H2020-SC6-GOVERNANCE-2018-2019-2020

DT-GOVERNANCE-05-2018-2019-2020

D5.4: ACROSS Platform Prototype and Applications - Intermediate

Project Reference No	959157 — ACROSS — H2020-SC6-GOVERNANCE-2018-2019-2020
Deliverable	D5.4: ACROSS Platform Prototype and Applications - Intermediate
Work package	WP5: ACROSS Applications and Platform Integration
Nature	Other
Dissemination Level	Public
Date	28.04.2023
Status	Final
Editor(s)	Elena Chrysikou (ATC)
Contributor(s)	Vincenzo Savarino (ENG), Marina Klitsi (ATC), Anna Opaska (FHG), Petros Christopoulos (GRNET)
Reviewer(s)	ENG, GRNET
Document description	This deliverable describes the intermediate release of the ACROSS platform.

About

The project is co-funded by the European Commission's Horizon 2020 research and innovation framework programme. Spanning through three years, ACROSS consists of a consortium of 10 partners from 7 countries: Athens Technology Center (coordinator), Tecnalia, Dataport, Engineering, Fraunhofer, GRNET, TimeLex, The Lisbon Council, Waag and VARAM.

DISCLAIMER

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Commission. The Commission does not guarantee the accuracy of the data included in this study. Neither the Commission nor any person acting on the Commission's behalf may be held responsible for the use, which may be made of the information contained therein.

© 2021 – European Union. All rights reserved. Certain parts are licensed under conditions to the EU.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
V0.1	20/03/2023	First skeleton of the deliverable and TOC	ATC
V0.2	21/03/2023	Minor updates of the skeleton	ATC
V0.3	20.04.2023	Final draft	ATC
V0.4	25.04.2023	Review	ENG, GRNET
V0.5	27.04.2023	Comments resolved	ATC
V1.0	28.04.2023	Submission	ATC

Executive Summary

This document describes the intermediate version of the ACROSS platform. This is the second working version of the prototype that offers an extra set of the expected functionalities compared to the first prototype and will act as the testbed for the ACROSS stakeholders to experience with a set of ACROSS provisions and assess the concepts and knowledge conveyed by the project.

The deliverable is describing the platform from two different aspects; from the internal point of view, in which the functional and architectural structure is presented briefly, and from the end-user point of view, in which the external appearance, look-and-feel and the overall high-level functionalities and interaction potentials are explained.

The ACROSS Platform, far from being a simple container for the individual modules, is a coherent application, where several different components reside and collaborate in harmony. This second version encapsulates most of the underlying technologies and gives a clear and easy to use graphical interface. The current architecture allows any additional functionality to be wrapped into a separate component and be added to the platform, provided that it abides by the basic communication standards exposed by the ACROSS platform architecture. The scope of this practice is to enable future extensions of the platform to arising functionalities, which may maximise the potentials for further exploitation and adoption of the platform beyond the project end.

Table of Contents

List of Figures

List of Terms and Abbreviations

Abbreviation	Definition
API	Application Programming Interface
ASR	AI-based automatic speech recognition
BPMN	Business Process Model and Notation
CDO	Citizen Data Ownership
CI/CD	Continuous Integration and Continuous Deployment
eIDAS	electronic IDentification, Authentication and trust Services
GB	Gigabyte
IMS	Identity Management System
K8S	Kubernetes
RAM	Random-access memory
REST	Representational state transfer
SC	Service Catalogue
SSO	Single Sign On
TD	Transparency Dashboard
UIC	User Interface Controller
UJMT	User Journey Modelling Tool
UJSE	User Journey Service Engine
VCPU	Virtual Central Processing Unit
WP	Work package

1 Introduction

1.1 Purpose and Scope

This deliverable aims to provide an overview of the intermediate release of the ACROSS prototype. To this end, the report describes shortly the implementation aspects and current functionality.

1.2 Approach for Work Package and Relation to other Work Packages and Deliverables

The WP5: ACROSS Application and Platform Integration aims at providing the implementation aspects for the delivery of the ACROSS components integration in a unified platform. The design of the ACROSS platform is driven by the user requirements definition and the technical specifications as delivered by the WP6: Use cases deployment, evaluation & impact assessment, WP3: ACROSS Data Governance Framework and WP4: ACROSS Modules Setup.

Following the implementation of the individual modules in WP3: ACROSS Data Governance Framework and WP4: ACROSS Modules Setup, this WP delivers an integrated view of the ACROSS platform to act as the testbed for setting the ACROSS pilots in WP6: Use cases deployment, evaluation & impact assessment and validating the results in real life scenarios. The content presented in this deliverable is subject to refinement and modifications, based on the progress of the technical work packages, as well as during the validation and evaluation phases of the project.

1.3 Methodology and Structure of the Deliverable

This document reports on the activities and effort placed in the integration of the various technologies and tools provided by the WP3: ACROSS Data Governance Framework and WP4: ACROSS Modules Setup towards delivering the second release of a functional ACROSS Integrated prototype. The integration effort is guided by the Agile Software Development methodology[1], aiming to progress the development work in parallel teams and regularly integrating their output, based on a well-defined design.

The scope of this document is to act as appendix to the current version of the ACROSS integrated prototype and, as such, it is structured as follows:

[1] https://en.wikipedia.org/wiki/Agile_software_development

- Section 2 provides an overview of the main principles of the technical design that have been applied to the development of the ACROSS solution;
- Section 3 presents the components integrated for the second version of the ACROSS prototype;
- Section 4 concludes this report and presents the next steps for the third release of the ACROSS integrated prototype.

1.4 Updates for the Intermediate Release

The Intermediate Integrated Prototype is the evolution of the First Integrated Prototype that was delivered at the end of the first year and reported at D5.3. The updates of this version are based on the received user feedback and also on the advancements of the WP3 and WP4 components.

These updates are listed below:

- The way of the interaction between the components has been updated
- Updates have been made to the components themselves to advance their functionality and the interaction among them
- Component APIs have also been updated to facilitate the updated functionalities

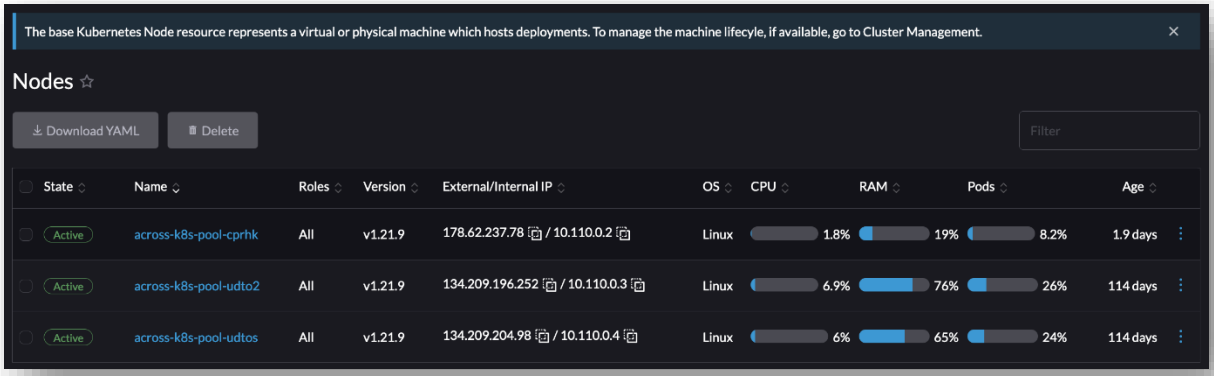
2 Software and System Design principles

This section contains some main principles of technical design which have been applied to the development of the ACROSS solution. Selecting design principles is critical for creating complex software structures and doing it properly at the initial stages of the project leads to better results in the long term in terms of scalability, availability, reliability, and reduced maintenance costs.

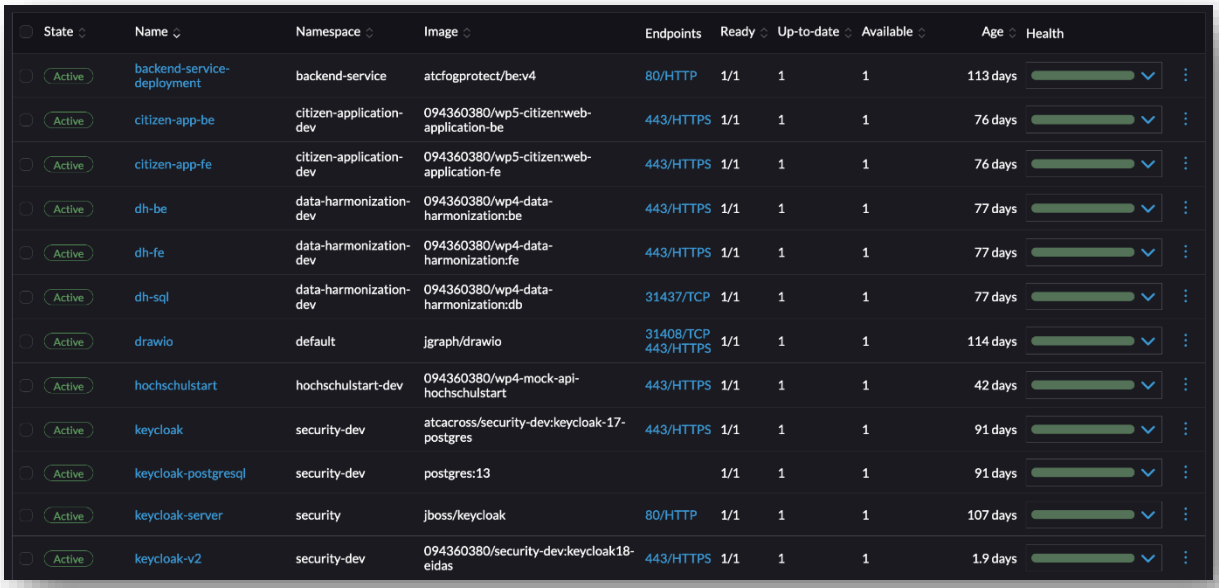
2.1 Operational Environment

In order to host and facilitate all the components and layers of the ACROSS ecosystem the operational environment was set upon a dedicated Kubernetes cluster (Version: v1.21.9). The cluster is hosted on Digital Ocean and consists of 3 Linux server nodes that act as cluster workers. Each cluster node consists of 4 VCPUs and 8 GB of RAM. Additionally, nginx-ingress, wildcard domain and cert manager have been configured to enable external encrypted access. Lastly a network file storage server has been configured to act as persistence volume storage and used by services that data must be persistent.

Every component is deployed, maintained, and scaled on these 3 nodes. Also, the Rancher framework was deployed to leverage the provision of a User Interface to administer the cluster.

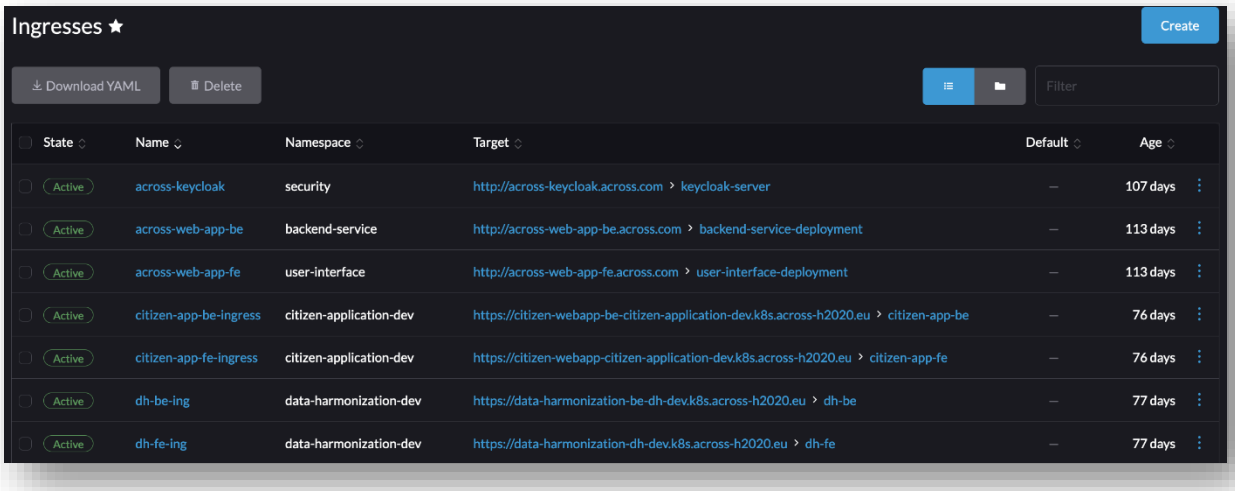


Through the provision of namespaces and projects (as a feature of K8S) the operational testbed is set to host different environments (development-staging-production). Each component is mapped to a dedicated deployment configuration that handles the continuous integration and deployment, scale up and monitoring of the component.



State	Name	Namespace	Image	Endpoints	Ready	Up-to-date	Available	Age	Health
Active	backend-service-deployment	backend-service	atcfogprotect/be:v4	80/HTTP	1/1	1	1	113 days	
Active	citizen-app-be	citizen-application-dev	094360380/wp5-citizen:web-application-be	443/HTTPS	1/1	1	1	76 days	
Active	citizen-app-fe	citizen-application-dev	094360380/wp5-citizen:web-application-fe	443/HTTPS	1/1	1	1	76 days	
Active	dh-be	data-harmonization-dev	094360380/wp4-data-harmonization:be	443/HTTPS	1/1	1	1	77 days	
Active	dh-fe	data-harmonization-dev	094360380/wp4-data-harmonization:fe	443/HTTPS	1/1	1	1	77 days	
Active	dh-sql	data-harmonization-dev	094360380/wp4-data-harmonization:db	31437/TCP	1/1	1	1	77 days	
Active	drawio	default	jgraph/drawio	31408/TCP 443/HTTPS	1/1	1	1	114 days	
Active	hochschulstart	hochschulstart-dev	094360380/wp4-mock-api-hochschulstart	443/HTTPS	1/1	1	1	42 days	
Active	keycloak	security-dev	atcacross/security-dev:keycloak-17-postgres	443/HTTPS	1/1	1	1	91 days	
Active	keycloak-postgresql	security-dev	postgres:13		1/1	1	1	91 days	
Active	keycloak-server	security	jboss/keycloak	80/HTTP	1/1	1	1	107 days	
Active	keycloak-v2	security-dev	094360380/security-dev:keycloak18-eidas	443/HTTPS	1/1	1	1	1.9 days	

On top of that, Rancher also handles provision of external access to all these services deployed by invoking nginx ingress instances. This way all the components of ACROSS are accessible to citizens.



