H2020-SC6-GOVERNANCE-2018-2019-2020

DT-GOVERNANCE-05-2018-2019-2020

D5.5: ACROSS Platform Prototype and Applications - Final

Project Reference No	959157 — ACROSS — H2020-SC6-GOVERNANCE-2018-2019-2020
Deliverable	D5.5: ACROSS Platform Prototype and Applications - Final
Work package	WP5: ACROSS Applications and Platform Integration
Nature	Other
Dissemination Level	Public
Date	30.04.2024
Status	Final
Editor(s)	Elena Chrysikou (ATC)
Contributor(s)	Vincenzo Savarino (ENG), Marina Klitsi (ATC), Ernst Thilo (FHG), Murua Belacortu, Idoia, Iturraspe Barturen, Urtza (TECNALIA)
Reviewer(s)	ENG, GRNET
Document description	This deliverable describes the final 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

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
V0.1	15.03.2024	First skeleton of the deliverable and TOC	ATC
V0.2	16.04.2024	Final draft	ATC
V0.3	17.04.2024	Updated Component Integration section (API)	TECNALIA, ENG
V0.4	18.04.2024	Updated version based on partners input	ATC
V0.5	25.04.2024	Input in Section 3	FOKUS
V1.0	26.04.2024	Final version to be submitted	ATC

Executive Summary

This deliverable is an accompanying report to the Demonstrator of the Final Integrated Prototype of the ACROSS platform. This prototype includes a full set of functionalities covering all three project pilots. It also incorporates feedback received during the evaluation periods and of course, encapsulates all the achievements and updated tools and components coming from the other work packages.

The WP5: ACROSS Applications & 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 prototype was driven from the user stories as delivered from the WP6: Use cases deployment, evaluation & impact assessment.

Following the implementation of the individual modules, this WP delivers an integrated view of the ACROSS platform to act as the test-bed for setting the ACROSS Pilots in WP6 and evaluating and validating the results in real-life scenarios. The content presented in this deliverable includes parts described in the previous version of the deliverable and all the updates of the final prototype. This way, the deliverable can serve as a standalone report describing the final ACROSS prototype via the use of a user guide of the system.

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
CFE	Citizen Front End
CDO	Citizen Data Ownership
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
VA	Virtual Assistant
VCPU	Virtual Central Processing Unit
WP	Work package

1 Introduction

1.1 Purpose and Scope

This deliverable aims to provide an overview of the final release of the ACROSS Prototype. To this end, the report briefly describes the implementation aspects of the prototype and elaborates on the updates with respect to the intermediate version. The current new and updated functionalities are explained in the form of a user guide, including screenshots of the prototype accompanied by descriptions of the offered functionalities.

The Final Integrated prototype is the evolution of the intermediate Integrated Prototype that was delivered at the end of April 2023 and reported in the Deliverable D5.4: ACROSS Platform Prototype and Applications – Intermediate. The updates of this version are based on the received user feedback and on the advancements of the available components.

The main updates are listed below:

- 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

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, and 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 evaluation of the ACROSS solution, as detailed in the "D6.3 Use case evaluation and impact assessment – Final" report, represents a significant milestone in understanding the platform's impact and effectiveness. The integration of feedback from both citizens and experts in mobility and service delivery has been instrumental in refining the platform's features and functionalities.

This collaborative approach ensures that the platform evolves to meet the needs of its users effectively. Moreover, the identification of requests beyond the project's scope, documented as best practices in D6.3, provides valuable insights for future implementations, highlighting the importance of flexibility and foresight in the development of such innovative solutions.

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 third 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:

- 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 final version of the ACROSS prototype;
- Section 4 concludes this report.

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

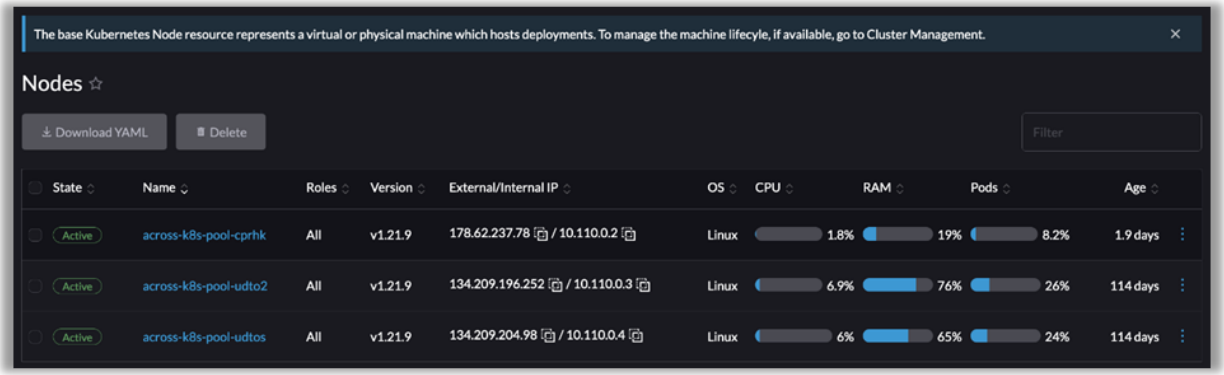
2 Software and System Design principles

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 having done 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.

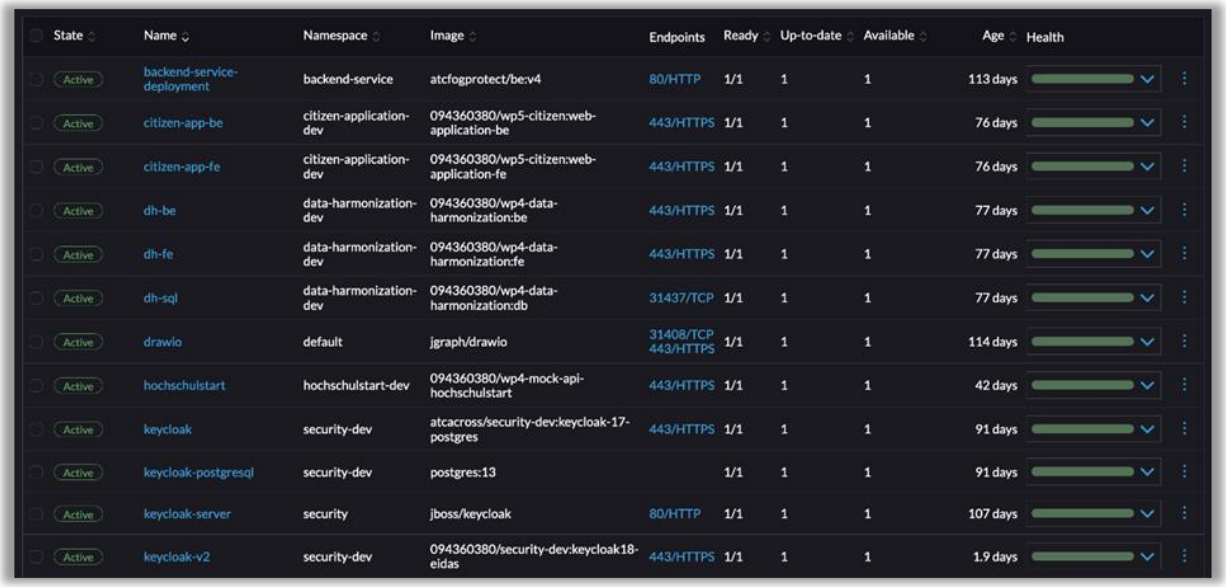
2.1 Operational Environment

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.

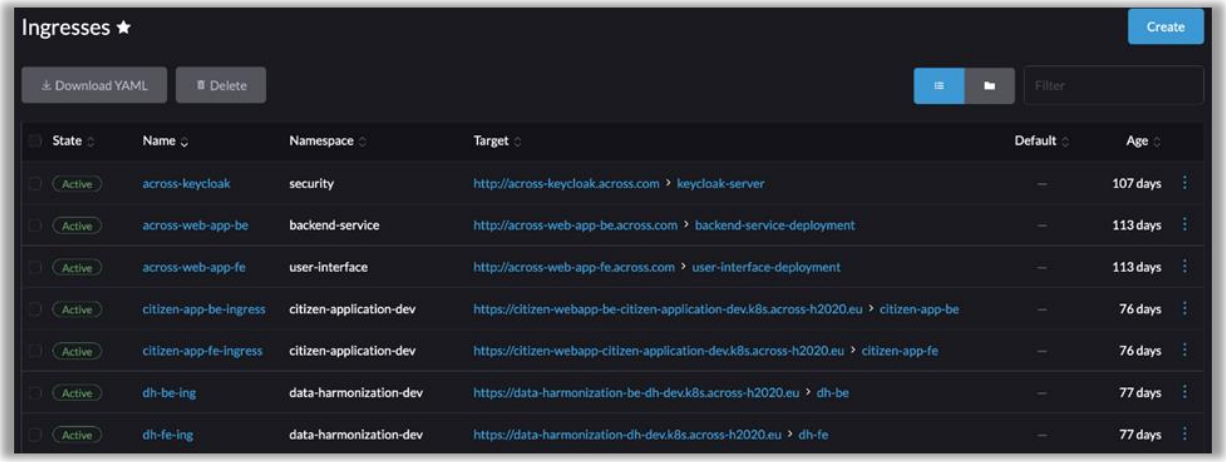


Through the provision of namespaces and projects (as a feature of K8S) the operational testbed is set to host different environments (development-staging-production). Each component is mapped to a dedicated deployment configuration that handles the continuous integration and deployment, scale up and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



Ingresses ★

Create

Download YAML | Delete | Filter

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:

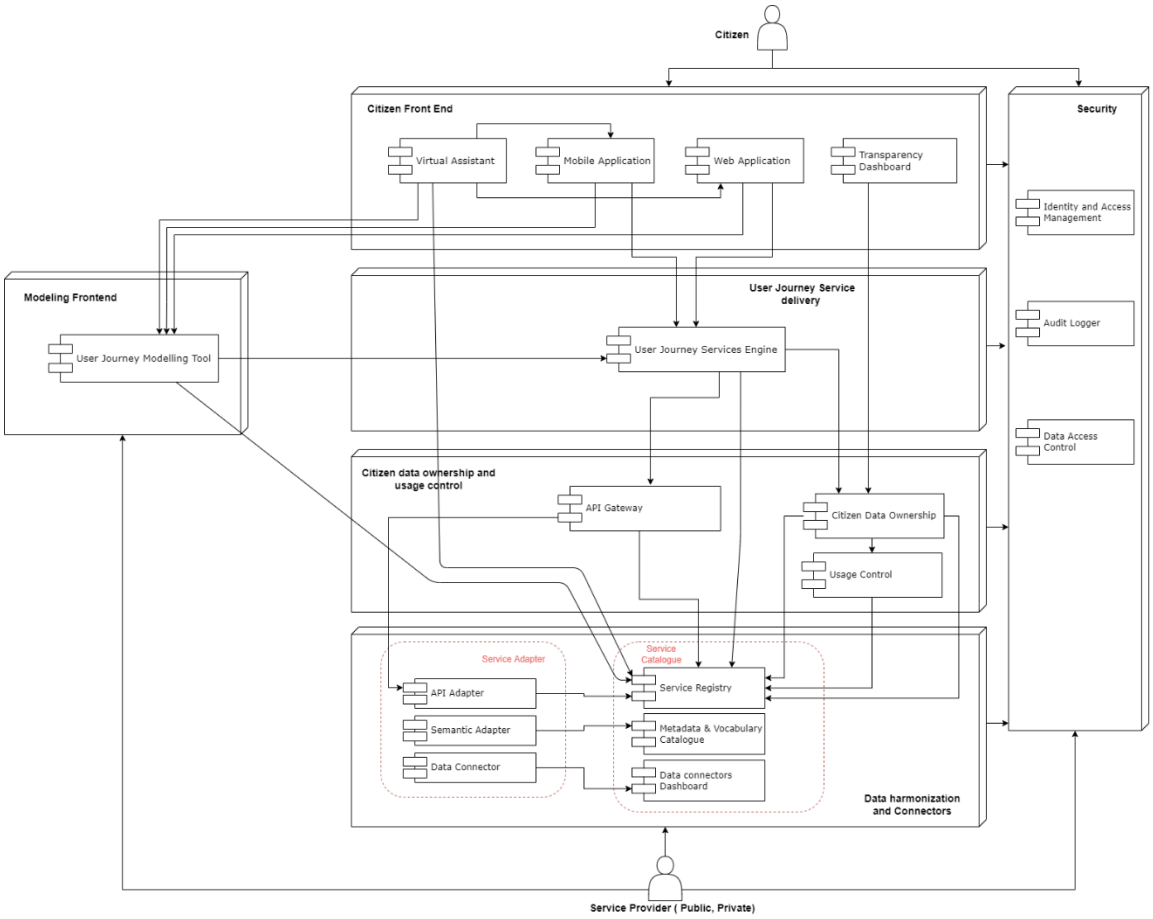


Figure 1: System architecture

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:

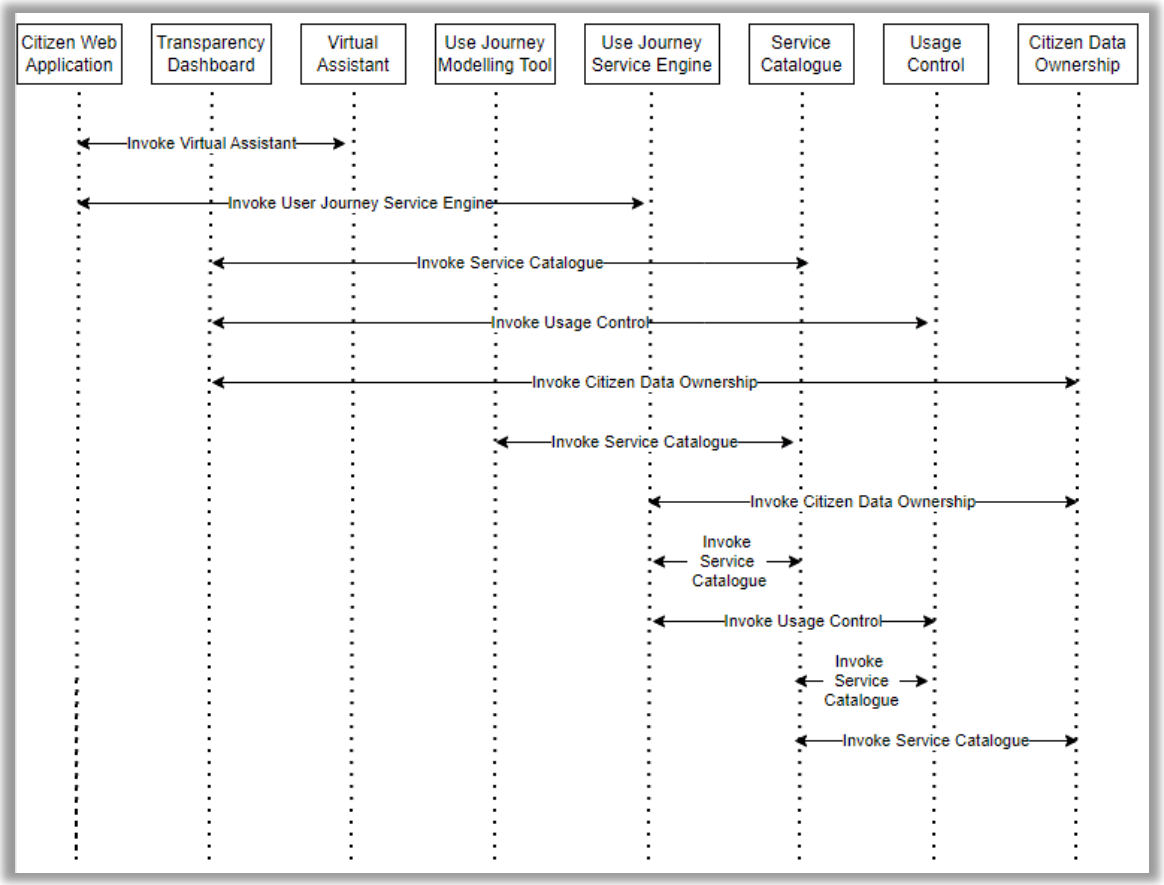


Figure 2: Components Interaction

The following table presents the interactions between the components in detail.

