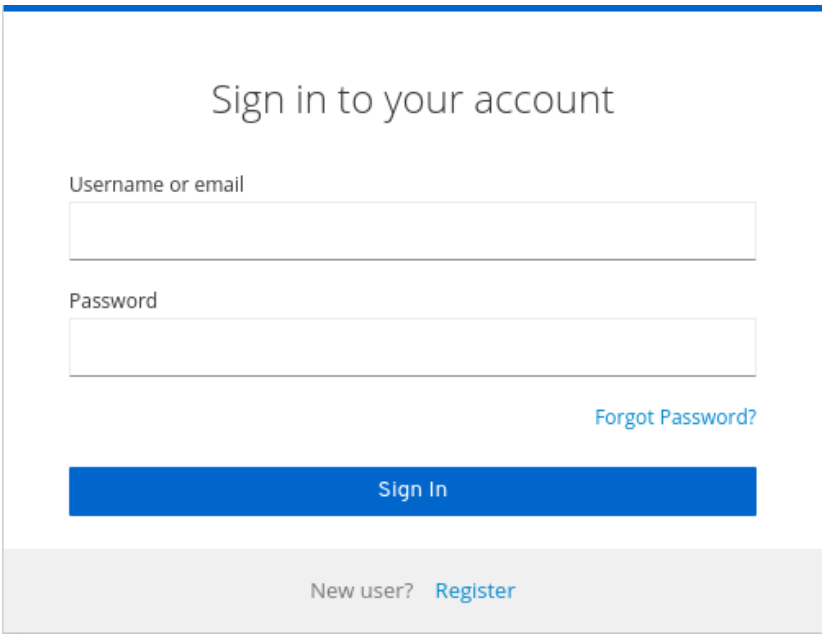

Figure 6: Keycloak signing page

- **User Login:** In case of verified account, the user will be able to log in on the Keycloak login page. They had to give the email address and password (shown in Figure 6).
- **Password reminder:** In case the user does not remember the created account's password, he can send a password change request to the account's email address. Through the link he will be able to set a new password for the account.

3.3. Dashboard implementation technologies

System requirements are defined specifications that outline the necessary hardware and software environments under which a software application or system operates. They are crucial for ensuring compatibility and optimal performance. These requirements are typically divided into two main categories: minimum and recommended.

Minimum system requirements are the basic specifications your computer must meet to run the software, but at the lowest settings or performance level. These include the processor speed, memory (RAM), storage space, operating system version, and, for certain applications, graphics capabilities. Meeting these requirements means the software will run, but not necessarily well or with all features enabled.

Recommended system requirements, on the other hand, are the specifications your computer should meet to experience the software's full potential. These are higher than the minimum and ensure the software runs smoothly, with faster loading times, better responsiveness, and higher quality graphics, where applicable. This category might specify a more powerful processor, greater RAM, more storage space, and a higher-end graphics card.

Adhering to system requirements is essential for users to avoid compatibility issues, crashes, or unsatisfactory performance. For developers and manufacturers, accurately defining these requirements helps users have a better experience with the software, fostering satisfaction and reducing support requests.

Vue.js Node version requirement:

Vue CLI 4.x requires Node.js version 8.9 or above (v10+ recommended). You can manage multiple versions of Node on the same machine with nvm or nvm-windows [7].

Keycloak requirements:

- Can run on any operating system that runs Java
- Java 8 JDK
- zip or gzip and tar
- At least 512M of RAM

- At least 1G of diskspace
- A shared external database like Postgres, MySQL, Oracle, etc. Keycloak requires an external shared database if you want to run in a cluster. Please see the database configuration section of this guide for more information.
- Network multicast support on your machine if you want to run in a cluster. Keycloak can be clustered without multicast, but this requires a bunch of configuration changes. Please see the clustering section of this guide for more information [7].

Dashboard frontend

Vue is a JavaScript framework for building user interfaces. It builds on top of standard HTML, CSS, and JavaScript and provides a declarative and component-based programming model that helps you efficiently develop user interfaces, be they simple or complex [8].