State	Name	Namespace	Target	Default	Age
Active	across-keycloak	security	http://across-keycloak.across.com › keycloak-server	–	107 days
Active	across-web-app-be	backend-service	http://across-web-app-be.across.com › backend-service-deployment	–	113 days
Active	across-web-app-fe	user-interface	http://across-web-app-fe.across.com › user-interface-deployment	–	113 days
Active	citizen-app-be-ingress	citizen-application-dev	https://citizen-webapp-be-citizen-application-dev.k8s.across-h2020.eu › citizen-app-be	–	76 days
Active	citizen-app-fe-ingress	citizen-application-dev	https://citizen-webapp-citizen-application-dev.k8s.across-h2020.eu › citizen-app-fe	–	76 days
Active	dh-be-ing	data-harmonization-dev	https://data-harmonization-be-dh-dev.k8s.across-h2020.eu › dh-be	–	77 days
Active	dh-fe-ing	data-harmonization-dev	https://data-harmonization-dh-dev.k8s.across-h2020.eu › dh-fe	–	77 days

Additionally, a Grafana and Prometheus instance is deployed to effectively monitor the performance of each component in the environment and report incidents. Also, Grafana and Prometheus can be used to monitor services on a more advanced level (ex. Process time, runtime

time etc.) using custom metrics on code level and notify developers using communication channels like slack, email etc. If an incident occurs.

Finally, combining the Gitlab CI/CD features along with the docker registry, an end-to-end Continuous Integration and Continuous Deployment mechanism is set up leading to instant automated deployments when code is pushed for each dedicated component.

2.2 System Architecture Overview

Based on the D5.2 System Architecture & Implementation Plan - Final, the ACROSS system architecture can be depicted in detail below:

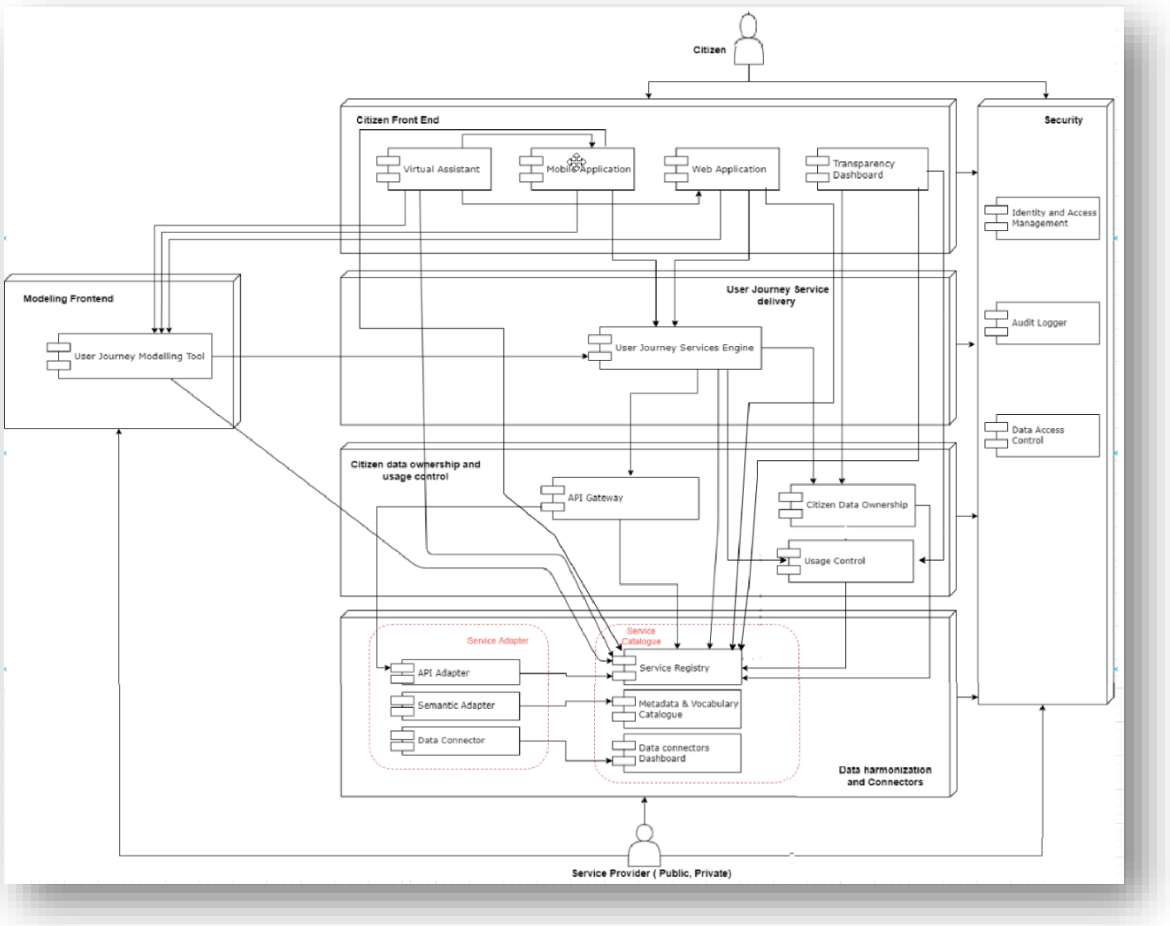


Figure 1: System architecture

From the ACROSS platform perspective, each component of these 6-layer architecture is mapped to a project containing the component deployment configuration. All the component interfaces have been set up either by the definition of internal cluster services or by the definition of external NGINX ingresses.

Regarding the Security of the ACROSS platform (vertical layer in architecture), a single common Identity management system is implemented authenticating users across all applications and leveraging functionalities such as Single Sign On principle.

2.3 Component Interaction

The ACROSS components interaction can be depicted in detail in the following interaction diagram:

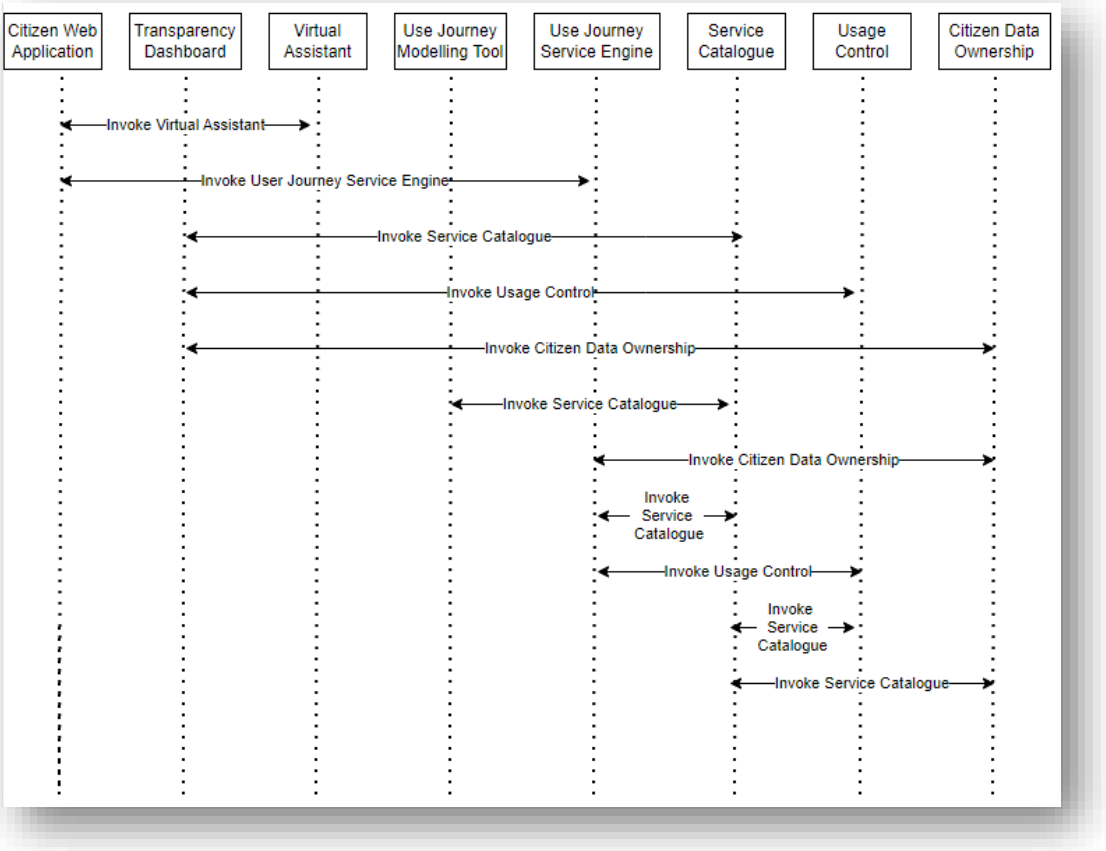


Figure 2: Components Interaction

The following table presents the interactions between the components in detail

Interface	Description
Invoke Virtual Assistant	During its startup, the citizen web/mobile application connects to the Virtual Assistant. Concretely, this is done by invoking the integrated User Interface Connector front-end component, via the UIC's API. As a result, the Virtual Assistant service is made available and, for the entire user session of the web/mobile application, is ready to provide conversational interfaces to the application's features towards the citizens. This means the citizen has the option of interacting with and navigating the application (as well as aspects of the virtual assistant itself) through conversational interaction (i.e. voice or chat utterances), relying on an automated, AI-supported process. The traditional mode of interaction (i.e. GUI interaction through screen, keyboard and pointing device such as mouse or touchscreen) remains fully available, and the user is free to switch forth and back between different interaction modes (GUI, chat, voice) on the fly.
Invoke User Journey Service Engine	The Citizen web/mobile application invokes the User Journey Service Engine (UJSE) to create, edit and delete user journeys. Additionally, the citizen can directly access services, be redirected towards the Transparency dashboard to provide consent policies, and finally execute services from the citizen web/mobile application. Through this interface the services are provided towards the user as a one-stop-shop catalog.
Invoke Service Catalogue (TD->SC)	The Transparency Dashboard (TD) invokes the Service Catalogue (SC) to get the list of available services which make use of personal data, so that the citizen can edit via the TD the consent he/she gives for the use of his/her personal data.

Invoke Citizen Data Ownership (TD->CDO)	The Transparency Dashboard invokes the Citizen Data Ownership (CDO) to get the list of given consents by each citizen and the status of them, to grant or withdraw the citizens' consents and to receive notifications about how their data is being used.
Invoke Service Catalogue (UJMT->SC)	The user journey modelling tool (UJMT) invokes the service catalogue to obtain the list of registered services, as well as their thematic areas, to be used to model the workflow. The invocation can filter/search for specific services and get the needed information in accordance with the service model provided by the Service Catalogue.
Invoke Service Catalogue (UJSE->SC)	The User Journey Services Engine (UJSE) invokes the Service Catalogue to get the description (required inputs, inputs considered personal data, info to invoke the service, etc.) of the services included in the corresponding workflow.
Invoke Usage Control (UJSE->UC)	The UJSE invokes the Usage Control (UC) to do the enforcement of the usage policies defined by the citizen.
Invoke Citizen Data Ownership (UJSE->CDO)	The User Journey Services Engine invokes the Citizen Data Ownership: - to inform about the services included in the new instanced workflow and that require data consent to be executed. - to verify if the citizen has already given consent for the use of the personal data involved in a specific service execution. If consent is not given, the UJSE won´t invoke the service execution. to inform that specific personal data has been used in a service invocation.

Invoke Usage Control (TD->UC)	The Transparency Dashboard invokes Usage Control to manage the usage policies to be defined by the citizen.
Invoke Service Catalogue (UC->SC)	The Usage Control invokes the Service Catalogue to get information about the services which make use of personal data.
Invoke Service Catalogue (CDO->SC)	The Citizen Data Ownership invokes the Service Catalogue to get the list of available services which make use of personal data, and which personal data they make use of.

All the participating components[2] are authenticated using Keycloak Identity management server. Through the definition of a common realm, users can be authenticated using the Single Sign On principle across all applications. Additionally, Keycloak is extended to include an eIDAS plugin to provide the option of eIDAS invocation using Keycloak as the central IMS authority.

[2] An exception is the UJMT, which has no interface to the citizens but to the Modeling Experts. A user authentication for the UJMT is considered in the future.

3 Components integrated for the Intermediate version

3.1 Scenario implementation

Main Scenario

The main scenario followed for the second release of the ACROSS platform involves a citizen from Latvia who interacts with the ACROSS One-Stop-Shop citizen application to create a journey to Greece for the purposes of studying. The citizen is presented with a straightforward way to invoke all the different services to submit and acquire all the data needed for this journey.

User Journeys intend to provide to citizens all needed information for their move between countries and also guide them to use relevant online services either by redirecting them to the proper websites or by offering some services directly within the ACROSS platform; in case there is no digital way to access a service, all the needed information about an offline interaction will be provided to the user (e.g. address, open hours, etc.)

In the above-described scenario, one of the indicative services that have been integrated is the 'Apply at University' service.

In order to be able to present journeys towards the citizens, the following preprocessing must take place in ACROSS.

Modelling Expert Interaction

Initially, by using the user interface of the User Journey Modeling Tool (UJMT), the initial workflow of the services needs to be designed by a Modelling Expert. The extensive details of the service descriptions (relevant inputs, invocation details, consent policies needed etc.) are registered in the Service Catalogue (Fig 3).

