"Building the Next Generation Personal Data Platforms"
G.A. n. 871370

DELIVERABLE D5.2
Release of the Cloud Controller, Open APIs, Personal Data Avatar, Transparency Tags and Data Marketplace

H2020-EU: **PIMCity**
Project No. 871370
Start date of project: 01/12/2019
Duration: 33 months

Revision: V.2
Deliverable delivery: 10/06/2022
Deliverable due date: 31/05/2022

Document Information

Document Name: Deliverable 5.2 - Release of the Cloud Controller, Open APIs, Personal Data Avatar, Transparency Tags and Data Marketplace
WP5 – Platform design and integration
Author: ERMES and all Partners

Dissemination Level

Project co-funded by the EC within the H2020 Programme		
PU	Public	☑
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Deliverable authors

	Name	Entity	Date
Author	Roberto Gonzales	NEC	31/05/2022
Author	Evangelos Kotsifakos	LSTECH	31/05/2022
Author	Xavi Olivares	LSTECH	31/05/2022
Author	Ruben Cueva Rumin	UC3M	31/05/2022
Author	Martino Trevisan	POLITO	31/05/2022
Author	Alvaro Garcia-Recuero	IMDEA	31/05/2022
Author	Santiago Azcoitia	IMDEA	31/05/2022
Author	Ioannis Arapakis	TID	31/05/2022
Author	Daniel Fernandez	Wibson	31/05/2022
Author	Rodrigo Irarrazaval	Wibson	31/05/2022
Author	Miguel Herranz	IAB	31/05/2022
WP Leader	Stefano Traverso	ERMES	31/05/2022
Coordinator	Marco Mellia	POLITO	31/05/2022

Document history

Revision	Date	Modification
Version 1	31/05/2022	V2

List of abbreviations and acronyms

Abbreviation	Meaning
G.A.	Grant Agreement
CA	Consortium Agreement
GA	General Assembly
PB	Project Board
PC	Project Coordinator
PrO	Project Office
IR	Interim Reports
PDK	PIMS Development Kit
PIMS	Personal Information Management Systems
PDA	Personal Data Avatar
DM	Data Marketplace
TT	Transparency Tags
P-CM	Personal Consent Manager
P-DS	Personal Data Safe
P-PPA	Personal Privacy Preserving Analytics
P-PM	Persona Privacy Metrics
DVTUP	Data Valuation Tools – End-user Perspective
DVTMP	Data Valuation Tools – Market Perspective
TE	Trading Engine
TM	Task Manager
DA	Data Aggregation
DPC	Data Portability and Control
DKW	Data Knowledge Extraction

Table of Contents

1 INTRODUCTION 9

2 DELIVERABLE OBJECTIVES 10

3 PLATFORM OVERVIEW 11

4 PHYSICAL PLATFORM, ENVIRONMENTS, AND DEPLOYMENT 13

4.1 INFRASTRUCTURE OVERVIEW 13
4.1.1 SECURITY AND ACCESS CONTROL 15
4.1.2 DEPLOYMENT 16
4.1.3 DEVELOPMENT AND PRODUCTION INFRASTRUCTURES 17

5 DESIGN OF THE CLOUD CONTROLLER 21

5.1 DESIGN CHOICES 21
5.2 AUTHENTICATION AND AUTHORIZATION FLOWS 23
5.2.1 USER TO APP AUTHORIZATION FLOW 23
5.2.2 MACHINE TO MACHINE 25

6 DESIGN OF THE OPEN APIS 26

6.1 DESIGN PRINCIPLES 26
6.1.1 RESTFUL DESIGN 27
6.2 DESIGN STANDARDS, SPECIFICATIONS AND TOOLS 30
6.2.1 STANDARDS AND SPECIFICATIONS 30
6.2.2 TOOLS 31
6.3 DESIGN OF APIS COMPONENTS 31
6.3.1 WP2 31
6.3.2 WP3 37
6.3.3 WP4 41

7 PERSONAL DATA AVATAR (PDA) AND USER DASHBOARD 48

7.1 GENERAL DESIGN 48
7.2 PDA COMPONENTS 49
INFORMATION SCREEN 50
7.2.1 REGISTRATION AND LOGIN 51
7.2.2 MAIN SCREEN 52
7.2.3 MY DATA 53
7.2.4 MY CONSENTS 54
7.2.5 SETTINGS 55