Table 1: Overview of frontend software dependencies

Npm package name	Version	Description	URL
vue	^2.6.12	The progressive JavaScript framework for building modern web UI.	https://www.npmjs.com/package/vue
axios	^0.1.9	Promise based HTTP client for the browser and node.js	https://www.npmjs.com/package/axios
jquery	^3.5.1	JavaScript library for DOM operations.	https://www.npmjs.com/package/jquery
vuex	^3.5.1	State management for Vue.js	https://www.npmjs.com/package/vuex
jsonwebtoken	^9.0.2	JSON Web Token implementation.	https://www.npmjs.com/package/jsonwebtoken

vue-router	^3.4.3		https://www.npmjs.com/package/vue-router
bootstrap-vue	^2.17.3	Implementations of the Bootstrap v4 component and grid system.	https://www.npmjs.com/package/bootstrap-vue
fortawesome/vue-fontawesome	^0.19.3	Official Vue component for Font Awesome 6	https://www.npmjs.com/package/@fortawesome/vue-fontawesome

Dashboard Backend

We use python 3.11 for the build of the backend code. The used python packages are listed below.

Table 2: Python packages dependencies

Pypi package name	Version	Description	URL
flask	3.0.x	Web framework, that lets develop web applications.	https://flask.palletsprojects.com/en/3.0.x/
gunicorn	20.0.4	WSGI HTTP Server for UNIX	https://pypi.org/project/gunicorn/
Python-keycloak	3.7.0	Providing access to the Keycloak API.	https://pypi.org/project/python-keycloak/
Flask-restful	0.3.9	Creating a REST API.	https://github.com/flask-restful/flask-restful/
werkzeug	2.2.2	Implements WSGI, the standard Python interface between applications and servers.	https://pypi.org/project/Werkzeug/

black	24.1.1	Python code formatter	https://pypi.org/project/black/

3.4. Interfacing of workflow orchestrators and web applications

In the following, we will go through each of the features to be developed in a workflow, to see how the user can access and use them. The workflows are provided with a sequence diagram for better clarity.

User registration

To access the functions of each web application, the user must have a profile. By clicking on the "Login" button on the Dashboard, the user will be taken to the Keycloak login interface, where they can click on the "Register" link at the bottom of the page to create their own profile. To do this, the user will need to enter a valid email address, a password and his name. After the successful registration, the Keycloak save the account into the database, but the user won't be able to log in until the email confirmation. The confirmation email will be sent to the email address the user provided, which will contain a confirmation link. The user must open it in order to activate their profile. Failing to do so, even though they have registered, the Keycloak will not allow them to log in.

