



Large-scale EXecution for Industry & Society

Deliverable D8.1

First Release of LEXIS Portal



Co-funded by the Horizon 2020 Framework Programme of the European Union
Grant Agreement Number 825532
ICT-11-2018-2019 (IA - Innovation Action)

DELIVERABLE ID TITLE	D8.1 First Release of LEXIS Portal
RESPONSIBLE AUTHOR	Ruben Garcia Hernandez (LRZ)
WORKPACKAGE ID TITLE	WP8 LEXIS Portal for Third Party Companies and SMEs
WORKPACKAGE LEADER	CYC
DATE OF DELIVERY (CONTRACTUAL)	31/03/2020 (M15)
DATE OF DELIVERY (SUBMITTED)	30/04/2020 (M16)
VERSION STATUS	V1.3 Final
TYPE OF DELIVERABLE	DEM (Demonstrator)
DISSEMINATION LEVEL	PU (Public)
AUTHORS (PARTNER)	Frederic Donnat (O24), Sean Murphy (CYC), Piyush Harsh (CYC), David Hruby (IT4I), Aaron Dees (BLABS), Ruben Garcia Hernandez (LRZ)
INTERNAL REVIEW	Paolo Pasquali (ITHACA), Ennio Spano (Avio)

Project Coordinator: Dr. Jan Martinovič – IT4Innovations, VSB – Technical University of Ostrava
E-mail: jan.martinovic@vsb.cz, **Phone:** +420 597 329 598, **Web:** <https://lexis-project.eu>

DOCUMENT VERSION

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	ToC	20/01/2020	Sean Murphy (CYC)
0.2	Inputs from CYC, LRZ, BLABS	10/02/2020	Sean Murphy (CYC), Ruben Garcia (LRZ), Aaron Dees (BLABS)
0.3	Review, clean ups by CYC	19/02/2020	Sean Murphy (CYC)
0.4	Add section about LEXIS AAI and Keycloak configuration, added screenshots, mockups and summary section	27/02/2020	Frederic Donnat (O24), Sean Murphy (CYC)
1.0	Clean ups to prepare for review - regularized figure captions, clean exec summary	06/03/2020	Sean Murphy (CYC)
review	Internal Review of Deliverable	10/03/2020	Paolo Pasquali (ITHACA), Ennio Spano (Avio)
1.1	Update of the document according to the review comments	15/03/2020	Sean Murphy (CYC)
1.2	Incorporation of EAB feedback and document clean up	27/03/2020	Sean Murphy (CYC)
1.21	Review and small fix to DDI section	03/04/2020	Ruben J. Garcia-Hernandez (LRZ)
1.22	Changes required to API definitions for portal API and userorg service	06/04/2020	Sean Murphy (CYC)
1.23	Added configuration parameters for the different component	14/04/2020	Sean Murphy (CYC)
1.3	Final check of the deliverable including minor clean ups	30/04/2020	Katerina Slaninova (IT4I), Sean Murphy (CYC)

GLOSSARY

ACRONYM	DESCRIPTION
JWT	JSON Web Token
DDI	Distributed Data Infrastructure
HPC	High Performance Computing
FE	Front-End
API	Application Programming Interface
AAI	Authentication & Authorization Infrastructure
URL	Uniform Resource Locator
OIDC	OpenID Connect
UUID	Universally Unique Identifier
JSON	JavaScript Object Notation
A4C	Alien4Cloud
IAM	Identity and Access Management
HEAPPE	High-End Application Execution
REST	Representational State Transfer
XML	Extensible Markup Language
CSV	Comma Separated Values

TABLE OF PARTNERS

ACRONYM	PARTNER
Avio Aero	GE AVIO SRL
AWI	ALFRED WEGENER INSTITUT HELMHOLTZ ZENTRUM FUR POLAR UND MEERESFORSCHUNG
BLABS	BAYNCORE LABS LIMITED
Bull/Atos	BULL SAS
CEA	COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES
CIMA	Centro Internazionale in Monitoraggio Ambientale - Fondazione CIMA
CYC	CYCLOPS LABS GMBH
ECMWF	EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS
GFZ	HELMHOLTZ ZENTRUM POTSDAM DEUTSCHESGEOFORSCHUNGSZENTRUM GFZ
IT4I	VYSOKA SKOLA BANSKA - TECHNICKA UNIVERZITA OSTRAVA / IT4Innovations National Supercomputing Centre
ITHACA	ASSOCIAZIONE ITHACA
LINKS	FONDAZIONE LINKS / ISTITUTO SUPERIORE MARIO BOELLA ISMB
LRZ	BAYERISCHE AKADEMIE DER WISSENSCHAFTEN / Leibniz Rechenzentrum der BAdW
NUM	NUMTECH
O24	OUTPOST 24 FRANCE
TESEO	TESEO SPA TECNOLOGIE E SISTEMI ELETTRONICI ED OTTICI

TABLE OF CONTENTS

EXECUTIVE SUMMARY	5
1 INTRODUCTION	7
1.1 PURPOSE OF THE LEXIS PORTAL	7
1.2 WHY ANOTHER PORTAL TO ACCESS HPC RESOURCES?	7
1.3 LEXIS PORTAL DESIGN CONSIDERATIONS	8
1.4 STRUCTURE OF THIS DOCUMENT	8
2 LEXIS PORTAL ARCHITECTURE.....	9
3 LEXIS PORTAL COMPONENTS AND INTEGRATIONS.....	11
3.1 LEXIS PORTAL FRONT-END.....	11
3.1.1 <i>Front-end Modular Structure</i>	11
3.2 LEXIS PORTAL FE SERVER	12
3.3 LEXIS PORTAL API	13
3.4 LEXIS USERORG SERVICE.....	14
3.5 LEXIS DATAMANAGEMENT INTERFACE SERVICE.....	18
3.6 LEXIS ALIEN4CLOUD INTERFACE SERVICE.....	19
3.7 LEXIS AAI CONFIGURATION	21
3.8 LEXIS CYCLOPS INTEGRATION	23
4 LEXIS PORTAL SCREENSHOTS AND MOCKUPS.....	26
4.1 SCREENSHOTS FROM LEXIS PORTAL.....	26
4.2 MOCKUPS OF FUTURE LEXIS PORTAL IMPLEMENTATION	30
5 LEXIS PORTAL DEPLOYMENT.....	33
5.1 DEPLOYING LEXIS PORTAL COMPONENTS	33
5.1.1 <i>System prerequisites</i>	33
5.1.2 <i>Bringing up the system</i>	34
5.1.3 <i>Bringing up the datasetmanagement-service</i>	34
5.1.4 <i>Bringing up the Alien4Cloud-integration service</i>	35
5.1.5 <i>Bringing up the user-org-service</i>	36
5.1.6 <i>Bringing up the Portal API</i>	37
5.1.7 <i>Bringing up the FE Server</i>	38
6 SUMMARY AND NEXT STEPS	40
6.1 SUMMARY OF CONTENT IN THIS DOCUMENT	40
6.2 NEXT STEPS IN DESIGN OF LEXIS PORTAL.....	40
REFERENCES.....	41
A EXAMPLES OF QUERYING CYCLOPS SERVICES	42

LIST OF TABLES

TABLE 1 ENDPOINTS PROVIDED BY THE LEXIS PORTAL API.....	14
TABLE 2 ENDPOINTS PROVIDED BY THE USERORG SERVICE.....	15
TABLE 3 THE USER DATA MODEL.....	16
TABLE 4 THE ORGANIZATION DATA MODEL.....	16
TABLE 5 PROJECTS DATA MODEL.....	17
TABLE 6 THE HPCRESOURCES DATA MODEL.....	17
TABLE 7 API ENDPOINTS PROVIDED BY THE DATASET MANAGEMENT INTERFACE.....	18
TABLE 8 EXAMPLES OF SIMPLE COMMAND-LINE USE OF THE APIS.....	18
TABLE 9 ENDPOINTS PROVIDED BY ALIEN4CLOUDINTERFACE SERVICE.....	19
TABLE 10 THE LEXIS WORKFLOW TEMPLATE MODEL.....	20
TABLE 11 THE LEXIS WORKFLOW MODEL.....	20
TABLE 12 THE LOGS MODEL.....	21
TABLE 13 THE CYCLOPS FRAMEWORK ELEMENTS.....	25
TABLE 14 CONFIGURATION PARAMETERS FOR DATASETMANAGEMENT INTERFACE.....	35
TABLE 15 CONFIGURATION PARAMETERS FOR ALIEN4CLOUD INTERFACE.....	36
TABLE 16 CONFIGURATION PARAMETERS FOR USERORGSERVICE.....	37
TABLE 17 CONFIGURATION PARAMETERS FOR LEXIS PORTAL API.....	38
TABLE 18 CONFIGURATION PARAMETERS FOR PORTAL FE SERVER.....	39

LIST OF FIGURES

FIGURE 1 HIGH LEVEL VIEW OF LEXIS PORTAL COMPONENTS.....	9
FIGURE 2 CONFIGURATION OF A CLIENT WITHIN A KEYCLOAK REALM.....	22
FIGURE 3 CLIENTS DEFINED WITH A KEYCLOAK REALM.....	22
FIGURE 4 CONFIGURING A GROUP WITH LIMITED PERMISSIONS.....	23
FIGURE 5 HIGH LEVEL VIEW OF CYCLOPS FRAMEWORK COMPONENTS.....	24
FIGURE 6 LEXIS PORTAL LANDING PAGE.....	26
FIGURE 7 LEXIS PORTAL LOGIN PAGE.....	26
FIGURE 8 LEXIS PORTAL POST LOGIN PAGE.....	26
FIGURE 9 LISTING USERS DEFINED WITHIN THE USERORG SERVICE.....	27
FIGURE 10 SHOWING USER PROFILE WITHIN THE LEXIS PORTAL.....	27
FIGURE 11 REGISTERING A NEW USER WITHIN THE LEXIS PORTAL.....	27
FIGURE 12 LISTING ORGANIZATIONS DEFINED WITHIN THE USERORG SERVICE.....	28
FIGURE 13 REGISTERING A NEW ORGANIZATION WITHIN THE LEXIS PORTAL.....	28
FIGURE 14 LISTING PUBLIC DATA SETS WITHIN THE LEXIS PORTAL.....	28
FIGURE 15 CREATING A LEXIS COMPUTATIONAL PROJECT.....	29
FIGURE 16 VIEWING DETAILS OF A LEXIS COMPUTATIONAL PROJECT.....	29
FIGURE 17 EDITING DETAILS RELATING TO A LEXIS COMPUTATIONAL PROJECT.....	29
FIGURE 18 VIEWING USAGE DETAILS RELATING TO A LEXIS COMPUTATIONAL PROJECT.....	30
FIGURE 19 MOCK-UP OF LANDING DASHBOARD IN LEXIS PORTAL.....	30
FIGURE 20 MOCK-UP OF DETAILED PROJECT INFORMATION WITHIN LEXIS PORTAL.....	31
FIGURE 21 MOCK-UP OF WORK-FLOW CREATION INTERFACE WITHIN THE LEXIS PORTAL.....	32

EXECUTIVE SUMMARY

An important aspect of the LEXIS project is to make HPC resources more accessible; this can be considered in two dimensions:

- Enable experienced users to have access to multiple sets of HPC resources such that they can distribute work across the available resources,
- Reduce friction in enabling less experienced users to gain access to HPC resources, see data sets and deploy computations to the available resources.

The LEXIS Portal is the primary vehicle by which this can be achieved.

Position of the deliverable in the whole project context

The deliverable showcases the preliminary results of Tasks 8.1, 8.2 and 8.3, which provide access to datasets, monitoring, accounting and billing. Synergies between WP8 on one side and WP2, 3, 4 ensure that the APIs provide the information needed for final users. A portal showcasing the expected look and feel of LEXIS, which starts by showcasing mockup-data, but which is modified to access real data from the other work packages as their APIs are stabilized and published, has been developed. The deliverable is related to other LEXIS deliverables as follows: D2.2 [1] and D2.3 [2] which describe the overall LEXIS architecture, D3.3 [3] which describes APIs developed within WP3 and D4.5 [4] which describes the AAI model developed for LEXIS.

This deliverable will be followed by D8.2 [5] and D8.3 [6] which will describe future releases of the LEXIS Portal.

Description of the deliverable

The deliverable describes the first public iteration of the LEXIS Portal, which provides users access to a user-centric view of their data, workflows, permissions, and other information handled by LEXIS on their behalf.

The deliverable is divided into the following chapters: Introduction, Portal Architecture (a high level view of the design of the LEXIS Portal), Components and Integration (a more detailed component level description), screenshots and mock-ups (screenshots of the LEXIS Portal), deployment, and current status and next steps.

The contributors to the deliverable were primarily those responsible for the development of the software components and the design and configuration of the AAI system. These are as follows:

- Reactive Front End (IT4I),
- User and organization management service and portal API, integration with accounting and billing functions (CYC),
- Dataset management interface (LRZ),
- Orchestration integration (BLABS),
- Configuration of AAI for development and test (O24).

1 INTRODUCTION

LEXIS WP8 is focused on developing a Portal to reduce friction when accessing HPC resources both for pilots within the project as well as third parties via the Open Call. This deliverable is the first deliverable from WP8 which comprises of a software release - a tagged version of the software in the LEXIS repository and some basic documentation relating to the functionality it provides and how it can be deployed and used.