7.3 DEPLOYED IMPLEMENTATIONS 55

8 DESIGN OF TRANSPARENCY TAGS (TT) 56

8.1 STATE OF THE ART 56
8.2 REQUIREMENTS 58
8.3 PRELIMINARY MOCKUPS 60
8.4 FEEDBACK CAMPAIGNS 63
8.4.1 FEEDBACK FROM FOCUS GROUPS 63
8.4.2 FEEDBACK FROM WIBSON USERS 63
8.4.3 FEEDBACK FROM DATA BUYERS 67
8.4.4 FEEDBACK FROM LEGAL PARTNERS 67
8.4.5 FEEDBACK FROM TECHNICAL PARTNERS 67
8.4.6 FEEDBACK FROM B2B MARKET 68
8.5 CONCLUSIONS ON DESIGN 68
8.6 IMPLEMENTATION IN EASYPIMS 68
8.7 IMPLEMENTATION IN ERMES FOR ENTERPRISE 70
8.8 CONCLUSIONS ON IMPLEMENTATION 72
8.9 DEPLOYED IMPLEMENTATIONS 72

9 DESIGN OF THE DATA MARKETPLACE 73

9.1 GENERAL DESIGN 73
9.1.1 DATA MARKETPLACE COMPONENTS 74
9.2 DEPLOYED IMPLEMENTATIONS 81

10 DESIGN OF THE TASK MANAGER 82

10.1 GENERAL DESIGN 82
10.2 COMPONENTS 83
10.2.1 API 83
10.2.2 SWEEPSTAKES BACK-OFFICE UI 85
10.2.3 LOGIN 86
10.3 DEPLOYED IMPLEMENTATIONS 90

11 CONCLUSIONS 91

12 BIBLIOGRAPHY 92

13 – APPENDIX – A1 93

14 – APPENDIX – A2 98

Disclaimer

Instructions to read this deliverable

This deliverable completes the design process started in Deliverable D5.1 [1]. As such, we decided to present this document as an updated version of D5.1 where we highlighted new and modified parts in blue color. That means that all parts highlighted in blue are either new or update and improve the corresponding sections present in Deliverable D5.1.

1 Introduction

The path towards the development of a novel platform for understanding, controlling and, possibly, monetizing personal data, requires an overall integration of the independent components that are each performing specific tasks.

To demonstrate the flexibility of the Personal Development Kit (PDK), PIMCity partners are designing and implementing EasyPIMS, a scalable, novel, holistic approach to personal data sharing on the Internet. The goal of this Work Package and, specifically, of this deliverable is to provide an overall design of such novel approach in data sharing, in which is key to provide to users the tools to concretely understand the value and nature of the data they share and consume. To simplify design and development of the tools, we try to follow the commonplace K.I.S.S.[1] principle by building a modular platform with separate, smaller components that reduce the chance of programming defects. The K.I.S.S term was introduced by the U.S Navy and at a high level it is also used in computing in systems development as the case of Unix[2]. For this, we introduce four fundamental blocks which will assist the user in achieving awareness and gaining control on personal data sharing. These blocks are: The Personal Data Avatar (PDA), the Transparency Tags (TT), the Data Marketplace, and the Dashboard. Each one of these builds on one or more of components offered by the PDK.

In this context, PIMCity's WP5 aims at designing, building, and implementing the set of software modules and deployment platforms to integrate all PDK components, and thus build the EasyPIMS holistic platform.

More specifically, this deliverable, the second and last of WP5, provides the final design of all fundamental bricks composing EasyPIMS, the platform which will allow users to trade their data with data buyers in an informed and controlled way. In this deliverable, we revise the high-level design of WP5 components, how the modules satisfy the requirements defined in Deliverable 1.1 and confirm or adjust the choices from both technical and integration perspectives.

Apart from defining the overall integration and system deployment for both development and production environment, this WP also provides the general guidelines to guarantee and enforce security and privacy uniformly all over the project based on state-of-the-art solutions and best practices. Moreover, for the sake of uniformity this WP also indicates the environments, the frameworks, and tools to ease development of components and accelerate their deployment.