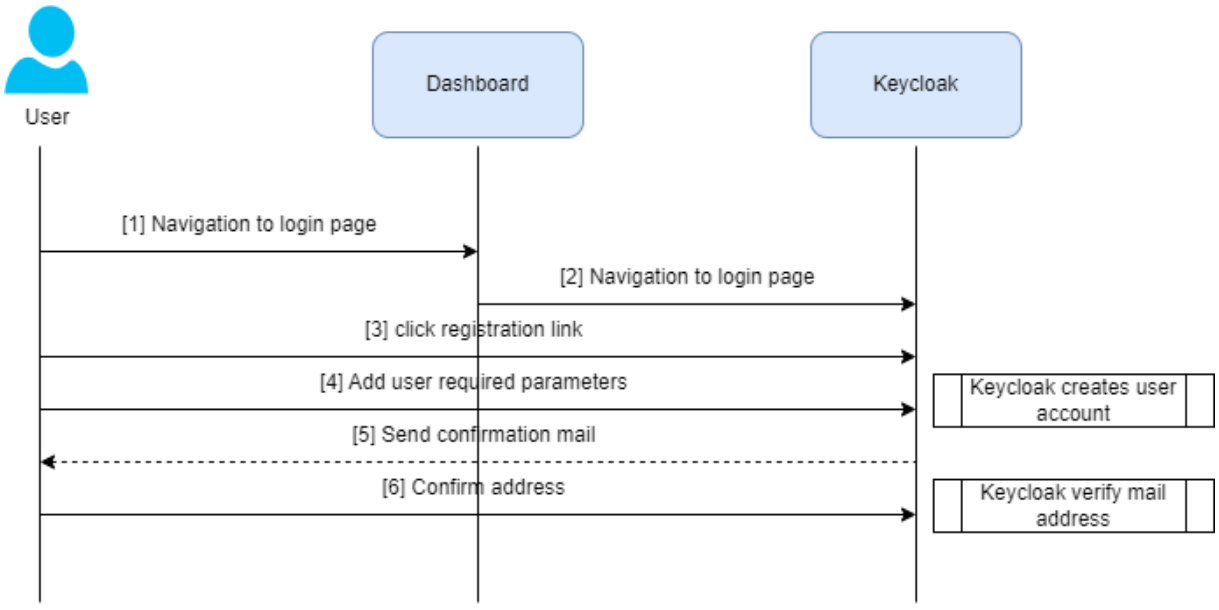

Figure 7: Sequence diagram of the user registration

Login with account

After the successful account registration and verification, the user can log in. The user can reach the login page on the same way as it was described in the previous part. On the Keycloak login page after the user give the proper credentials, the Keycloak will navigate back to the Dashboard site. Now they will see all the connected web-applications.

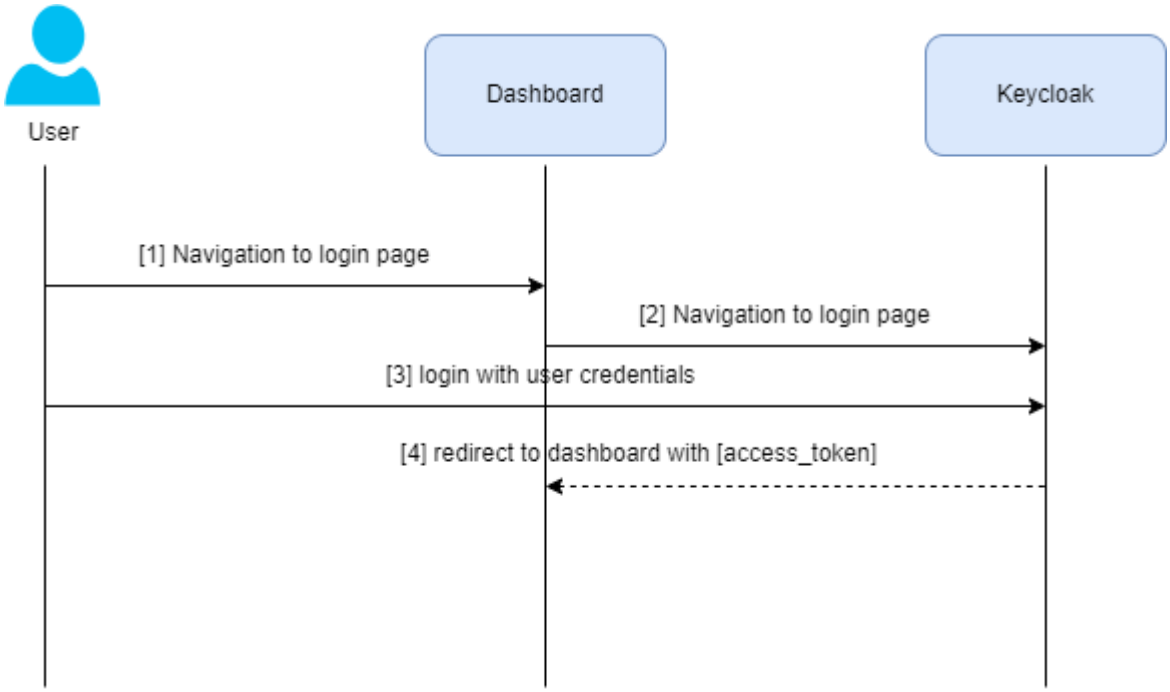

Figure 8: Sequence diagram of the login

Redirect to other web applications

It is a prerequisite that the user has a registered and verified profile to which they are already logged in. The user selects the Portal web application from the Dashboard page already logged in. The backend under the Dashboard determines which SSO login endpoint is associated with that web application and then navigates the user to that endpoint. The Keycloak generates the necessary tokens for the user and redirects them to the Portal site, as logged in user. The user gets all the basic permissions.