The LEXIS Portal comprises of four distinct releases¹ R1, R2, R3, R4 - each with increasing levels of capability/functionality. R1 was an initial internal release within the project at M9 but was delayed a little until M10. R2 is the focus of this deliverable and R3 and R4 will be delivered in M24 and M30, respectively.

1.1 PURPOSE OF THE LEXIS PORTAL

The LEXIS Portal is an open portal which will provide access to distributed HPC/Cloud resources to support operations relevant to users of such services. The primary function is to make it easier for non-experienced users of HPC resources to engage with HPC resources; enabling them to work with big data sets, create and deploy jobs and see the results of the jobs. Also, within a bigger context of making HPC resources accessible to users outside the classical scientific community, the LEXIS Portal will track usage and provide accounting and billing for usage of the platform.

The LEXIS Portal will ultimately support the following capabilities:

- Portal sign up,
- Log in to the Portal,
- Add an Organization to the Portal,
- Create a LEXIS Computational Project within the Portal,
- Request access to HPC resources, specifying intended use, type of resources and amount of resources; if the request is approved, these will be added to the LEXIS Computational Project,
- View available resources within the LEXIS Computational Project,
- Add users to/delete users from a given LEXIS Computational Project,
- View data sets that are available within the LEXIS Computational Project,
- View DDI resources that are visible to the user,
- View public DDI resources,
- Upload data to the DDI via the Portal,
- Create a LEXIS Workflow based on a set of compute elements,
- Create a LEXIS Job based on the LEXIS workflow, adding input data sets and parameterization,
- Monitor and manage the LEXIS job during its operation, including viewing status and log messages that have been produced by the job,
- View the output produced by the LEXIS Job,
- View the accounting information for a specific LEXIS Computational Project including the accounting information for potentially multiple HPC Computational Projects that constitute the LEXIS Computational Project,
- View billing information for a specific LEXIS Computational Project - formalized bills or invoices associated with the LEXIS Computational Project.

1.2 WHY ANOTHER PORTAL TO ACCESS HPC RESOURCES?

There has, of course, been other work on portals to support access to HPC resources; indeed, one of the LEXIS Partners - Bull/Atos - provides a portal to facilitate their HPC as a service offering [7]. Another significant related

¹ Note: three distinct releases were defined in the Grant Agreement; it was decided to have an earlier internal release to address integration problems early

technology is the Open OnDemand project [8]. Open OnDemand offers access to job management, file system browsing and terminal sessions; as such, there is some overlap with the functionality provided by LEXIS. LEXIS, however, has a different focus which means that a different design is required. More specifically, LEXIS needs to support resources across multiple HPC centres and facilitate access to data sets and compute resources across multiple infrastructures; further, an important aspect of LEXIS is that it supports integration of cloud and HPC technologies - it does not focus exclusively on established HPC technologies. Finally, LEXIS supports advanced systems such as the Alien4Cloud/Ystia orchestration tool and technologies such as Smart Burst Buffers. For these reasons, the LEXIS portal was developed from scratch.

1.3 LEXIS PORTAL DESIGN CONSIDERATIONS

The approach to designing the LEXIS Portal was based on a compromise between:

- The Agile Development approach which is well suited to many web applications and is characterized by an understanding that requirements evolve as usage of a system evolves and the development process should reflect this by being adaptive,
- The classical requirements, design, implementation approach which is more compatible with the structure of an H2020 R&D project.

The basic approach was to have a quite informal requirements gathering phase based on an understanding of typical data management solutions and HPC workflows. Some implementation work could start early based on approximate requirements and the project team could gain some feedback and perspective of the design from the pilots within the project. This feedback could be taken into the next phase of the design and would be reflected in future releases of the Portal.

1.4 STRUCTURE OF THIS DOCUMENT

This document is structured as follows. Section 2 provides a high-level view of the design of the LEXIS Portal, highlighting the main architectural components, their respective functionalities, and their interfaces. Section 3 provides a more detailed, component level description, including the specific interfaces provided by the components and the design of the components. Section 4 provides some screenshots of the LEXIS Portal so the reader can understand the user experience. Section 5 describes the deployment and configuration of the different constituent components. Section 6 provides a summary of the current state of the development of the LEXIS Portal and identifies the next steps in the development process.

2 LEXIS PORTAL ARCHITECTURE

The design of the LEXIS Portal is evolving based on feedback from the implementation process and a deeper understanding of key supporting technologies (e.g. iRODS, Keycloak, Ystia); as such the final LEXIS Portal design may differ slightly from the current design.

The current design of the LEXIS Portal - as of LEXIS Portal R2 - is shown in Figure 1.

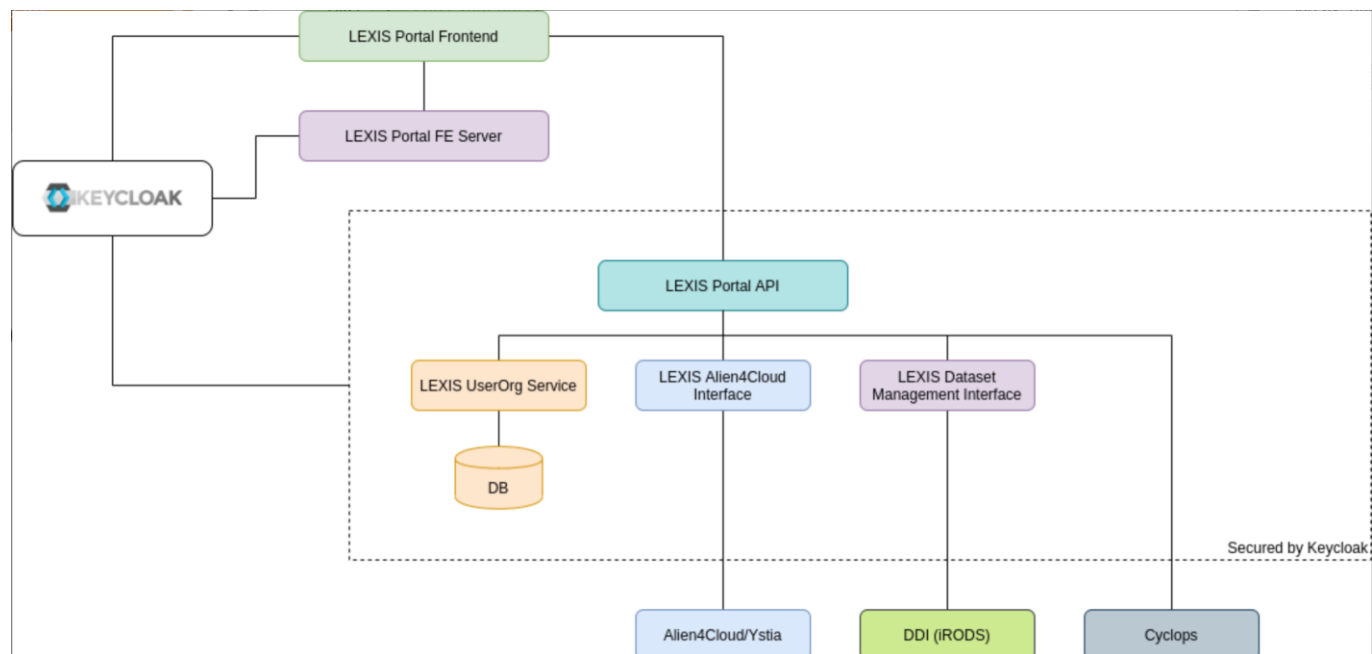


Figure 1 High Level View of LEXIS Portal Components

As can be seen from the Figure 1, the Portal comprises of the following components:

- The LEXIS Portal Front End: this is a rich React based Front End that interacts with the LEXIS Portal Front End Server (FE Server) and primarily the LEXIS Portal API
- The LEXIS Portal FE Server: this is a simple process that serves the Front End content and obtains a token from Keycloak
- The LEXIS Portal API: this is the main entry point by which the Front End can access functionality offered by the Portal - a valid token is necessary to access any functionality offered by the Portal API
- The LEXIS UserOrg service: this is a service which contains the set of users, organizations and projects defined within the system - it persists this information in a local database
- The LEXIS Alien4Cloud interface: this service performs interaction with Alien4Cloud to support definition of LEXIS Workflows and deployment of LEXIS Jobs
- The LEXIS Data Management Interface: this service supports interaction with the DDI to support listing of directories on the DDI and uploading and download files to and from the DDI

Keycloak has been selected to implement the LEXIS AAI capabilities as discussed in Deliverable D2.2 [1] and for this reason, services interact with Keycloak² to perform token verification and determine user authorizations on the backend services.

² Keycloak: <https://www.keycloak.org/>

The Portal services must interact with the lower level services that constitute the LEXIS Platform including the DDI - implemented with iRODS³ - the job management and orchestration services - provided via Ystia/Alien4Cloud⁴ - and the accounting and billing service - realized via the Cyclops accounting and billing software⁵ provided by CYC.

In general, the Portal services interact via HTTP/HTTPS requests. OpenID Connect⁶ is the primary authentication mechanism and HTTP/HTTPS requests must contain valid tokens to access the secured endpoints offered by the backend services. The Portal API offers a single entry point to the backend services which exposes this functionality on an as needs basis, thus helping to minimize the exposed attack surface of the system.

³ iRODS: <https://irods.org/>

⁴ Alien4Cloud: <https://alien4cloud.github.io/>

⁵ Cyclops: <https://github.com/Cyclops-Labs/cyclops>

⁶ OpenID Connect: <https://openid.net/connect/>

3 LEXIS PORTAL COMPONENTS AND INTEGRATIONS

3.1 LEXIS PORTAL FRONT-END

The LEXIS Portal Front-end (FE) is a rich browser-based web application which communicates primarily with the LEXIS Portal API. The FE is modular in structure, making it easy extensible; further views are tightly coupled to data obtained from the Portal API providing further isolation of concerns and providing support for contributions from different members of the LEXIS development team.

The LEXIS Portal FE builds on widely used and mature front-end web technologies including the following:

- React⁷: modern, efficient JavaScript⁸ library which provides abstractions over the Document Object Model (DOM) to easily capture events coming from the browser and update state within the visible part of the application with ease,
- Redux⁹ (including `redux-sauce` and `redux-saga`): a set of JavaScript libraries which are widely used and integrated well with React which provides support for managing application state within the browser and obtaining state as necessary from the server side - in this case, the Portal API - including asynchronous communications with the server,
- Router5¹⁰: this provides a means to enrich the link between a client side application state and a hierarchy of server-side endpoints which serve application data; the basic notion is that this hierarchy is a good way to represent application data will result in reduced interaction between the client and the server.

3.1.1 Front-end Modular Structure

The application runs within a web page which is defined within a basic `index.html` file; this file simply provides a container for the JavaScript application. The application itself comprises of a set of modules - these modules are divided into general modules and domain modules. The general modules are used by the domain modules and the domain modules provide the application state and rendering information.

Modules consist of following file types:

- `saga` - `*-saga.js` - Redux saga files
- `actions` - `*-actions.js` - Redux + Redux sauce files
- `reducer` - `*-reducer.js` - Redux reducer files
- `selectors` - `*-selectors.js` - Reselect files

And they also have:

- `ui` - folder with UI parts of the module
- `style` - folder containing CSS files specific to the module

The general modules are as follows:

- `root` - common functionality
- `api` - common API functionality
- `auth` - common authentication functionality

⁷ React: <https://reactjs.org/>

⁸ JavaScript is the language that runs within a standard Web browser - it is sometimes called ECMAScript in this context. It can also run in server side applications using some specific frameworks such as node.js: <https://nodejs.org>

⁹ Redux: <https://redux.js.org/>

¹⁰ Router5: <https://router5.js.org/>

- entity repository - handles data which comes from LEXIS Portal FE Server
- forms - forms common functionality (form's fields, etc.)
- routing - handling of URLs with router5 library

The domain modules are as follows:

- user - logged in user management functionality
- users - users management functionality
- data sets - data sets management functionality
- organizations - organizations management functionality
- projects - functionality to manage create and administration of projects
- workflows - functionality to manage creation and execution of workflows

3.2 LEXIS PORTAL FE SERVER

The LEXIS Front End Server is a simple service that comprises of a simple Golang based HTTP/HTTPS server which serves the following endpoints:

- `/auth/login`
- `/auth/logout`
- `/auth/callback`
- `/auth/session-info`
- `/datasets`
- `/organizations`
- `/users`
- `/projects`
- `/account_info`
- `/workflows`
- `/`

The first 4 endpoints relate to the authentication process. The `/auth/login` and `/auth/callback` are part of the standard OpenID Connect authentication flow: `/auth/login` redirects to the OpenID Connect service - Keycloak in this instance - and passes a code. The client browser performs an authentication flow and is redirected from Keycloak back to the `/auth/callback` endpoint. This endpoint takes the returned authentication token from the OIDC server and requests a bearer token which has a longer validity and can be used to communicate with other services on behalf of the user. The `/auth/logout` endpoint removes all authentication and user information from the current session; however, the user's Keycloak session may still be active, so logging in again may be possible without needing to reenter a password.

The FE Server supports persistent session management such that the authentication status of all sessions is defined within the application. The `/auth/session-info` enables a so-called bearer token to be provided to the FE so it can communicate directly with the Portal API in an authenticated manner. This bearer token is generated by Keycloak and is recognized by Keycloak as being associated with a particular user: it is passed between the different microservices to perform actions on behalf of the user. The session information - including the bearer token - is relayed over a HTTPS connection and hence is not a significant security risk.