Table 1: Components Interaction

Interface	Description
Invoke Virtual Assistant	During its startup, the citizen web 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 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 Front End 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 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.

Interface	Description
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.

Interface	Description
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 Final version

3.1 Scenario implementation

Main Scenario

The main scenario followed for the final 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 'European Health Insurance Card' 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 (Figure 3).

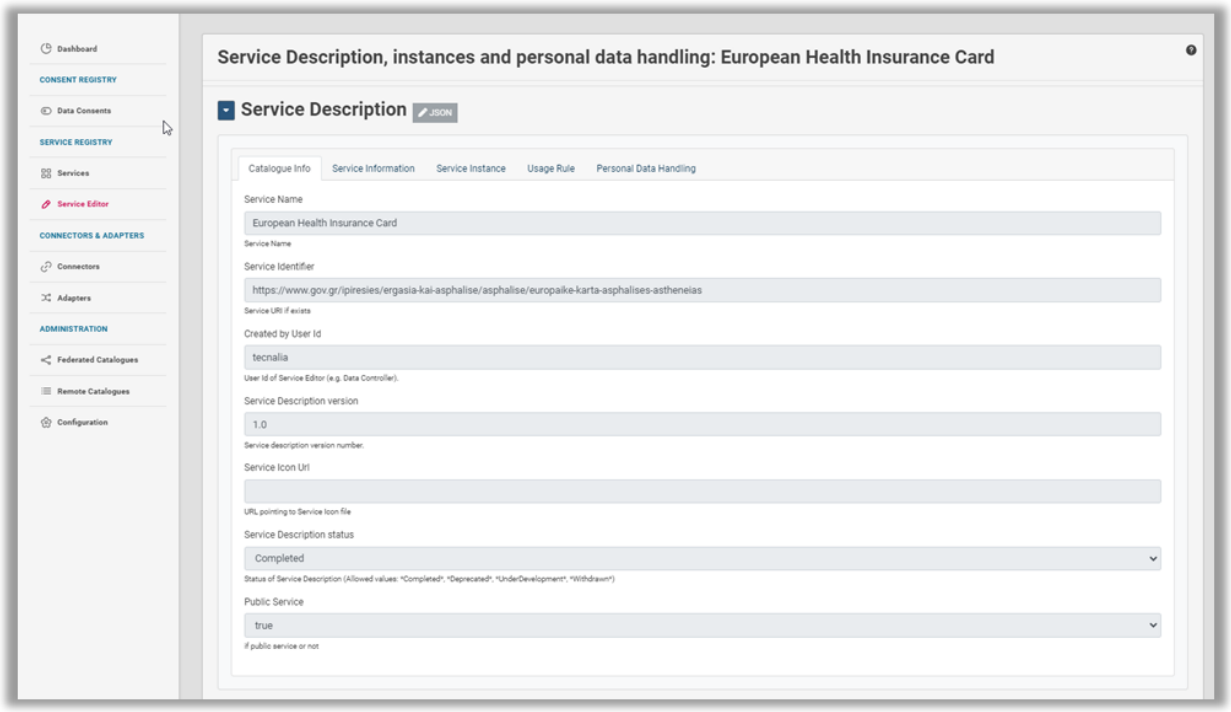


Figure 3: Service information sections included in the service catalogue

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).

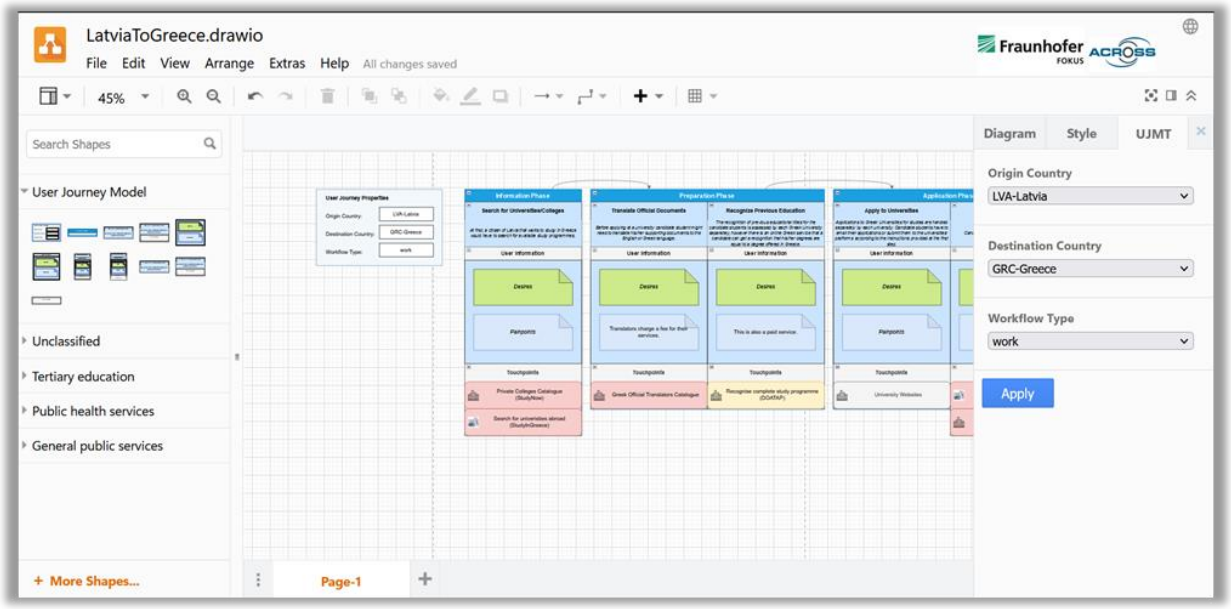


Figure 4: User Interface of the UJMT

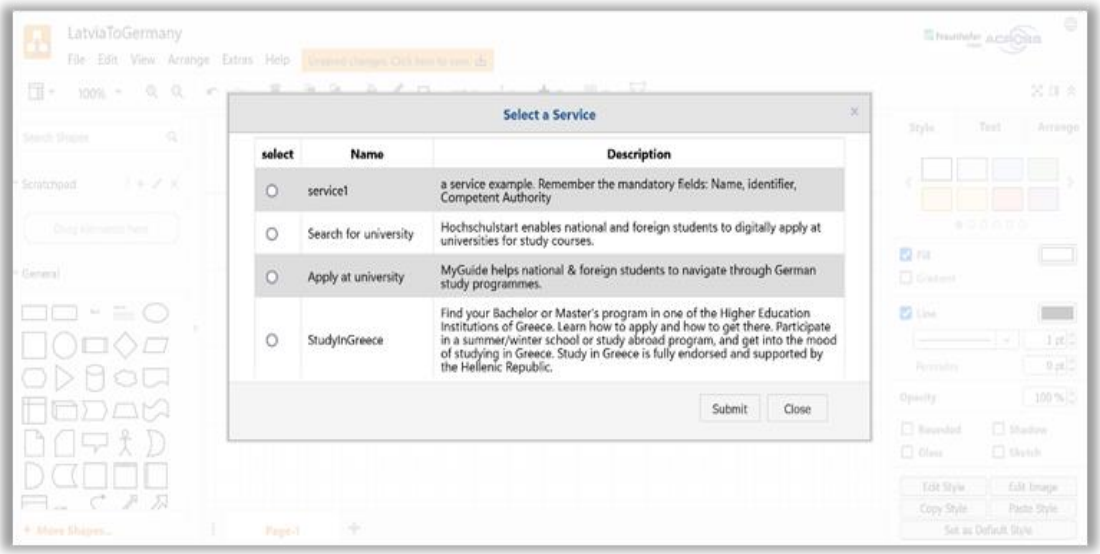


Figure 5: Service Selection for an Action in the UJMT

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.