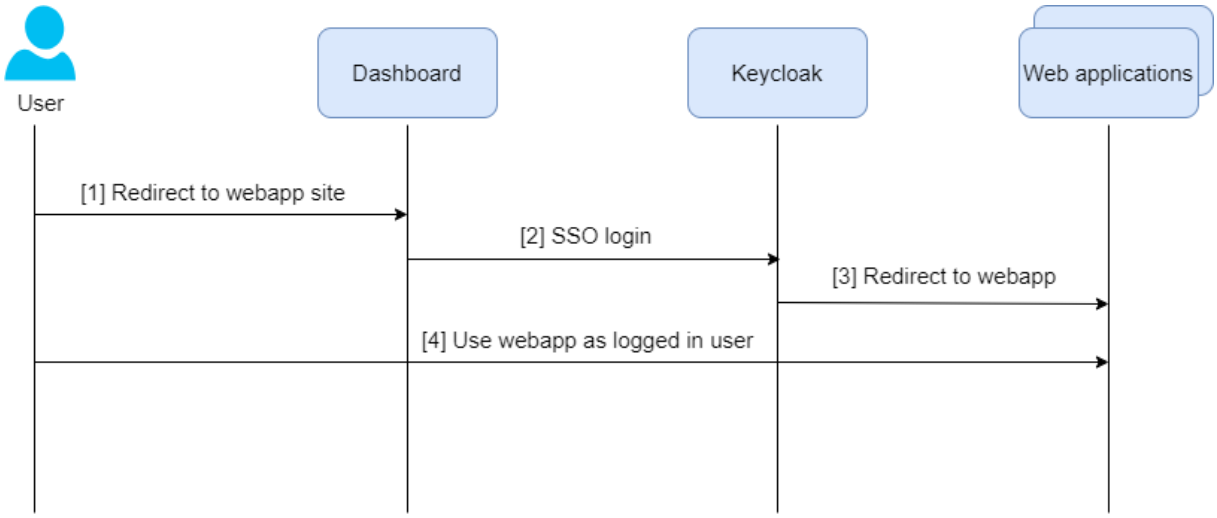

Figure 9: Sequence diagram of the redirection

Further integration of WFO applications

After the initial release we would like to integrate the dashboard feature (not the dashboard as a page) of the MathSO portal into the dashboard, to enable user to see all the necessary information about running jobs. The MathSO portal's backend API offer features to use as a Third-party tool.

Similarly, through the usage of QCG API, HiDALGO2 Dashboard is going to display a bunch of information provided by the QCG system that can be interesting for a user, including:

- list of resources a user has access to,
- for each resource, a detailed information about its assets, configuration, and status,
- type of available CPU/GPU, total number of cores,
- list of available partitions/queues,
- node statistics (total/allocated/free),
- number of queued tasks per partition,
- perhaps some MOTD from the access cluster/news from the cluster website,
- number of uncompleted tasks submitted by a user (queued/running),
- number of submitted tasks by user in 24h,
- number of completed user's tasks in 24h,

The proposed set of displayed information can be extended or limited depending on circumstances. In particular, the provisioning of individual information will depend on

the progress of development of QCG API and discovered needs of HiDALGO2 Dashboard's users.