The FE content is packaged such that all assets are self-contained: `npm install`¹¹ is used to gather all dependencies and assets of the FE into a single directory such that they can be served to the browser.

¹¹ npm is the widely used package management tool for many JavaScript applications; `npm install` is a specific command which is supported by npm to resolve dependencies.

This service requires a configuration file containing standard configuration information such as location of log files, port to operate on, debug level, location of SSL certificates for HTTPS security etc. It also requires configuration of Keycloak parameters, including location of Keycloak service, the Realm that should be used, the Client ID and Secret that are required¹² etc. The service is intended to be packaged and run in a Docker¹³ container - more details on configuration and deployment are provided below.

3.3 LEXIS PORTAL API

The LEXIS Portal API is the main entry point for the Portal FE; in principle, it can facilitate API based access to LEXIS functionality from other clients/tools, but that has not been considered in detail at this time. The LEXIS Portal acts as a proxy to the backend services and hence the endpoints offered by the LEXIS Portal can be seen primarily as an aggregation of the endpoints offered by the other services (UserOrg service, Alien4Cloud interface and DDI interface).

The LEXIS Portal API offers a Swagger/ OpenAPI v2¹⁴ based API which supports the following HTTPS endpoints (see Table 1) - all HTTPS requests must provide a valid bearer token or will receive a HTTP 403 Unauthorized response.

ENDPOINT	DESCRIPTION
/dataset	GET lists all known datasets POST supports creation of a dataset
/dataset/<doi>	GET a dataset with a given DoI PUT updates a specific dataset
/dataset/search/year	GET datasets published in a given year
/dataset/search/metadata	GET datasets based on a metadata search
/user	GET lists all users POST can be used to create a user
/user/<id>	GET gets a user with a given ID PUT is used to update the user with the given ID DELETE deletes a user with a given ID
/organization	GET lists all organizations POST creates a new organization
/organization/<id>	GET organization with given ID PUT updates organization with given ID DELETE deletes organization with given ID
/organization/<id>/user	GET users within organization with the given ID
/organization/<id>/projects	GET projects within a specific organization
/project	GET lists projects POST is used to create a project
/project/<id>	GET information relating to a specific project PUT updates a specific project DELETE deletes a specific project

¹² Realms, Client IDs and Secrets are well-defined Keycloak concepts detailed in the Keycloak documentation.

¹³ Docker: <https://www.docker.com/>

¹⁴ OpenAPI v3 was briefly considered, but the tooling relating to OpenAPI v3 is not sufficiently mature, specifically when it comes to Golang implementations; for this reason, Swagger 2 was used within the project. More information on Swagger API definitions can be found at <https://swagger.io/docs/specification/2-0/basic-structure/>

/project/<id>/hpcresources	GET HPC Computational Projects associated with a given LEXIS project
/hpc/resource	GET lists all HPC Computational Projects POST creates a new HPC Computational Project
/hpc/resource/<id>	GET information relating to a specific HPC Computational Project PUT updates information relating to a specific HPC Computational Project DELETE deletes information relating to a HPC Computatioal Project
/account-info/<id>	GET account usage for given LEXIS Computational Project
/workflow/template	GET returns list of available Workflow Templates
/workflow/template/<id>	GET returns detailed inform on a specific workflow template
/workflow	GET returns list of available LEXIS Workflows POST creates a new LEXIS workflow
/workflow/<id>	GET returns detailed info on given workflow DELETE deletes the workflow with the given id
/workflow/execution	GET lists currently executing workflows
/workflow/<workflowid>/execution/<executionid>	DELETE removes the execution of the given workflow
/workflow/<workflowid>/execution/<executionid>/logs	GET returns logs associated with given workflow execution
/workflow/<workflowid>/execution/<executionid>/status	GET returns the status of the given workflow execution
/workflow/<id>/execution/run	POST creates a new execution of a given workflow

Table 1 Endpoints provided by the LEXIS Portal API

Appropriate data models are defined and returned by the endpoints: as the Portal API is primarily a proxy, the data models defined echo those defined in the service that sit behind the Portal API; for this reasons, details on the data models defined for all endpoints are not included here.

A set of valid return codes is defined for all calls - on success HTTP 200¹⁵ is typically returned although 201 is returned on successful creation of an entity. HTTP 500, 400 and 409 are also possible, depending on the input parameters and the state of the system.

The LEXIS Portal API is implemented in Golang¹⁶, uses a variant of the go-swagger¹⁷ code generation tool to provide its own API; it uses the same tool for generating the client libraries which are used to communicate with the respective services.

3.4 LEXIS USERORG SERVICE

The LEXIS UserOrg service was originally a service to manage the mapping between users and organizations but evolved to become a service which also contains mappings to LEXIS Computational Projects and HPC Computational

¹⁵ It is assumed the reader has some familiarity with HTTP response code - a full list of valid HTTP response codes is here: <https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

¹⁶ Also referred to as the Go programming language: <https://golang.org/>

¹⁷ Go-swagger project: <https://github.com/go-swagger/go-swagger>; LEXIS uses a variant developed by Stratoscale as it offers some greater flexibility: <https://github.com/Stratoscale/swagger>

Projects. The UserOrg service offers a Swagger/OpenAPI v2 interface over HTTPS to support creation of users, creation of organizations, creation of projects and the various mappings between these different entities, e.g. mappings from users to organizations and mappings from users to projects. It is supported by a persistent storage: a Postgres¹⁸ DB is used for persisting data.

The endpoints provided by the service are described in Table 2.

ENDPOINT	DESCRIPTION
/status	GET operation returns the health status of the service
/user	GET lists all users allowed by the user privileges POST is used to create a user
/user/<id>	GET returns the user with the given id if allowed by user privileges PUT updates the user with the given id if allowed by user privileges DELETE deletes the user with the given id if allowed by user privileges
/organization	GET llists all organizations allowed by user privileges POST is used to create a user, depending on user privileges
/organization/<id>	GET returns an organization with a given id, depending on user privileges PUT updates an organization with a given id, depending on user privileges DELETE deletes an organization with a given id, depending on user privileges
/organization/<id>/users	GET lists all users in a given organization depending on user privileges
/organization/<id>/projects	GET lists all LEXIS Computational Projects within a given organization
/project	GET lists all LEXIS Computational Projects available to the user POST creates a LEXIS Computational Project
/project/<id>	GET information about a specific LEXIS Computational Project PUT updates a specific LEXIS Computational Project DELETE deletes information relating to a specific LEXIS Computational Project
/project/<id>/hpcresources	GET returns HPC Projects associated with a given LEXIS Computational Project
/hpc/resources	GET returns a list of HPC Projects which are accessible by the user POST creates a new HPC Project
/hpc/resource/<id>	GET returns information about a specific HPC Project PUT updates information about a specific HPC Project DELETE removes all information relating to a specific HPC Project

Table 2 Endpoints provided by the UserOrg service

The above endpoints return well defined data types with models for each of the different entities they operate on. The User data model is shown in Table 3.

FIELD NAME	FIELD TYPE	DESCRIPTION (WHERE NECESSARY)
Id	string/UUID	This is the Keycloak ID of the user
FirstName	string	
LastName	string	
RegistrationDateTime	string/DateTime	Date in UTC

¹⁸ PostgreSQL: <https://www.postgresql.org/>

EmailAddress	string/Email	
OrganizationID	string/UUID	
PGPKeyID	string	Optional
UserStatus	enum	Can be ENABLED, DISABLED (simply mapped to Keycloak status)
AgreedToTermsOfUse	boolean	
TermsOfUseVersion	string	String of format "vx.y.z"
DateOfAgreementToTermsOfUse	string/DateTime	
AgreeToUseOfCookies	boolean	
DateOfAgreementToUseOfCookies	string/DateTime	

Table 3 The User Data Model

The Organization Data Model is shown in Table 4.

FIELD NAME	FIELD TYPE	DESCRIPTION (WHERE NECESSARY)
ID	string/UUID	
FormaltName	string	
RegisteredAddress1	string	
RegisteredAddress2	string	
RegisteredAddress3	string	
RegisteredCountry	string	
CreationDate	string/DateTime	Date in UTC
CreatedBy	string/UUID	This is they Keycloak user id of user which created the record
Website	string	
OrganizationEmailAddress	string/Email	
PrimaryTelephoneNumber	string	In standard ITU E.164 number format
VATRegistrationNumber	string	
OrganizationStatus	enum	PENDING_APPROVAL, APPROVED, DISABLED, TERMINATED

Table 4 The Organization Data Model

The Projects Data Model is shown in Table 5.

FIELD NAME	FIELD TYPE	DESCRIPTION (WHERE NECESSARY)
ProjectName	string	The name of the project
ProjectID	string/UUID	A LEXIS specific UUID for the project
ProjectDescription	string	A short description of the project - may or may not be used
ProjectCreationTime	string/DateTime	Date and Time of project creation (UTC)
ProjectCreatedBy	string/UUID	ID of user who created the project
LinkedOrganization	string/UUID	The ID of the organization to whom the project belongs
ProjectStatus	enum	The project can be one of PENDING, ACTIVE, DISABLED, TERMINATED
ProjectContactPerson	string/UUID	User ID of the person who is responsible for the project
ProjectStartDate	string/DateTime	
ProjectTerminationDate	string/DateTime	
ProjectMaxPrice	number/double	Maximum budget associated with the project
NormCoreHours	integer	Normalized Core Hours associated with Project
ProjectContactEmail	string/email	
ProjectDomain	string	

Table 5 Projects Data Model

The HPCResources Data Model is shown in Table 6.

FIELD NAME	FIELD TYPE	DESCRIPTION (WHERE NECESSARY)
HPCResourceID	string	Identifier of HPC Project - should be recognizable within realm of HPC Centre
AssociatedLEXISProject	string/UUID	UUID of LEXIS Computational Project to which this project is linked
HPCProvider	string	Name of HPC Centre which is providing HPC Project
ResourceType	string	Type of resources provided by HPC Centre for this project - this could be Slurm, PBS or Openstack

Table 6 The HPCResources Data Model

The service is implemented in Golang. It uses a variant of the go-swagger code generation tool to generate stub code for the Swagger servers and go libraries for the Swagger clients; as a Swagger API is defined, libraries in other languages can easily be generated. The service uses the gorm library to map from models defined in the Swagger definition to a specific data schema used in the Postgres database. The service performs authentication and access

controls using Keycloak: every valid request must provide a bearer token and the service verifies this bearer token with Keycloak to ensure that it is valid and the user has the correct authorizations.

3.5 LEXIS DATAMANAGEMENT INTERFACE SERVICE

The LEXIS Data Management’s backend uses iRODS, an open-source data management software which supports data virtualization and discovery, workflow automation and secure collaboration, alongside the Keycloak AAI solution, to support LEXIS data management goals. REST APIs over HTTPS are being developed to decouple the DDI architecture from the rest of the LEXIS system.

One specific issue that arose relating to Portal-DDI interaction related to token processing within iRODS: iRODS communications protocols do not permit tokens of the length generated by Keycloak; this caused issues with the use of OIDC tokens when interoperating with iRODS. To address this, a solution for integrating Keycloak and iRODS was developed. The details are available in Deliverable D3.3 Section 3.2.4 [3].

The AAI to the REST API uses the “Authorization: Bearer” header with a Keycloak token, with standard JSON packets for parameter passing and return values. API endpoints for data upload, query and transfer, in addition to user permission management, are included.

Examples of the API endpoints are provided in Table 7.

ENDPOINT	DESCRIPTION
/dataset	GET provides a list of datasets metadata available to the user, including DataCite fields and access level. POST allows insertion of dataset. Parameters include required metadata, selected upload method and access level.
/dataset/search/metadata	GET allows search of available datasets by multiple metadata fields.

Table 7 API Endpoints provided by the Dataset Management Interface

Examples of simple command-line use of the APIs is presented in Table 8.

<pre>curl -d '{}' \ -H "Content-Type: application/json" \ -H "Authorization: Basic \$1" \ -X GET \ https://<SERVER PATH>/dataset</pre>	<pre>export LC_CTYPE="en_US.UTF-8" base64 < mydata > mydata.asc awk -v ORS='\n' '1' mydata.asc > mydata.esc echo -en '{"push_method": "directupload", "name": "dataset1", "access": "user", "project": "", "group": "", "metadata" : [...], "file" :"' > dat.json cat mydata.esc >> dat.json echo -en '}' >> dat.json</pre>
<pre>curl -d \ '{"publicationYear": 2010, "relatedIdentifier": "doi://lexis- datasets/wp5/datasetpublicx1"}' \ -H "Content-Type: application/json" \ -H "Authorization: Basic \$1" \ -X GET \ https://<SERVER PATH>/dataset/search/metadata/</pre>	<pre>curl --data-binary "@dat.json" \ -H "Content-Type: application/json" \ -H "Authorization: Basic \$1" \ -X POST \ https://<SERVER PATH>/dataset</pre>

Table 8 Examples of simple command-line use of the APIs

3.6 LEXIS ALIEN4CLOUD INTERFACE SERVICE

The LEXIS Alien4Cloud Interface has been designed to provide the LEXIS portal with the functionality it requires from Alien4Cloud/Ystia Orchestrator (Yorc). The LEXIS Alien4Cloud Interface Service provides an API exposing endpoints that will allow the portal to interact with Alien4Cloud. To generate a server in Golang we used a variant of the go-swagger library with custom templates created by Stratoscale. Alien4Cloud provides a REST API, the Alien4Cloud Interface interacts with this API via a Golang client library¹⁹. This client library is still under active development and only started during the lifetime of the project; LEXIS is providing a good opportunity to give a feedback and requirements for this library.