As expected at the beginning of WP5, proposals described in Deliverable 5.1 have evolved during the development of the project based on the feedback obtained from developers, users and data buyers, and depending on possible modification of the technical requirements. For instance, one major design change is represented by the introduction of a novel component, named Task Manager. This has been introduced because of technical requirements emerged during the integration phase.

Apart from describing final design choices and implementation, this deliverable releases the software of key components of EasyPIMS platforms. These are the Cloud Controller, the PDA, the Data Marketplace, the Transparency Tags, and the Task Manager. All code building these components is available on PIMCity's official Gitlab repository in WP5 folder.[3]

This document is structured as follows: section 2 describes the objectives of this deliverable; Sections 3 and 4 describe the platform which will host and run EasyPIMS; Section 5 details the design choices for the Cloud Controller; Section 6 describes the general requirements for the design of Open APIs and provides the specifications for all PDK components; Section 7 details the structure of Personal Data Avatar design and Section 0 describes the design choices made for Transparency Tags. Finally, Section 9 describes the structure of the Data Marketplace, and Section 11 describes the Task Manager and Section 11concludes the document.

2 Deliverable objectives

This deliverable is the second and the last of WP5. The goals of this deliverable are:

1. Describing the final design and implementation the production platform which will run EasyPIMS and make it easy for components to be integrated in the systems.
2. Confirming the design and implementation choices at the basis of EasyPIMS fundamental components. These are Open APIs and SDKs developed to ease the integration among modules, developed within and outside PIMCity.
3. Documenting the final design of the controller that allows the execution and interoperability of the building blocks in some existing computing platform.
4. Describing the final design and implementation of the components at the basis of EasyPIMS, I.e., Transparency Tags, Personal Data Avatar, Data Marketplace and Task Manager.
5. Releasing the code so far produced to develop the Open APIs, the Transparency Tags, the Personal Data Avatar, the Data Marketplace and the Task Manager.

More in detail, this deliverable provides the final design of 1) the **Cloud Controller**, the component responsible for orchestrating, coordinating and authorizing the interactions among components; 2) the **Open APIs**, technically the glue allowing each component to interact with the others using standard and uniform interfaces; and iii) the four integrated components building EasyPIMS, the **Personal Data Avatar,** the **Transparency Tags,** the **Data Marketplace** and the **Task Manager** with a special attention to their relations with the PDK components. For the sake of simplicity, the User Dashboard and the Personal Data Avatar have been merged in a unique component.

In this document we provide the final complete design for each of such component, providing insights on technical requirements that have been considered during both preliminary design phase and implementation.

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

[2] https://en.wikipedia.org/wiki/The_Art_of_Unix_Programming

[3] https://gitlab.com/pimcity/wp5

3 Platform overview

The EasyPIMS platform is composed of a hardware infrastructure hosting several virtual machines in a cloud environment, and a set of software modules running on those machines. The software components interact with each other using open APIs designed by PIMCity partners or industry-standard open APIs.

One fundamental component is the **Cloud controller**, which manages end-user accounts and provides end-user authentication to each one of the other modules, as well as inter-module authentication and authorization. The Cloud controller will also provide end-user management and authentication to future software modules, to ensure central control over user access and authorizations. The Cloud controller is an instance of the open-source Keycloak software, installed on EasyPIMS virtual machines and managed by PIMCity partners.

These components are the **Personal Data Avatar** (PDA in short), the **Data Marketplace** and the **Transparency Tags** (TTs in short). All these components synthetize and integrate the PDK components developed in WP2, WP3 and WP4. In addition to these components, Deliverable 5.2 introduces the **Task Manager**, a novel component. The Task Manager is an EasyPIMs component that interacts with Trading Engine and Cloud Controller to raise the level of engagement of end-users by introducing the gamification concept.

Fundamental to glue together the EasyPIMS components and guarantee their interoperability are the Cloud Controller and the **Open APIs**. We depict the overall platform overview in the following picture.