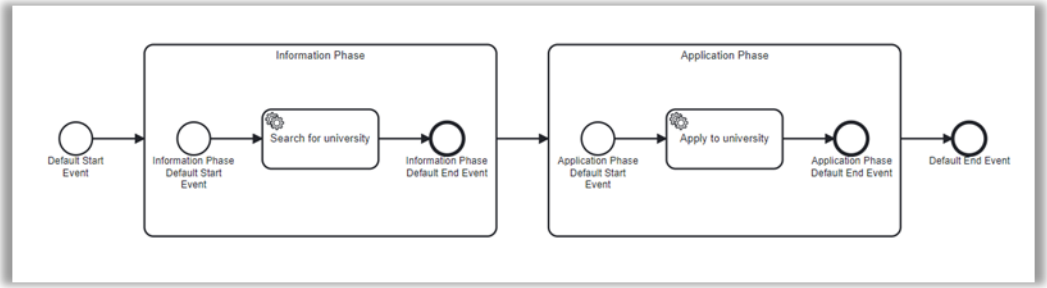


Figure 6: Generated BPMN diagram

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).


```
{
    "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 app and Transparency dashboard) have all the relevant information needed to deliver this journey towards the citizens.

Citizen Interaction

Citizens can visit the Citizen Front End and access useful information about its functionalities directly from the Landing page (Figure 8). The website is verified to be WCAG 2.1 compliant, as confirmed by the AChecker+ tool (check the footer of the Landing page https://citizen-webapp-citizen-application-dev.k8s.across-h2020.eu/login). Additionally, the UserWay accessibility widget has been integrated into the application to facilitate accessibility adaptations, with further details provided later in this document. Moreover, the integration of the Virtual Assistant component supports website exploration through both vocal and written commands in multiple languages. Lastly, it is worth mentioning that the website is multilingual, with support for English, German, Latvian and Greek at all levels and components.

Regarding the Landing page, in the 'What We Offer' section, users can find a high-level overview of what the ACROSS application offers. Here, they can explore:

- The User Journeys which are available in the application, offering tailored experiences and guidance for users in various scenarios.
- The Countries participating in this project, showcasing the European reach and collaboration involved in the ACROSS initiative.
- The available languages of the application, ensuring accessibility and usability for users of the participating countries in addition to English speakers.
- The Services that have been integrated into the existing User Journeys.

Users can access the details of each category by clicking the relevant statistics button, allowing them to dive deeper into the features and offerings of the ACROSS application.

A brief application tutorial is available in the 'Start To Discover' section of the Landing Page, providing users with an overview of how the application functions. Within this section, an interactive menu is also provided, enabling users to explore the available journeys and experience some functionality of the application firsthand. This interactive feature enhances user engagement and facilitates a better understanding of the application's capabilities.

Users' feedback on the application is displayed on the Landing Page, including information about journey details such as the country of origin, the country of destination and the purpose, along with the date of feedback submission.

In the 'Who We Are' section, the ACROSS consortium members are listed, providing information about the collaborative partners involved in the project.

Finally, in the 'Learn More' section, users can access a descriptive video of the ACROSS project, offering a brief overview of its objectives and scope.

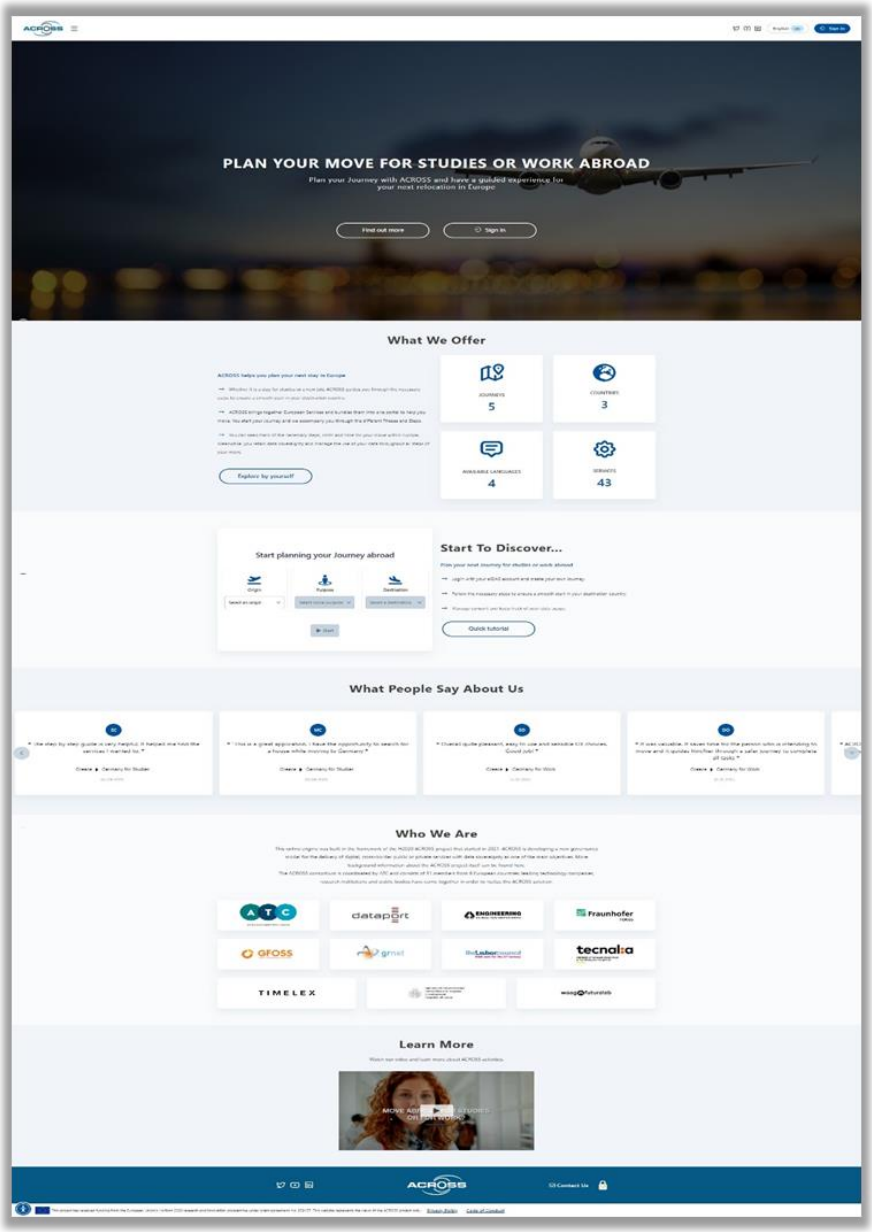


Figure 8: Landing page of ACROSS Citizen Web Application

The user can log into the application by clicking on one of the ‘Sign in’ buttons located either in the top right corner of the screen or on top of the hero image. Alternatively, they can also click on the 'Explore by Yourself' button in the 'What We Offer' section. Upon clicking, the user will be redirected to the Keycloak Sign-In page (Figure 9).

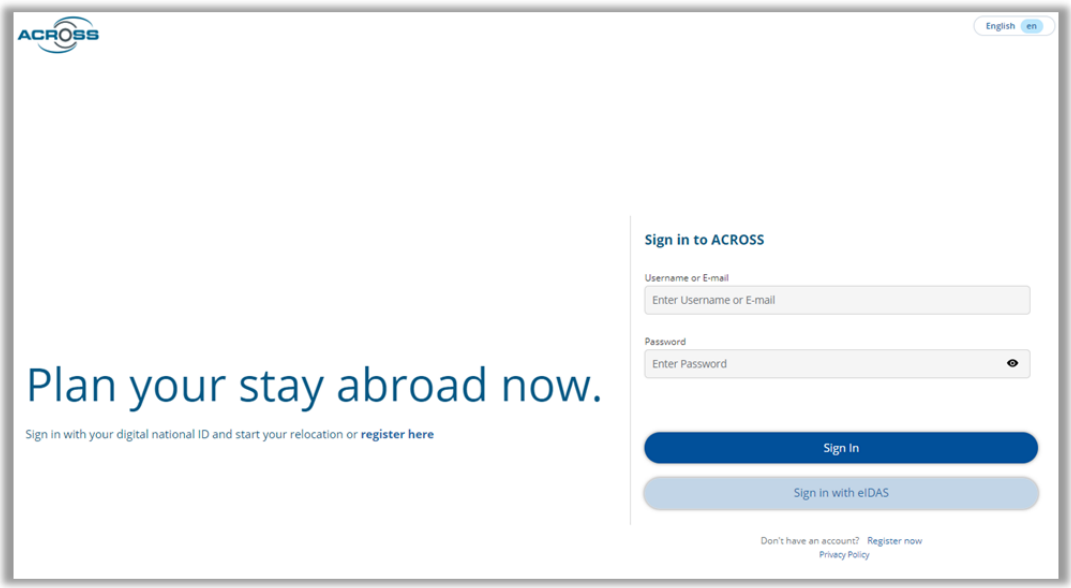


Figure 9: Keycloak Sign-In page.

After logging in, the user will be directed to the Home page (Figure 10). Here, they are provided with a quick application tutorial and presented with options to either start a new User Journey or view their previously initiated Journeys.

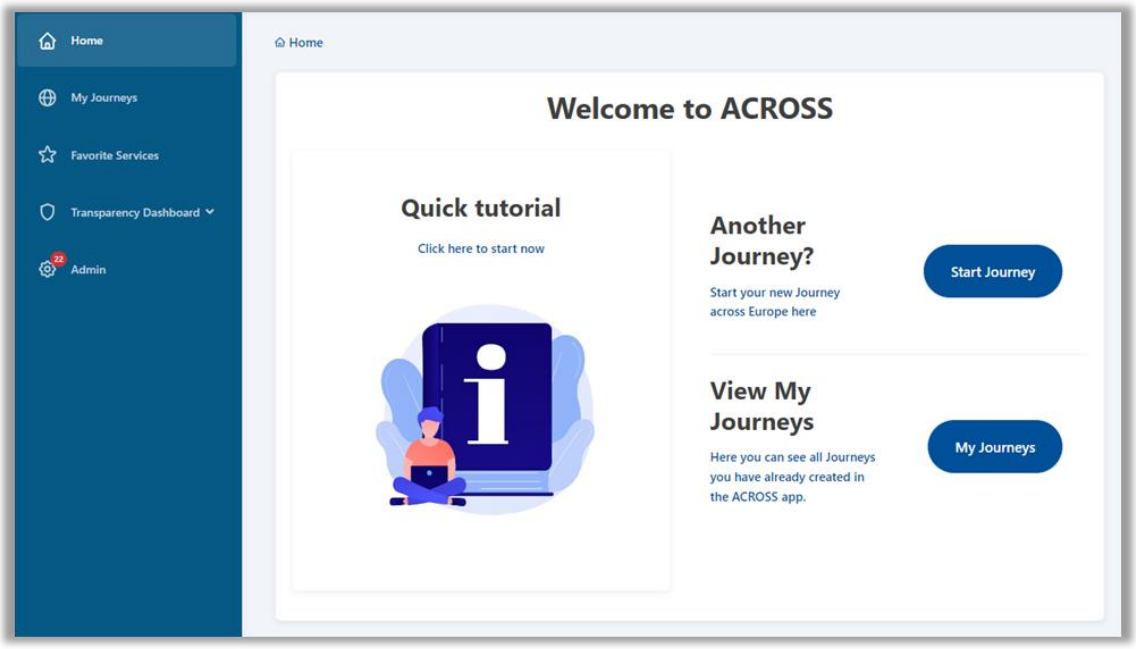


Figure 10: ACROSS Home page.

The application's quick tutorial will guide the user through the steps necessary to start and complete a Journey. Each step of the guide contains a screenshot and a brief description to assist users in understanding the concepts of the application's workflow, as shown in Figure 11.

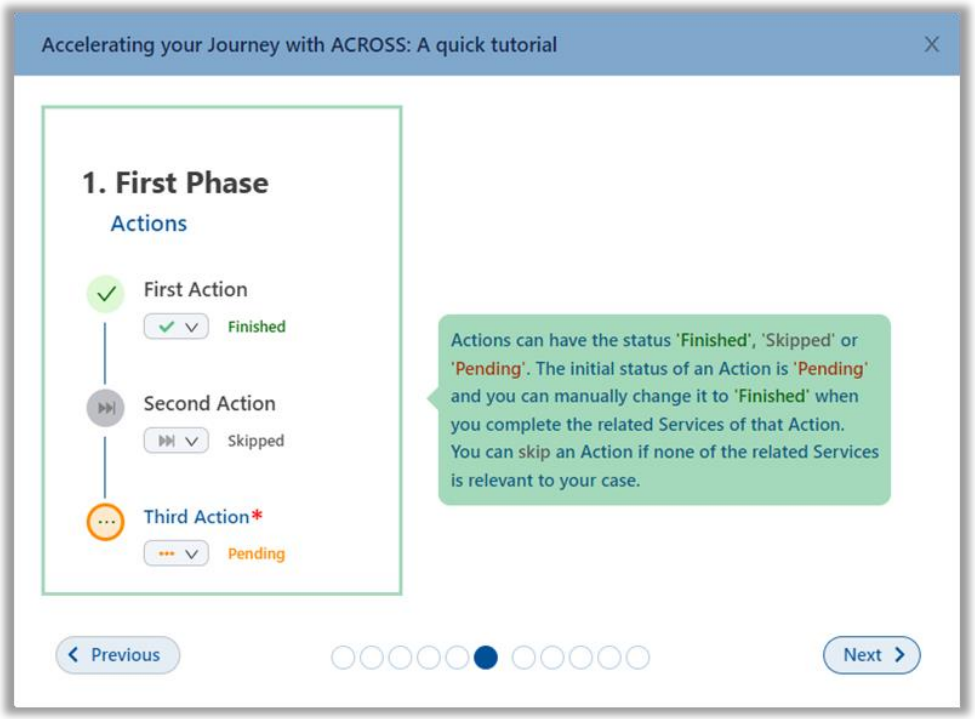


Figure 11: Step example of quick tutorial.

By clicking on the 'Start Journey' button, citizens can initiate the journey of their choice and proceed by selecting the journey details from the journey creation menu that appears on the screen. (Figure 12). Only one journey with a specific combination of parameters (i.e., 'Origin', 'Purpose', and 'Destination') is permitted to be initiated by the user at a time. If the user attempts to start a journey identical to one that is already running, a warning message will inform them about the necessary actions to be taken, as illustrated in Figure 13.

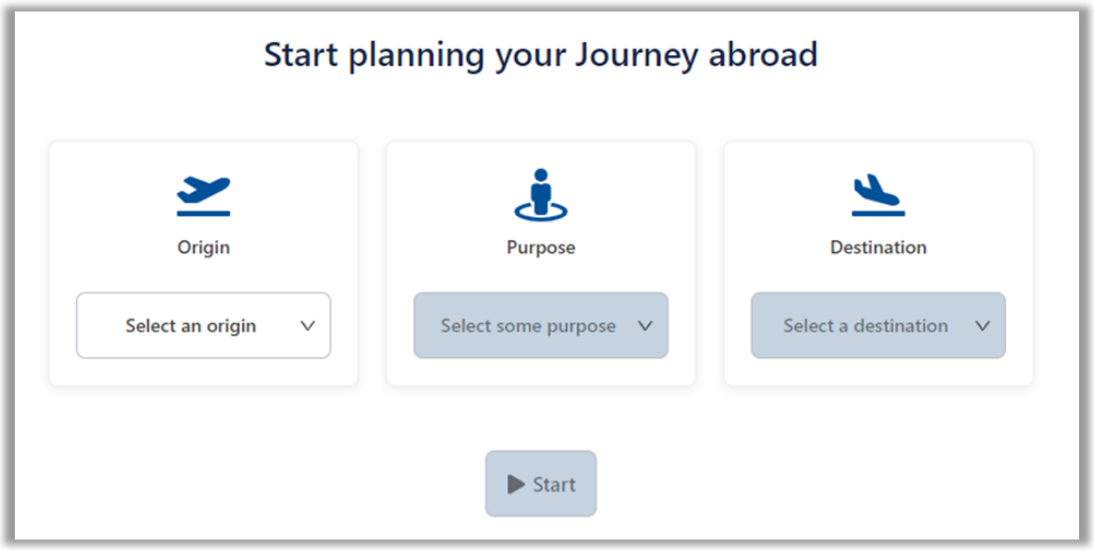


Figure 12: Create Journey menu.

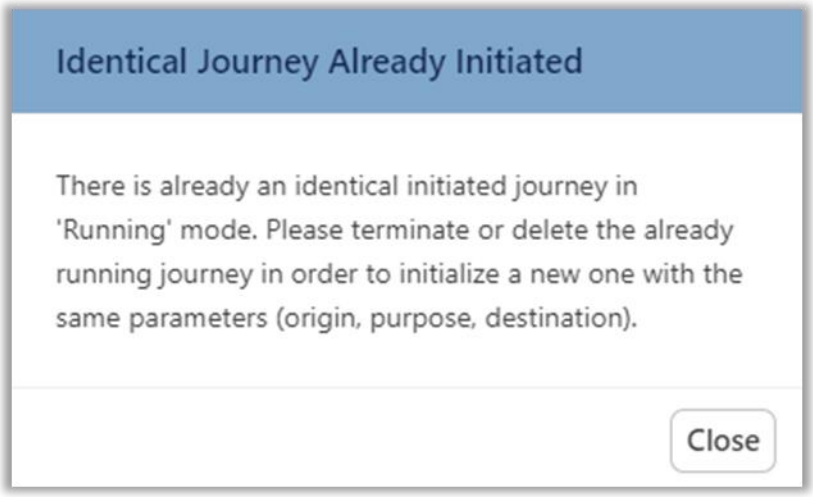


Figure 13: Warning message for identical initiated journey.

Conversely, upon clicking the ‘My Journeys’ button on the homepage, users are redirected to the corresponding page where a list of previously initiated Journeys, categorized as either "Running" or "Finished", is prominently displayed (refer to Figure 14). Each item on this list comprises a journey card (Figure 14, Figure 15), featuring primary characteristics of the Journey such as ‘Purpose’, indicated by icons like a student icon for ‘Studies’ and a person with a gear icon for ‘Work’. The ‘Country of Origin’ is displayed below the purpose icon, or below the journey’s friendly name, as depicted in Figure 15, while the ‘Country of Destination’ is situated below the airplane icon. Additionally, each list item includes a journey description, along with a colored