Table 9 lists the currently endpoints provided by this service.

ENDPOINT	DESCRIPTION
/workflow/template	GET lists all LEXIS Workflow Templates for given user.
/workflow/template/{id}	GET returns LEXIS WorkflowTemplate structure for given ID.
/workflow	GET returns list of all LEXIS Workflows for a given user. POST creates a new LEXIS Workflow on the system.
/workflow/{id}	GET returns LEXIS Workflow structure for given ID DELETE removes a LEXIS Workflow from the system.
workflow/execution	GET returns a list of the LEXIS Workflow Executions for a user
workflow/{workflowId}/execution/{id}	DELETE cancels a given LEXIS Workflow Execution.
/workflow/{workflowId}/execution/{workflowExecutionId}/logs	GET returns the logs for a given LEXIS Workflow Execution
/workflow/{workflowId}/execution/{workflowExecutionId}/status	GET returns the basic status for a LEXIS Workflow Execution
/workflow/{workflowId}/execution/run	POST runs a LEXIS Workflow Execution for the given LEXIS Workflow

Table 9 Endpoints Provided by Alien4CloudInterface Service

The interface should present endpoints to the portal that coincide with the LEXIS terminology and concepts. Under the hood the Alien4Cloud Interface service will deal with the translation of these concepts between LEXIS Portal and Alien4Cloud. Table 10, Table 11, and Table 12 show some of the data structures returned related to status of Alien4Cloud Applications.

FIELD NAME	FIELD TYPE	DESCRIPTION
workflowTemplateName	string	Name of LEXIS Workflow Template
workflowTemplateId	string	ID of LEXIS Workflow Template
inputParameters	array	Array of input parameters for LEXIS Workflow Template
inputParamName	string	Name of input parameter

¹⁹ Alien4Cloud Golang client library: <https://github.com/alien4cloud/alien4cloud-go-client>

inputParamType	string	Type of input parameter
inputParamRequired	boolean	Is input parameter required
description	string	Description of input parameter
inputFiles	array	Array of input files for LEXIS Workflow Template
inputFileName	string	Name of input file
inputFileType	string	Type of input file
description	string	Description of input file
nodeTemplates	array	Array of A4C Node Templates
nodeName	string	Name of A4C Node Template
nodeType	string	Type of A4C Node Template
tags	string	Tag of A4C Node Template

Table 10 The LEXIS Workflow Template Model

FIELD NAME	FIELD TYPE	DESCRIPTION
workflowName	string	Name of LEXIS Workflow
workflowId	string	ID of LEXIS Workflow
description	string	Description of LEXIS Workflow
workflowTemplateName	string	Name of LEXIS Workflow Template
workflowTemplateId	string	ID of LEXIS Workflow Template
uploadedInputFiles	array	Array of uploaded input files for LEXIS Workflow
inputFileName	string	Name of input file
inputFileType	string	Type of input file
description	string	Description of input file
nodeTemplates	array	Array of A4C Node Templates
nodeName	string	Name of A4C Node Template
nodeType	string	Type of A4C Node Template
tags	string	Tag of A4C Node Template

Table 11 The LEXIS Workflow Model

FIELD NAME	FIELD TYPE	DESCRIPTION
applicationID	string	A4C Application ID
deploymentID	string	A4C Deployment ID
deploymentPaaSID	string	Deployment PaaS ID
level	string	Level
timeStamp	string	Timestamp of log
workflowID	string	Workflow ID
executionID	string	Execution ID
nodeID	string	Node ID
instanceID	string	Instance ID
interfaceName	string	Interface Name
operationName	string	Operation Name
content	string	Content of Log message

Table 12 The Logs Model

The data models defined above are for the initial variant of the integration with the LEXIS orchestration system; as such, they may be subject to change in future versions. The system has been implemented and tested using a running instance of Alien4Cloud on LRZ. The current implementation requires further investigation of the security aspects as the Alien4Cloud security model has some compatibility issues with the model used throughout the rest of the project: Alien4Cloud does not support OIDC authentication at present. For the purposes of this work, credentials are provided by the service configuration file as well as details about current running instance of Alien4Cloud. The service then uses this information to log into Alien4Cloud each time a request is handled.

3.7 LEXIS AAI CONFIGURATION

LEXIS AAI relies on the Keycloak IAM opensource solution for managing authentication and authorization of LEXIS users. A single Keycloak Realm²⁰ is used for the LEXIS users, but it is necessary to support different services within LEXIS (LEXIS DDI, LEXIS Orchestration, HEAppE and LEXIS Portal) with common information that is configured within the Keycloak Realm and added at Keycloak Client level according to the needs (see Figure 2). Keycloak is also configured with a Realm named “Portal Testing” with basic options for a development purpose that will be reviewed later in order to improve security.

²⁰ A Realm is a clearly defined concept within Keycloak - it comprises of a set of users, credentials, roles and groups with associated authorizations and policies. A single Keycloak deployment can comprise of multiple Realms which are fully isolated from one another (for instance a user belongs to only one Realm and is only managed by this Realm).

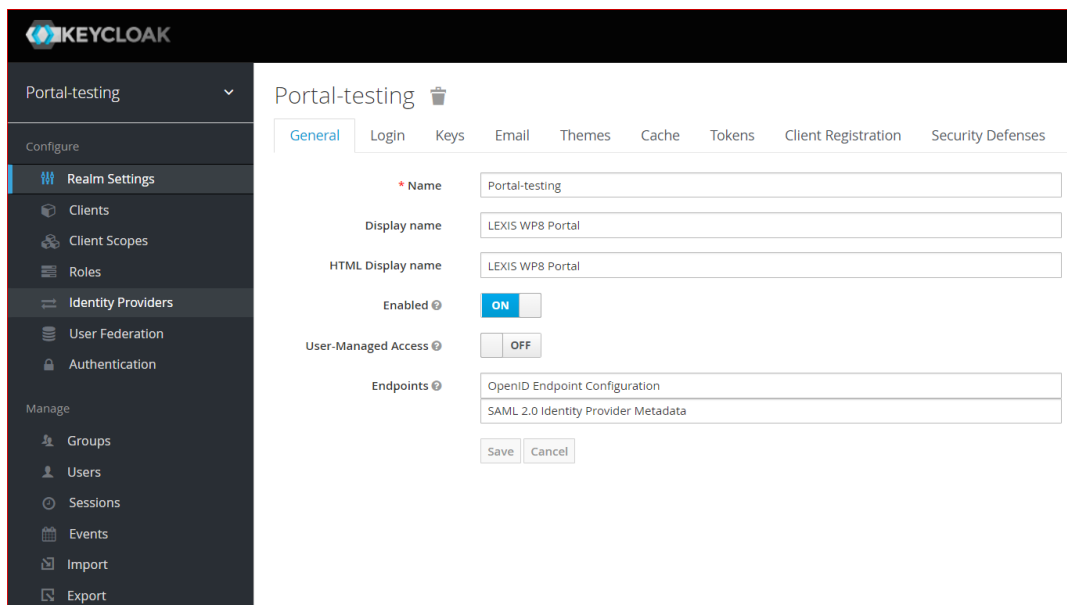


Figure 2 Configuration of a Client within a Keycloak Realm

This LEXIS Realm contains several Keycloak Clients²¹ dedicated to LEXIS services (see Figure 3). For instance, one named “Portal-client” is dedicated to the Portal and one name “broker” is dedicated to the LEXIS DDI and its underlying software technology implementation, iRODS.

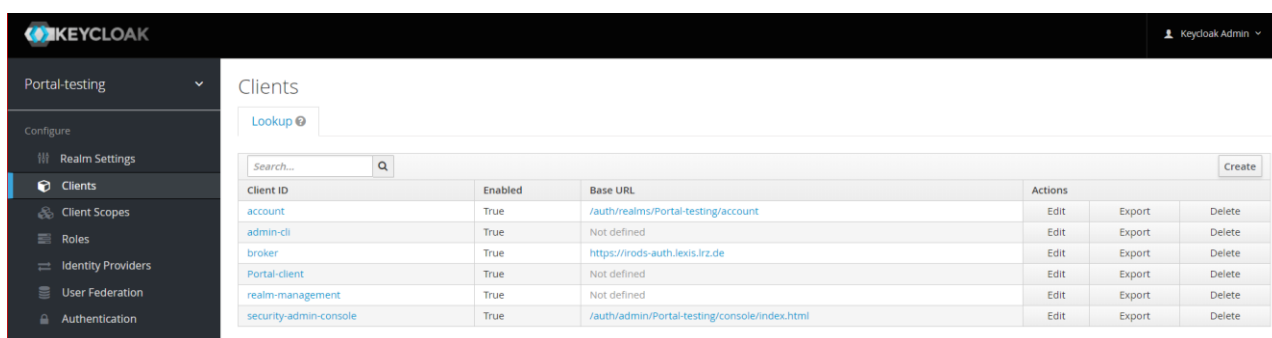


Figure 3 Clients defined with a Keycloak Realm

The detailed authorization configuration of LEXIS AAI in Keycloak is currently a work in progress, as we are finalizing an RBAC Matrix which maps roles to authorizations and will be implemented in Keycloak. The main idea is to use Groups²² and Users (key, value) attributes for specifying authorization that will be configured at the Keycloak Realm level; then at Keycloak Client level, the configuration will consist of adding the Groups or Users attributes that are relevant for the LEXIS Service. As an example the Realm configuration will contain an “Org_A” Group with 2 sub-groups “Org_A_list” and “Org_A_write” with the following attributes “org_list=Org_A” and “org_write=Org_A” (key, value) respectively. Then at Keycloak Client level, the configuration for the LEXIS Portal service will contain mapping for those 2 Groups whereas the configuration for LEXIS DDI service (and iRODS) will not contain mapping for those 2 Groups.

The LEXIS project is evaluating the use of Group attributes for all LEXIS permissions/authorizations to facilitate both administration and API usage. One potential drawback of this approach is the number of Groups in the Keycloak

²¹ The notion of a Client within Keycloak is an arbitrary remote entity which presents with a Client ID and Secret; a Client can then query Keycloak for information relating to users, groups, roles, and attributes. The authorizations of the Client are fully configurable - some Clients may have read only access to a subset of Keycloak data, while others could have full read/write access to the entire Realm.

²² Keycloak supports quite arbitrary grouping of users into Groups

Realm could increase rapidly and there is a need for automation to create all necessary Groups when creating an Organization. For instance, configuring a Group for an Organization A with limited permissions such as listing - the Organization itself will be configured with a Group for Org_A and attribute specifying the authorization “org_list” and that applies to “Org_A” with "list" permission as presented in Figure 4:



Figure 4 Configuring a group with limited permissions

This design makes it easy to grant or deny permissions to/from a user, by simply adding or removing the user from the Group. It also makes it search for sets of users that can access a certain resource faster and the JWT²³ (JSON Web Token) token size should be limited²⁴ as it will only contain the valid/active attributes for the user (with key value set) and empty attributes will not be included. Moreover according to Keycloak documentation we should be able to “concatenate” attributes such a way that for instance there will be no need to duplicate “list_org” key but just extend the resource it applies to as follow “org_A##org_B”. The counterpart of mainly using Groups will be that it requires some automation to create all Groups necessary when creating LEXIS Organisation, LEXIS Computational Project, etc.

3.8 LEXIS CYCLOPS INTEGRATION

From the Portal perspective, integration with Cyclops is via the Portal API: the Portal API communicates with Cyclops depending on user privileges and obtains information on resource consumption and accounting for given projects. Cyclops provides an API for usage records and charge records (based on a predefined charging policy). The Portal API service sends queries to Cyclops requesting usage and charge records for given projects on request from the FE and returns the resulting usage and charge details to the FE for display.

As noted elsewhere, a LEXIS Computational Project comprises of one or more projects associated with HPC centres (HPC Computational Projects): resource and charging information for HPC Computational Projects are collected by Cyclops within the specific HPC centres and this can be queried by the Portal API. Prior to querying Cyclops, the Portal API must obtain a mapping from LEXIS Computational Projects to HPC Computational Projects: this is provided by the UserOrg service; the resulting set of HPC Computational Projects are sent to Cyclops which returns a set of itemized usage and charge records.

²³ JWT - JSON Web Token as defined in RFC7519 by IETF and later updates, is a JSON based token that contains a set of claims and allows to pass identity and authorization of authenticated user, process from an Identity Provider to a Service Provider.

²⁴ There is a known issue that JWT tokens can become large if not managed carefully.