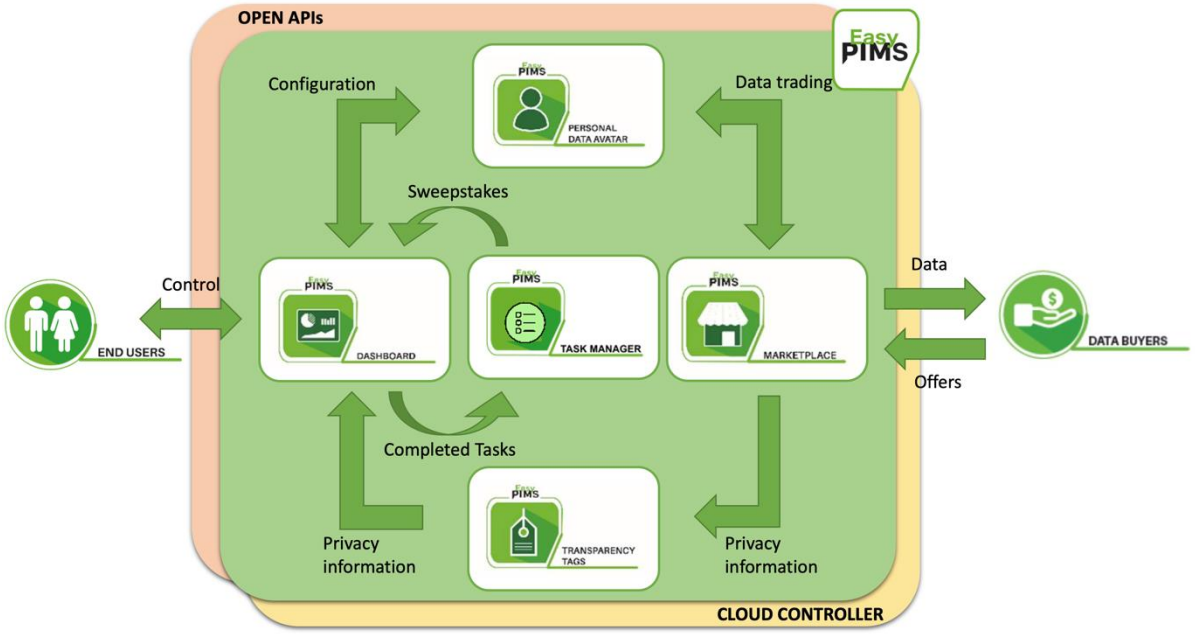


Figure 1 - High-level overview of EasyPIMS platform

The picture above depicts the fundamental EasyPIMS components and their interactions. For the sake of completeness, we report here a short description of each component and refer the reader to Deliverable D1.1 [1] for a more complete description.[4]

[4] Task Manager description is not available in Deliverable D1.1. Indeed, it has been introduced, designed and implemented in the last months of WP5.

- **Cloud Controller and Open APIs:** As shown above, the Cloud Controller and the Open APIs act as distributed sub-layers operating under the surface of EasyPIMS. Indeed, their role is to guarantee a common and uniform API layer to deliver the functionalities offered by PDK components with state-of-the-art security and privacy measures. In particular, security and privacy orchestration (e.g., authentication and authorization functions) will be defined and implemented at the Cloud Controller. On the other hand, Open APIs will be responsible for enforcing the security and privacy policies defined by the Cloud Controller on the interface endpoints for both human-to-machine and machine-to-machine interactions. Open APIs will operate uniformly across all PDK components, thus allowing correct integration, interoperability, and re-usability. The designs of the Cloud Controller and the Open APIs are provided in Sections 5 and 6, respectively.
- **Dashboard:** It assists the end user at controlling the personal data in the Personal Data Safe, check the corresponding value via the PDA, controlling service reputation via the Transparency Tags, etc. Thanks to the Dashboard, users are able to control the EasyPIMS, their data, and their preferences. The Dashboard builds on the design of the Personal Data Avatar (Section 7) and the Transparency Tags (Section 0).
- **Personal Data Avatar:** the interface between the user and the services operating in the Data Marketplace. Thanks to the PDA, the user becomes the only owner of her personal data and acquires the power of deciding which data to share with which service. The PDA lets the user know the actual monetary value of exposed data, increasing awareness about her privacy. In turn, the PDA provides services (e.g., data buyers) with personal information built and validated by the user. We describe its design in Section 7.
- **Transparency Tags:** The Transparency Tags communicate in an easy-to-understand way the nature of the service (may it be a website, a mobile app or data buyer) the user is interacting with. For each service EasyPIMS exposes the information computed by the Privacy Metrics, such as its owner, its purpose, the personal data it collects, etc. Part of the Transparency Tags builds on information directly provided by the Data Buyer as part of the GDPR-compliance process. The design of the Transparency Tags is described in Section 0.
- **Data Marketplace:** Similar to a physical marketplace, the EasyPIMS Data Marketplace facilitates the trading of user data (via the PDA) and data buyers. EasyPIMS shows in the Marketplace data profiles obtained by PDA made available by users. Then, the Data Marketplace provides Data Buyers the tools to present offers and buy the data. The resulting compensation will be then delivered via the PDA to the end user who will be able to manage and control the transaction through the Dashboard. The design of the Data Marketplace is provided in Section 9.
- **Task manager:** As we were approaching the integration phase, the consortium has agreed on the need of introducing a novel component, responsible of introducing tools to engage the users. Thanks to the Task Manager, EasyPIMS administrators can design challenges and sweepstakes to allow users compete with the final goal of bootstrapping the platform.