overview indicating the status of 'Total', 'Pending', 'Finished', and 'Skipped' Actions, alongside the creation date of the Journey.

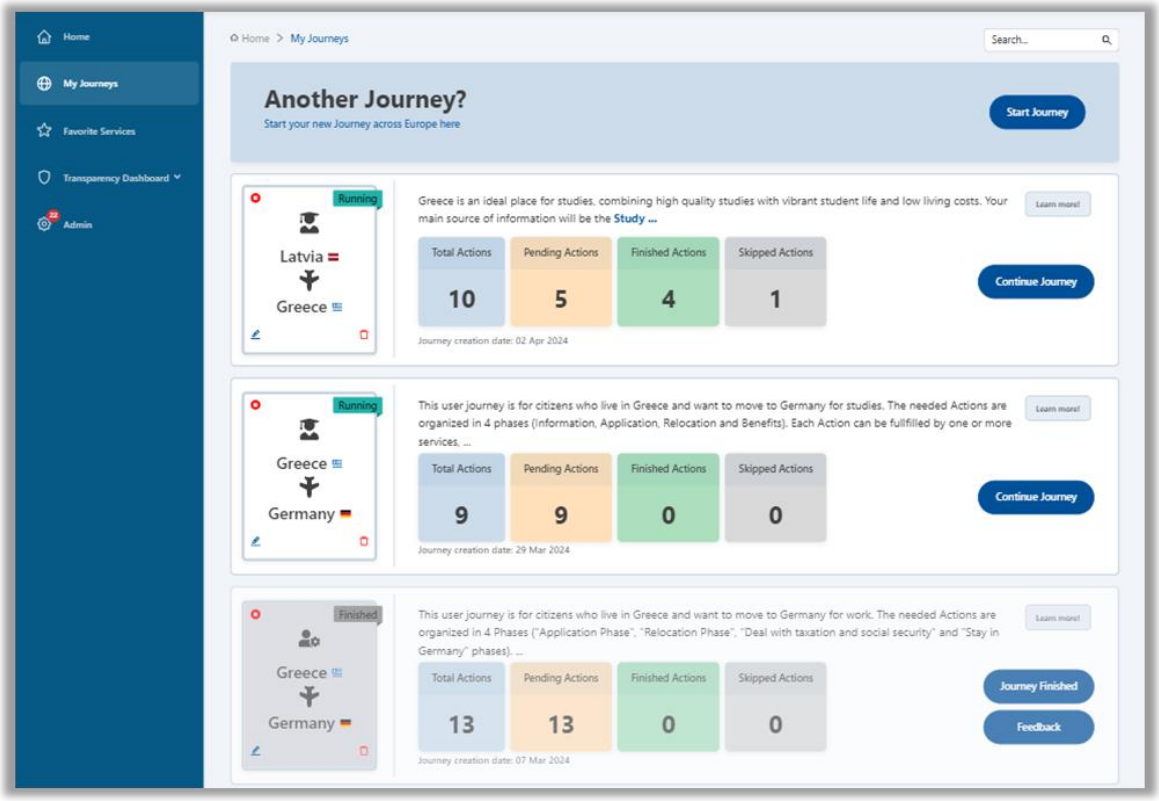


Figure 14: My Journeys page listing the user's initiated Journeys.

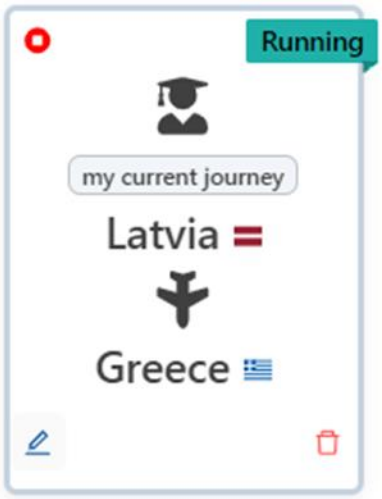


Figure 15: Journey card.

Users have the option to personalize their journey by renaming it with a friendlier name (e.g. 'my current journey' as depicted in Figure 15). By clicking on the pencil icon located at the bottom left corner of the card, users can easily edit the journey's name. This feature enables users to conveniently search for specific journeys using the search bar located at the top right corner of the 'My Journeys' page. Additionally, users can utilize the origin or destination of the journey as search terms to filter the displayed journeys on this page.

Furthermore, a red 'stop' button is available at the top left of the Journey card, enabling users to terminate the relevant Journey if needed. Should a user wish to delete a journey entirely, they can do so by clicking on the trashcan icon button positioned at the bottom right corner of the card.

Upon clicking on the journey card or the 'Continue Journey' button (Figure 14), users are seamlessly redirected to the 'My Journey' overview page (Figure 16). Here, they can embark on navigating through the steps necessary to finalize the journey and prepare for moving to their chosen destination. Users will also be automatically redirected to this page immediately after successfully creating the journey of their choice. In this scenario, a quick journey guide is prompted, offering users the option to either view it or skip it before proceeding to navigate through the journey steps. This guide is readily accessible at all times, allowing users to initiate it again by clicking on the info ('i') button situated next to the Journey's title at the top of the page.

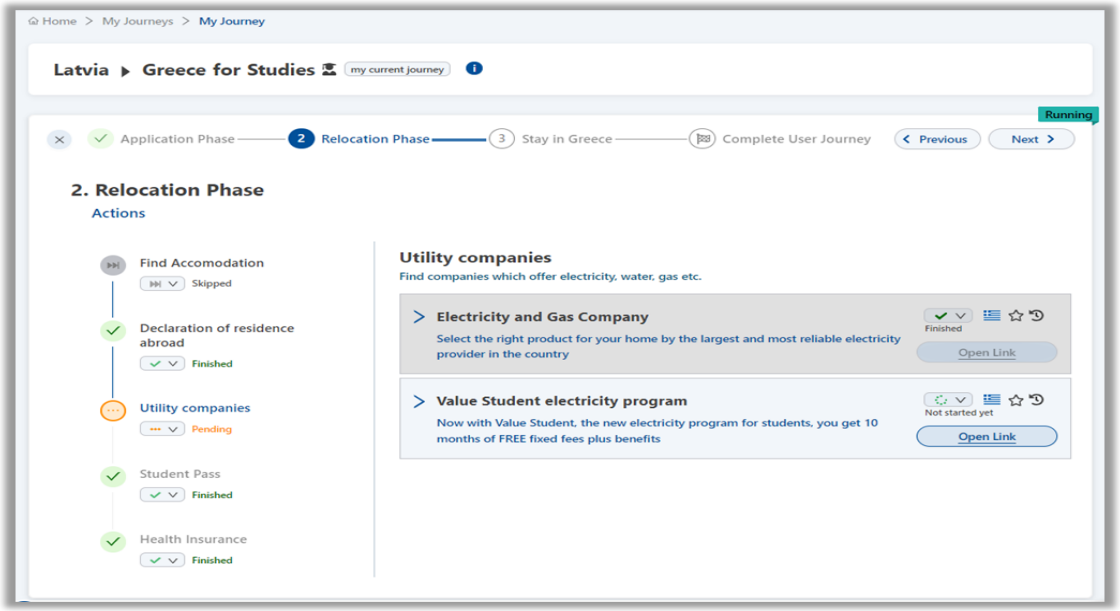


Figure 16: Journey overview page.

A User Journey comprises several Phases that the user must navigate through. Each Phase includes a series of Actions that need to be completed. To view the Actions included in a Phase, users can either click on the Phase name (or number) or navigate using the next/previous buttons of the wizard. An Action, such as 'Utility companies', is defined as a group of Services, which may include 'Electricity and Gas Company' and 'Value Student electricity program', for example. To view the Services included in an Action, users simply need to click on the Action name. In Figure 16, the journey overview page is depicted, where the user can access all the information required to complete the necessary steps of the journey.

Upon clicking on a Service, various metadata are displayed organized in tabs, as illustrated in Figure 17 . The 'Service details' tab provides information such as the service points, digital delivery points, estimated cost and time needed to complete the service, etc. In the 'Basic Info' tab, users can find details about the type of service (public or private) and the country offering the service. The 'Supporting Documents' tab lists possible supporting documents needed to complete the service, while the 'Contact Information' tab contains details about the website, email address, telephone and fax numbers, as well as the opening hours and availability hours of the Service, when available.

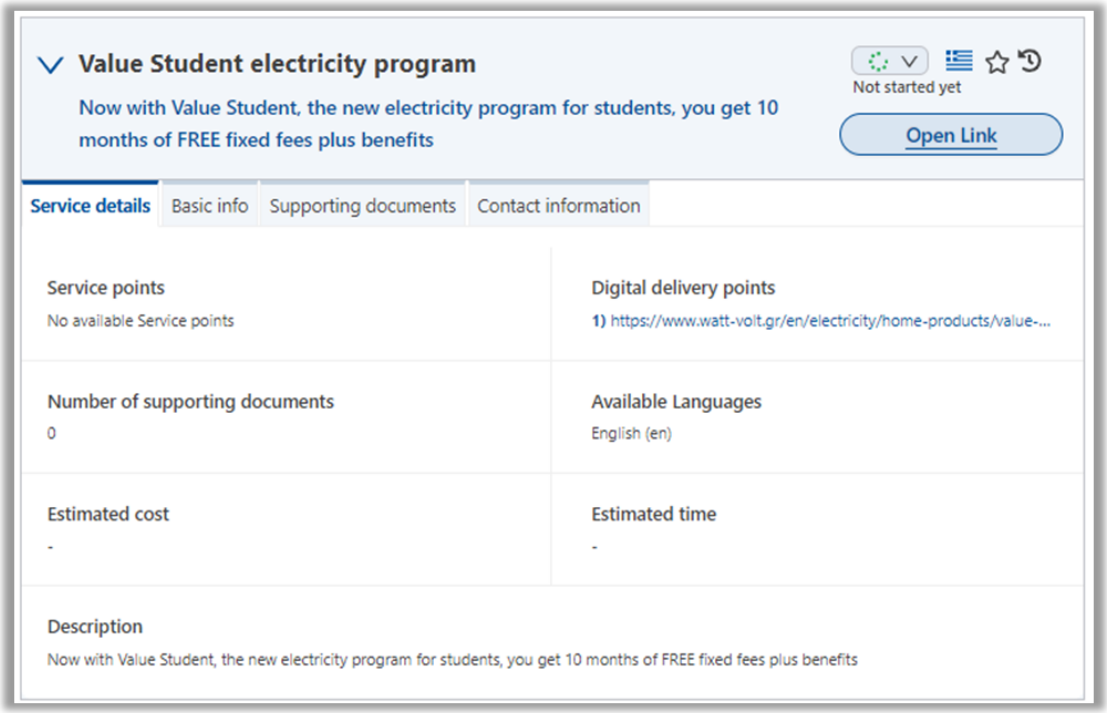


Figure 17: Service Metadata tabs.

As depicted in Figure 18, the user may manually update the status of an Action or a Service to keep track of the progress made so far, using the relevant dropdown menus. The available statuses for Actions (steps) are as follows:

a. Pending: There are still services that have not been completed yet.
b. Skipped: The step is considered unnecessary by the user.
c. Finished: The Action has been completed; all necessary services of this step have been concluded.

Similarly, the available statuses for Services are:

a. <empty>: No action has been taken yet by the user for this service.
b. Submitted: The application form for 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 for this service has been successfully processed by the service provider.

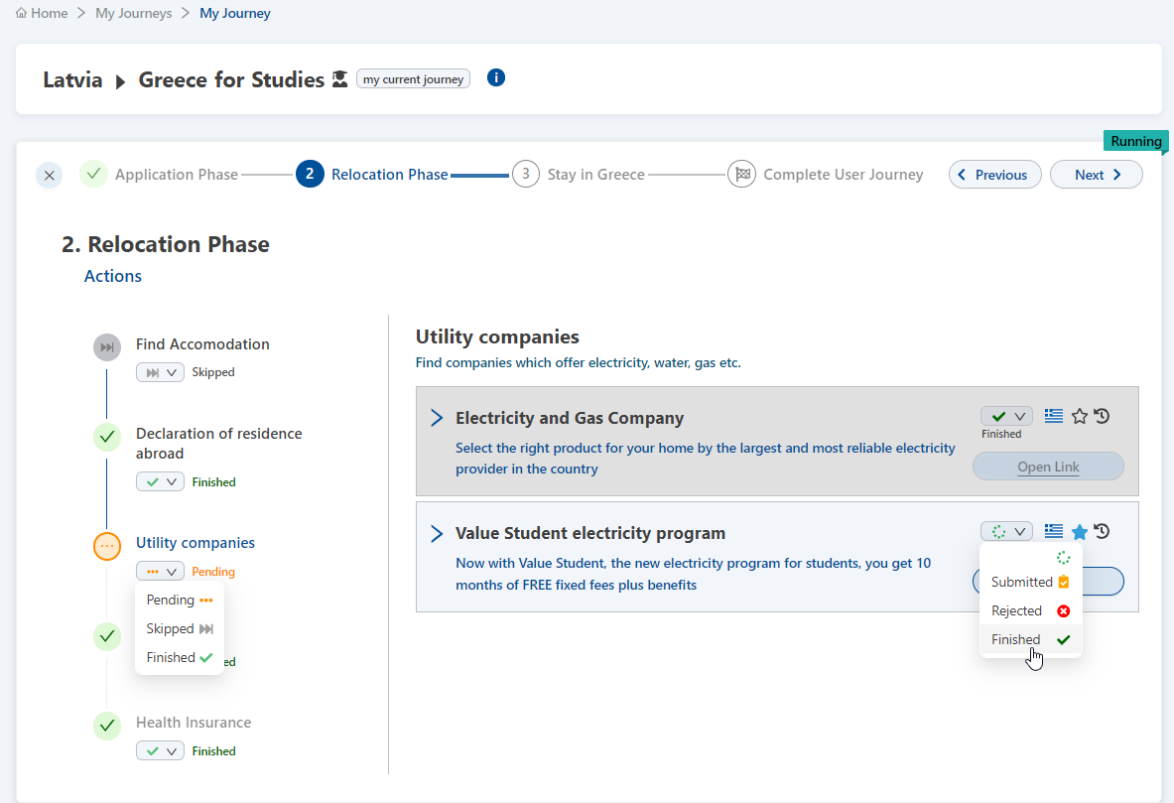


Figure 18: Action & Service status update dropdown menus / greyed out Service/ completed Phase.

For asynchronous REST Services, the service provider can automatically update the status of the service. For instance, the status will transition to 'Finished' once the submitted form has been successfully processed. Even if the user is not logged in to the application when the service provider updates the status, they will still observe this status update the next time they access the Citizen Web Application. A notification will also be seen in the notification panel on top of the screen, informing the user about the status update. Conversely, in the case of synchronous REST Services, the status of the service will be promptly updated based on the service provider's response.

If the status of a Service is set to 'Rejected' or 'Finished', the relevant Service will be greyed out (see Figure 18, service ‘Electricity and Gas Company’), but its metadata will still be accessible. However, in the case of a REST Service where an application form is provided via ACROSS, the application form itself will not be accessible in such cases.

Furthermore, if the status of all Actions within a Phase is marked as 'Finished' or 'Skipped', the Phase itself will be adorned with a green 'tick', indicating completion and signaling that no Actions remain unfinished within that Phase (see Application Phase in Figure 18).