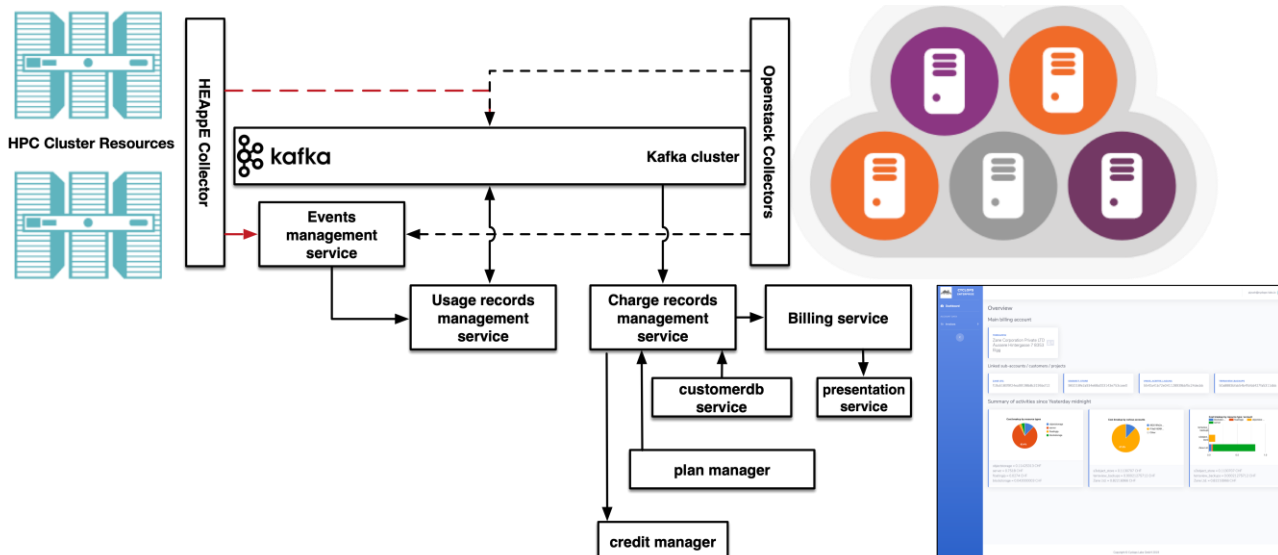


Figure 5 High Level View of Cyclops Framework Components

Figure 5 shows the main components that make up Cyclops accounting and billing framework. Summary of modules are described in Table 13.

FRAMEWORK SERVICE	DESCRIPTION
HEAppE collector	Process that periodically collects and sends resource consumption metrics from HPC Jobs into Cyclops framework
OpenStack compute collector	Background process that uses OpenStack’s REST APIs to assess state of virtual machines and report those to Cyclops
OpenStack network collector	Background process that uses OpenStack’s REST APIs to assess state of allocated floating IPs and report those to Cyclops
OpenStack block storage collector	Background process that uses OpenStack’s REST APIs to assess state of allocated block storage and report those to Cyclops
OpenStack object storage collector	Background process that uses OpenStack’s REST APIs to fetch the size of object storage containers and report those to Cyclops
Kafka cluster	Messaging middleware which facilitates efficient communication among various Cyclops microservices and also allows various collectors to send in collected data
Events management service	Tracks lifecycle status reports for various tracked resources and computes time-based usage reports
Plan manager	Allows creation and management of pricing plans which govern the invoicing process
Customerdb service	Allows tracking of billable entities, billed organizations, and plans associated with such entities
Credit manager	Allows tracking and management of allocated prepaid credits into an organization’s account
Usage records management service	This service aggregates all usage reports for various services used as a part of a single account

Charge records management service	This service transforms all aggregated usage reports into charge records using the elements of the linked plan data and the SKUs
Billing service	This service aggregates all the charge records from different linked sub accounts belonging to an organization based on the linkage-map reported from customerdb service, applies relevant discounts etc.
Presentation service	The main purpose of this service is to transform the bill reports (invoices) into a final form anticipated by Cyclops users - be it JSON to XML, or JSON to CSV, happens in this service

Table 13 The Cyclops framework elements

The Portal exposes data from Cyclops framework via the REST APIs. In the latest release of the portal (at the time of writing of this document), the following information is exposed in the UI:

- resources consumed data,
- cost information on resources used in a given time frame for a LEXIS organization.

The actual payable invoices are not going to be integrated in the interim release of the portal but will be enabled in a subsequent release.

Examples of API calls used by the Portal to obtain accounting and billing information from Cyclops services are shown in Appendix A.

4 LEXIS PORTAL SCREENSHOTS AND MOCKUPS

4.1 SCREENSHOTS FROM LEXIS PORTAL

Screenshots from the Portal Web Interface are shown in Figure 6 - Figure 18.