4 Physical platform, environments, and deployment

4.1 Infrastructure overview

In this section we describe the cloud platforms that we use for both developing (Development Environment) the PDK components of PIMCITY and hosting EasyPIMS platform (Production Environment).

The development and production platforms support the following general requirements:

- **Availability:** The platform shall be available 24/7 in order for the partners to be able to develop, deploy and test their applications and prevent service outages.
- **Continuous integration**: The PIMCITY project follows a lean methodology, where elements of the architecture and infrastructure will evolve based on the continuous feedback from developers and users.
- **Security:** The environments are secured with restricted access and no connection to the external world. If it is necessary for applications to connect to external applications, this is done through secure procedures, such as proper firewall exceptions.
- **Extensibility and interoperability:** The environments are easily extensible. New applications are easily integrated. The environments allow connections to external services when needed.
- **Administration and manageability:** The environment allow easy administration and management. The infrastructures build on dynamic, powerful, and robust centralized environments, able to provide PIMCITY developers a system to access and share their different components without affecting each other, within their own workspace. To this end, each technical partner has at least one user to manage their workspace, with the proper access rights.

For both development and delivery of EasyPIMS platform we rely on Dockerized environment for most of the components. This allows ease of implementation, extensibility, portability, and security. Indeed, a container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker[5] container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine. Available for Linux, macOS and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from the hosting environment and ensure that it works uniformly despite differences for instance between development and staging.

[5] https://www.docker.com/

The main advantages of using Docker containers running on Docker Engine are:

- **Standardization:** Docker created the industry standard for containers, so they could be portable anywhere.
- **Lightweight:** Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs.
- **Secure:** Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry.
- Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less resources than Virtual Machines (VM), (container images are typically tens of MBs in size). See a comparison of Container and VM based approaches in the figure below.