Beside the status dropdown menu of a Service, users can find the flag of the country offering the service, with the name of the country displayed on hovering over the flag icon. Adjacent to the country flag, a star button allows users to mark a service as 'Favorite' (the icon changes to a light blue color when selected - Figure 18, ‘Value Student Electricity program’ Service). If the Service is already marked as 'Favorite', clicking on the button will remove the Service from the 'Favorite Services' page. Additionally, a clock icon positioned to the right side of the star icon, upon clicking, triggers a popup containing information about the status update history of the service (refer to Figure 19).

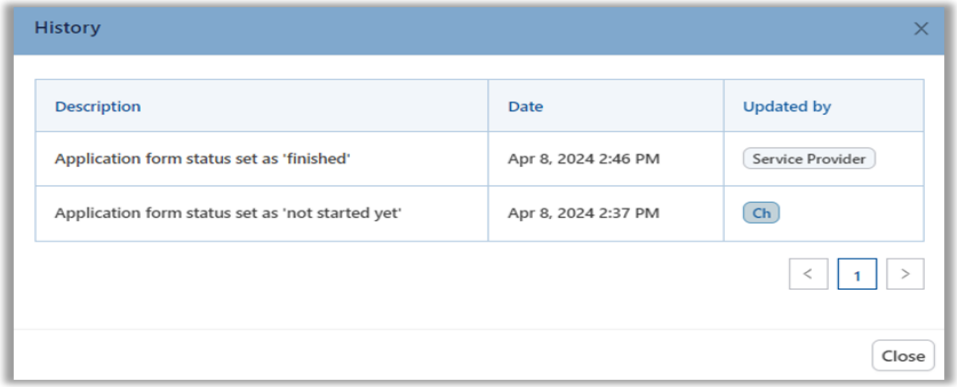


Figure 19: Service status History table

If the Service can be invoked via the ACROSS Application (REST Service), users will find an 'Apply' button available (refer to Figure 21). However, if the Service cannot be invoked via the ACROSS Application but is digitally available, an 'Open Link' button will be provided instead. Clicking on this button will redirect the user to the Service website. If the Service is not digitally available at all, the Service Metadata will provide all the necessary information to access the Service.

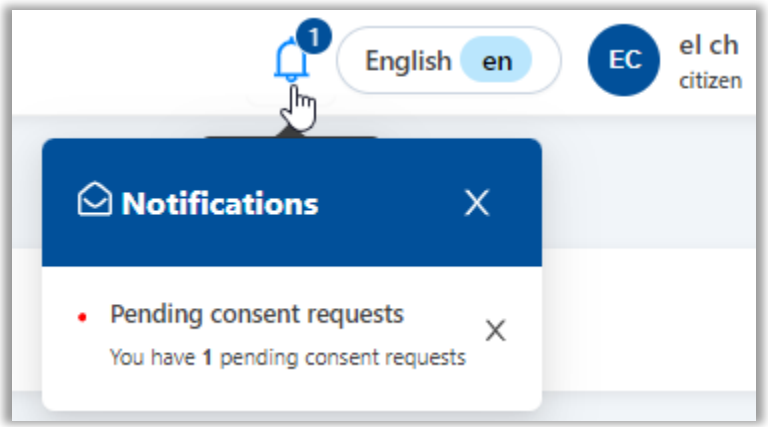


Figure 20: Pending consent notification.

In instances where Services accessible though the ACROSS Application require user consent for data requested within the service application form, users will get a notification (Figure 20) and also notice a distinctive yellow shield icon adorned with a diagonal strike-through line displayed alongside the Service name (Figure 21).

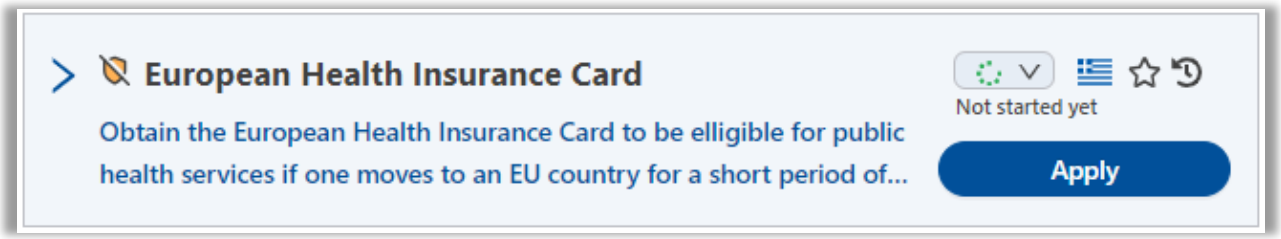


Figure 21: REST Service requiring policies consent.

Clicking on the notification will redirect users to the Transparency Dashboard page, where they can review any missing consents required to proceed with the relevant service application and manage their consents accordingly.

Upon clicking on the Service 'Apply' button, a new popup window emerges (Figure 22), offering users the option to grant consent for all data (mandatory and optional) or solely for mandatory data. The lists detailing mandatory and optional data are conveniently presented below toggle switches, simplifying the consent process. Alternatively, users have the choice to navigate to the Transparency Dashboard (Figure 23), for more comprehensive consent management options. Granting consent for these policies is imperative for the services to be invoked.

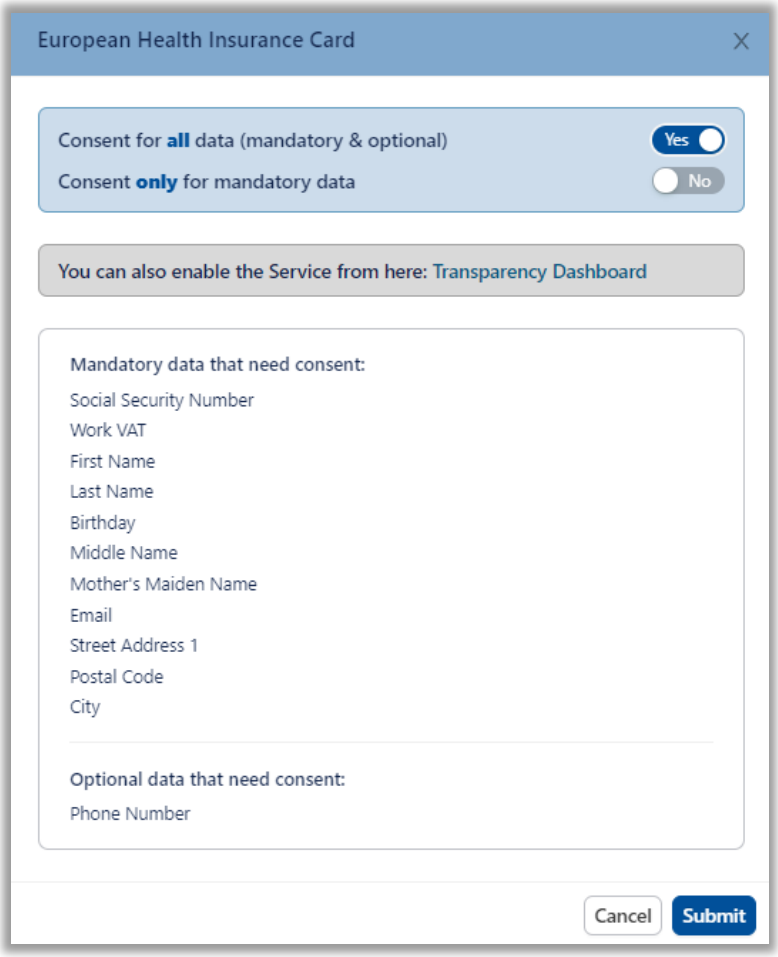


Figure 22: Data consent pop-up window.

At any time, users can manage the consent for all services through the Transparency Dashboard by simply clicking on the relevant section located in the left-side menu of the ACROSS Application.

In addition to granting or denying consent for various service data, users may also withdraw their consent at any time, ensuring their data is no longer utilized by the corresponding service(s).

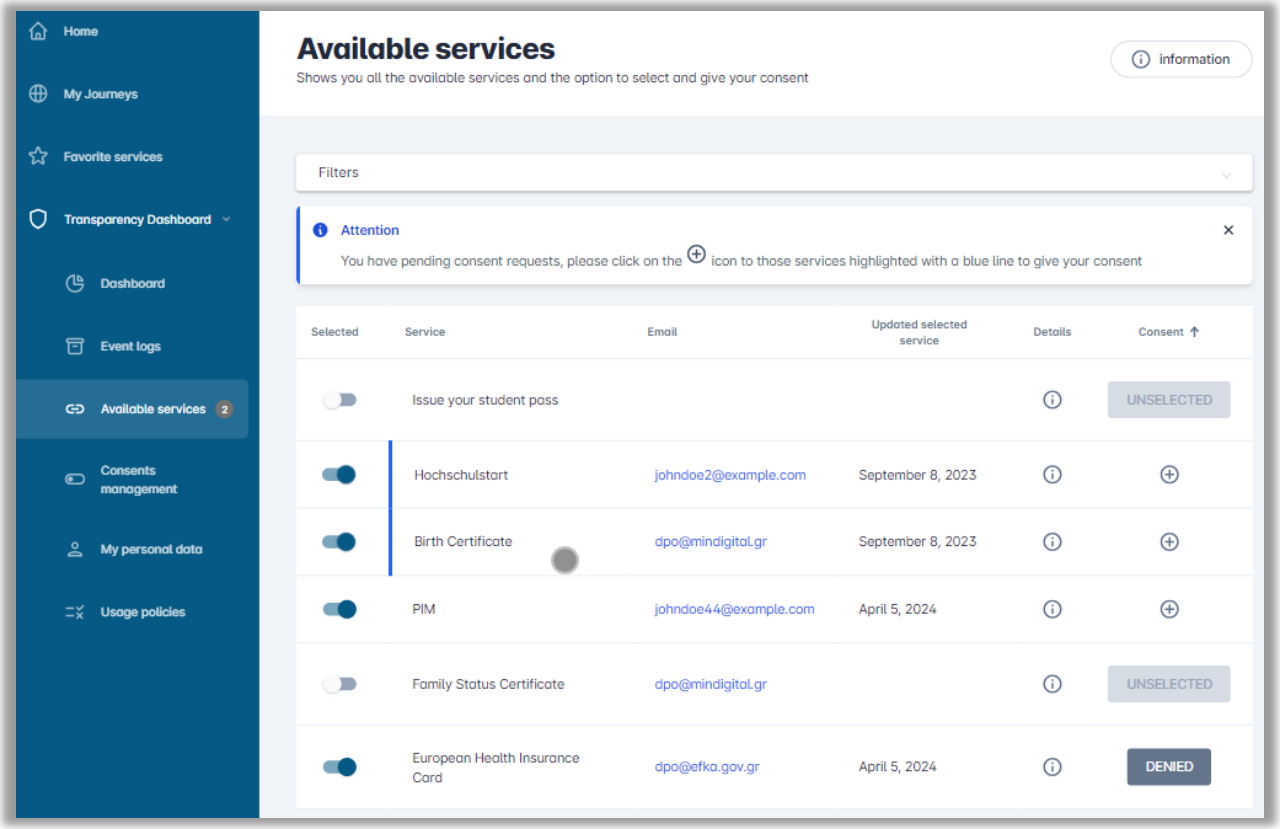


Figure 23: Transparency Dashboard consent management.

After granting consent via the relevant pop-up, the service application form will appear on the screen (refer to Figure 24). The same process occurs in cases where consent has already been granted for a service, or no consent is required at all, and users click on the 'Apply' button. Users can then fill in the application form and press the 'Submit' button to invoke the Service by transmitting the relevant data to the Service Provider. User details such as the first or last name, which users have defined during registration, are prefilled and non-editable. The Service Provider will process the application data and respond to the user synchronously or asynchronously, as described earlier.

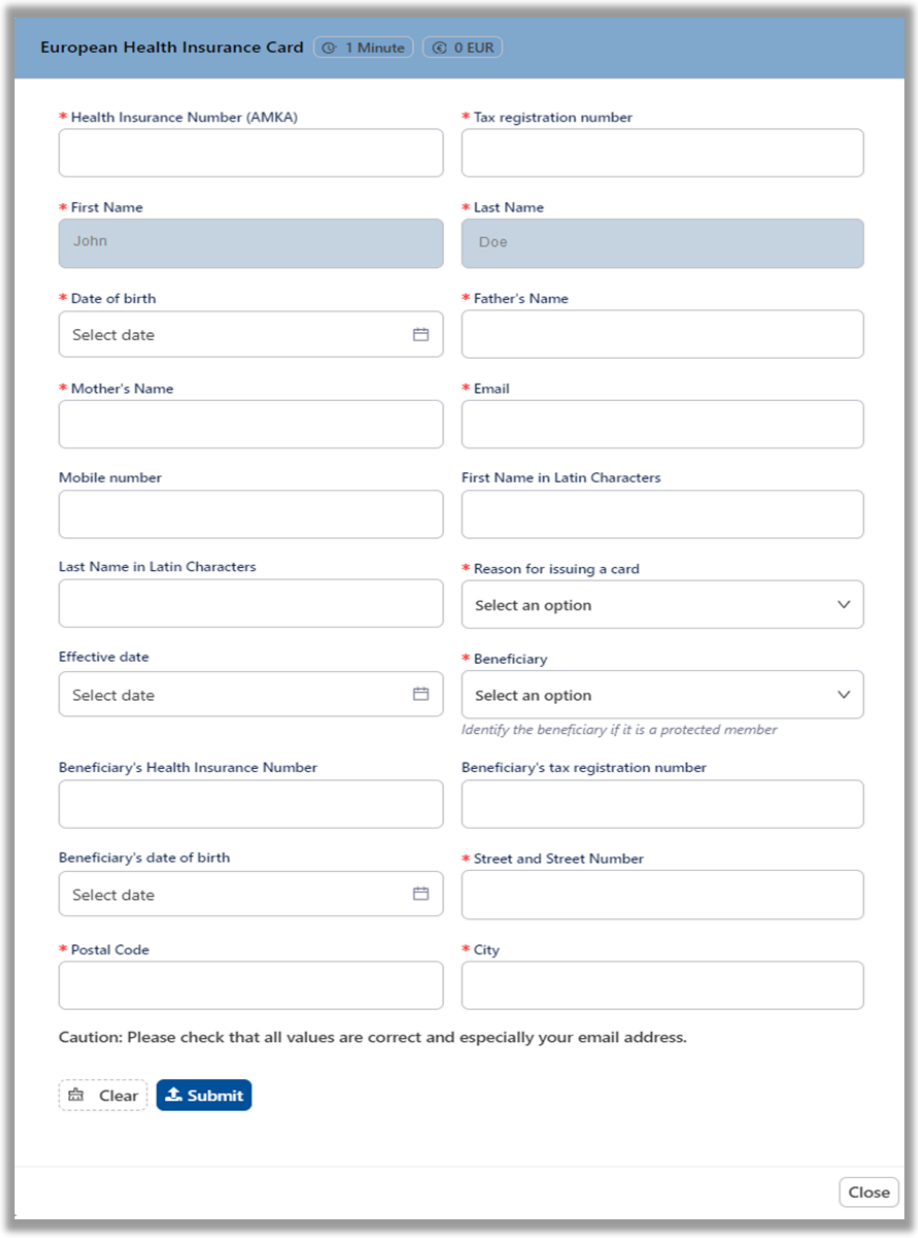


Figure 24: Service application form example

If the service is synchronous and the provider generates one or more files after processing the application form, users will be prompted to download these files. They can choose to download each file individually by clicking on the relevant buttons displayed in the popup window or download all files together in a zip file by clicking the 'Download Response' button (Figure 25). Users are clearly informed that they must save the response before closing the popup, as the ACROSS application does not store any personal data. If the response files are not downloaded,

users will need to resubmit the application form. The status of the service will also be updated as described above.

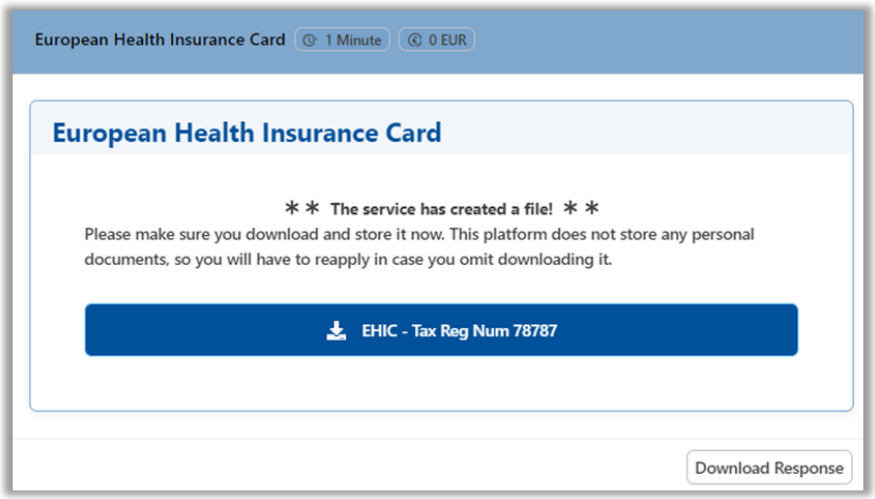


Figure 25: Service provider response including files.

In cases of asynchronous services where the service provider's response includes one or more files, users will receive notification through the ACROSS platform, as depicted in Figure 26. A numbered badge will appear atop the notifications icon (bell) to alert users of any service status updates. Clicking on the bell icon will provide detailed information about the status of the service and the response of the service provider, if any. If there is a response, this will be provided as a URL, ensuring that no user data is stored in the ACROSS database in this scenario.

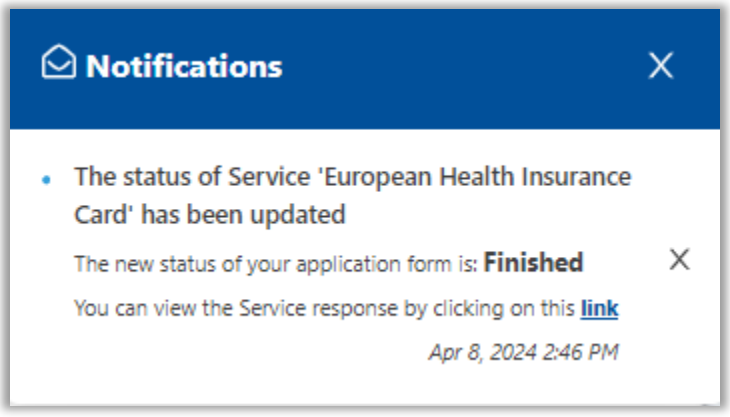


Figure 26: Asynchronous Service status update and response URL.

Aside from the notification, another button appears below the service description, as shown in Figure 27, when there is an asynchronous service response. Clicking this button redirects users to the response URL, while the status of the service is automatically updated accordingly.

Figure 27: Asynchronous Service response button.

To access the last Phase of the Journey, i.e. the 'Complete User Journey' Phase, users will have to complete all previous phases and click on the 'Finish' button on the top right while on the next to last Phase of the Journey (refer to Figure 28).