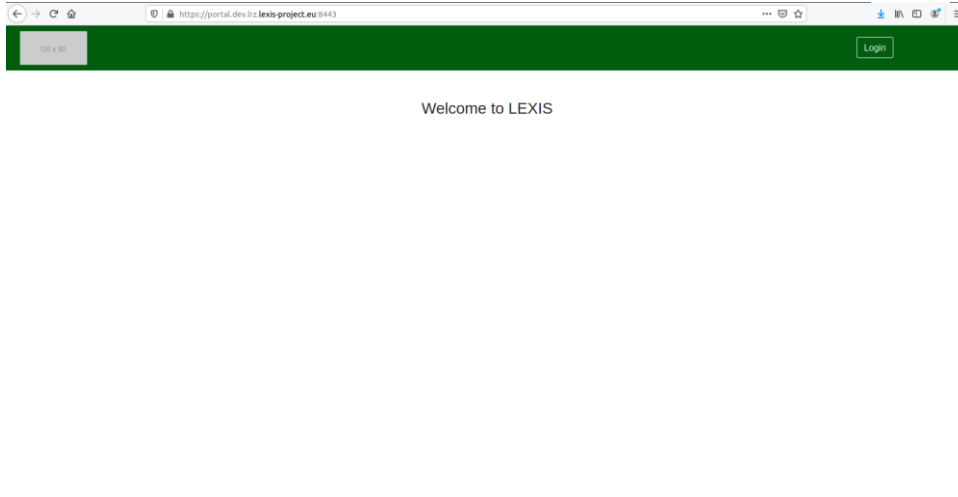


Figure 6 LEXIS Portal Landing Page

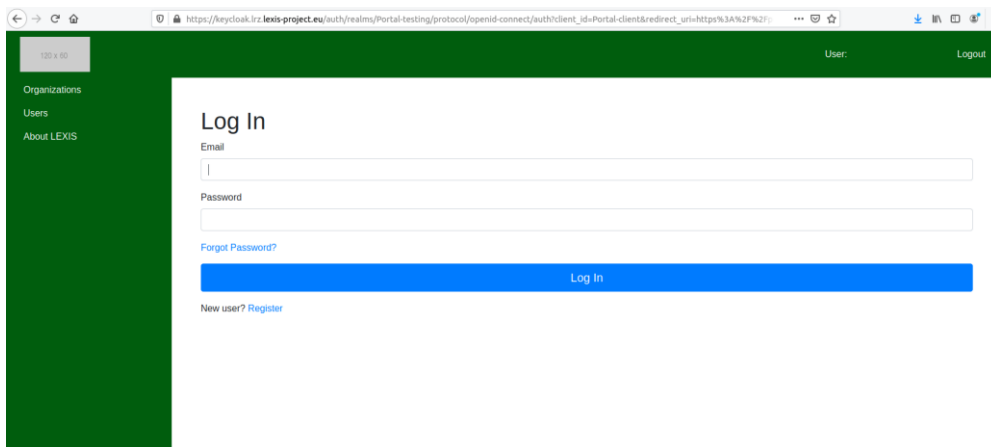


Figure 7 LEXIS Portal Login Page

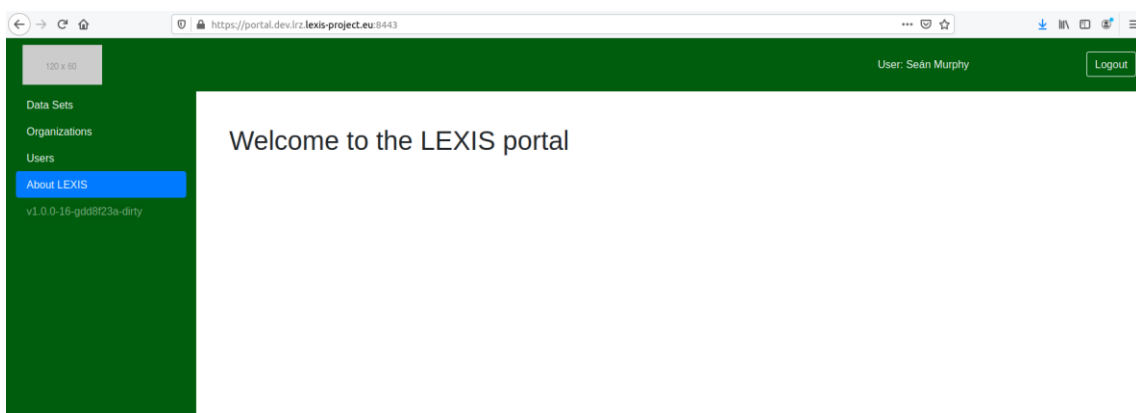


Figure 8 LEXIS Portal Post Login page

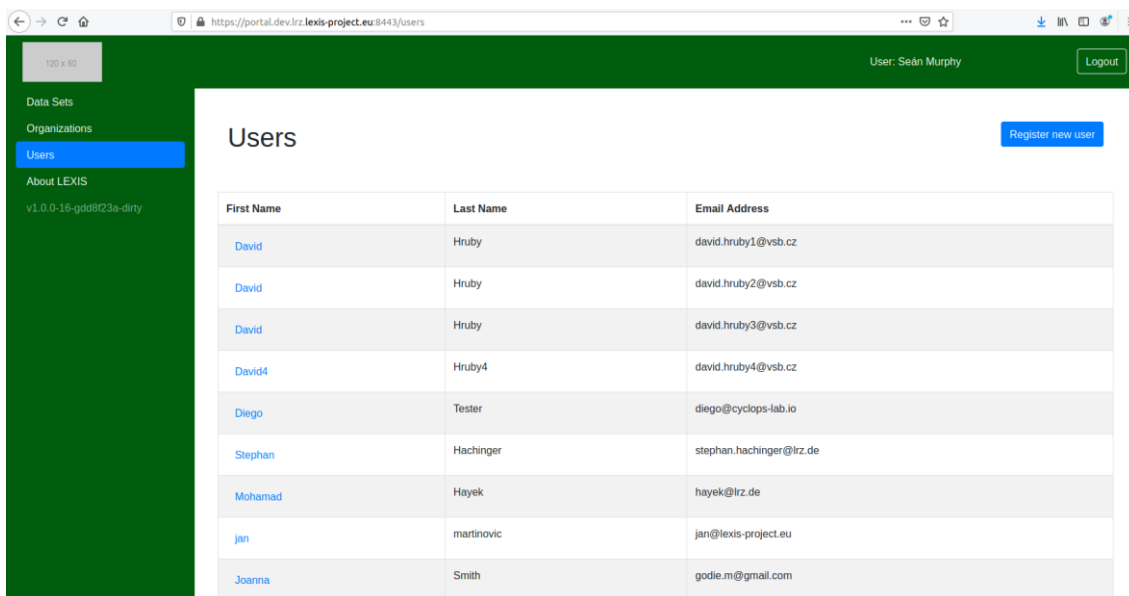


Figure 9 Listing Users defined within the UserOrg service

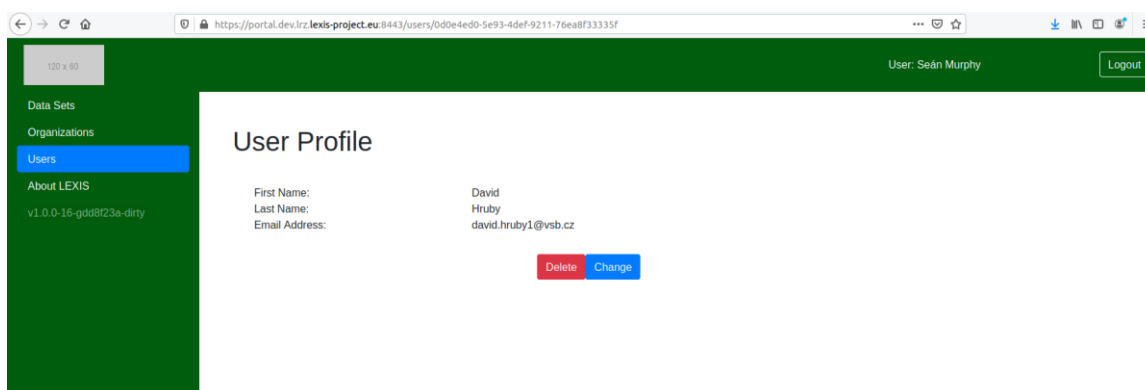


Figure 10 Showing user profile within the LEXIS Portal

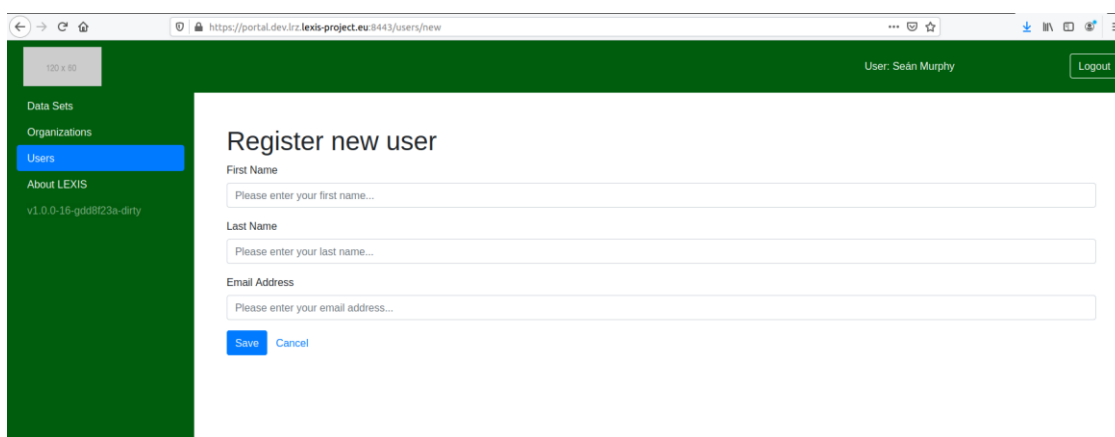


Figure 11 Registering a new user within the LEXIS Portal

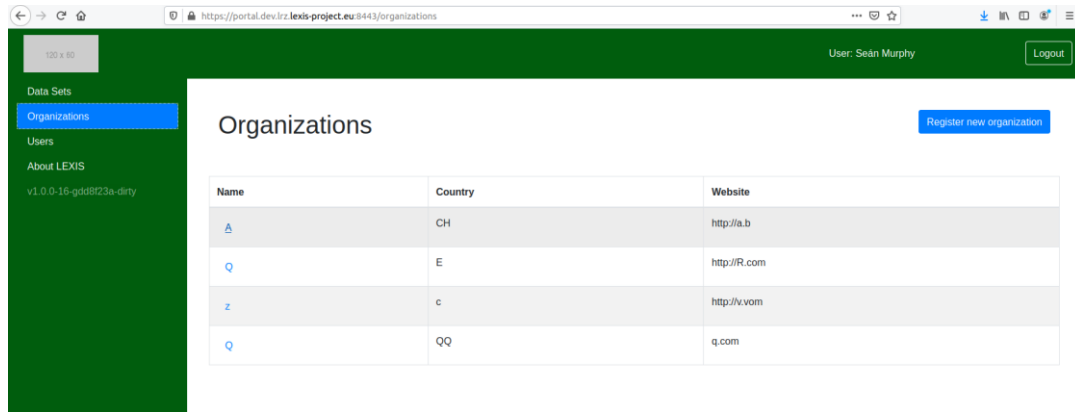


Figure 12 Listing Organizations defined within the UserOrg service

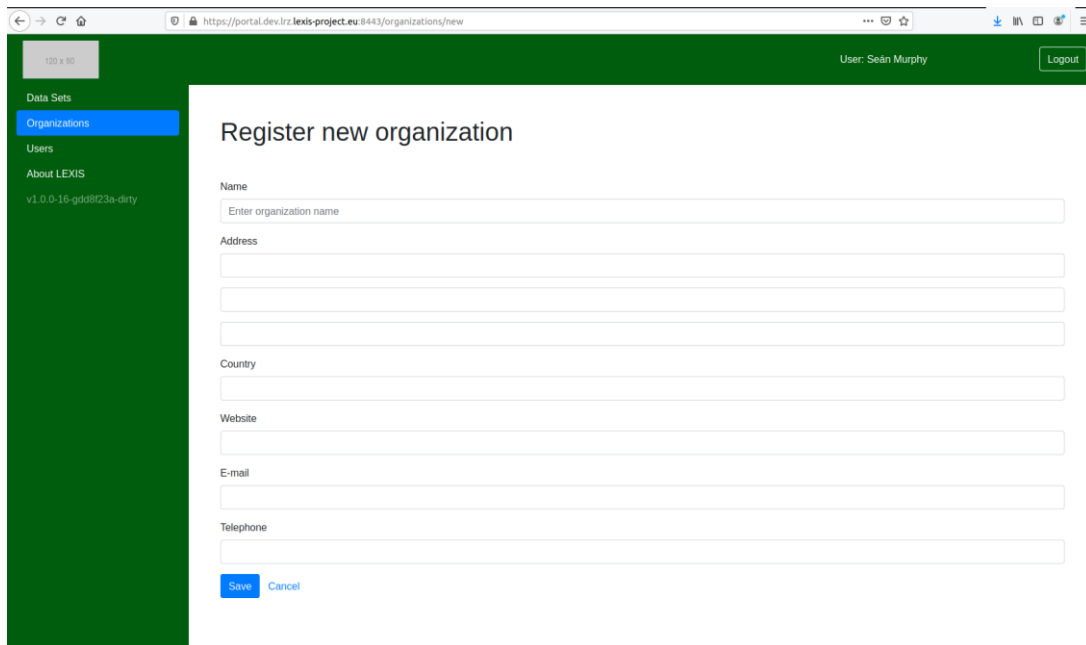


Figure 13 Registering a new organization within the LEXIS Portal

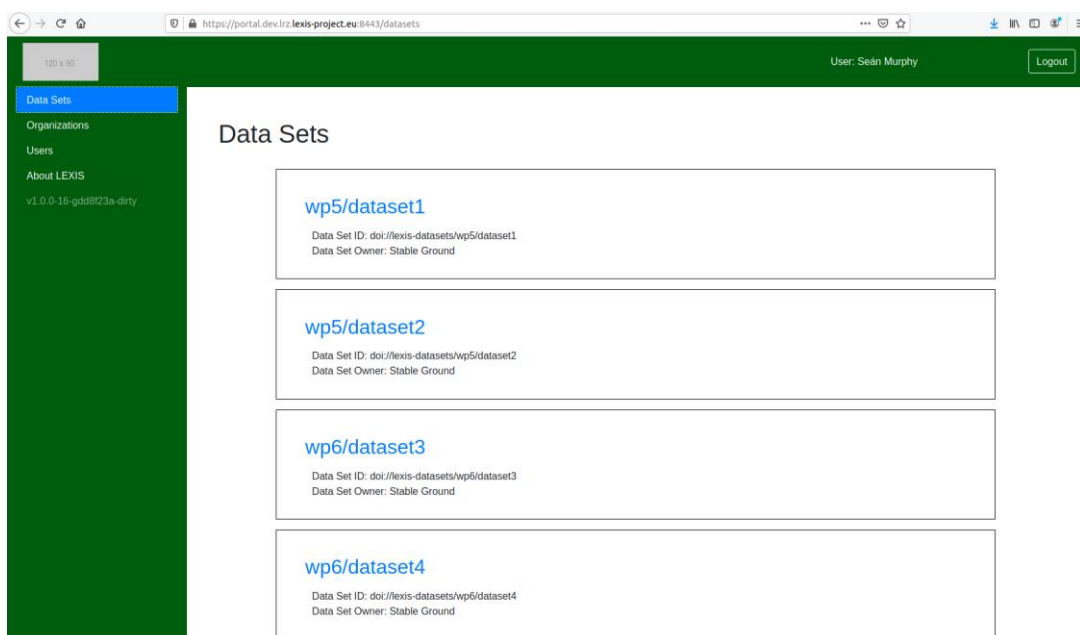


Figure 14 Listing public data sets within the LEXIS Portal

Create: LEXIS project

Name: LEXIS Earthquake and tsunami pilot project ✓

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. ✓

LEXIS project manager (email): abc@cde.com ✓

Domain: Weather forecast

Start: 27. Mar. 2020

Termination: 31. Dec. 2020

I declare that the information provided by me is correct, that I have read the contents of the Contract on the use of high performance cluster and agree to its terms.

[Create](#) [Cancel](#)

Figure 15 Creating a LEXIS Computational Project

Detail: LEXIS Earthquake and tsunami pilot project

ID of the Project:	LEXIS_ID_5
Status:	PENDING
LEXIS project manager (email):	abc@cde.com
Norm. core hours:	0
Max price (euro):	0
Created by:	david.hruby1@web.cz
Created at:	Friday, January 31, 2020 at 5:07:39 AM
Start:	27 March 2020
Termination:	31 December 2020
Domain:	Weather forecast

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

[Edit project info](#)

Figure 16 Viewing details of a LEXIS Computational Project

Edit: LEXIS Earthquake and tsunami pilot project

Name: LEXIS Earthquake and tsunami pilot project

LEXIS project manager (email): abc@cde.com

Start: 27. Mar. 2020

Termination: 31. Dec. 2020

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Domain: Weather forecast

I declare that the information provided by me is correct, that I have read the contents of the Contract on the use of high performance cluster and agree to its terms.

[Save](#) [Cancel](#)

Figure 17 Editing details relating to a LEXIS Computational Project

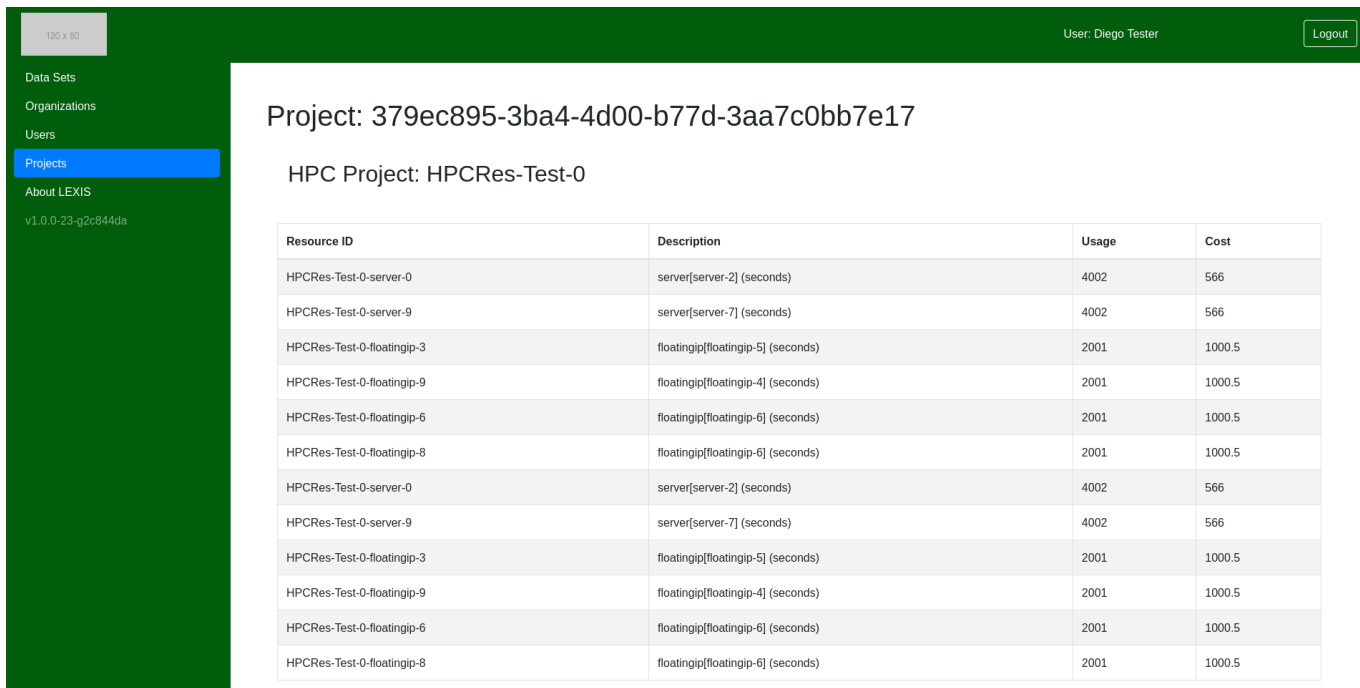


Figure 18 Viewing usage details relating to a LEXIS Computational Project

4.2 MOCKUPS OF FUTURE LEXIS PORTAL IMPLEMENTATION

LEXIS WP8 performed a design of functionalities of the LEXIS Portal which still need to be implemented. This was quite an extensive exercise which took some weeks and produced mockups for quite a comprehensive set of capabilities. It does not make sense to include all these capabilities here; rather a selection of these mockups are provided to convey key aspects of the design in Figure 19 – Figure 21.