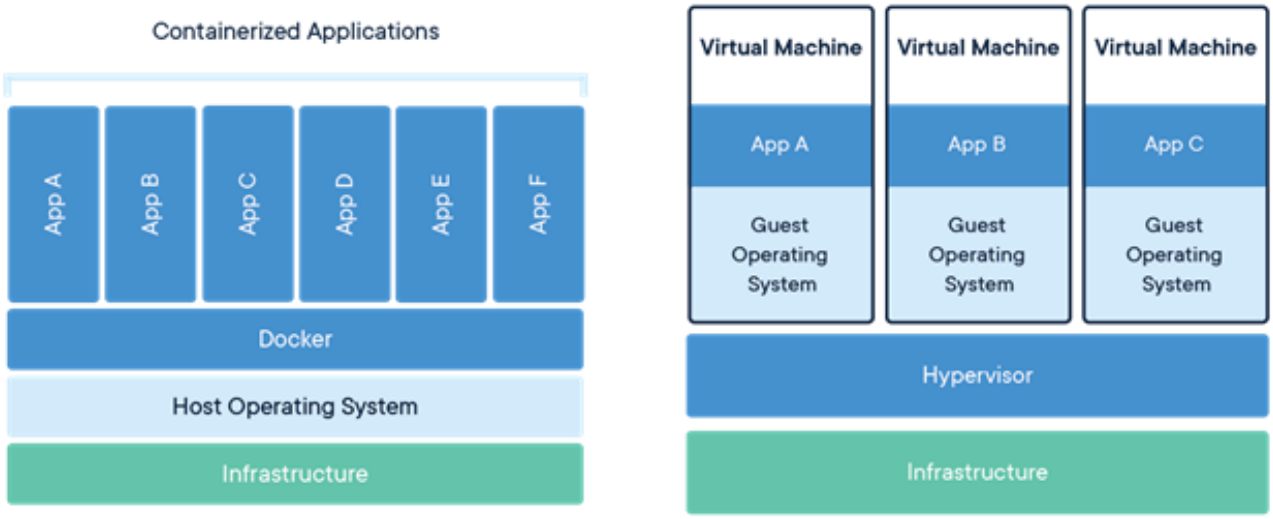


Figure 2 - Virtualization architectures – courtesy of Docker

Each content provider has created containerized version of PDK modules, that will be run on the part of the infrastructure hosting the machines. In case a module has only Python dependencies that can be completely satisfied using a regular Python virtual environment, we run it natively on the infrastructure machines. The following figure illustrates this setup.

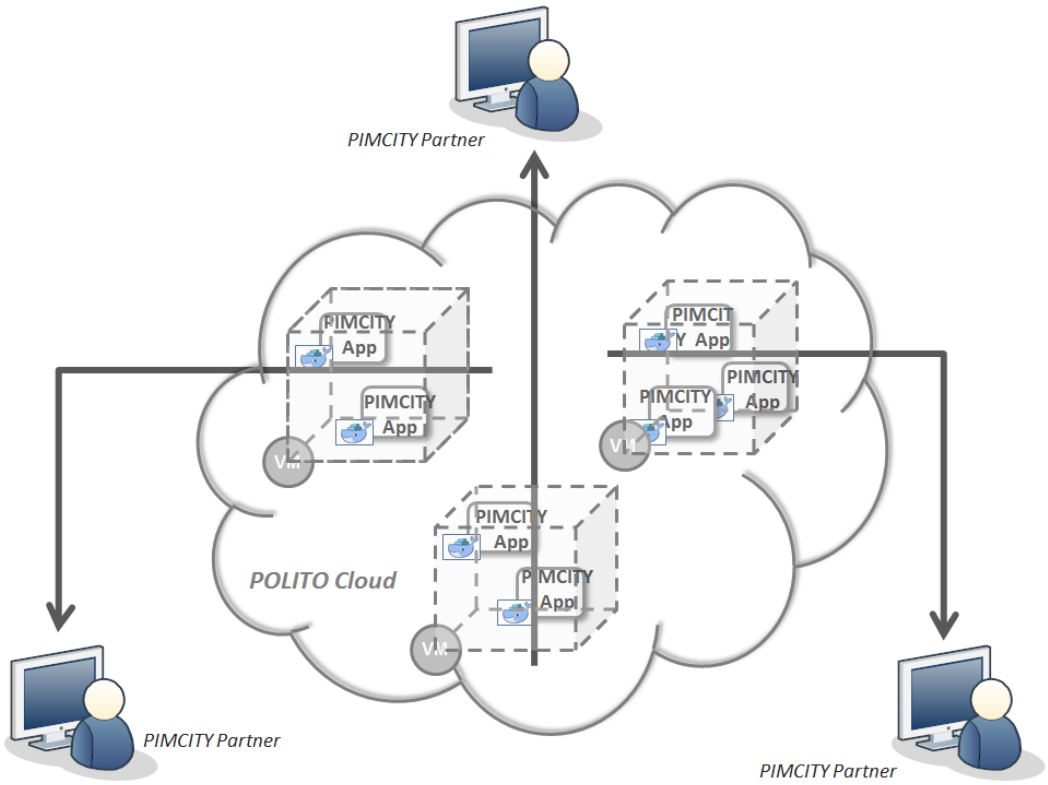


Figure 3 - PIMCity physical and virtual infrastructure setup

To accommodate to the size progression of the project, new tools have been used in order to have better control of the infrastructure. We might use Ansible automation tool to manage the infrastructure as the number of Virtual Machines and computational demand keep growing.