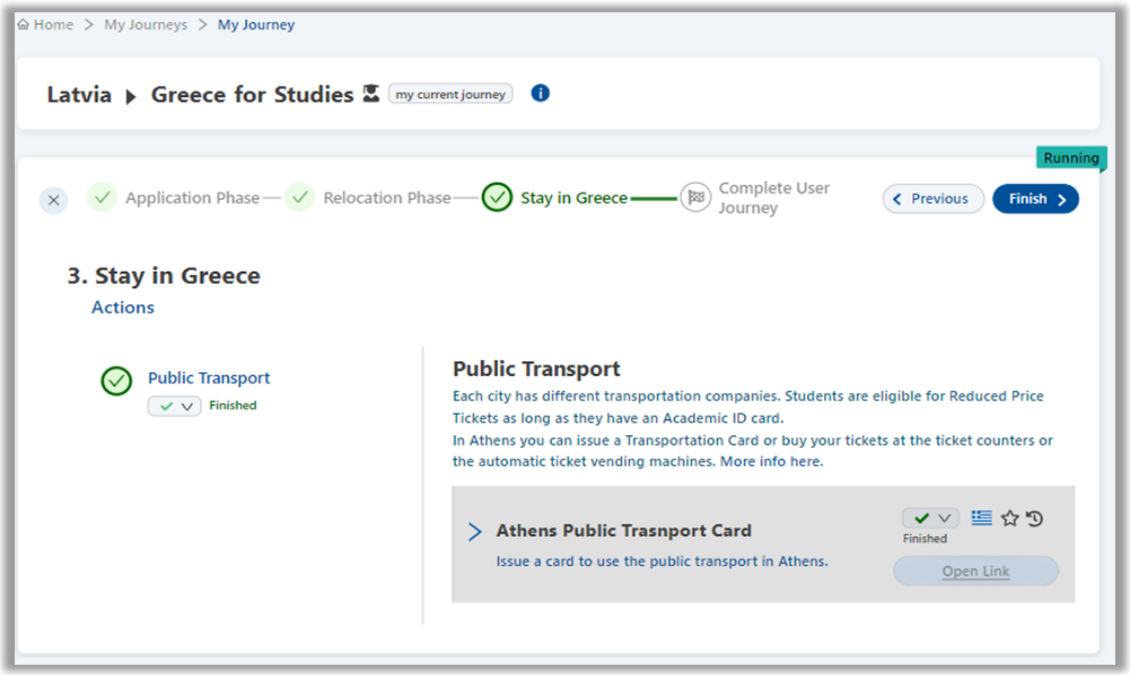


Figure 28: Finish button shown on top right when next to last Phase.

A confirmation dialog will appear (Figure 29), informing users that terminating a journey means it cannot be restarted. However, users can initiate a new journey with the same parameters (origin, destination, and purpose) at any time, provided there is no other 'Running' Journey with identical parameters.

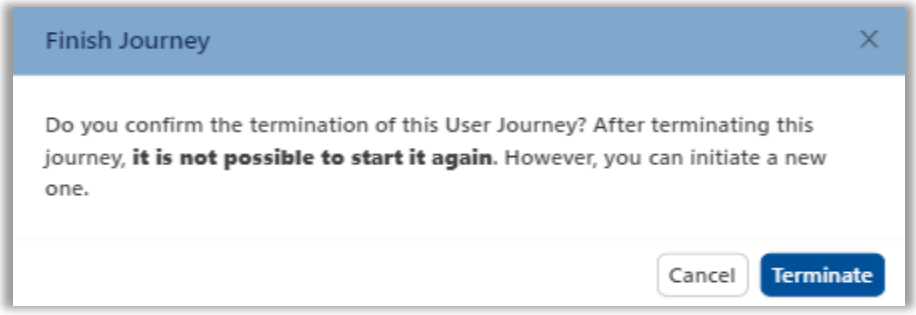


Figure 29: Confirmation dialog for User Journey termination.

After terminating the journey, either as described above or by clicking the red 'stop' button on the Journey card while on the 'My Journeys' page, users will be prompted to provide feedback about their experience on that specific journey. A 'Share Experience' button will appear on the last Phase of the journey (Figure 30). Clicking this button will trigger a popup window with the journey parameters prefilled and non-editable (Figure 31). The feedback provided will be displayed on the Landing Page after it has been reviewed by the administrators of the ACROSS Citizen Web Application.

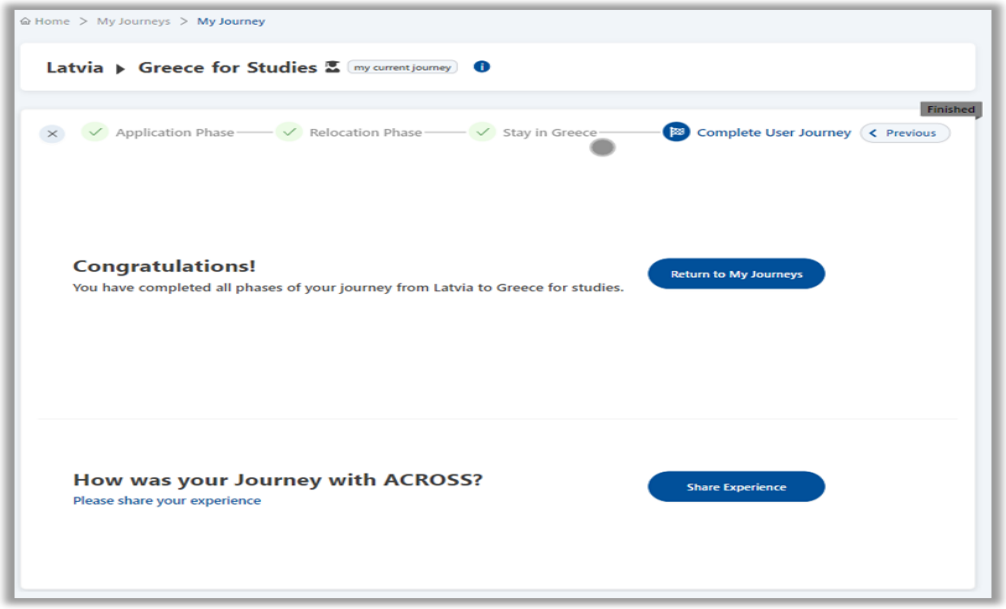


Figure 30: Complete User Journey Phase.

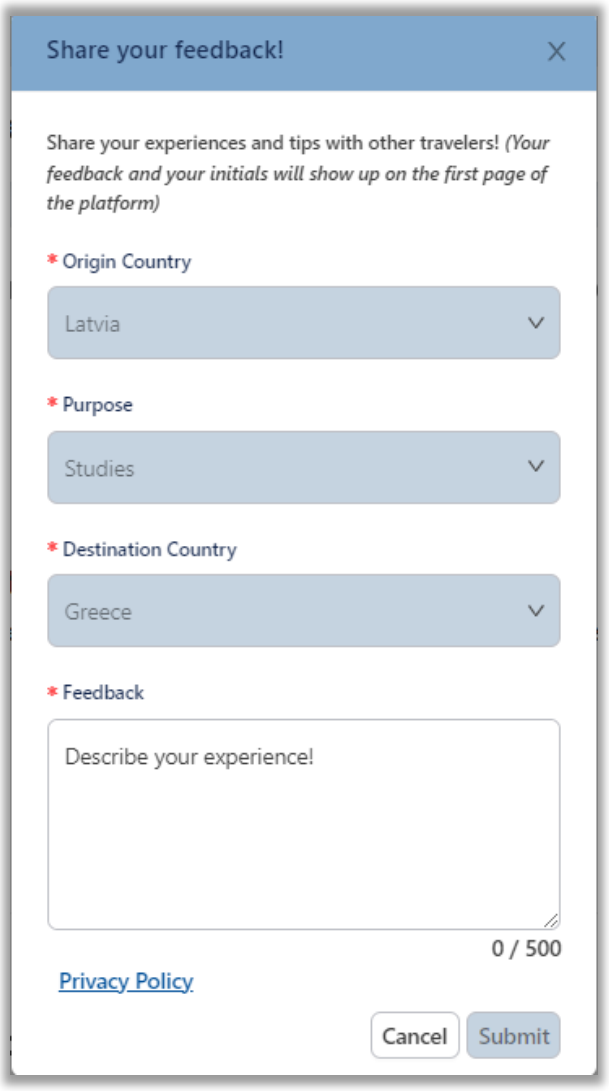


Figure 31: Feedback dialog.

Users can access a list of their favorite services on the 'Favorite Services' page, accessible via the left-side menu of the platform (Figure 32). Here, users can view their favorite Services organized by initiated journey. The Phase and Action where the Service belongs are provided below the Service description. Clicking on them transfers users to the relevant spot on the User Journey overview page. Additionally, users can update the Service status, check the status history table, view the offering country, or apply directly for the Service. Clicking on the star icon button, will remove the Service from this list.

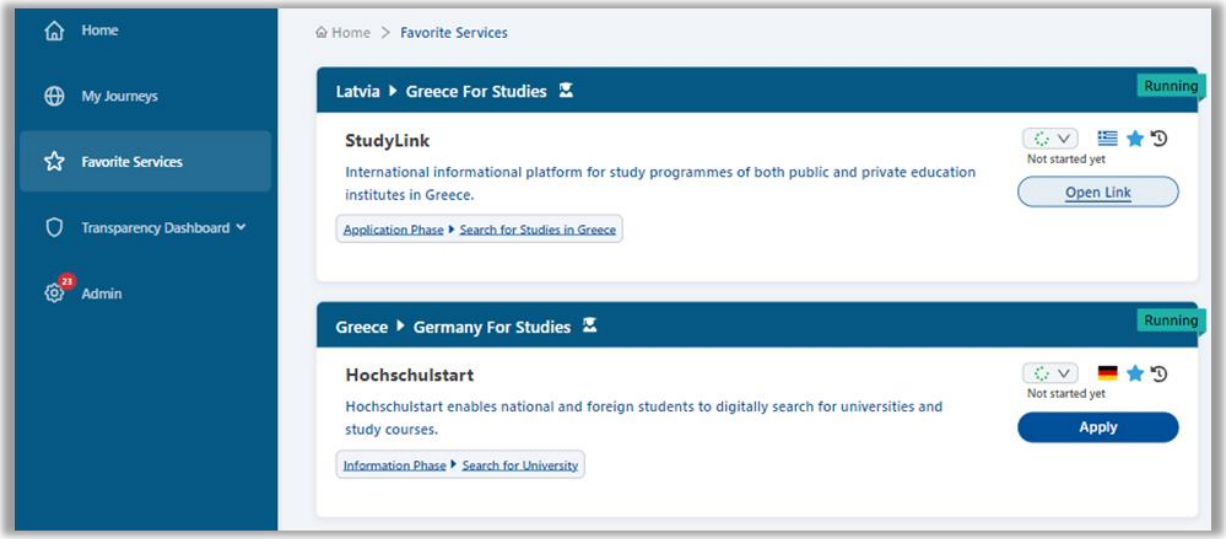


Figure 32: Favorite Services page.

Administrator users of the platform can oversee and regulate the feedback displayed on the Landing Page through the Admin page (Figure 33). When new feedback items await review, a numbered badge appears above the gear icon in the left-side menu's Admin submenu. Within the 'Pending User Stories' tab, administrators can either approve or reject the feedback. All feedback items, regardless of approval status, are cataloged in the 'All User Stories' tab.

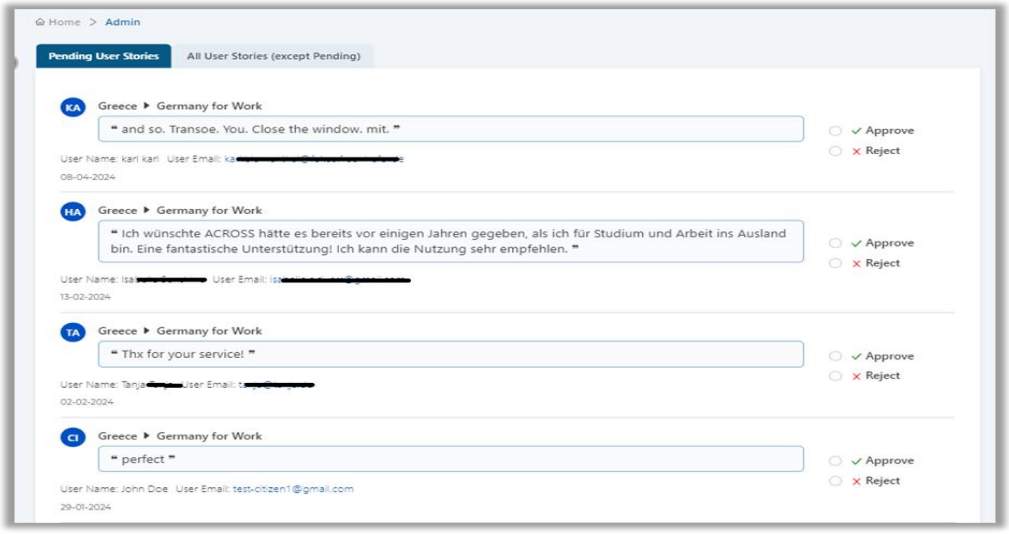


Figure 33: Admin page.

As mentioned earlier, an accessibility widget is conveniently accessible within the application, located at the bottom left of the screen as a blue and white icon featuring a human figure. Upon clicking this icon, users can access the UserWay accessibility widget menu (Figure 34), a comprehensive tool designed to enhance inclusivity and usability across the platform. Its presence signifies the commitment to ensuring equal access to the application for all users, including those with disabilities. Through this widget, users gain access to a variety of accessibility features and settings tailored to their individual needs, fostering a more inclusive online experience. For example, users can adjust text size, contrast and spacing, choose dyslexia-friendly fonts, enable link highlighting, customize cursor size, and more. These features empower users to customize their browsing experience according to their specific accessibility needs, ensuring that everyone can access and engage with the platform comfortably and effectively.

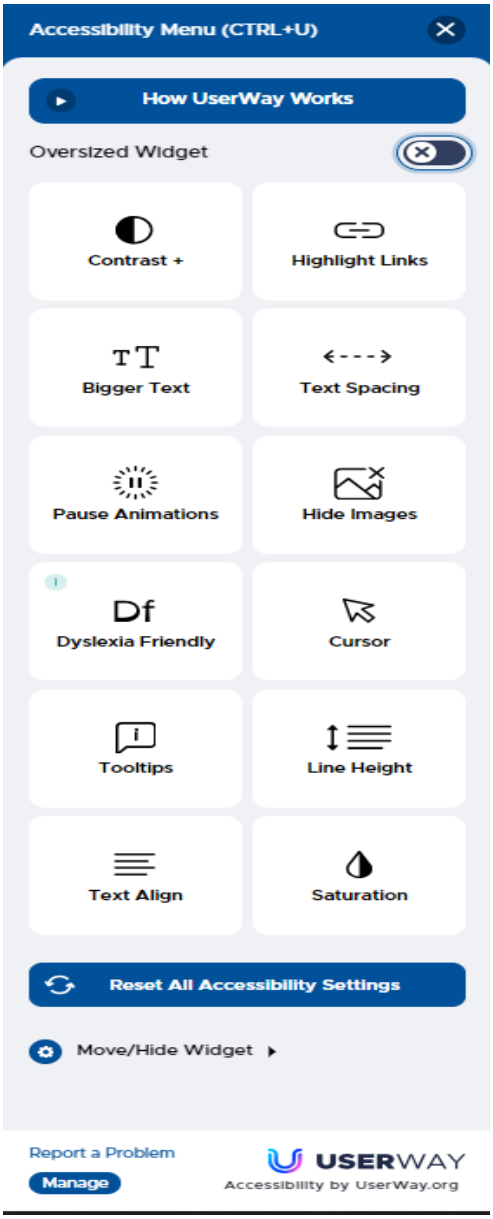


Figure 34: Accessibility widget menu.

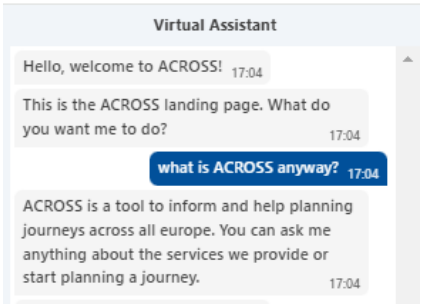
The platform's administrators benefit from the implementation of the Matomo analytics tool, which offers comprehensive insights into user interactions, traffic flow, and engagement metrics. Through this tool, administrators gain access to real-time visit details, including timestamps, geographic locations, browser types, and operating systems used by visitors. Furthermore, administrators can delve into historical visit data, analyzing trends on a weekly, monthly, or yearly basis. Visual representations such as visitor maps provide intuitive insights into the origins of unique visitors, empowering administrators to make informed decisions and optimize the platform to better serve its users.

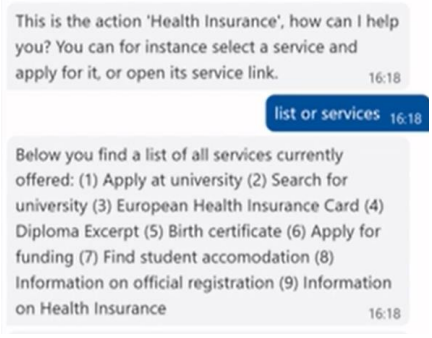
Virtual Assistant

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 Application (CWA) 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
- The VA also contains a simple info chatbot (also known as "Q&A tool") with which the user can obtain information about the ACROSS platform and the services available in it – see the following screenshots (in the second one, the user actually said "list all services", the "or" was an ASR misrecognition).

  The VA automatically and seamlessly switches between *WebAssist mode*, where the user controls the Citizen Web App conversationally, and this *InfoChat mode*, within a single conversation.





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 all project languages when interacting with the Application UI. The user is free to use the language to which the UI is currently set or English. 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). For two languages (German and English), voice output (through text-to-speech synthesis components) is available as well.

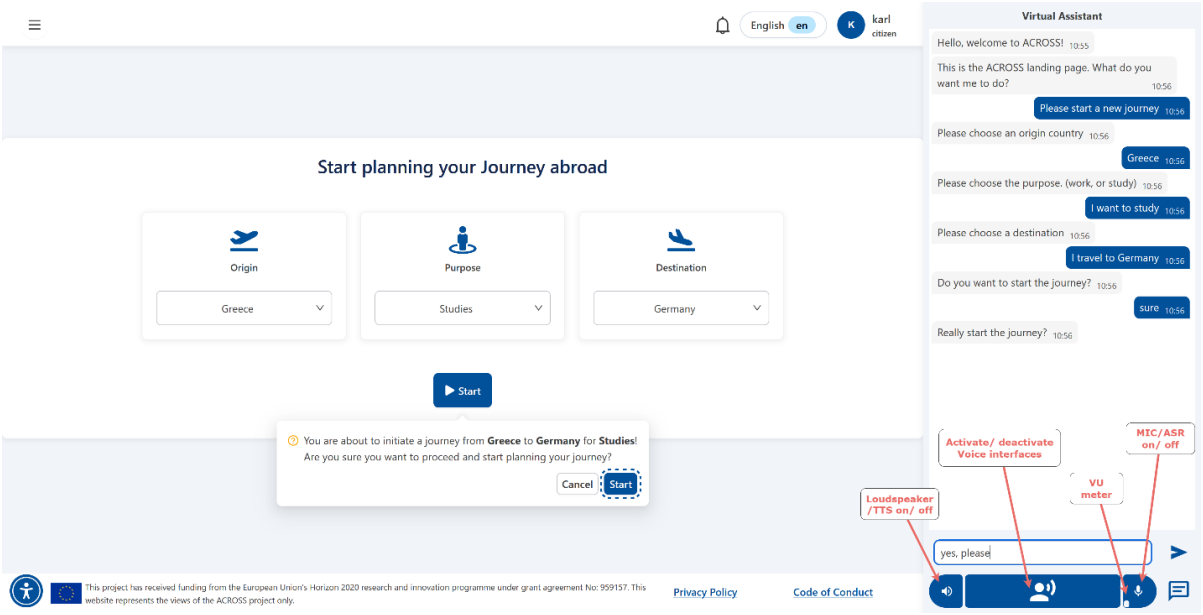


Figure 35: Chat User Interface of Virtual Assistant

The VA provides a chat "sidebar" widget which can be dynamically opened and closed using a button made available in the CWA. Pressing the button opens the VA sidebar and simultaneously activates the Virtual Assistant, and the conversation between user and VA is then shown within this widget (see right side of Fig. 35 above).