3.5. Deployment and development roadmaps

Initial Release (end of 2024):

- 2024/02/01 – 2024/02/16: Deliverable documentation reading and writing
- 2024/02/19 - 2024/03/29: Initialize Dashboard – Web-application endpoints, Keycloak setup
- 2024/04/01 - 2024/05/31: Handling of the user interactions (registration, login)
- 2024/06/03 - 2024/08/30: SSO related navigation to the web-applications
- 2024/09/02 - 2024/12/31: Testing, and feedback

2nd release (end of 2025):

- 2025/01/02 - 2025/01/31: Aggregation and evaluation of feedback, recording of tasks resulting from the evaluation
- 2025/02/03 - 2025/10/31: Develop defined tasks
- 2025/11/01 – Testing with internal and external users of round 2 required features

4. Overview of service offer

In the following sections, a short overview of the services making up the HiDALGO2 ecosystem is provided, including the purpose of the services, and their inclusion in the HiDALGO2 dashboard, as well as the system of ensuring the reliable deployment of these services.

4.1. Deployed services

The HiDALGO2 environment centres around the pilot codes and their simulation use-cases, which are supported by web services and tools that aim to help developers as well as future users and customers in utilizing the pilot code applications in an intuitive way. The services provided can be divided into the following broad categories:

1. Compute Services
2. Data Management Services
3. Supporting Services

Each category will be further explained in the next subsections, and an overall list of all available services is presented in Table 3.

Table 3: Overview of the web services

	Service	Description	Domain Link
1	Website	Project Website	https://hidalgo2.eu
2	MathSO Portal	Workflow Orchestrator	https://portal.hidalgo2.eu
3	QCG Portal	Workflow Orchestrator	https://qcg.hidalgo2.eu
3	Prototype	Compute Cluster	https://prototype.hidalgo2.eu
4	JupyterHub	Jupyter Notebooks	https://jupyter.hidalgo2.eu
5	IDM	Identity Management	https://idm.hidalgo2.eu
6	Askbot	User Forums	https://askbot.hidalgo2.eu

7	CKAN	Data Management System	https://ckan.hidalgo2.eu
8	Bitbucket	Git Repository	https://git.man.poznan.pl/stash/projects/HIDALGO2
9	Zammad	User Support	https://ticket.hidalgo2.eu
10	Wiki	Knowledge Management	https://wiki.hidalgo2.eu
11	Open Project	Project Management	https://project.hidalgo2.eu
12	Moodle	Learning Platform	https://moodle.hidalgo2.eu

4.1.1. Computing services

HiDALGO2 needs HPC and other compute infrastructure, to provide its pilots with a testbed to run, test, execute and benchmark their codes and newer features. This infrastructure is also helpful in testing out pilot integrations and workflows with other needed tools, like orchestrators and visualisers.

Access to these computing services is also planned to be provided through the HiDALGO2 dashboard:

- **Altair** - the main HPC System at PSNC.
- **Eagle** - the second HPC System at PSNC.
- **Prototype Cluster** – mini-cluster composed of four compute nodes provided to project partners for fast prototyping purposes for prompt access.
- **JupyterHub** - Jupyter Notebook services are provided to project Partners (and future users if requested).
- **EuroHPC** - pilots are assisted in gaining access to all available EuroHPC JU systems.

4.1.2. Data management services

HiDALGO2 platform includes services for Data Management (DMS) that enable end-users and platform administrators to:

i) Provision (from external Cloud providers or from the HiDALGO2 central data storage), catalogue and manage (e.g., transfer them to the target HPC centres) the data-sets required, by the pilots, to execute their simulations in those HPC centres.

ii) Collect the simulation results back to the central HiDALGO2 data storage for further post-processing and analytics.

Some DMS services are intended to be used by end-users (e.g. pilot's simulation and data-set owners) and others by administrators of the DMS. Current DMS architecture and planned services have been described in D4.1. In this current architecture, the following DMS services are offered for end-users:

- Data Storage:
  - **CKAN**: offers a data-set catalogue system with its own data storage. It also offers a Web-based GUI.
  - **Hadoop HDFS**: offers a highly distributed, high performance data storage system. It includes a HDFS front-end (targeting administrators, but not for end-users), a WebFS REST API (for client interoperability) and an CLI tooling (for end-user interaction). Complementing HADOOP HDFS, we have included in the architecture:
    - **HUE**: a general-purpose end-user centric, Web GUI for Hadoop and other services of the Hadoop ecosystem (e.g. Hive, etc). HUE will offer main GUI data management for end-users.
    - **Apache Atlas**: a catalogue and meta-data management system, with also offers a Web GUI for end users to catalogue the HDFS data-sets with meta-data.
    - Eventually, the DMS architecture can evolved, by addressing the needs of HiDALGO2 pilots. As examples, other services of the Hadoop Ecosystem or related can be included in the architecture,

such as Apache Knox[1], Hive[2], Ranger[3] to improve aspects like authentication access, meta-data management and analytics or authorization policies, respectively.