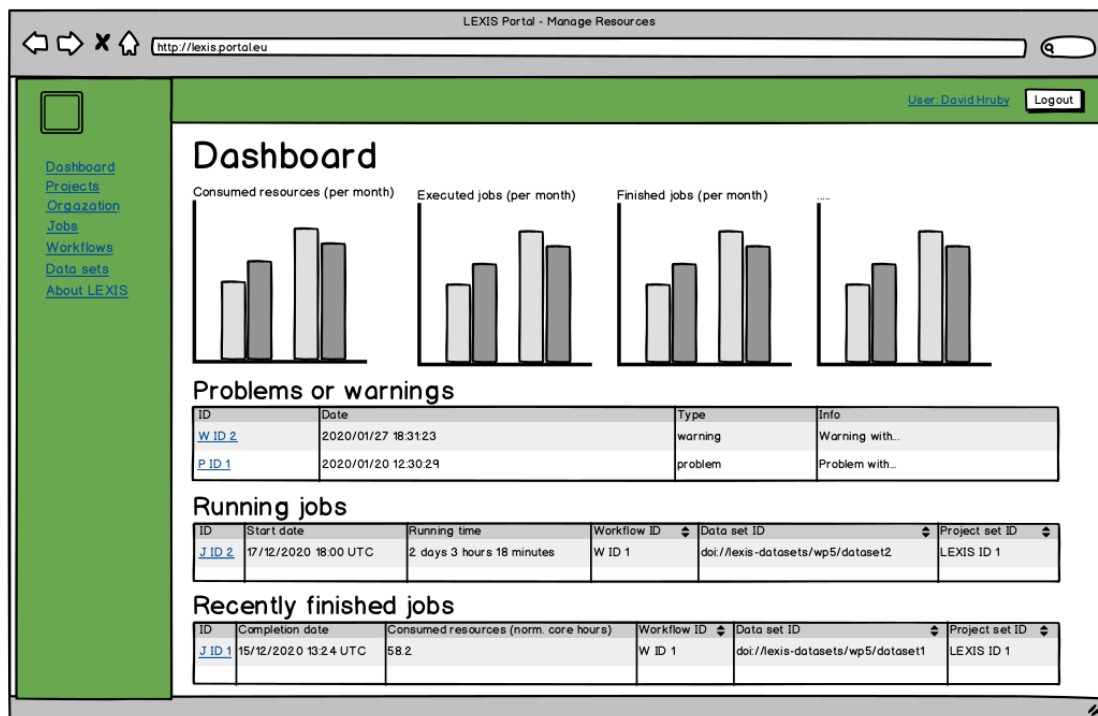


Figure 19 Mock-up of landing dashboard in LEXIS Portal

http://lexisportal.eu
LEXIS Portal - Manage Resources
User: David Iribay Logout

Dashboard

Projects

Organization

Jobs

Workflows

Data sets

About LEXIS

Detail: LEXIS Earthquake and tsunami pilot project

ID of the Project: LEXIS ID 1

Status: active

LEXIS project manager (email): abc@cedec.com

Norm. core hours: 5.000.000

Max price (euro): 240.000

Created by: [redacted]

Created at: 4:07:39 PM Tuesday 31st December 2019

Start: 17th January 2020


Termination: 31st December 2021

Domain: Weather forecast

Description:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Available/spent core hours:



Workflows: Add existing workflow Create new workflow

ID	Name	Status
W ID 1	Open building map update	active
W ID 3	Open building map update	obsolete
W ID 4	Tsunami detection	active
W ID 5	Early tsunami	active
W ID 6	Precise tsunami	active
W ID 7	SEM	active

Resource orders: Add resources (static allocation) Add resources (dynamic allocation) Add resources (approved)

ID	IPC center	Date	From	Until	Norm. core hours	Budget (euro)	Status
XYZ-1	IT41	12.2.2020	1.6.2020	31.12.2020	1.000.000	50.000	approved
XYZ-2	IT41	12.3.2020	1.6.2020	31.12.2020	1.000.000	50.000	approved
XYZ-3	IT41	12.4.2020	1.7.2020	31.12.2020	1.000.000	50.000	approved
XYZ-4	IT41	12.4.2020	1.7.2020	31.12.2020	1.000.000	50.000	cancelled
XYZ-5	IT41	12.5.2020	1.8.2020	31.12.2020	1.000.000	40.000	approved
XYZ-6	IT41	12.6.2020	1.9.2020	31.12.2020	1.000.000	50.000	approved

Data sets: Add existing data set

Title	Resource type	Publication year	DOI
wp5/dataset1	190	2020	doi://lexis-datasets/wp5/dataset1
wp5/dataset2	22156	2020	doi://lexis-datasets/wp7/dataset2

Jobs: Run new job

ID	Workflow ID	Data set ID	Project set ID	Status
J ID 1	W ID 1	0193948585	LEXIS ID 1	finished
J ID 2	W ID 1	0193948585	LEXIS ID 1	running

Edit project info
Add user to project

Figure 20 Mock-up of detailed project information within LEXIS Portal

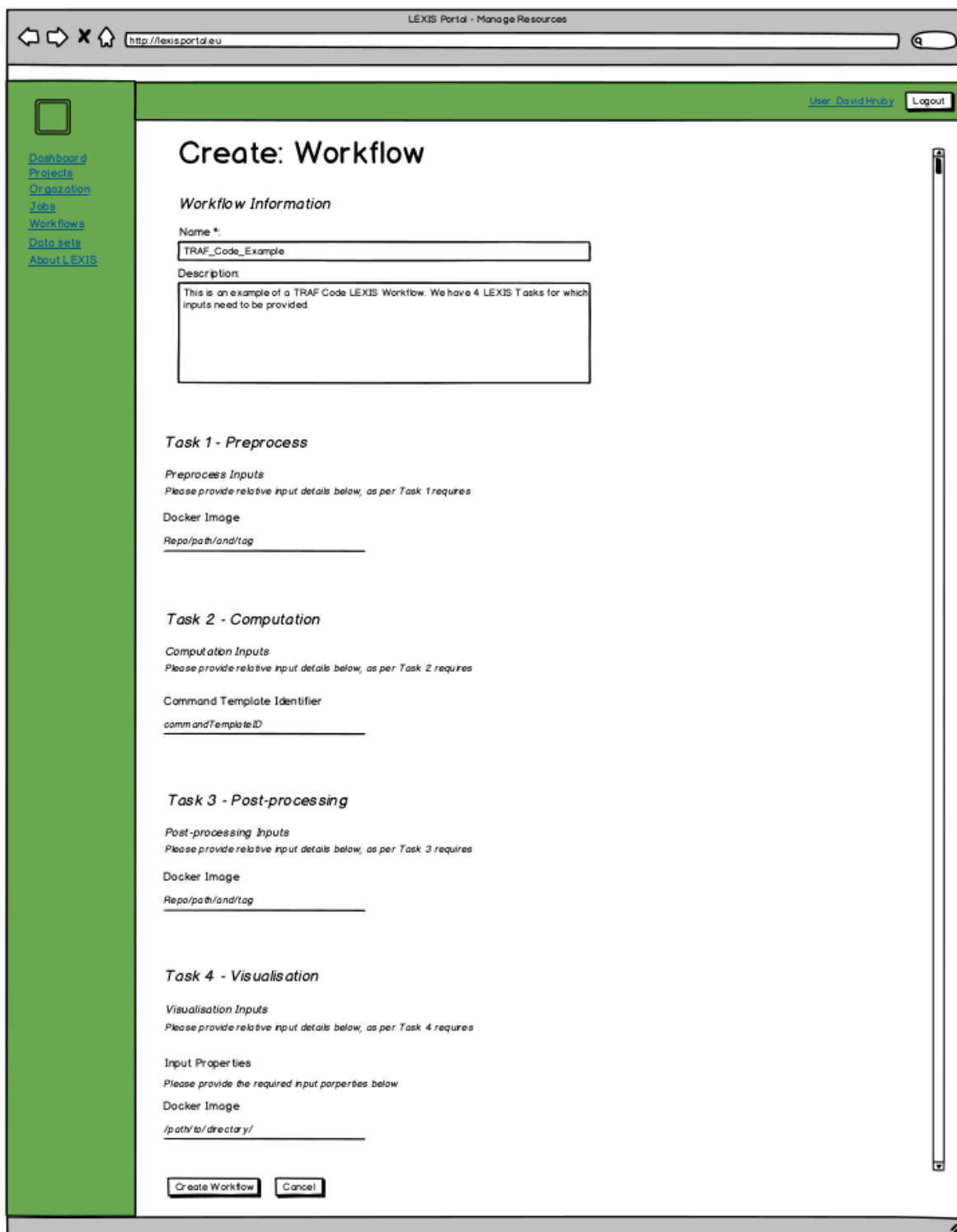


Figure 21 Mock-up of work-flow creation interface within the LEXIS Portal

5 LEXIS PORTAL DEPLOYMENT

The default operating environment for the LEXIS Portal is in Docker containers. All of the individual components contain instructions on how to build Docker containers which can be run on a wide range of different platforms. To date, the containerization process has not been integrated into the software release process, but it is expected that this will happen in the future releases.

To deploy the LEXIS Portal, it is necessary to deploy and configure each of these components:

- go-lexis-portal²⁵,
- user-org-service²⁶,
- portal-api²⁷,
- datamanagement-service²⁸,
- alien4cloud-integration²⁹.

Source code for all of these services is stored on the IT4I git repository at `code.it4i.cz` at the links provided in the footnote and can be obtained using valid credentials.

The LEXIS Portal has dependencies on other services - specifically Keycloak, iRODS and Alien4Cloud and Cyclops - it is necessary to have running deployments of these systems before it is possible to bring up the system. Information on how these are deployed and configured in the LEXIS context is provided in Deliverable D3.3 [3].

Each of the services comprising the LEXIS Portal contains a configuration file with basic information necessary to run the service. Some details on all of these configuration files are provided in the subsections below. All configuration files use the toml³⁰ configuration file format and all configuration files contain examples and default values are specified when appropriate. Configuration files are typically mounted from a host filesystem into a running container as is standard practice when working with Docker containers.

5.1 DEPLOYING LEXIS PORTAL COMPONENTS

5.1.1 System prerequisites

The system has been tested on the following environment:

- Virtual machine with 40 GB storage, 8GB RAM, 4 vCPUs
- Ubuntu 18.04
- Docker Engine (v18.09)
- docker-compose v1.24

If everything is installed inside a single VM, two ports need to be opened to the external world: one for the Frontend server and one for the Portal API.

It is recommended that no other applications be running in this VM; it would be expected that standard systemd managed services should be running (journald, resolved, ssh, cron, network.service etc).

²⁵ Portal Front End Server: <https://code.it4i.cz/lexis/wp8/go-revel-test>

²⁶ UserOrgService: <https://code.it4i.cz/lexis/wp8/user-org-service>

²⁷ Portal API: <https://code.it4i.cz/lexis/wp8/portal-api>

²⁸ Data Management Interface: <https://code.it4i.cz/lexis/wp8/dataset-management-interface>

²⁹ Alien4Cloud Interface: <https://code.it4i.cz/lexis/wp8/alien4cloud-interface>

³⁰ TOML: <https://github.com/toml-lang/toml>

5.1.2 Bringing up the system

There are in principle dependencies between different entities in the system; however, the current implementation does not have a tested boot ordering for all components - further in many containerized environments the standard solution is to repeatedly bring up services until the dependencies are stable. For now, the basic process is as follows:

- Bring up the datamanagement-service and the Alien4Cloud-integration services first as these have dependencies on the DDI and Alien4Cloud orchestrator,
- Bring up the user-org-service - this has a dependency on a database, but this has no dependency on other LEXIS services,
- Bring up the Portal API which has dependencies on the other services above,
- Bring up the service which serves the FE and supports the OIDC authentication flow.

5.1.3 Bringing up the datasetmanagement-service

To build this service, it is necessary to use the Dockerfile provided. Using this Dockerfile, the repository for this service is cloned and a set of dependent libraries is installed; in particular, this service has a dependency on iRODS libraries which in turn have dependencies on specific Operating System versions³¹. Note that building this component can take some time - depending on the platform used it could take some minutes to download all of the correct dependencies and build the software.

Once the container is built, a configuration file can be provided as follows:

```
# toml file for configuration of the dataset-management-interface
[General]
logfile = "logfile.log"
logtoconsole = true
loglevel = "Info"
serverport = 80

[Irods]
host = "1.2.3.4"
port = 1247
zone = "tempZone"
username = "rods"
password = "pass1234"
```

The detailed specification of toml files is not provided here, but to aid understanding, the following points are noted:

- Toml files support different data types - string types are terminated by quotes, number types are not quoted,
- Toml files can be divided into sections - sections are started using a line containing the name of the section in square brackets as with General and iRODS in the above,
- Toml files support hierarchies - subsections can be defined by including the name of the section followed by a period and then the name of the subsection. For example, [General.logging] could contain a subsection containing information specific to logging,
- Toml files are case sensitive.

³¹ iRODS has known issues with newer versions of SSL and hence is currently constrained to work with older versions of Ubuntu (v16.04, the xenial release).

In the Table 14, a brief description of the configuration parameters for this service is provided.

CONFIGURATION PARAMETER	DESCRIPTION
General.logfile	Name of the file to which logging output will be written
General.logtoconsole	Parameter which controls whether log information will be written to stdout or just to the logfile
General.loglevel	Loglevel to use for log output - can be Fatal, Warning, Info, Debug or Trace
General.serverport	The port that this service should run on. If the service is run inside a container, this port should be somehow made accessible outside the container
Irods.host	The host that provides the iRODS service
Irods.port	The port on which the iRODS service is running on
Irods.zone	The name of the zone as defined in iRODS
Irods.username	The username used to connect to the iRODS service
Irods.password	The password used to connect to the iRODS service

Table 14 Configuration Parameters for Datasetmanagement Interface

5.1.4 Bringing up the Alien4Cloud-integration service

As with the other services, the Alien4Cloud integration service can be deployed and built into a Docker. At present, the container packaging is not completely standalone in the sense that it does not clone the latest version from the git repository for a production build; rather, it is necessary to build the component using the Golang toolchain on the local machine and the resulting binary will be inserted into a lightweight container.

To run the service, the following configuration parameters (see Table 15) need to be specified in a configuration file mounted into the container. Toml file format is used.