At the bottom of the VA widget, there are audio buttons for activating and deactivating the microphone, loudspeaker, or both of them simultaneously. The audio buttons only become active after inputting a special command "/voice" into the chat input area. This is because the speech components impose comparatively heavy resource requirements which the current prototypical ACROSS infrastructure is not ready to bear, at least for multiple concurrent user sessions. Therefore, this feature needs to be explicitly activated.

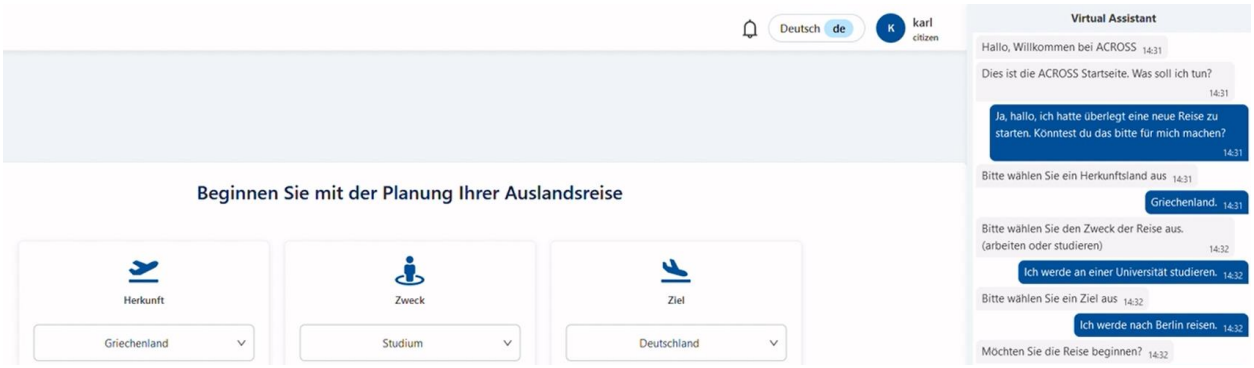
The small "VU meter" vertical bar visualizes the current incoming audio level and thus supports a simple form of control over whether the microphone is "open" and working. Both for data protection and in order to avoid misrecognitions due to background noise, It is recommended to activate the microphone only for as long as the user actually intends to speak.

In addition to this main chat widget (i.e., its end-user interface), the VA also provides a hidden debugging and testing widget. When activated (by clicking the VA button five times in a row, in quick succession), this widget allows the user to inspect important internal aspects of the VA's inner workings during its execution. That way e.g., when the system does not react to a user utterance as expected, 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.

Given the ongoing revolution around generative AI models exemplified by OpenAI's ChatGPT, user expectations around conversational systems have skyrocketed over roughly the last year. Towards that end, an initial integration between the ACROSS VA and a generative AI model has been pursued. Because strict privacy and data sovereignty requirements are fundamental to the ACROSS platform, interfacing with a cloud-based AI service (perhaps even operated outside Europe) would not have been a viable option. Consequently, after careful evaluation, an open-source large language model (LLM) from a European vendor (Mistral AI) was selected and has been integrated in such a way that secure deployment (on-premises or in a private cloud) is supported. (Note that the LLM version of the ACROSS VA currently is not fully deployed in the ACROSS infrastructure because it needs GPU resources for LLM inferencing, which are currently not available in this prototypical infrastructure – therefore a provisional deployment at Fraunhofer FOKUS needs to be used)

However, as this activity could only be started when the ACROSS project approached its conclusion, its results are very promising but still limited:

Currently the generative AI model is only used for NLU, i.e. for understanding the intention of the user's (voice or chat) utterances – and this indeed has considerably improved the NLU behavior of the ACROSS Virtual Assistant. The following screenshot shows a snippet of a conversation with the VA (in German) where the user uses quite colloquial and wordy language and answers a question only rather indirectly (asked for the journey destination country, he replies with the name of its capitol instead) – nevertheless, the VA correctly identifies the user's intentions and executes all associated actions in the ACROSS Citizen Web App correctly, which was not possible in previous VA versions without generative AI.



Now on the one hand side, there is still substantial room for further optimization regarding generative AI usage for NLU – such as by fine-tuning the LLM in order to further decrease misrecognitions, or by optimizing the LLM inferencing service in order to reduce latency and thus improve interactive responsiveness.

On the other hand, NLU is only one (albeit essential) part of the Virtual Assistant. The VA also contains an info chatbot, also known as the “Q&A tool”, aimed at answering user questions that might arise regarding the ACROSS platform being used, or regarding the external Services which the ACROSS platform integrates (and lets the user access as well). This info chatbot is rather limited in its current implementation, not just because of its prototype character but because it recently had become clear as well that a new approach based on generative AI would be needed anyway to meet the user expectations such systems now face due to the ChatGPT “hype wave”. However, the considerable risks involved with AI-generated text (hallucinations, factual errors, generation of unwanted or even illegal content) require very complex mitigation measures (such as maintaining and interfacing with traditional knowledge representations for fact checking) which would have been not only outside the scope of the ACROSS project but also way beyond its budget.

Fully integrating generative AI into a platform as deeply rooted in the Public Administration sector as ACROSS, while both unleashing the power of these new models and adequately taming their inherent risks, and fulfilling the long and strict list of requirements implied by this application domain (regarding privacy, data sovereignty, compliance and reliability) remains a hugely challenging and complex task, worthy perhaps of not one but several successor R&D projects.

3.2 Components Integration

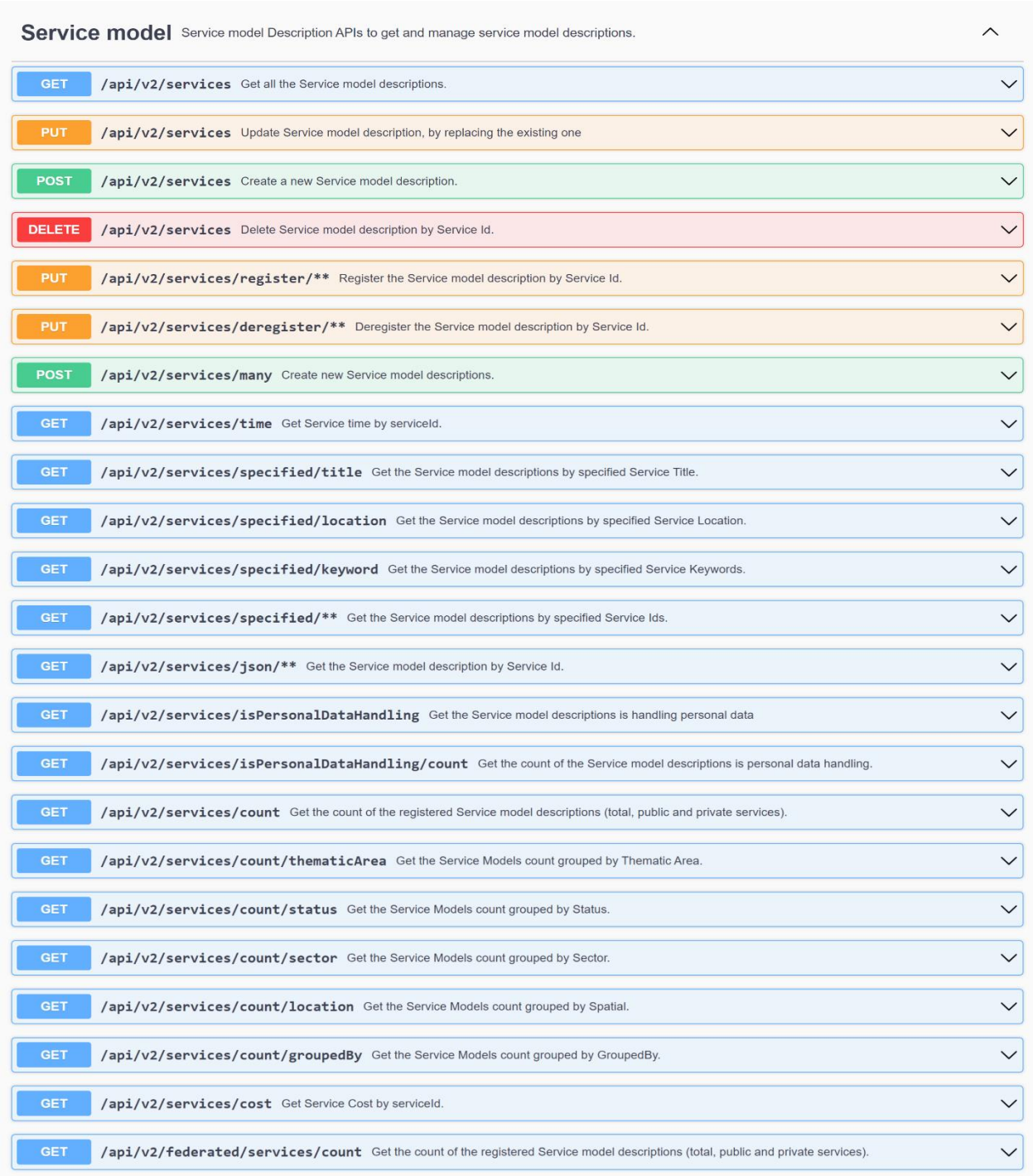
Table 2: Components Interaction

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:



Connector model

GET	/api/v2/connectors	Get all the Connector descriptions.
PUT	/api/v2/connectors	Update Connector model description, by replacing the existing one
POST	/api/v2/connectors	Create a new connector.
DELETE	/api/v2/connectors	Delete Connector model description by connectorId.