- Data Transfer:
  - **Apache NIFI**: this is the main data transfer engine in the HiDALGO2 DMS. It includes a Web GUI for designing, executing and monitoring data transfer pipelines. This GUI is intended for pipeline designers and administrators, but not for end-users, which will be offered with a CLI and programmatic API (for integration by other HiDALGO2 services, notably the Workflow Orchestrators) for requesting data transfer jobs, which will be executed by pipelines installed in NIFI.

Some of these DMS services can be integrated into the HiDALGO2 dashboard, by links presented in the dashboard toolbars (or menu entries). By clicking on them, the corresponding DMS service GUI would be opened in another browser tab. The current lists of DMS services to be integrated into the dashboard are, split into services for end-users and administrators:

A. End-users:
   a. CKAN
   b. HDFS HUE
   c. Other (second release): Atlas, Rancher
B. DMS administrators:
   a. Hadoop HDFS GUI
   b. NIFI GUI

By offering a single access point to the DMS data storage (based on CKAN or HDFS) we facilitate pilot's owners the management of their data (either required as input for their simulations or resulting from them). In D4.1 we argued the reasons to adopt two different data storage systems:

- Promoting sharing and cataloguing of the datasets with CKAN

[1] Apache Knox website https://knox.apache.org
[2] Apache Hive website https://hive.apache.org
[3] Apache Ranger website https://ranger.apache.org

- Facilitating the scaling up on demand of the data storage and speeding up the data transfer rates (required by T4.6 Uncertainty Quantification and T3.3 Ensemble Scenarios) and pre/post processing analytics (required by T4.2 HPDA).

4.1.3. Supporting services

The HiDALGO2 ecosystem consists of a wide array of services, ranging from learning platforms to data management platforms. A detailed description of the current state of services was provided in the Deliverable D2.4, “Infrastructure Provisioning, Workflow Orchestration and Component Integration.” A summarized description is provided in section 4.1, “Overview of the Dashboard”, including the relevance of the service to the HiDALGO2 dashboard, as to why these services would become part of the HiDALGO2 dashboard.

4.2. Deployed services in production/testing environment

HiDALGO2 has designed its infrastructure environment for services by keeping in mind the general Infrastructure requirements outlined in Deliverable D2.4:

REQ-HPII-001: Access to HPC infrastructure

- Dedicated Resources for Pilot Simulations as well as for Benchmarking are available internally in the project from PSNC, as well as externally through EuroHPC JU systems.

REQ-HPII-02: Platform Scalability

- Allow for scalability to the number of managed applications and the load/data that each one produces and consumes. Both PSNC and USTUTT commit to increasing resources, e.g. for Data Management.

REQ-HPII-003: Platform Robustness

- High levels of robustness against hardware and software failures, utilise redundancy, load-balancing, and checkpoints. This requirement is fulfilled for web

services through the CI/CD system, as well as by using strategies like separating Development and Production environments.

In order to maintain the robustness of provided services, and to allow for scalability and modifications in the future, HiDALGO2 has established separate development and production Virtual Machine instances for the necessary services. In order to increase the robustness, these development and production instances have been split up between the two main Infrastructure providers of the project: PSNC and USTUTT.

Each Virtual Machine (VM) is provisioned through a robust custom scripting service, depending on the institute, and has measures in place for security, regular updates, and backups.

4.3. Deployment and development roadmaps

In Year 2 of the project, the main focus is to follow and fulfil the deployment requirements and guidelines outlined for all Web Services in the earlier Deliverable D2.4. A summary of these requirements is repeated here:

REQ-CICD-001: The Infrastructure behind the Web Service must be described and specified.