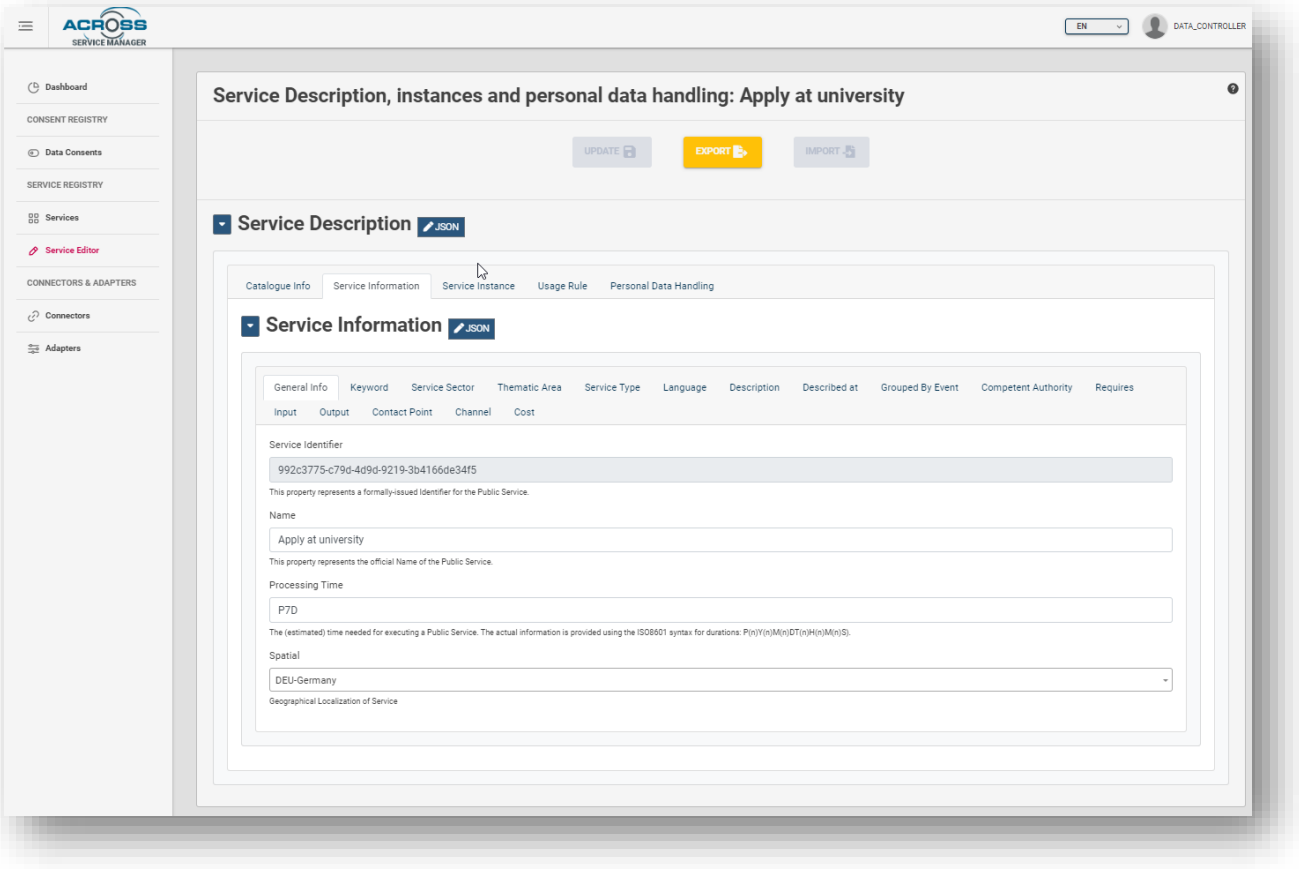


Figure 3: Service information sections included in the service catalogue

Figure 4 shows the user journey for the proof-of-concept scenario in the UJMT. The UJMT provides the Modelling Expert with a palette of graphical elements for user journey modelling, including elements for the services from the Service Catalogue in the left sidebar. The services from the Service Catalogue can also be assigned to graphical elements via right-click (see Figure 5).

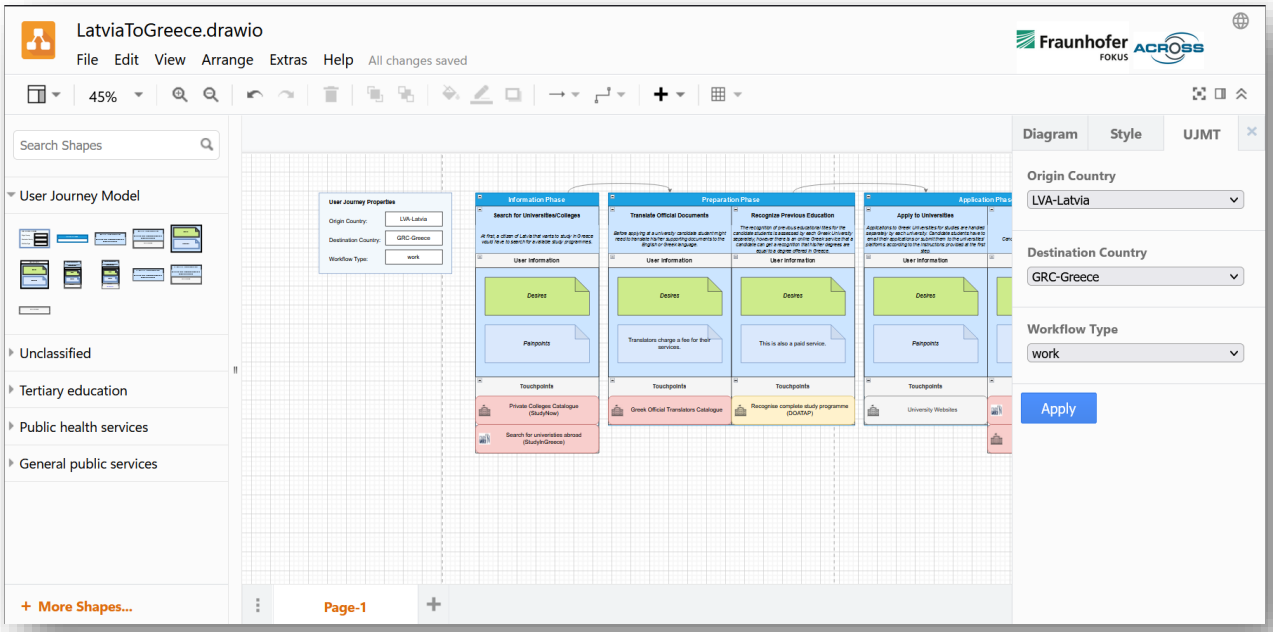


Figure 4: User Interface of the UJMT

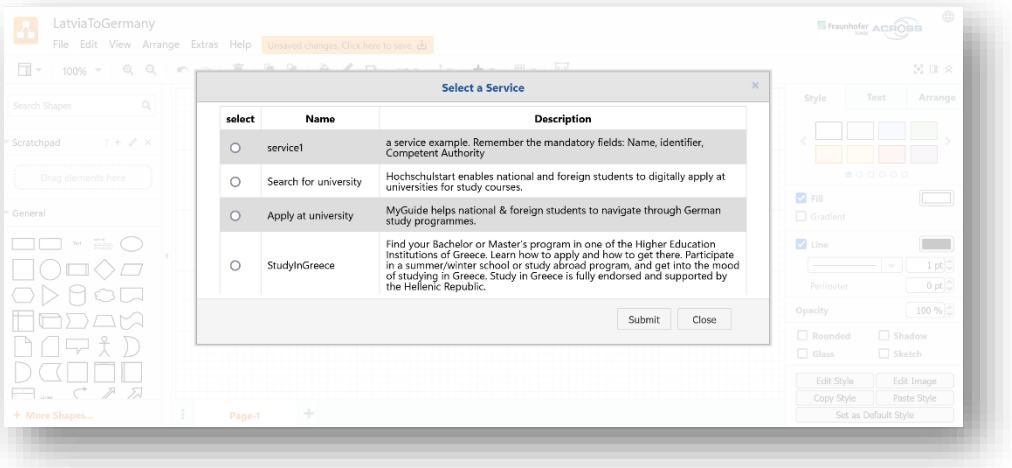


Figure 5: Service Selection for an Action in the UJMT

The BPMN model produced by the User Journey Modeling Tool (see Figure 6) was subsequently also ingested by the User Journey Service Engine. Hence, a new workflow representing a journey in ACROSS was created.

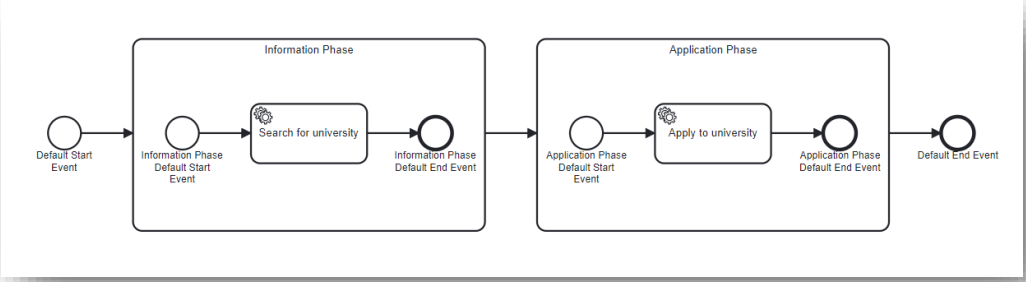


Figure 6: Generated BPMN diagram

In addition, the UJMT produced a JSON representation of the User Journey, which is sent to the UJSE and from there forwarded to the Citizen Frontend (see Figure 7).


```json
{
    "origin_country": "Latvia",
    "destination_country": "Greece",
    "workflow_type": "Study",
    "description": "This is a User Journey for a student from Latvia that wants to study in Greece.",
    "phases": [
        {
            "order": 0,
            "drawio_id": "uHRuqZ58I5v9CffmCqbi-80",
            "name": "Information Phase",
            "actions": [
                {
                    "order": 0,
                    "drawio_id": "uHRuqZ58I5v9CffmCqbi-81",
                    "name": "Search for university",
                    "description": "First, the citizen wants to search for universities.",
                    "touchpoints": [
                        {
                            "drawio_id": "uHRuqZ58I5v9CffmCqbi-107",
                            "service_id": "https://atlas.grnet.gr/DefaultEn.aspx",
                            "service_url": "https://atlas.grnet.gr/DefaultEn.aspx",
                            "location": "GRC-Greece",
                            "name": "ATLAS",
                            "description": "“Atlas” is a centralized online service which interconnects
                        }
                    ],
                    "is_mandatory": false
                }
            ]
```


Figure 7: JSON description of an example User Journey

From that point and forward, all applications of the Citizen Front End layer (Citizen web/mobile app and Transparency dashboard) have all the relevant information needed to deliver this journey towards the citizens.

Citizen Interaction

The citizen can visit the Citizen Web Application and find useful information about the available functionalities of the application, while being on the pre-login page. This will include the description of the available journeys, a tutorial video of the application, some statistics about the usability of the application itself, a list of useful user stories, etc. The user is also able to select the language and the theme (light or dark) of their preference, before logging into the app.

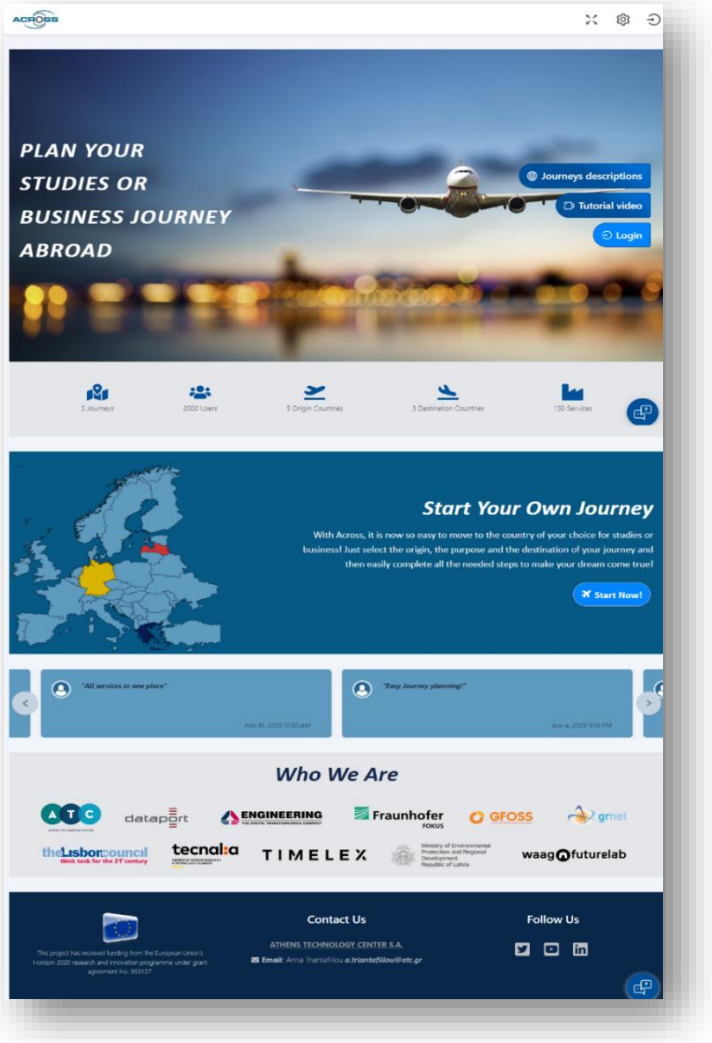


Figure 8: Login page of ACROSS Citizen Web Application

After logging-in, the user will be landed on the Dashboard page, where the choices of starting a new journey or reviewing any already initiated journeys will be available.

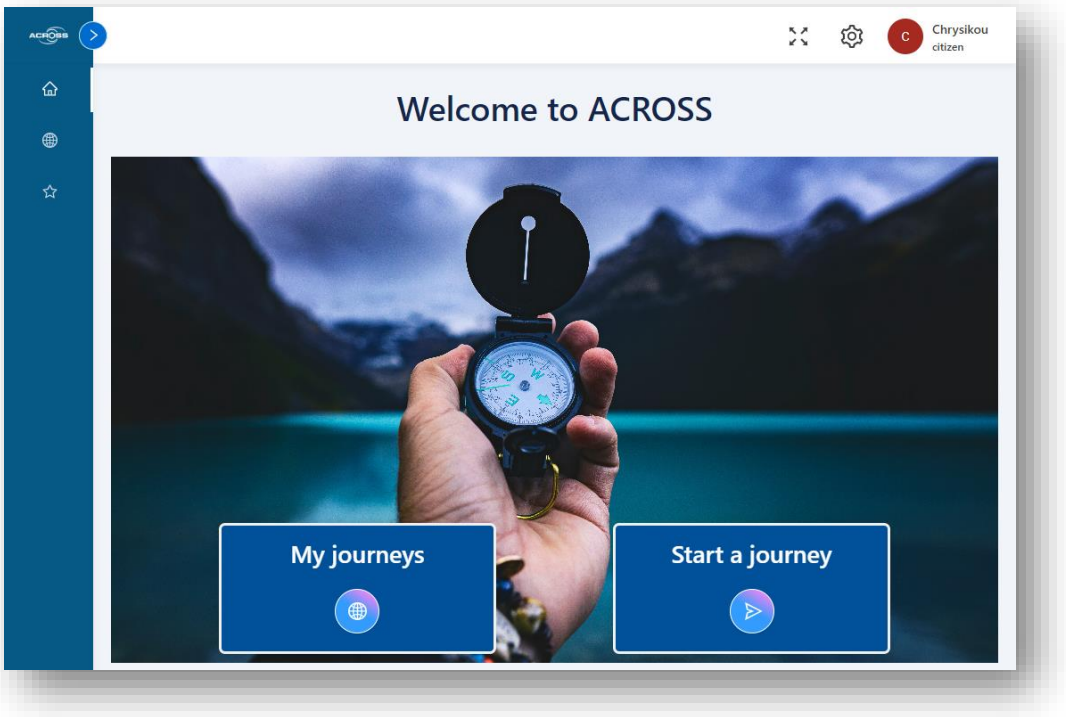


Figure 9: Citizen Web Application dashboard page

The citizen can initiate the journey of their choice by clicking on the 'Start a journey' button and selecting the journey details in the journey creation menu that appears on the screen (Figure 10). Only one journey with a specific combination of parameters (i.e. 'Origin', 'Purpose' and 'Destination') is allowed to be initiated by the user per time. In case the user attempts to start a journey that is identical to an already existing and running one, a warning message will inform them about the needed actions to be taken, as shown in Figure 11.