GET	/api/v2/connectors/json	Get Connector description by connectorId.
GET	/api/v2/connectors/count	Get the count of the registered Connector descriptions.

Adapter model

GET	/api/v2/adapters	Get all the Adapter model descriptions.
PUT	/api/v2/adapters	Update Adapter model description, by replacing the existing one
POST	/api/v2/adapters	Create a new adapter.
DELETE	/api/v2/adapters	Delete Adapter model description by AdapterId.
GET	/api/v2/adapters/json	Get Adapter description by adapterId.
GET	/api/v2/adapters/count	Get the count of the registered Adapter descriptions.

Connector log

GET	/api/v2/connectors/logs	Get Connector Logs description by connectorId.
POST	/api/v2/connectors/logs	Create a new connector log.
DELETE	/api/v2/connectors/logs	Delete Connector Log description by connectorId.
GET	/api/v2/connectors/logs/all	Get all Connectors Logs descriptions.

Adapter log

GET	/api/v2/adapters/logs	Get Adapter Logs description by adapterId.
POST	/api/v2/adapters/logs	Create a new adapter log.
DELETE	/api/v2/adapters/logs	Delete Adapter Log description by adapterId.
GET	/api/v2/adapters/logs/all	Get all Adapters Logs descriptions.

Status

- GET `/api/v2/status` Get Service catalogue's status.
- GET `/api/v2/federated/status` Get Service catalogue's status.

Federated query

- GET `/api/v2/federated/services` Get all the Federated Federated query descriptions.
- GET `/api/v2/federated/services/time` Get Service time by serviceId.
- GET `/api/v2/federated/services/specified/title` Get the Federated query descriptions by specified Service Title.
- GET `/api/v2/federated/services/specified/location` Get the Federated query descriptions by specified Service Location.
- GET `/api/v2/federated/services/specified/keyword` Get the Federated query descriptions by specified Service Keywords.
- GET `/api/v2/federated/services/specified/**` Get the Federated query descriptions by specified Service Ids.
- GET `/api/v2/federated/services/json/**` Get the Federated query description by Service Id.
- GET `/api/v2/federated/services/isPersonalDataHandling` Get the Federated query descriptions is handling personal data
- GET `/api/v2/federated/services/isPersonalDataHandling/count` Get the count of the Federated query descriptions is personal data handling.
- GET `/api/v2/federated/services/count/thematicArea` Get the Federated querys count grouped by Thematic Area.
- GET `/api/v2/federated/services/count/sector` Get the Federated querys count grouped by Sector.
- GET `/api/v2/federated/services/count/location` Get the Federated querys count grouped by Spatial.
- GET `/api/v2/federated/services/count/groupedBy` Get the Federated querys count grouped by GroupedBy.
- GET `/api/v2/federated/services/cost` Get Service Cost by serviceId.

Catalogue dataset model

- PUT `/api/v2/catalogueDatasets` Update Catalogue dataset model description, by replacing the existing one
- POST `/api/v2/catalogueDatasets` Create a new catalogue dataset .
- DELETE `/api/v2/catalogueDatasets` Delete Catalogue dataset model description by catalogue dataset ID.
- GET `/api/v2/catalogueDatasets/public` Get all the catalogue datasets descriptions.
- GET `/api/v2/catalogueDatasets/json` Get catalogue dataset description by catalogue dataset ID or name.
- GET `/api/v2/catalogueDatasets/count` Get the count of the registered CatalogueDatasets descriptions (total, public and private services).

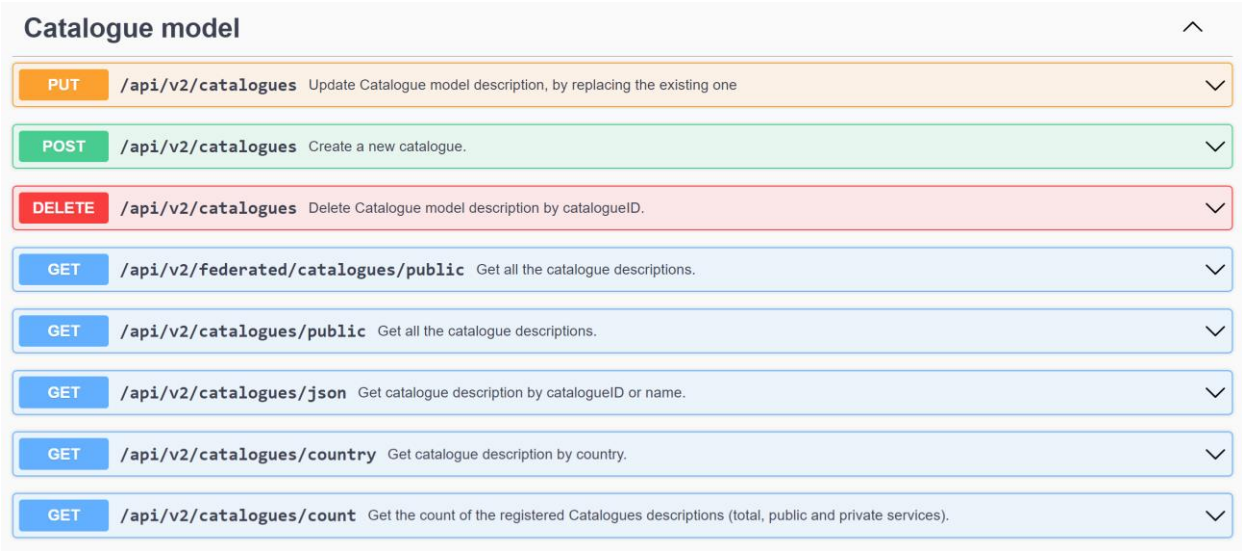


Figure 36: Service Catalogue APIs

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 available for the citizen. After saving a User Journey Model to the Storage, it will only be made available to the UJSE once the "Approved" flag has been set in the UJMT UI. Thus it is possible to work on different versions of a model, and finally release it to practical use.

3.2.3 Virtual Assistant.

Although it supports the ACROSS Citizen Web 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 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 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 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 37: 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 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.

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

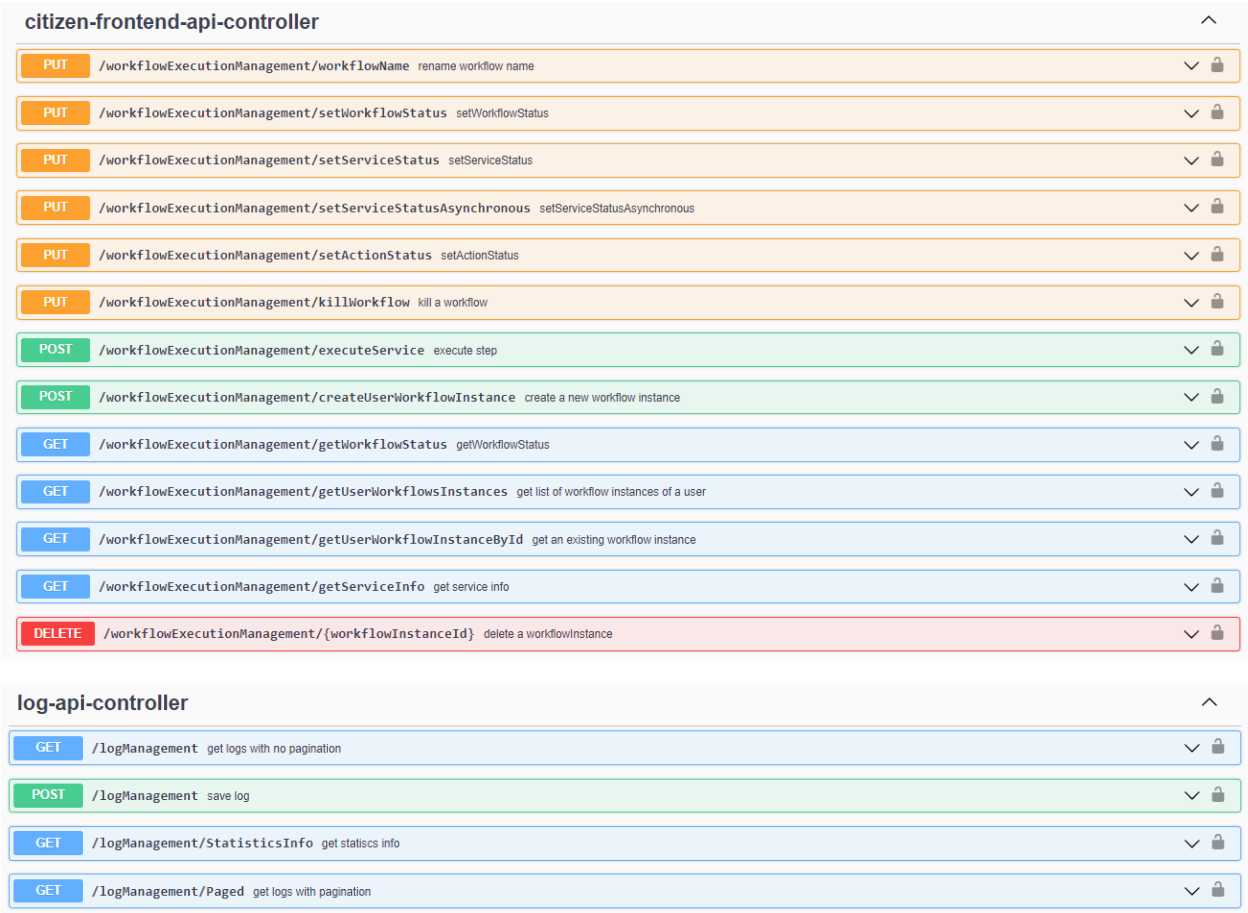
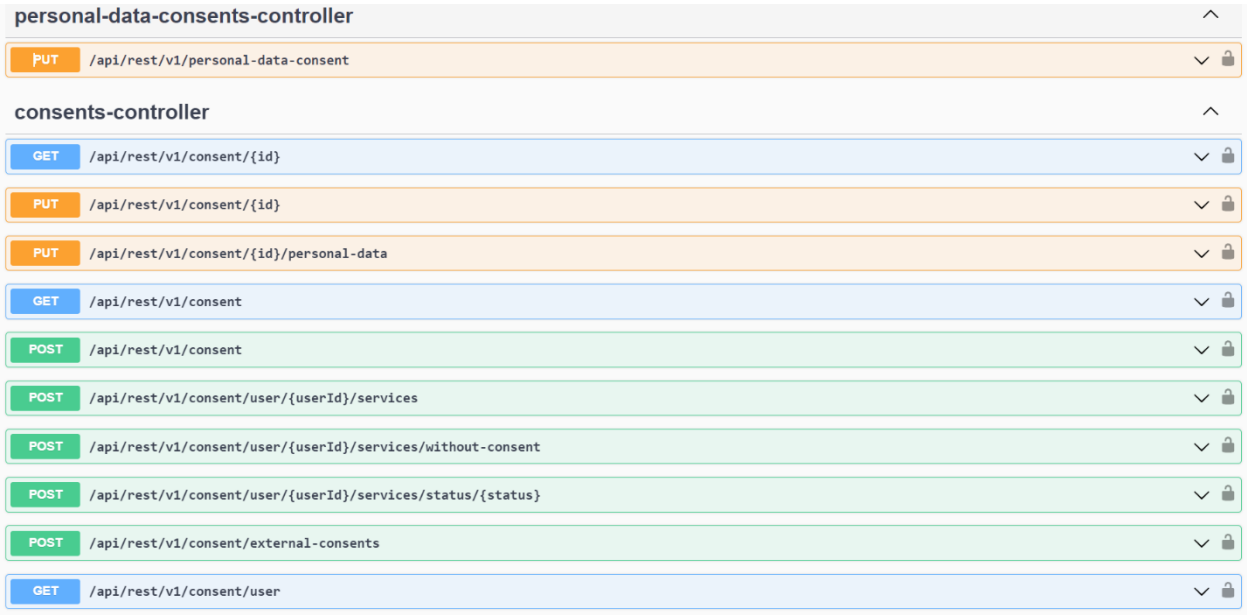


Figure 38: UJSE APIs

3.2.5 Transparency Dashboard

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

- personal-data-consents-controller and consents-controller: 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.
- event-logs-controller: it provides the API for the Transparency Dashboard to retrieve the events (applied actions on consents).



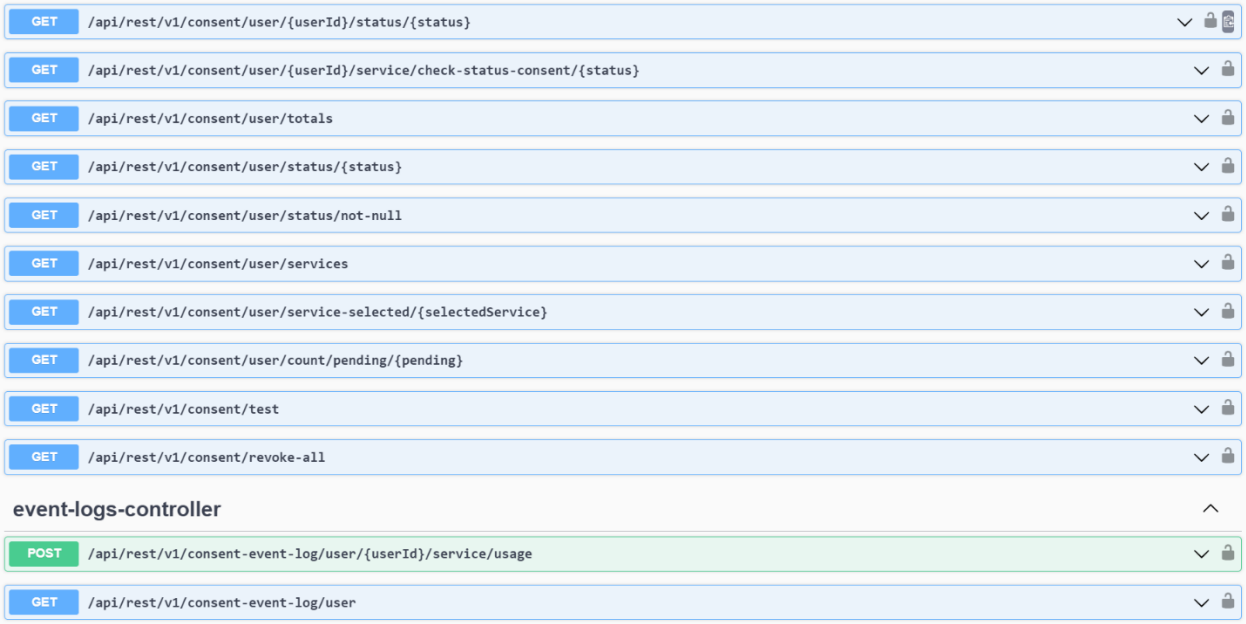


Figure 39: Transparency Dashboard APIs

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:

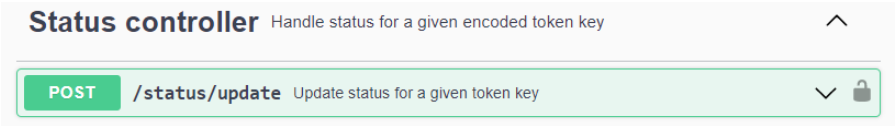


Figure 40: Citizen Web Application API

3.2.7 Usage Control

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

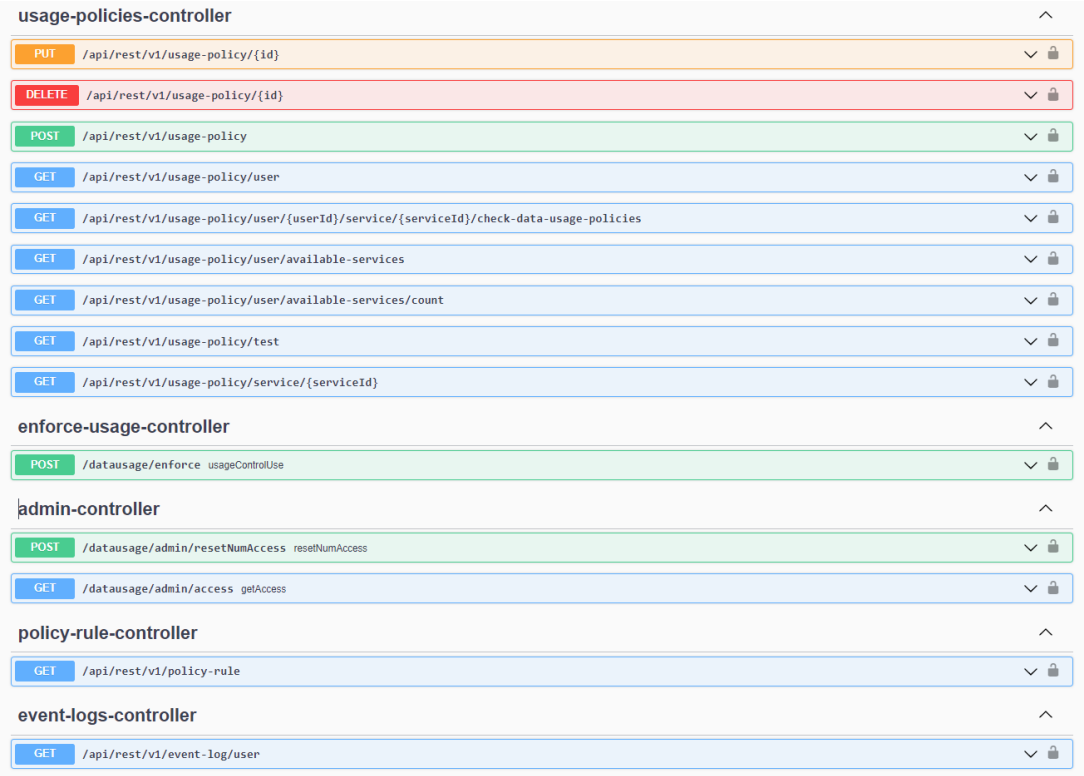


Figure 41: Usage Control APIs

4 Conclusions

This deliverable is the accompanying report of the final 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.

5 References

[1] D4.3 Components adaptation for SDG, OOP, Eidas for National public services (final)

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

[3] D4.6 Micro proxies and service catalogue (final)

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

[5] D4.9 User Support tools (final)

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

[7] D5.3: ACROSS platform prototype and applications (initial)

[8] D5.4 ACROSS platform prototype and applications (intermediate)

[9] D6.3 Use case evaluation and impact assessment – Final

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