REQ-CICD-002: All Infrastructure dedicated to a Service must be allocated through a CI/CD automation pipeline.

REQ-CICD-003: Services themselves must be built and deployed through a CI/CD automation pipeline.

REQ-CICD-004: Dependencies and additional sub-systems for running the above Services must be handled by the CI/CD automation pipeline.

REQ-CICD-005: Separate Development and Production Instances must be provided and handled by the CI/CD pipeline.

REQ-CICD-006: The setting up and deployment of Services and Infrastructure must be reliable and idempotent (replicable).

The key component of the road map is to extend the automated deployment of the components of the HiDALGO2 dashboard. This is achieved through the automation

platform, Ansible [10] which helps to convert any manual process of deploying services to automated, reusable scripts in the form of Ansible Playbooks [10].

These Ansible playbooks can then be triggered manually, as would be preferred for Production instances, or also automated through CI/CD provided through runners, e.g.

i) Gitea Runners [11]
   used for the initial automation of services so far.
ii) Gitlab Runners [12]
   potential replacement in Year 2.
iii) Bitbucket Pipelines [13].
iv) Other systems like Jenkins [14] and Woodpecker CI [15]

The following time line breaks down the steps to achieve the above requirements:

- **Month 14:** Target of having 4 Web Services ready for automated deployment to Dev Instances, through Ansible Playbooks.
- **Month 16:** Adding 4 more Web Services, and adding the automated running of these playbooks through Runners or another CI/CD system.
- **Month 18**: Target of remaining supporting Web Services to be covered be automated.
- **Month 22**: Extend automated deployments to Development instance for the Dashboard itself, as well as the Workflow Orchestrators.
- **Month 24**: Have manual triggering of the automated deployments for Production Infrastructure.

5. Conclusions and future work

This deliverable addresses the challenges of putting together all the different services HiDALGO2 project will deliver to partners and customers. The infrastructure is strengthened by setting up and maintaining various web services that provide support to developers and internal project members, as well as lay a platform for future external users and customers. This includes a brief look ahead at the envisioned HiDALOG2 dashboard, and an in-depth look at the advancement and integration of HiDALGO2's workflow orchestrators, MathSO and QCG. Requirements for the workflow orchestrators have been carefully collected from pilot partners, to ensure that the development of the workflow orchestrators aligns as closely as possible to the simulation use cases and project goals.

The deliverable contributors have agreed that the HiDALGO2 dashboard will be a single point of entry to all available and planned HiDALGO2 services.

From ticketing to e-Learning platform all services must have a single-sign-in mechanism, that allows to use all services without further authentication.

HiDALGO2 will continue working on achieving its technical infrastructure and component integration goals by following the roadmaps developed through this deliverable, and report on the advancement of its efforts in the follow up deliverables, D2.8 (M28) and D2.9 (M42).

References

[1] https://ckan.org/ - The world's leading open source data management system – last opened: 2024/02/05

[2] https://zammad.com last opened: 2024/02/12

[3] https://js.wiki last opened: 2024/02/18

[4] https://moodle.org last opened: 2024/02/18

[5] https://www.openproject.org last opened: 2024/02/18

[6] https://bitbucket.org last opened: 2024/02/18

[7] https://vuejs.org/guide/introduction.html - Introduction – last opened: 2024/02/05

[8] https://wjw465150.gitbooks.io/keycloak-documentation/content/server_installation/topics/installation/system-requirements.html - System Requirements – last opened: 2024/02/05

[9] https://cli.vuejs.org/guide/installation.html - Installation – last opened: 2024/02/06

[10] https://www.ansible.com/ - Use Cases – last opened: 2024/02/06

[11] https://docs.gitea.com/usage/actions/act-runner – Gitea Act Runner – last opened: 2024/03/12

[12] https://docs.gitlab.com/runner/ – Gitlab Runner – last opened: 2024/03/12

[13] https://bitbucket.org/product/de/features/pipelines – Bitbucket Pipelines - last opened: 2024/03/12