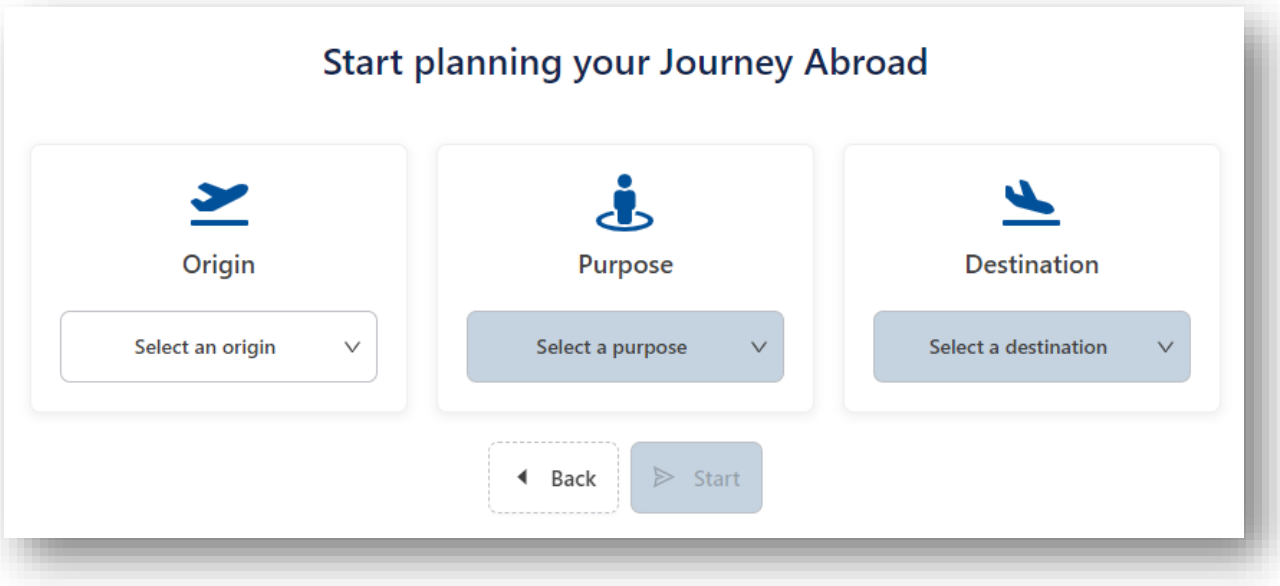


Figure 10: Journey creation menu

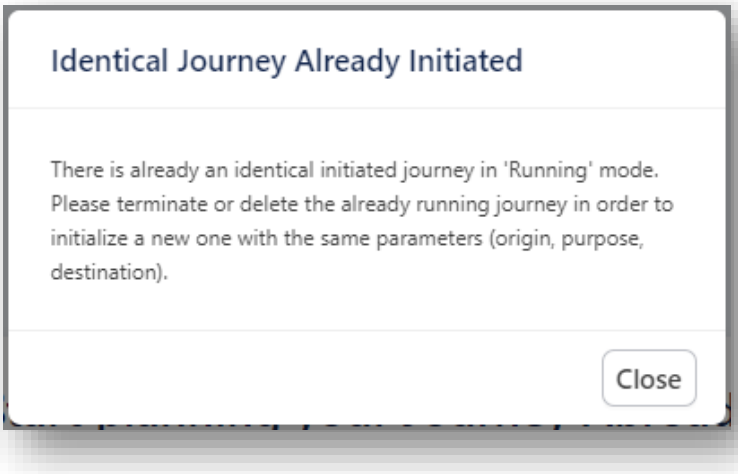


Figure 11: Warning message for identical initiated journey

A journey should either be deleted or marked as 'Finished' before the user initiates a new journey with the exact same parameters. To mark a journey as 'Finished', the user may click on the 'Terminate' button of the relative journey, as shown in Figure 12. A confirmation dialog will appear to inform the user about the 'effects' of a journey termination (Figure 13).

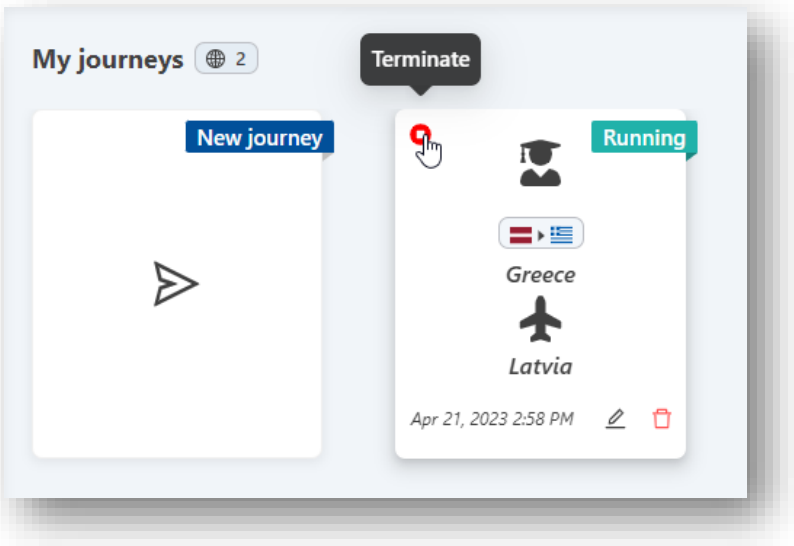


Figure 12: Initiated Journeys page

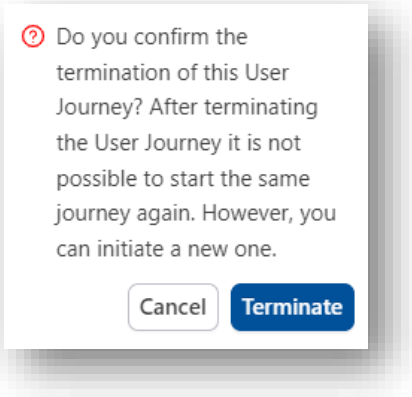


Figure 13: Confirmation dialog for journey termination

After the journey creation, the user can start navigating through the steps that need to be completed so that the journey is finalized and the citizen is ready to move to the destination of their choice. A journey can be considered as finalized when all related Actions are completed.

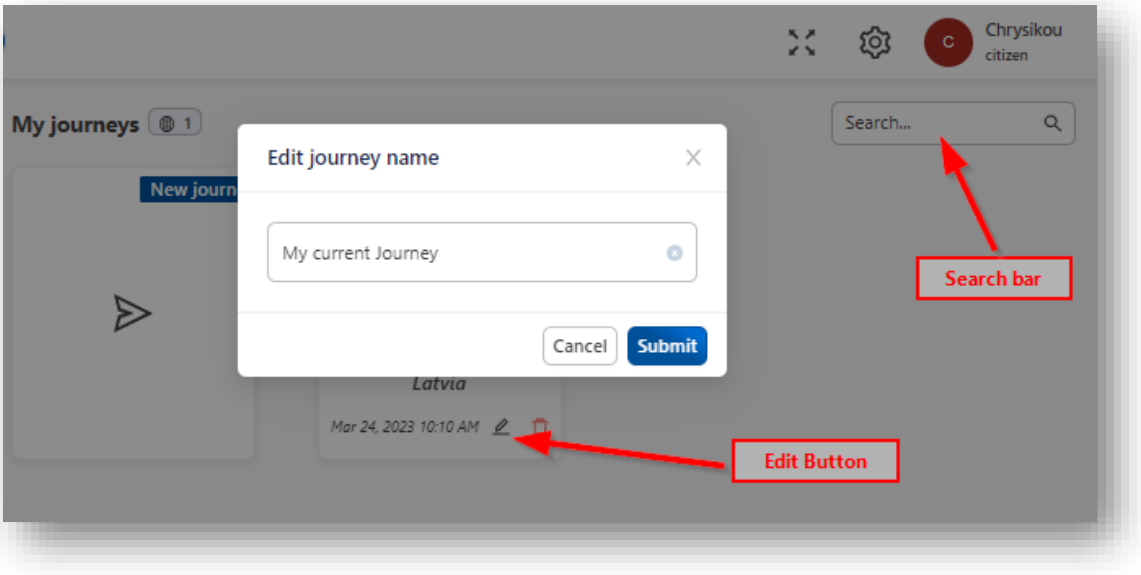


Figure 14: Editing and searching a journey

The user can rename a journey and give it a friendlier name (as shown in the figure above) so that they can easily search for it via the search bar appearing in the 'Journeys' page. The origin or the destination of the journey can also be used as searching terms to filter the initiated journeys shown on this page. If the user wishes to delete an already initiated journey, they can do so by pressing the trashcan icon next to the 'Edit' button.

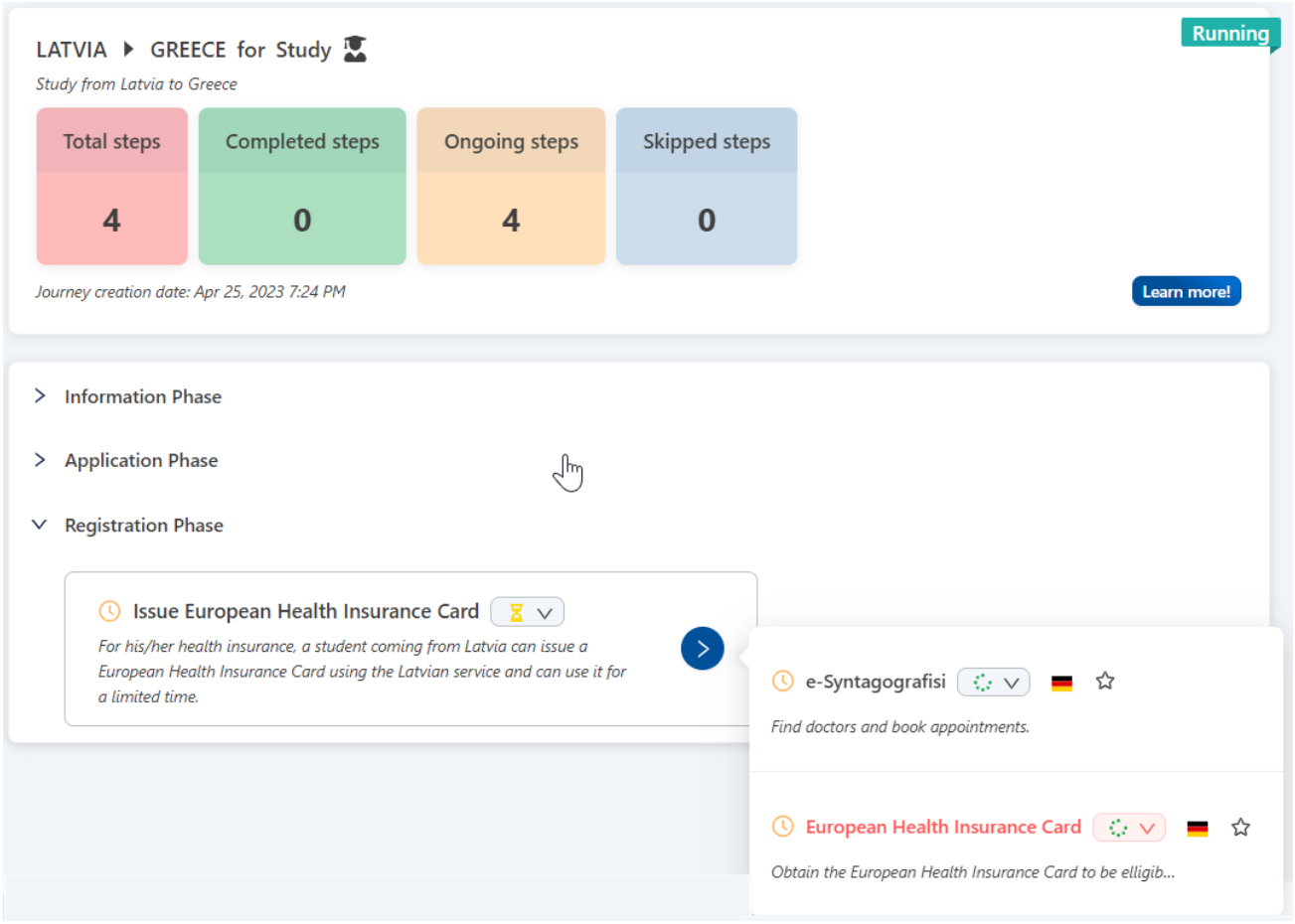


Figure 15: Journey Overview Page

An Action (e.g., 'Issue European Health Insurance Card') is defined as a group of Services (e.g., 'European Health Insurance Card', 'e-Syntagografisi'). Actions are also grouped into categories named Phases. In Figure 15 we can see the journey overview page, where the user will find all the information needed to complete the needed steps of the journey. There, the user can also see a colored overview of the statuses of the Actions (steps).

As seen in the Figure 16, the user can manually update the status of an Action or a Service in order to keep track of the progress made so far.

The available statuses for Actions (steps) are the following:

a. Pending (there are still services that have not been completed yet)
b. Skipped (the step is considered as not necessary by the user)
c. Finished (the Action has been completed; all needed services of this step have been concluded.)

The available statuses for Services are the following:

a. <empty> (no action has been taken yet by the user for this service)
b. Submitted (the application form of this service has been submitted by the user)
c. Rejected (the application form that has been submitted for this service has been rejected by the service provider)
d. Finished (the application form of this service has been successfully processed by the service provider)

In the case of asynchronous REST Services, the status of the service can be automatically updated by the service provider, e.g., the status will be updated to 'Finished' after the submitted form has been successfully processed. Even if the user is not logged in to the application at the moment the service provider updates the status, they will see this status update the next time they enter the Citizen Web Application. In the case of synchronous REST Services, the status of the service will be instantly updated, according to the service provider response.

If the status of an Action is set to 'Skipped' or 'Finished', the relevant Action will be greyed out and its content will not be accessible. The same applies for a Service, the status of which is set to 'Rejected' or 'Finished'. If the status of an Action or Service is updated to a value different than the above-mentioned values, the content will be accessible again.

The user is also informed about the country/countries offering a specific service, by the flag icon(s) appearing on the right of the Service name in the Service list, and also mark/unmark a Service as their favorite or not, by clicking on the 'star' icon on the right of the country flag icon, as shown below.