CONFIGURATION PARAMETER	DESCRIPTION
General.ServerPort	Port that this service should run on
General.Host	Address that this service should bind to; this can be '0.0.0.0' to bind to all addresses on a host
General.ReadTimeout	Timeout value for communicating with dependent services
General.WriteTimeout	Timeout value for communicating with dependent services
General.LogFile	Name of file to which log messages are sent
General.LogToConsole	Boolean indicating whether logging to stdout should be enabled
General.LogLevel	Loglevel to use for log output - can be Fatal, Warning, Info, Debug or Trace
General.HttpsEnabled	Boolean indicating whether the service should use HTTPS or not
General.CertificateFile	Certificate file in case HTTPS is enabled

<code>General.CertificateKey</code>	Certificate key in case HTTPS is enabled
<code>General.InsecureSkipVerify</code>	Standard parameter used in Go to disable strict HTTPS security checking; useful for test and development environments
<code>a4c.A4cUrl</code>	The host that provides the iRODS service
<code>a4c.A4cUser</code>	The port on which the iRODS service is running on that port
<code>a4c.A4cPassword</code>	The name of the zone as defined in iRODS

Table 15 Configuration Parameters for Alien4Cloud Interface

5.1.5 Bringing up the user-org-service

The user-org-service has a build process which pulls down the latest version of the software and builds inside a container. As such, the process for building this component is to make a copy of this repository, enter the *build* folder and run the *docker build* command. This will generate a container image. This needs to be run with a provided configuration file which contains the parameters in Table 16:

CONFIGURATION PARAMETER	DESCRIPTION
<code>General.ServerPort</code>	Port that this service should run on
<code>General.Host</code>	Address that this service should bind to; this can be '0.0.0.0' to bind to all addresses on a host
<code>General.ReadTimeout</code>	Timeout value for communicating with dependent services
<code>General.WriteTimeout</code>	Timeout value for communicating with dependent services
<code>General.LogFile</code>	Name of file to which log messages are sent
<code>General.LogToConsole</code>	Boolean indicating whether logging to stdout should be enabled
<code>General.LogLevel</code>	LogLevel to use for log output - can be Fatal, Warning, Info, Debug or Trace
<code>General.HttpsEnabled</code>	Boolean indicating whether the service should use HTTPS or not
<code>General.CertificateFile</code>	Certificate file in case HTTPS is enabled
<code>General.CertificateKey</code>	Certificate key in case HTTPS is enabled
<code>General.InsecureSkipVerify</code>	Standard parameter used in Go to disable strict HTTPS security checking; useful for test and development environments
<code>database.UserName</code>	Username to use for storing information relating to this service
<code>database.Password</code>	Password associated with above username
<code>database.DBName</code>	Name of the database which is used for storing the information for this service
<code>database.Sslmode</code>	Boolean indicating whether the database uses SSL or not

database.Host	Name of the host where the database is running
database.Port	Port on which database is running
keycloak.Enabled	Boolean indicating whether Keycloak authentication checks are enabled
keycloak.Host	Host on which the Keycloak service is operating
keycloak.Port	Port on which the Keycloak service is operating
keycloak.Realm	The Keycloak realm that should be used for authentication and authorization checks
keycloak.ClientID	The client ID specified within Keycloak for making privileged queries to Keycloak
keycloak.ClientSecret	The secret associated with the above client ID as defined within Keycloak

Table 16 Configuration Parameters for UserOrgService

Note that bringing up this service requires an operational Postgres database: the schema to be used for the database is documented in the swagger model for the service³². The service will automatically create a database in the case it does not exist or check for schema compliance before running the service in the case that the database does exist.

5.1.6 Bringing up the Portal API

The Portal API build process follows that of the other services: the Dockerfile in the repository obtains a copy of the repository within the build container and builds it. The Portal API has dependencies on multiple services and hence the configuration file has a larger parameter set; also, for the service to operate properly more dependent services need to be active and properly operational.

The configuration parameters for this service are shown in Table 17.

CONFIGURATION PARAMETER	DESCRIPTION
General.serverport	Port that this service should run on
General.logfile	Name of file to which log messages are sent
General.logtoconsole	Boolean indicating whether logging to stdout should be enabled
General.loglevel	Loglevel to use for log output - can be Fatal, Warning, Info, Debug or Trace
Keycloak.host	Host on which the Keycloak service is operating
Keycloak.port	Port on which the Keycloak service is operating
Keycloak.realm	The Keycloak realm that should be used for authentication and authorization checks
Keycloak.clientid	The client ID specified within Keycloak for making privileged queries to Keycloak

³² Database schema: <https://code.it4i.cz/lexis/wp8/user-org-service/-/raw/master/swagger.yaml>

<code>Keycloak.clientsecret</code>	The secret associated with the above client ID as defined within Keycloak
<code>UserOrgService.host</code>	Host on which the UserOrgService is running
<code>UserOrgService.port</code>	Port on which the UserOrgService is running
<code>UserOrgService.baseurl</code>	Base URI below which the REST interface operates
<code>DataCatalogService.host</code>	Host on which the datasetmanagement interface is running
<code>DataCatalogService.port</code>	Port on which the datasetmanagement interface is running
<code>DataCatalogService.baseurl</code>	Base URI below which the REST interface operates
<code>WorkflowService.host</code>	Host on which the Alien4Cloud interface is running
<code>WorkflowService.port</code>	Port on which the Alien4Cloud interface is running
<code>WorkflowService.baseurl</code>	Base URI below which the REST interface operates

Table 17 Configuration Parameters for LEXIS Portal API

5.1.7 Bringing up the FE Server

The final component which needs to be activated is the FE server. Building this component is more complex than most of the other components as it has dependencies on two git repositories - one for the React/Javascript frontend and one for the lightweight Golang FE server. The build process takes the React/Javascript frontend and compiles it to optimized, production ready Javascript which is served to the browser via the FE server. The instructions for building the container are contained in the Dockerfile within the git repository.

As with the other components, this component has a set of configuration parameters shown in Table 18.

CONFIGURATION PARAMETER	DESCRIPTION
<code>General.serverport</code>	Port that this service should run on
<code>General.logilfe</code>	Name of file to which log messages are sent
<code>General.logtoconsole</code>	Boolean indicating whether logging to stdout should be enabled
<code>General.loglevel</code>	Loglevel to use for log output - can be Fatal, Warning, Info, Debug or Trace
<code>General.httpsenabled</code>	Boolean indicating whether the service operates a HTTP or HTTPS interface
<code>General.certificatefile</code>	Certificate file to be used in case the service provides a HTTPS interface
<code>General.certificatekey</code>	Certificate key for associated certificate file in case the service provides a HTTPS interface
<code>Keycloak.host</code>	Host on which the Keycloak service is operating
<code>Keycloak.port</code>	Port on which the Keycloak service is operating

<code>Keycloak.realm</code>	The Keycloak realm that should be used for authentication and authorization checks
<code>Keycloak.clientid</code>	The client ID specified within Keycloak for making privileged queries to Keycloak
<code>Keycloak.clientsecret</code>	The secret associated with the above client ID as defined within Keycloak
<code>Keycloak.redirecturl</code>	The endpoint to which Keycloak should redirect at the end of an authentication flow; this should be an endpoint provided by this service and should be known a priori by Keycloak and associated with the client

Table 18 Configuration Parameters for Portal FE Server

6 SUMMARY AND NEXT STEPS

6.1 SUMMARY OF CONTENT IN THIS DOCUMENT

Release R2 is a significant step within the development of the LEXIS Portal, providing some integration with all major subsystems within LEXIS. This document provided details on the current status of the LEXIS Portal R2. This release of the LEXIS Portal includes functionality to perform user and organization management, view information on available datasets, deploy jobs and view accounting and billing information.

The architecture of the system design was provided, identifying modules within the system with associated capabilities and modules they need to interact with. A more detailed description of each of these modules was then provided which focused mostly on the interfaces and functionalities provided by each of the modules with higher-level references to how the modules were implemented - languages used, key libraries, etc.

Screenshots of the current release of the system were then provided in the next section to give the reader some understanding of the interaction modus. Also, mock-ups of new capabilities are included to highlight how user interaction will work for new capabilities.

Following this, a description of the setting up the system is provided. It is worth noting that there are significant dependencies: the iRODS DDI is required and the Alien4Cloud orchestration system.

6.2 NEXT STEPS IN DESIGN OF LEXIS PORTAL

With the release of the version of the LEXIS Portal which integrates all the major LEXIS technology components, clear understanding of APIs between all entities and User Interface designs, the next steps in implementing the LEXIS Portal will happen quickly. With the release of R2, the project team will turn its focus towards the delivery of R3 which will progress the initial work done in all directions, including:

- Providing support for a more comprehensive AAI model as described in Deliverable D4.2 [9],
- Providing support for more powerful and flexible job deployment and management including viewing current status, viewing log output, detecting error conditions etc.,
- Providing support for data upload and download to and from the DDI for files of limited size - these can be small data sets or for example, parameters used as input to a job,
- Providing support for the process of requesting resources from the HPC centres and monitoring overall resource consumption on a per project basis.

Support for all of these capabilities will be reported in Deliverable D8.2 [5] which will be provided with Release R3 of the LEXIS Portal.

REFERENCES

- [1] LEXIS Deliverable, *D2.2 Key parts LEXIS Technology Deployed on Existing Infrastructure and Key Technologies Specification*.
- [2] LEXIS Deliverable, *D2.3 Report of LEXIS Technology Deployment - Intermediate Co-Design*.
- [3] LEXIS Deliverable, *D3.3 Mid-Term Infrastructure (Deployed System Hard/Software)*.
- [4] LEXIS Deliverable, *D4.5 Definition of Mechanisms for Securing Federated Infrastructures*.
- [5] LEXIS Deliverable, *D8.2 Second Release of LEXIS Portal*.
- [6] LEXIS Deliverable, *D8.3 Final Release of LEXIS Portal*.
- [7] *HPC and AI as a Service, Atos White Paper, 2019*.
- [8] D. Hudak, D. Johnson, A. Chalker and et al., "Open OnDemand: A web-based client portal for HPC centers," *Journal of Open Source Software*, vol. 3, no. 25, p. 622, 2018.
- [9] LEXIS Deliverable, *D4.2 Design and Implementation of the HPC-Federated Orchestration System - Intermediate*.

A EXAMPLES OF QUERYING CYCLOPS SERVICES

The following APIs from UDR and CDR microservices are used for the interim integration within LEXIS portal -

UDR query sample:

```
GET /udr/usage/737097af82e1498ebce1ddd8675c4b29?from=2019-12-09T00:00:00Z&to=2019-12-13T00:00:00Z
```

CDR query sample:

```
GET /cdr/usage/737097af82e1498ebce1ddd8675c4b29?from=2019-12-09T00:00:00Z&to=2019-12-13T00:00:00Z
```

UDR sample output:

```
[
  {
    "AccountId": "737097af82e1498ebce1ddd8675c4b29",
    "TimeFrom": "2019-12-09T00:00:00.000Z",
    "TimeTo": "2019-12-13T00:00:00.000Z",
    "Usage": [
      {
        "Metadata": {
          "flavorid": "79ce7ce6-6f2e-407a-8e3c-9440c5877314",
          "flavorname": "c1.xxlarge",
          "imageid": "69a9df94-3f80-45c6-b0c5-0a41ece646d0",
          "imagename": "Ubuntu Xenial 16.04 (SWITCHengines)-2017-09-07T13:48:34Z",
          "imageosflavor": "ubuntu",
          "region": "S2"
        },
        "ResourceId": "5e54b1d1-ba5f-4ecb-ba56-f996824380a8",
        "ResourceName": "GRID02",
        "ResourceType": "server",
        "UsageBreakup": {
          "active": 3601
        }
      },
      {
        "Metadata": {
          "flavorid": "3",
          "flavorname": "m1.medium",
          "imageid": "5176f9a5-3c12-4c14-8bc0-1284dbb2c08c",
          "imagename": "Ubuntu Bionic 18.04 (SWITCHengines)",
          "imageosflavor": "ubuntu",
          "region": "S2"
        },
        "ResourceId": "693f9506-b2fe-450b-8c70-e44b1aba963d",
        "ResourceName": "GRID09",
        "ResourceType": "server",
        "UsageBreakup": {
          "active": 3601
        }
      },
      {
        ...
      }
    ]
  },
  {
    ...
  }
]
```

CDR sample output:

```
[
  {
    "AccountId": "737097af82e1498ebce1ddd8675c4b29",
    "TimeFrom": "2019-12-09T00:00:00.000Z",
    "TimeTo": "2019-12-13T00:00:00.000Z",
    "Usage": [
      {
        "Metadata": {
          "flavorid": "79ce7ce6-6f2e-407a-8e3c-9440c5877314",
          "flavorname": "c1.xxlarge",
          "imageid": "69a9df94-3f80-45c6-b0c5-0a41ece646d0",
          "imagename": "Ubuntu Xenial 16.04 (SWITCHengines)-2017-09-07T13:48:34Z",
          "imageosflavor": "ubuntu",
          "region": "S2"
        },
        "ResourceId": "5e54b1d1-ba5f-4ecb-ba56-f996824380a8",
        "ResourceName": "GRID02",
        "ResourceType": "server",
        "Cost": {
          "appliedDiscount": 566,
          "costBreakup": [
            {
              "sku": "vcpu",
              "sku-cost": 8,
              "sku-discount": 0,
              "sku-net": 8
            },
            {
              "sku": "ram",
              "sku-cost": 2048,
              "sku-discount": 0,
              "sku-net": 2048
            },
            {
              "sku": "rootdisk",
              "sku-cost": 200,
              "sku-discount": 0,
              "sku-net": 200
            },
            {
              "sku": "license",
              "sku-cost": 8,
              "sku-discount": 0,
              "sku-net": 8
            }
          ],
          "netTotal": 1698,
          "totalFromSku": 2264
        }
      },
      {
        ...
      }
    ]
  }
]
```