[14] https://www.jenkins.io/ – Jenkins - last opened: 2024/03/12

[15] https://woodpecker-ci.org/ – Woodpecker CI - last opened: 2024/03/12

Annexes

Annex 1. List of requirements

#ID	Title	Description	Required by	Release date
DASH-REQ-001	Access all pilot	Access to all pilots using only dashboard, and linking to the proper WFO portal	SZE	Initial release (2024.11.30)
DASH-REQ-002	Access CKAN directly from dashboard	After login, users will be able to create datasets and upload data to CKAN	SZE	Initial release (2024.11.30)
DASH-REQ-003	SSO auth	Using dashboard as a single entry point for HiDALGO2 infrastructure	SZE	First Iteration release (2025.06.01)
DASH-REQ-004	User levels in Dashboard infra	Different users with different roles can access different services (example: admin users can access to Keycloak to create new user)	SZE	Final release (2026.01.01)
DASH-REQ-005	Access from dashboard to HiDALGO2 DMS frontends	Multiple DMS frontends will be provided, each on its own frontend service (different URLs), This list of DMS services includes services for end-users (not exhaustive lists, could be increased in future): - CKAN (already posted in above requirement) - HUE frontend (Hadoop frontend for users) - Atlas (Catalogue) - to be confirmed. - Apache Knox (Gateway) - to be confirmed Other DMS services, intended for administration are not required to be integrated within the frontend, they can be accessed by administrators in their own URL: - Hadoop UI - NIFI	ATOS	Initial release (2024.11.30)

DASH-REQ-006	Access to CKAN	-interaction with data management platform ;	UNISTRA	Initial release (2024.11.30)
DASH-REQ-007	Data preparation frontend	run Ktirio-GUI in wasm mode in the dashboard, to prepare the dataset before upload to CKAN	UNISTRA	First Iteration release (2025.06.01)
DASH-REQ-008	Access documentation website of Ktirio urban building (KUB)	the KUB pilot has a documentation explaining how to configure and run it, the documentation is hosted on a website than should be accessible	UNISTRA	First Iteration release (2025.06.01)
DASH-REQ-009	Monitor pilot health	use reframe JSON files to produce a dashboard of the health of the pilot over time and over the different machines used; this should be generated automatically e.g., through CI/CD or portal	UNISTRA	First Iteration release (2025.06.01)
DASH-REQ-010	Visualisation	Enable visualisation through the dashboard either 3D interactive but also through Jupyter files and automated report generation	UNISTRA	Initial release (2024.11.30)
DASH-REQ-011	Embed seamlessly webpages associated to KUB	KUB provides webpages using ascii doc that are rendered into websites. it would be super to be able to integrate seamlessly these pages into the dashboard	UNISTRA	Initial release (2024.11.30)
DASH-REQ-012	Access to CKAN	interaction with data management platform, including historic data provided by other pilots for reanalysis and/or using this as initial input data for RES. Browse historic data of RES runs.	PSNC (RES)	Initial release (2024.11.30)
DASH-REQ-013	QCG	interaction with QCG for ensemble runs and uncertainty quantification	PSNC (RES)	Initial release (2024.11.30)
DASH-REQ-014	Ticketing system	access to ticketing system to solve issues users encounter	PSNC (RES)	Initial release (2024.11.30)
DASH-REQ-015	Visualisation	visualisation of RES results using in-house scripts and/or other HiDALGO2 visualization tools	PSNC (RES)	First Iteration release (2025.06.01)
DASH-REQ-016	Monitoring	monitoring of pending jobs, status of hidalgo2 tools and status of EuroHPC JU sites	PSNC (RES)	First Iteration release (2025.06.01)
DASH-REQ-017	Boundary conditions download	Select model for boundary conditions / Select date / Select regional area -> DMS (Hadoop?) Combustibles type	MTG (WF) Landscape	Final release (2026.01.01)

DASH-REQ-018	Simulation environment definition	Name list configuration / submit scripts	MTG (WF) Landscape	First Iteration release (2025.06.01)