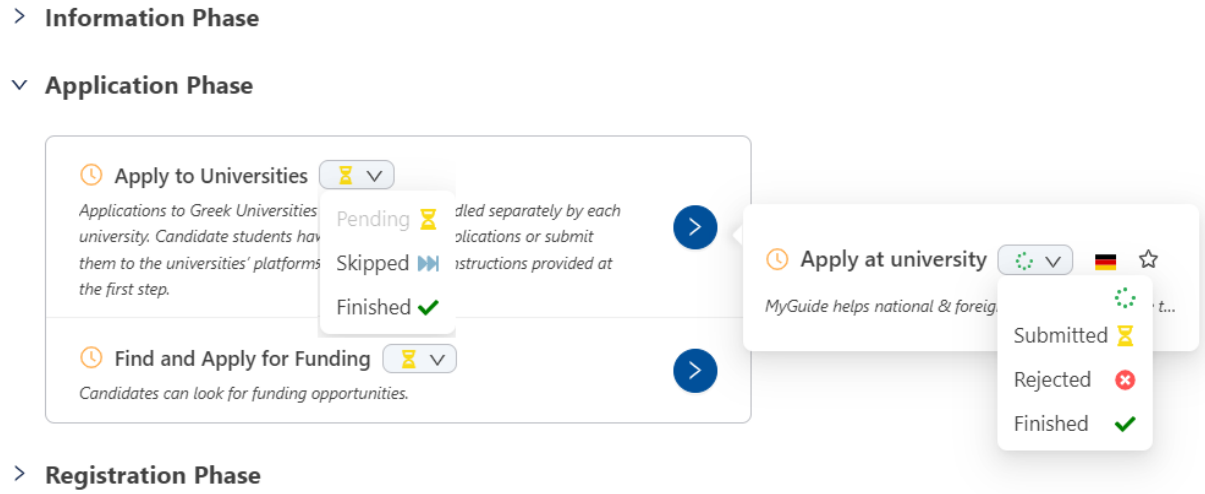


Figure 16: Action/Service status options

A history button is also available for the Action and also the Service level. The user is able to see the status history of an Action/Service, after clicking the orange clock button on the left of the Action/Service header.

When clicking on a Service, a pop-up window is opened where the user can see some useful metadata of the Service, such as a brief description of the Service, the estimated cost of the Service, the estimated time needed to conclude it, etc. In case the Service can be invoked via the Citizen Web Application, an 'Apply' button will be shown in this window (Figure 17). In case the Service cannot be invoked via the Citizen Web Application, a 'Redirect' button will be provided instead, which if clicked will redirect the user to the Service site, provided that the service is digitally available. If the Service is not digitally available, an 'Offline' tag indication will inform the user that the service can be accessed via other means (physical presence, etc.).

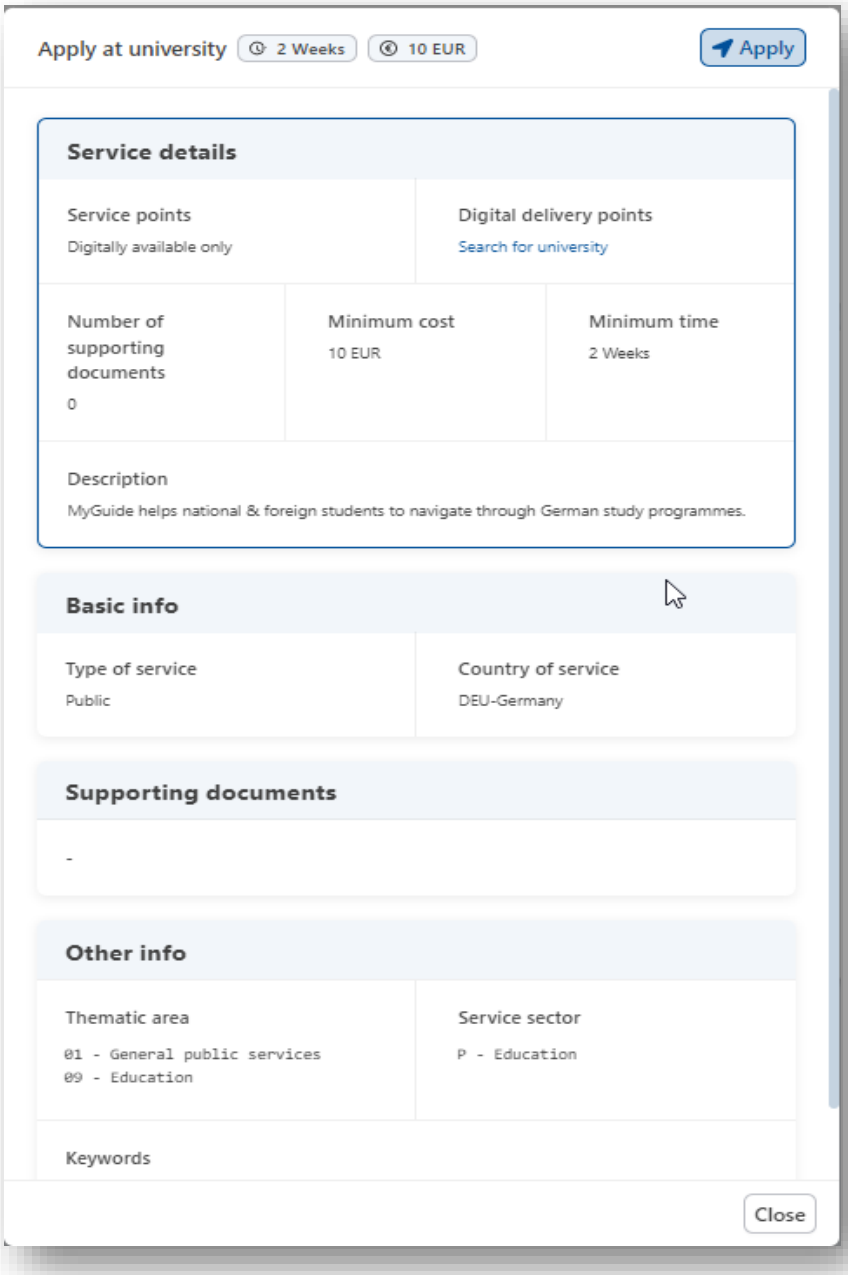


Figure 17: Service metadata info

In cases where Services that can be invoked by the Citizen Web Application require consent policies, the color of this button ('Apply') will be of red color. After clicking on it, the citizen will be given the opportunity to grant consent via a new pop-up window directly, or via being redirected to the Transparency Dashboard, where a more detailed consent management is offered. It is imperative for these consent policies to be granted by the user for the services to be invoked.

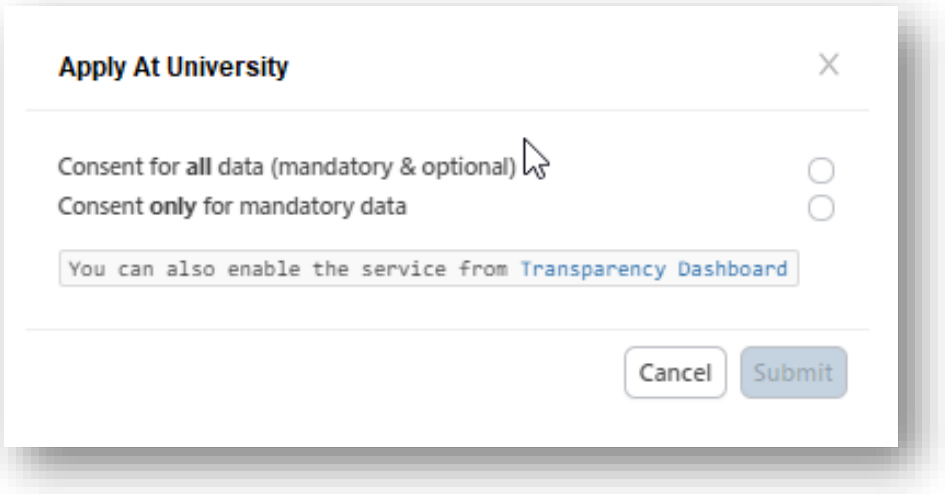


Figure 18: Consent granting pop-up window

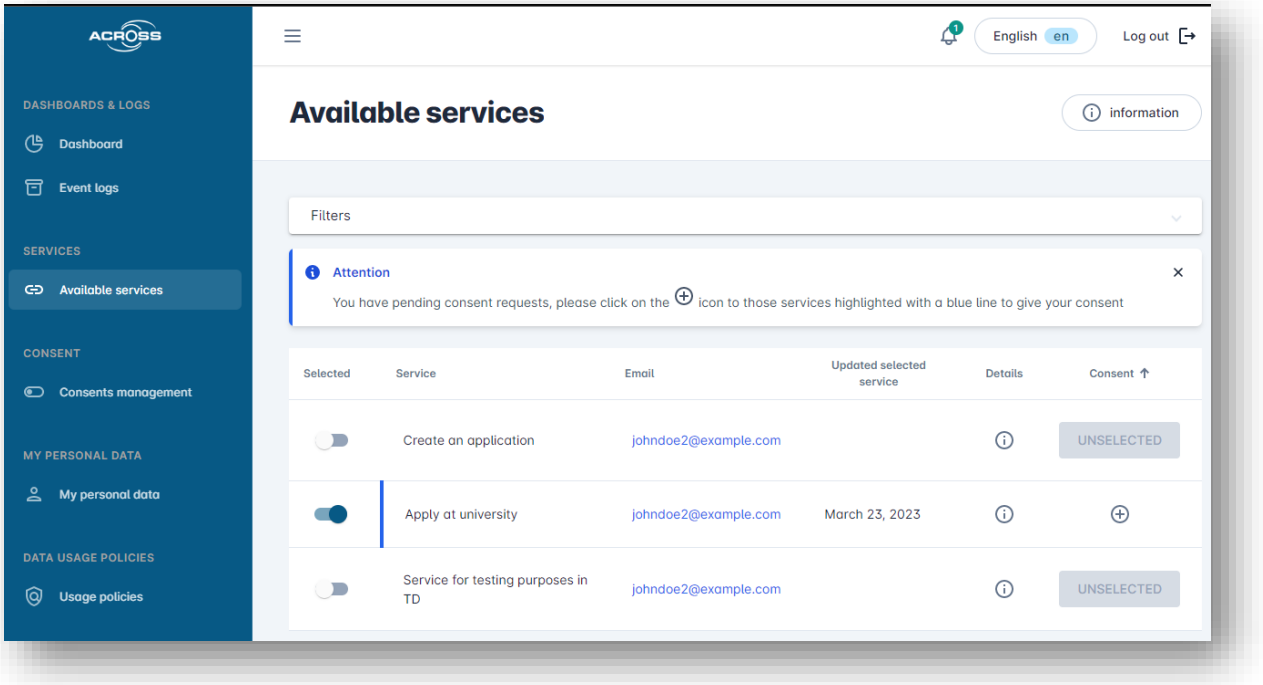


Figure 19: Transparency dashboard

In case the user chooses to be redirected to the Transparency Dashboard, as shown in the figure above, the citizen can grant or/and revoke the consent towards services of ACROSS. For this scenario, the "Apply at university" service requires certain consent provision from the citizen. The citizen agrees to these consent policies and then proceeds to the Citizen Web Application to start completing the application form of this service.

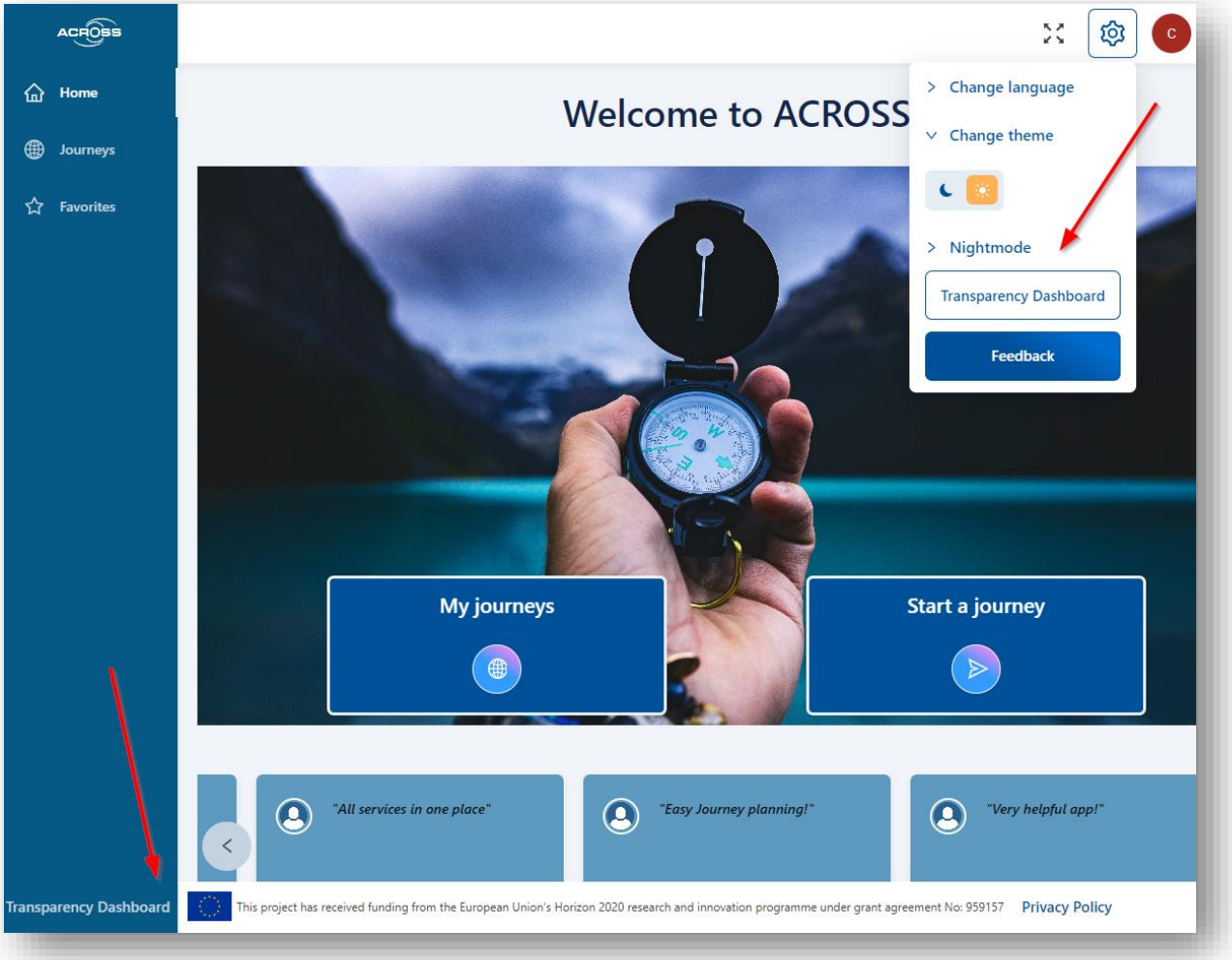


Figure 20: Redirections to Transparency Dashboard

In cases where consent is already granted for a Service (or no consent is needed at all), the 'Apply' button will be of blue color, and after the user clicks on it, the Service application form will be shown on the screen. Consent can be given at any time, since the user can also be redirected to the Transparency Dashboard by clicking on this option either in the bottom of the side menu of the Citizen Web Application, or in the menu appearing after clicking on the gear icon of the navigation bar on the top right of the screen (Figure 20).

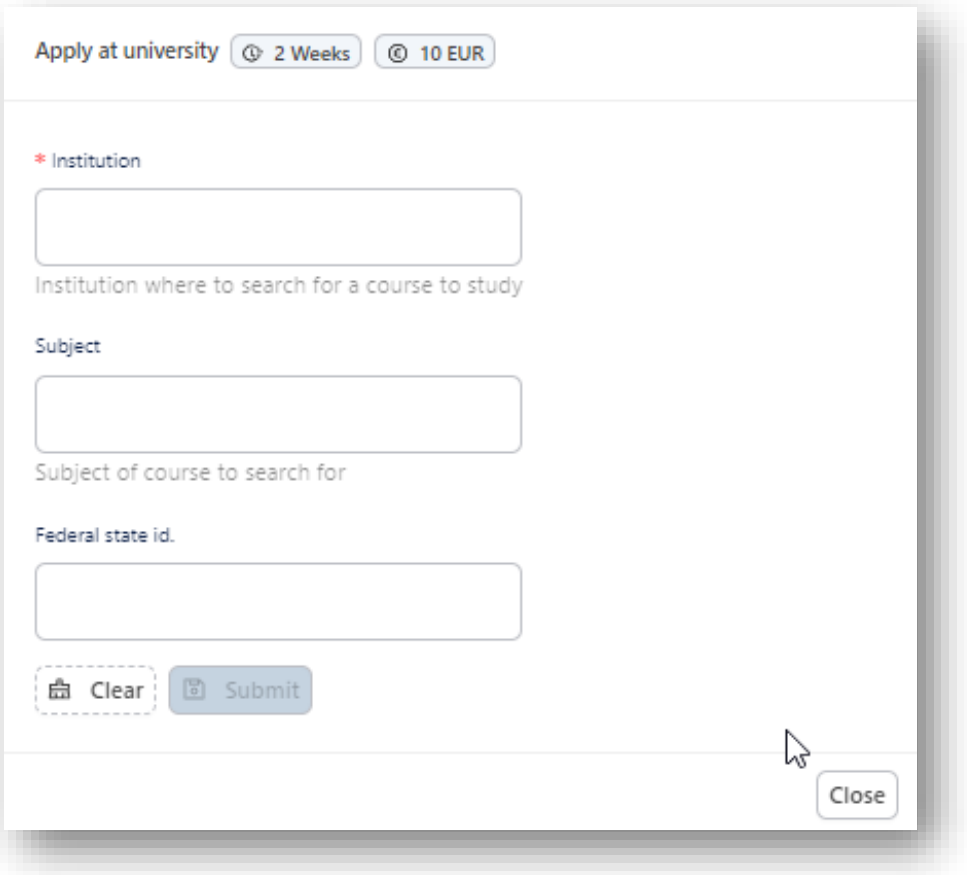


Figure 21: Service application form

As described earlier, after consent is granted, the citizen is free to provide all the relevant information required by the service provider, filling in the Service application form via the Citizen Web Application (Figure 21). After that, the citizen can invoke this service through the ACROSS application in a seamless way.

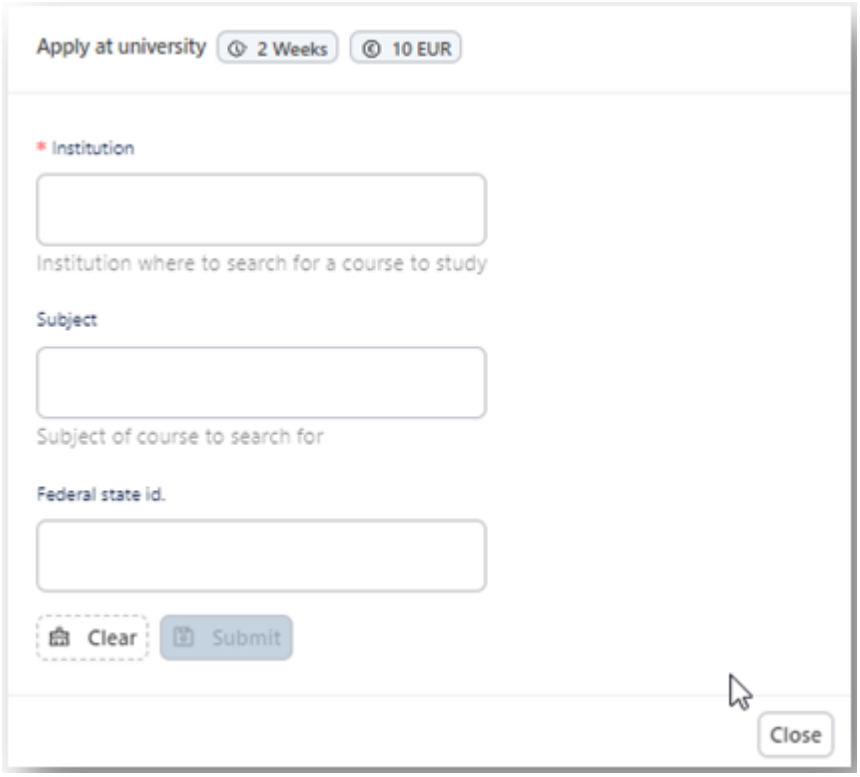


Figure 22: Colored overview of steps status

As mentioned earlier, the citizen can overview the progress of the journey and the statuses of the journey steps at any moment, as shown in the figure above. Each colored card informs the user about the number of the steps having a specific status. If a card is clicked, it will trigger a dialog containing the list of the related steps; if one of the steps of this list is also clicked, the user will be navigated to the specific step, for their convenience.

A list of the user's favorite services can be found in the 'Favorites' page, accessed via the left side-menu of the platform (Figure 23). There, the user can see their favorite services per initiated journey. If the user clicks on a service of this list, a dialog will appear on the screen, making the service directly accessible from this page.

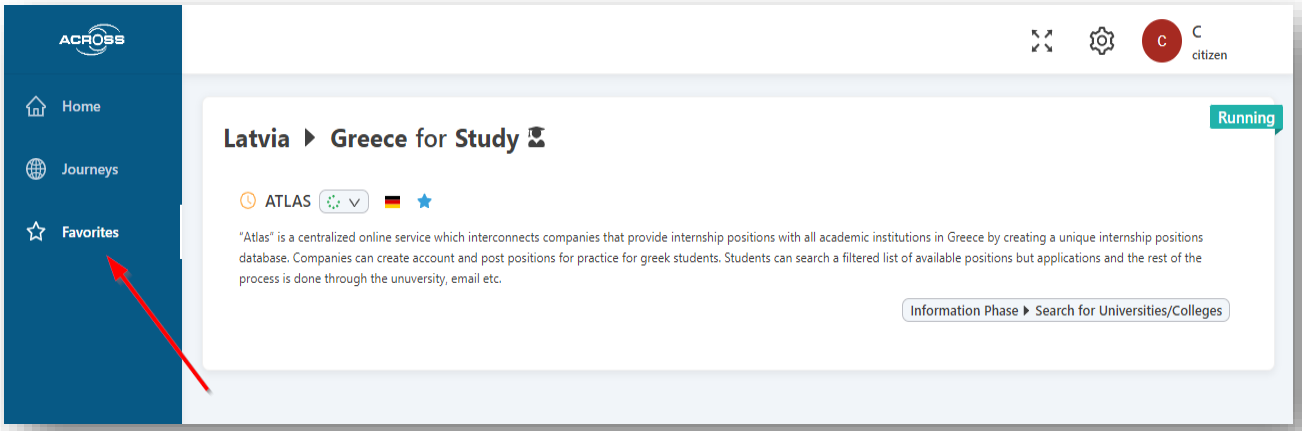


Figure 23: Favorite services page

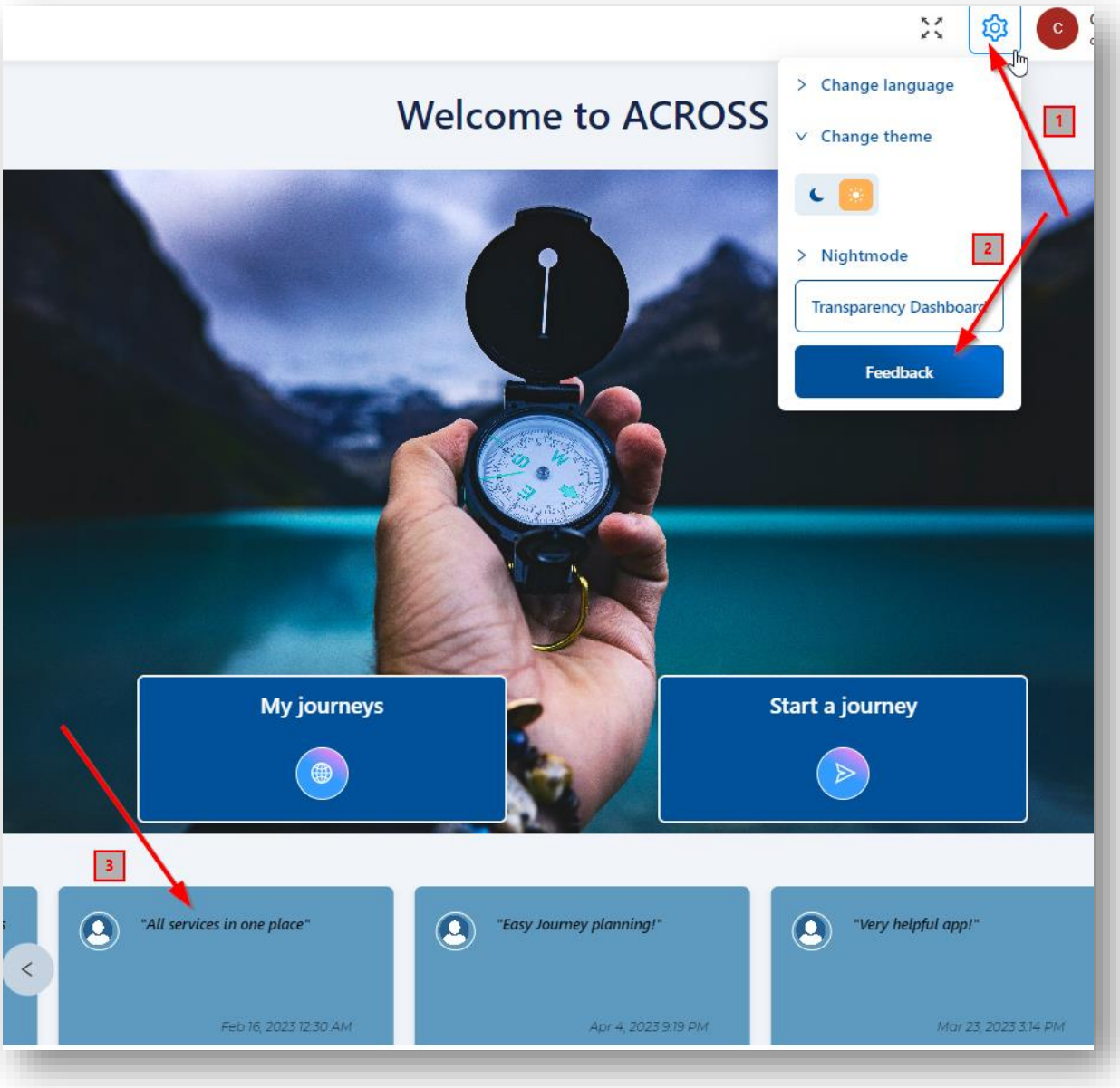


Figure 24: User feedback

The users of the Citizen Web Application are also free to describe their experience regarding the usage of the app and see their feedback in the list of the user stories. A 'Feedback' option is available on the top right of the screen, after expanding the menu of the 'gear' button. The user will also be prompted to provide their feedback, if they wish so, after a journey is manually terminated by them.

Virtual Assistant

Complementary to the above scenario, the integration and invocation of the Virtual Assistant was implemented. The citizen can control a substantial part of the UI elements of the Citizen Web/Mobile Application conversationally (i.e. through natural-language chat text input, or through spoken language), such as:

- To create a journey, including selection of source and destination country and the purpose of the journey (work or travel).
- To control Application's settings such as user/display language and select an UI theme
- To control the execution of individual services within a user journey workflow, e.g. by enabling conversational input into a web form that request additional data (needed for the service) from the user

As AI-based automatic speech recognition (ASR), Text-to-speech (TTS) synthesis and machine translation (MT) components have been integrated, the Virtual Assistant is able to accept typed and spoken user utterances in both English and German language when interacting with the Application UI. This works both for controlling the application (e.g. issuing command-like statements like "go to next page or "show previous page" for navigation) and for inputting data (for example, entering a name such as for the user's city of birth).

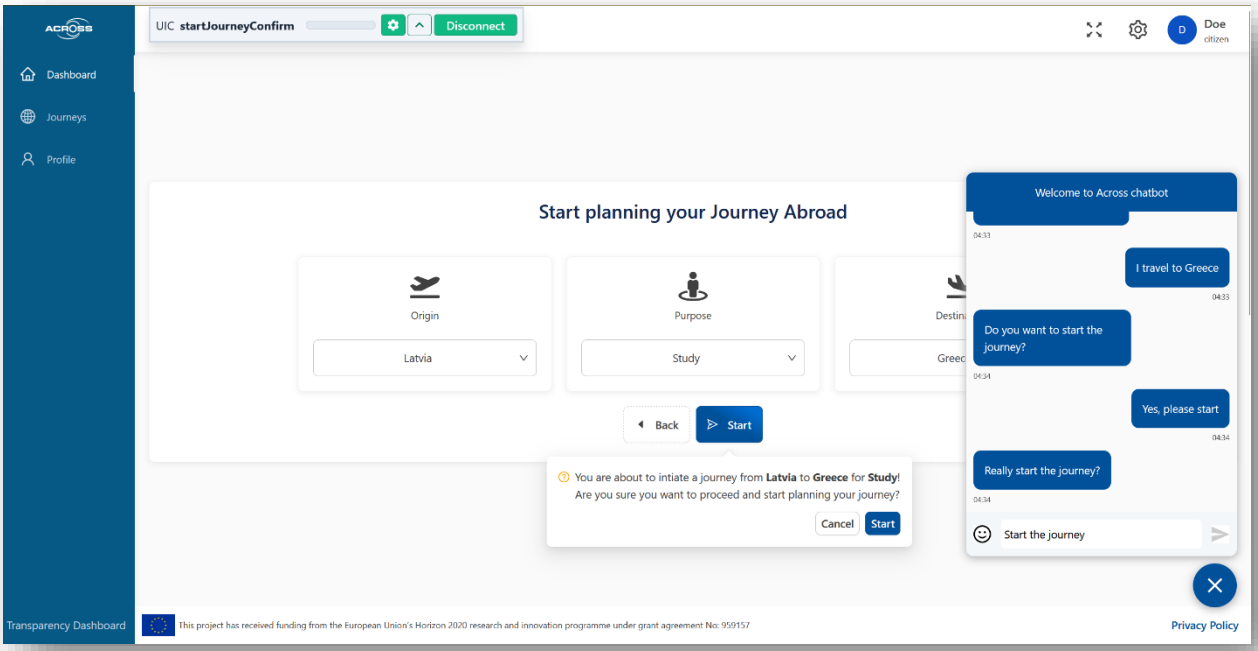


Figure 25: Chat User Interface of Virtual Assistant

The ACROSS Citizen Web/Mobile App provides an integrated chat widget which can be opened using a button. This chat widget has been coupled to the VA, so pressing the button simultaneously activates the Virtual Assistant, and the conversation between user and VA is then shown within this widget (see right side of Fig. 21 above).

In addition to this native chat widget (i.e., its end-user interface), the VA also provides a hidden debugging and testing widget. When activated, this widget allows the user to inspect important internal aspects of the VA during its execution. That way e.g., when misrecognitions of user utterance occur, it is possible to trace whether the error is due to the ASR not recognizing specific words or caused by too few training data for the intent recognition ML model.

3.2 Components Integration

Component	Git Registry
Service Catalogue	Main release: https://github.com/OPSILab/Service-Catalogue Documentation: https://service-catalogue.readthedocs.io/en/latest/
User Journey Modelling Tool Frontend (UJMT.F)	https://git.code.tecnalia.com/across/private/modeling-frontend/user-journey-modeling-tool/ujmt-frontend
User Journey Modelling Tool Proxy	https://git.code.tecnalia.com/across/private/modeling-frontend/user-journey-modeling-tool/ujmt-proxy
User Journey Modelling Tool Backend (UJMT.B)	https://git.code.tecnalia.com/across/private/modeling-frontend/user-journey-modeling-tool/ujmt-backend
Virtual Assistant Front-end (UI Connector)	https://git.code.tecnalia.com/across/private/citizen-front-end/web-application/web-application-UI/-/tree/virtual-assistant
Virtual Assistant Back-end (FDM)	(background material)
User Journey Service Engine (UJSE)	https://git.code.tecnalia.com/across/private/user-journey-service-delivery/userjourneyservicesengine
Transparency Dashboard	https://git.code.tecnalia.com/across/private/citizen-front-end/transparency-dashboard/transparency-dashboard-ui/-/tree/main/transparency_dashboard_frontend
Citizen Data Ownership	https://git.code.tecnalia.com/across/private/citizen-front-end/transparency-dashboard/transparency-dashboard-ui/-/tree/main/transparency_dashboard_backend
Usage Control	https://git.code.tecnalia.com/across/private/citizen-data-ownership-and-usage-control/usage-control/usagecontrol

Citizen Web Application Frontend	https://git.code.tecnalia.com/across/private/citizen-front-end/web-application/web-application-UI/-/tree/kubernetes
Citizen Web Application Backend	https://git.code.tecnalia.com/across/private/citizen-front-end/web-application/web-application-BE/-/tree/kubernetes

3.2.1 Service Catalogue

The following figure provides the available API used in component interactions:


Service Model
Service Model Description APIs to get and manage service model descriptions.
GET /api/v2/services Get all the Service Model descriptions.
PUT /api/v2/services Update Service Model description, by replacing the existing one
POST /api/v2/services Create a new Service Model description.
DELETE /api/v2/services Delete Service Model description by Service Id.
GET /api/v2/services/time Get Service time by serviceId.
GET /api/v2/services/specified/title Get the Service Model descriptions by specified Service Title.
GET /api/v2/services/specified/location Get the Service Model descriptions by specified Service Location.
GET /api/v2/services/specified/keyword Get the Service Model descriptions by specified Service Keywords.
GET /api/v2/services/specified/** Get the Service Model descriptions by specified Service Ids.
GET /api/v2/services/json/** Get the Service Model description by Service Id.
GET /api/v2/services/isPersonalDataHandling Get the Service Model descriptions is handling personal data
GET /api/v2/services/isPersonalDataHandling/count Get the count of the Service Model descriptions is personal data handling.
GET /api/v2/services/count Get the count of the registered Service Model descriptions (total, public and private services).
GET /api/v2/services/count/thematicArea Get the Service Models count grouped by Thematic Area.
GET /api/v2/services/count/sector Get the Service Models count grouped by Sector.
GET /api/v2/services/count/location Get the Service Models count grouped by Spatial.
GET /api/v2/services/count/groupedBy Get the Service Models count grouped by GroupedBy.
GET /api/v2/services/cost Get Service Cost by serviceId.
GET /api/v2/connectors/count Get the count of the registered Connector descriptions (total, public and private services).
GET /api/v2/adapters/count Get the count of the registered Adapter descriptions (total, public and private services).

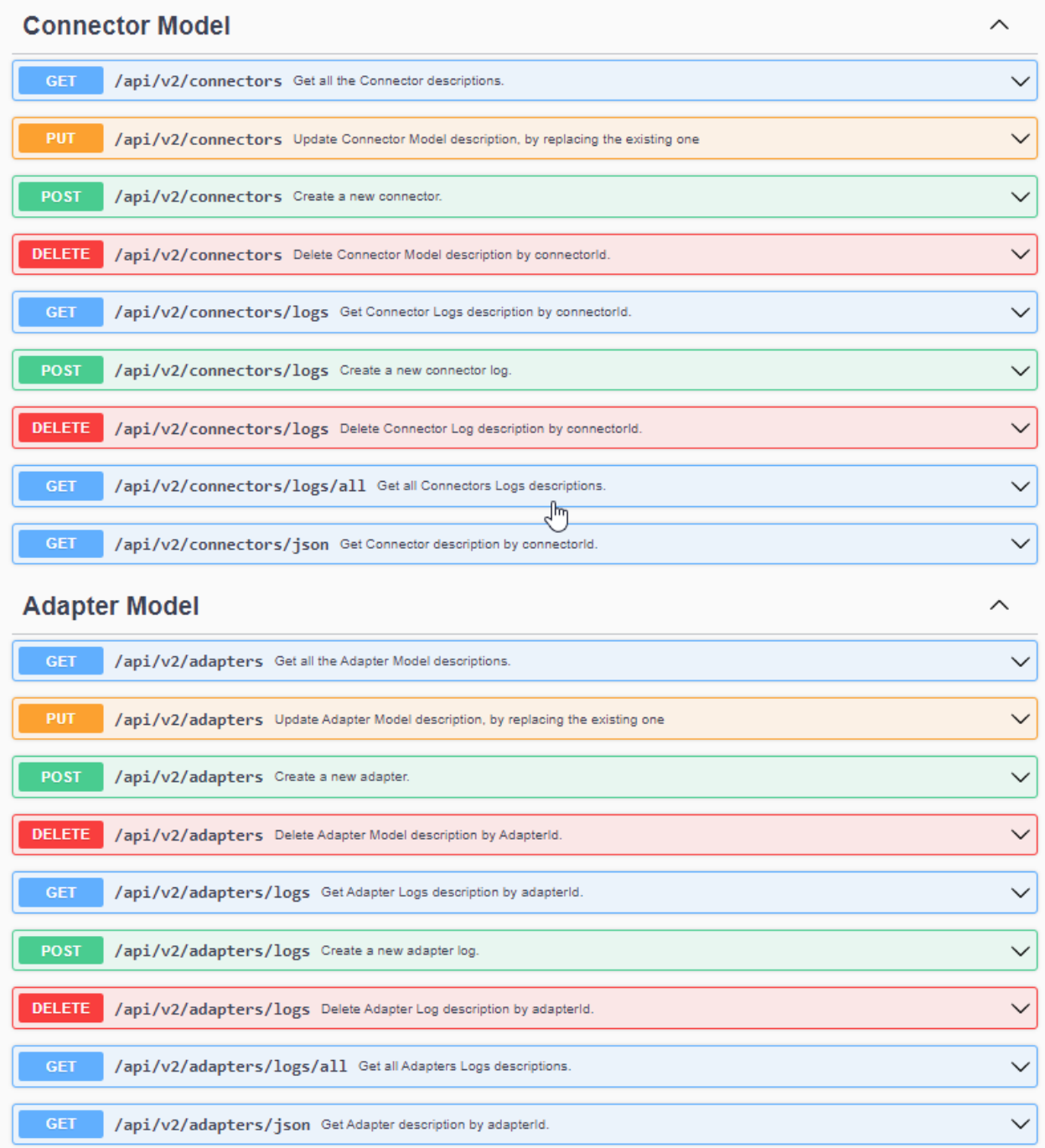


Figure 26: Service Catalogue APIs

3.2.2 User Journey Modeling Tool

The UJMT communicates with other components in the platform, but is not a service provider itself and therefore does not offer an API. As a modeling tool, the UJMT has a user interface to the modeling expert.

The UJMT retrieves the entered services via the defined API of the Service Catalogue and offers them to the modeling expert as graphic elements. Furthermore, additional information such as the locations and thematic areas of the services are retrieved and used in the tool for service categorization and filtering.

The UJMT also interacts with the UJSE. It passes the user journey information in various formats via the defined API, thereby creating new user journeys for the citizen. In the future, changing and deleting user journeys via the UJMT will also be supported.

3.2.3 Virtual Assistant

Although it supports the ACROSS Citizen Web/Mobile App, providing additional functionality to the App, the VA technically is not in a service provider role within the ACROSS architecture. Therefore, no service APIs are documented here.

Instead, the VA is a user-exposed component – it provides an additional, *conversational user interface* for the ACROSS Citizen Web/Mobile App to the citizen. Interestingly, the VA is not in a service *consumer* role in the technical sense either: It does control the ACROSS Citizen Web/Mobile App, but not through an actual service API. Instead, this control is exercised directly through the standardized Javascript programming interface offered by the browser's DOM (document object model) [3] together with the internal Javascript programming interfaces provided by the ACROSS Citizen Web/Mobile App's front-end code.

virtual-assistant / web-application-UI / src / App.tsx

Find file | Blame | History | Permalink

App.tsx 3.57 KiB

Open in Web IDE | Replace | Delete

```
1  import type { AuthClientTokens } from '@react-keycloak/core';
2  import { ReactKeycloakProvider } from '@react-keycloak/web';
3  import { ConfigProvider } from 'antd';
4  import { Locale } from 'antd/lib/locale-provider';
5  import deDe from 'antd/lib/locale/de_DE';
6  import elGR from 'antd/lib/locale/el_GR';
7  import enUS from 'antd/lib/locale/en_US';
8  import lvLV from 'antd/lib/locale/lv_LV';
9  import { KeycloakInitOptions } from 'keycloak-js';
10 import React from 'react';
11 import { Toaster } from 'react-hot-toast';
12 import { useTranslation } from 'react-i18next';
13 import 'typeface-lato';
14 import 'typeface-montserrat';
15 import { setSession } from './api/config/http.api';
16 import { AppRouter } from './components/router/AppRouter';
17 import { notificationController } from './controllers/notificationController';
18 import { useAppSelector } from './hooks/reduxHooks';
19 import { useAutoNightMode } from './hooks/useAutoNightMode';
20 import { useLanguage } from './hooks/useLanguage';
21 import { useThemeWatcher } from './hooks/useThemeWatcher';
22 import keycloak from './Keycloack';
23 import GlobalStyle from './styles/GlobalStyle';
24 import { notificationDarkColorsTheme } from './styles/themes/dark/darkTheme';
25 import { notificationLightColorsTheme } from './styles/themes/light/lightTheme';
26 import { themeObject } from './styles/themes/themeVariables';
27
28 import UIConnector from './@uic';
29 UIConnector.toggleChat();
```

Figure 27: UIC integration into CWA font-end (line 28)

This is possible because the VA's UIC (User Interface Controller) is directly integrated into the Web/Mobile App's front-end (see figure above). On initialization, the VA immediately opens a websocket connection to its back-end (the FISA Dialogue Manager, FDM), enabling the processing of conversational user utterances during the just-started session. Each time a user utterance has been processed by the FDM and the results transferred back to the UIC, the UIC can then directly access the DOM and all of the CWA frontend's functionality, e.g. to programmatically fill in a form field, press a button, or open a hyperlink. Along the same lines, the UIC uses the Javascript browser interfaces standardized by the W3C for accessing the user's microphone and speaker, in order to enable the VA's bi-directional voice communication.

More details on this mode of integration, the VA's function and on the connection and protocol used for communication between the UIC and the FDM can be found in [4, Chapter 3].

3.2.4 User Journey Service Engine

The following figure provides the available API used in component interactions. This API is divided into 3 sections:

- citizen-frontend-api-controller: it provides the API for the Citizen Front-End for executing workflows.
- modelling-tool-api-controller: it provides the API for the modelling tool for uploading new defined workflows.
- log-api-controller: it provides the API for the Citizen Front-End to get the UJSE logs.

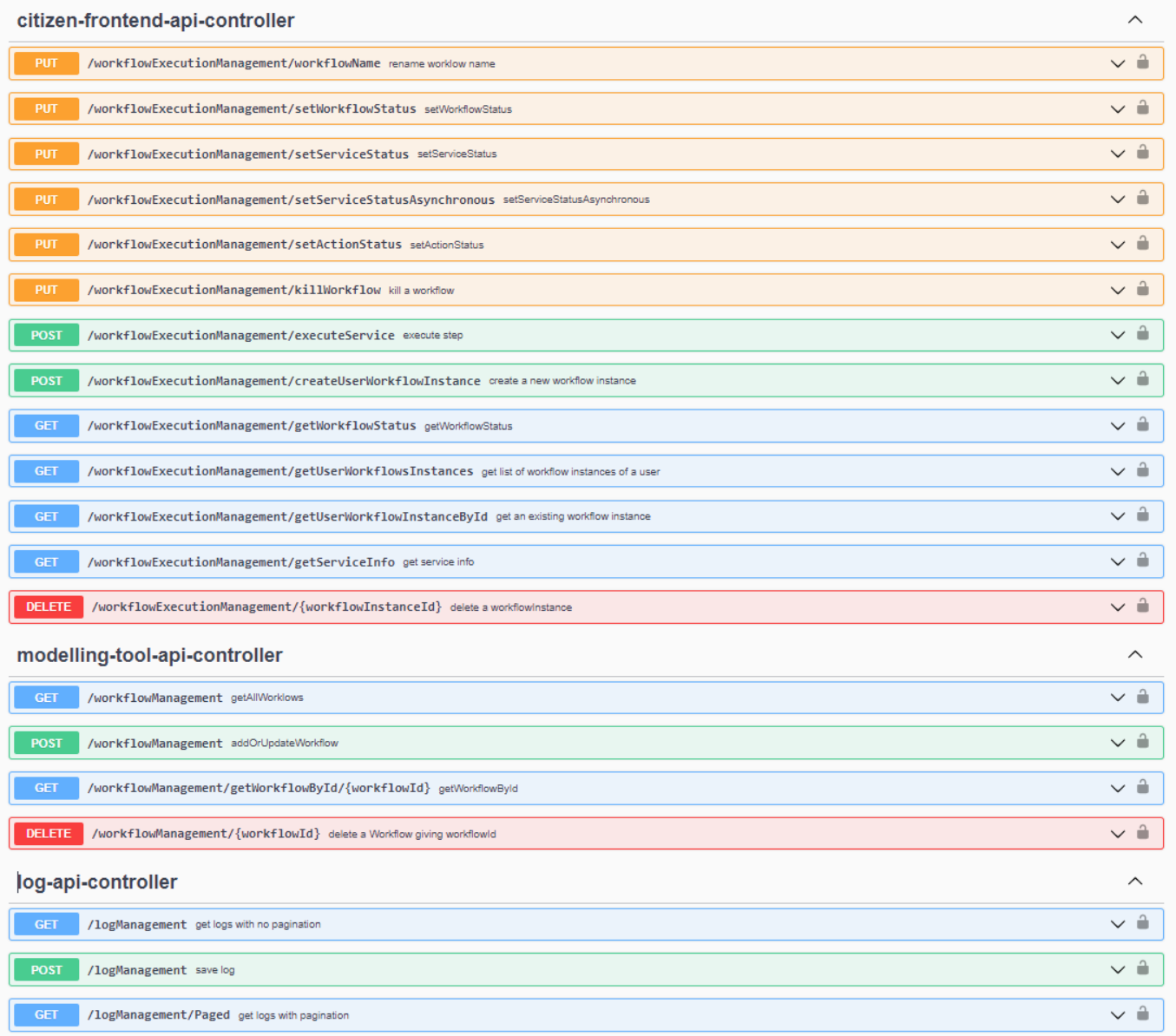
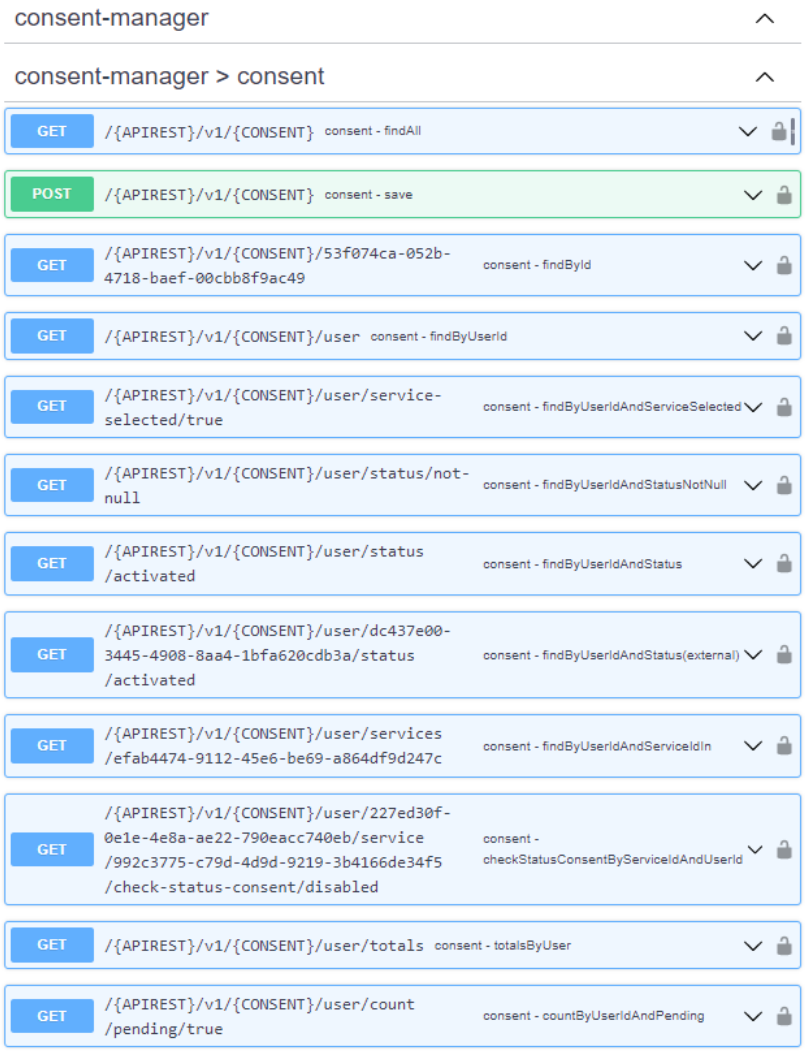


Figure 28: UJSE APIs

3.2.5 Transparency Dashboard

The following figure provides the available API used in component interactions. This API is divided into 2 sections:

- consent-manager > consent and consent-manager > personal-data-consent: it provides the API for the Transparency Dashboard to manage the consents, that is, to give, deny or withdraw consents and to get information about the consents already given, by different criteria.
- Consent-manager > event-log: it provides the API for the Transparency Dashboard to retrieve the events (applied actions on consents).



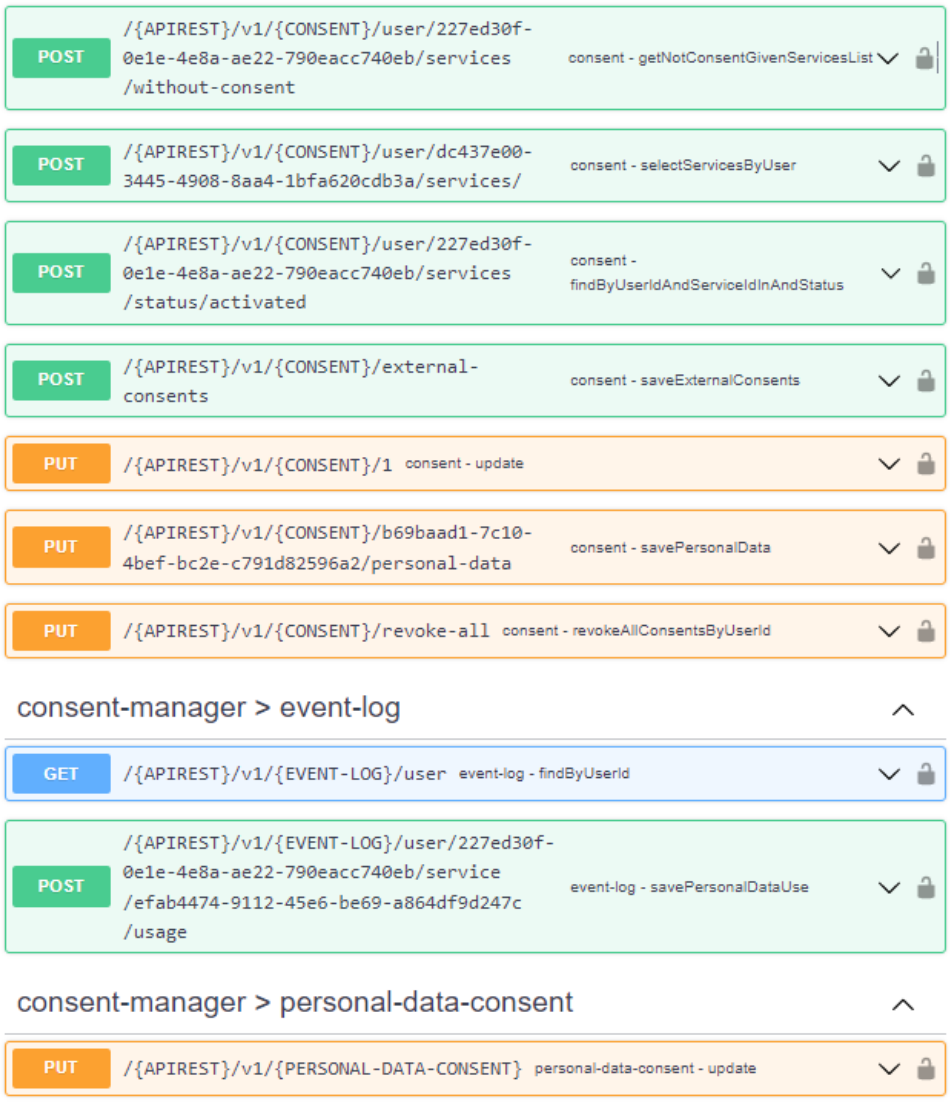


Figure 29: Transparency Dashboard APIs

3.2.6 Citizen Web Application

The following figure provides the available API used for the interaction between the service providers of Asynchronous REST Services and the Citizen Web Application:

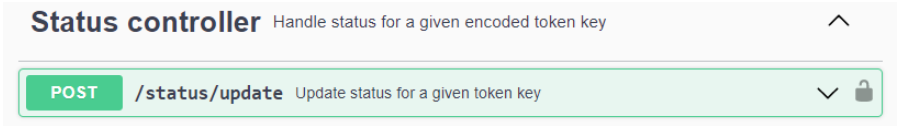


Figure 30: Citizen Web Application API

3.2.7 Usage Control

The following figure provides the available API used in component interactions. This API is divided into 2 sections:

- usage-policies-controller: it provides the API for the Transparency Dashboard to manage the usage policies defined by the citizen.
- enforce-usage-controller: it provides the API for UJSE to do the enforcement of the defined usage policies, when executing a service of the workflow.
- admin-controller: it provides the API of the PIP endpoint to be used when doing the enforcement of the usage policies, to get the number of times that the service provider has accessed personal data of a user.
- policy-rule-controller: it provides the API to get the policy patterns supported by the Usage Control.
- log-api-controller: it provides the API for the Transparency Dashboard to get the Usage Control logs.

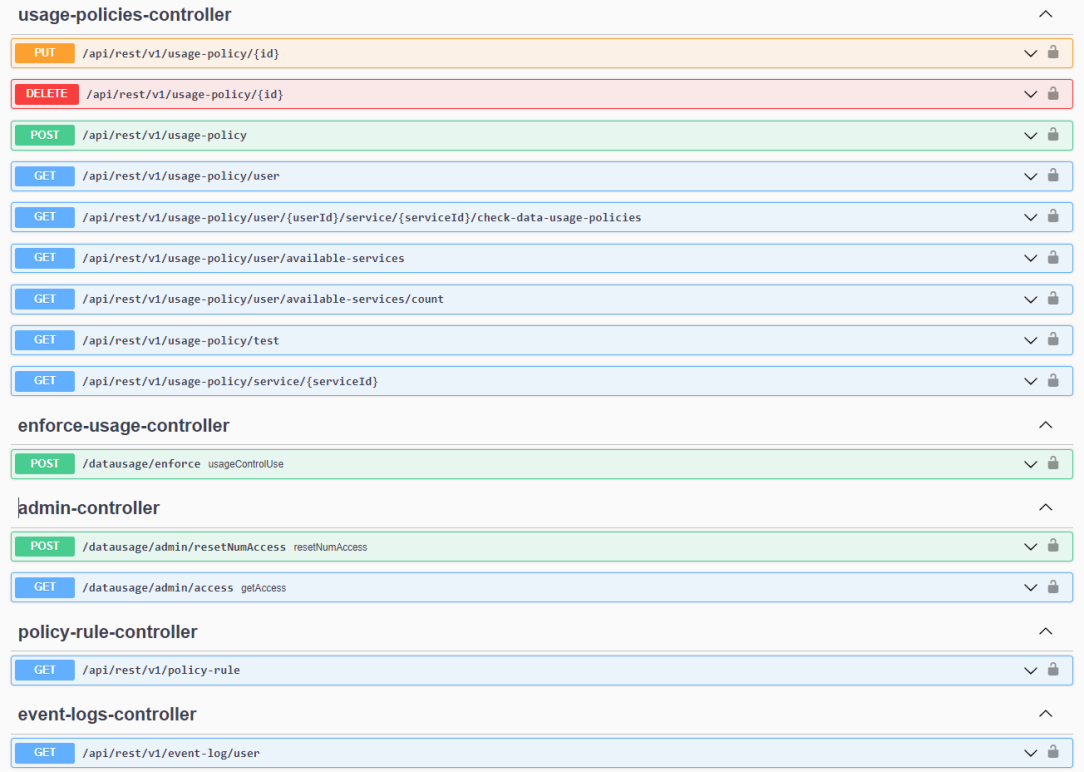


Figure 31: Usage Control APIs

4 Conclusions

This deliverable is the accompanying report of the second version of the ACROSS Integrated Prototype. This version includes a complete set of functionalities covering the needs of all the pilots and enabling the ACROSS stakeholders to test and evaluate at a great extent the concepts and knowledge conveyed by the project.

The integration of new components, as well as the updates of existing ones was a straightforward process and proved the flexibility and extensibility of the platform. The architecture allowed for easy adaption of new capabilities and functionalities developed in the context of WP3 and WP4 and made it possible to quickly present the users with the latest updates through a continuous delivery of new releases. This way, most of the user feedback was addressed and the platform capabilities evolved to match most of the user needs.

The next steps include continuous updating of the ACROSS platform prototype according to the feedback that will be received during the intermediate evaluation period, as well as the ongoing work in the technical WPs. Intermediate releases are planned to continue during the final year of the project so as to facilitate the project time plan and the needs of the other WPs in view of the final release which is officially planned by M36 of the project (January 2024).

5 References

[1] D5.2 - System Architecture & Implementation Plan – Final, ACROSS project, November 2022

[2] D4.5 Micro Proxies and services catalogue – Intermediate, ACROSS project, January 2023

[3] World Wide Web Consortium: Document Object Model (DOM) Technical Reports https://www.w3.org/DOM/DOMTR

[4] D4.8 User Support Tools – Intermediate, ACROSS project, February 2023