4.1.1 Security and access control

The environments are accessible using SSH and HTTPS protocols and restricted to specific people related to the technical work packages and their applications.

Before the administrator of the site grants access to the shared space, developers had to fill in an excel file with the relevant information to manage their permissions and allocate the needed resources. The required information is the following:

- Partner name
- Module/component
- Contact person
- Email of the contact person
- Gitlab email
- Dependences
- Main technology/framework used
- Minimum requirements (CPU/RAM/disk)

The infrastructures are monitored to identify possible inconsistencies, conflicts or security issues based on system, application, and access logs.

Applications and programming interfaces (APIs) have been designed, developed, deployed, and tested in accordance with leading industry standards and adhere to applicable legal, statutory, or regulatory compliance obligations.

All the servers will be hosted in EU countries and no data will be transferred to other countries outside the EU.

In case special agreements were needed between the partners to ensure secure access to the developed software and to disclosure of information or data, these have been drafted and signed according to the counselling of the project's legal partners and implemented following the appropriate measures.

The technical partners have also implemented additional special security and privacy requirements for their components. Indeed, we remark that all partners have completed a DPIA analysis, which is kept updated in [2] [1]. This allowed partners to be conscious of eventual problems, and keep control on the process so to prevent security and privacy issues and data leakages.

4.1.2 Deployment

We use the same deployment design and policies for both the Development and Production infrastructures. We describe them in the following.

User management and roles: In both Development and Production infrastructure, we use a cluster of virtual machines (VMs). The accounts on the VMs are stored on the master VM, using an LDAP server. Each machine authenticates users remotely via a private network internal to the cloud data center we use. Each user has a non-shareable account. The authentication server is kept secure and runs behind (i) the internal firewall of the cloud datacenter, allowing only SSH and web access, and (ii) the intrusion protection system deployed at the network level. During the development phase, each partner is provided a VM, and the partner's developers will have complete control of that machine. Users access their machines via SSH.

Gitlab project access and users: For what concerns the versioning of the code, we rely on Gitlab. Our setup is the following**:**

- The Group PIMCITY Platform has been created at https://gitlab.com/pimcity
- Each project member has been invited, and developers have created a code repository for the version of their components.
- Each repository corresponds to a PIMCity component, and apart from the code, it contains the documentation and instructions to use the code.

Recommendation for deployment: The easiest way to deploy all the necessary files in the Git repository is Docker. If a Dockerized version of the code is made available in the code repository of each component, partners (and external users) can then easily use Docker to get updates and deploy components, avoiding the struggling of building components from scratch.

Continuous integration: There exist many tools that allow developers to automatically integrate and new code developments and deploy it in production environment. Jenkins is possibly the most popular tool for implementing continuous integration, but its usage is limited to large projects managed by advanced developers, and installation requires a discrete amount of time and resources. For this project, we encourage the usage of continuous integration tools, and, in particular, we recommend Jenkins, but given its complexity, we do not consider it as a must-have requirement.

System health monitoring and restart procedures: all components deployed on the platform have required to introduce some gear to monitor their status, and automatically restart the component in case of reboot of the machine hosting it. The tool that has been adopted for monitoring the status of components is Grafana (https://grafana.com/), which allows to create Synthetic Monitoring Alerts for HTTP endpoints. Taking as input a URL to check periodically, the returning code expected in the response (e.g., 200 OK) and the mail address of the alert receiver, it periodically "probes" the endpoint with HTTP requests and notify the receiver in case responses do not return the expected code or no response at all is received.

For what concerns the restart procedure, each component provider has developed some scripts to make components automatically start at boot. For this, we used Linux's scheduling tool named Cron. Finally, all developers have shared with other partners the instructions to restart their components in case of failures. These have been uploaded on PIMCity's private file storage system as they may contain information critical for the infrastructure (e.g., admin credentials) and, thus, impossible to store on the official code repository.

4.1.3 Development and Production Infrastructures

To guarantee continuous service availability and reliability, we created two infrastructures, completely independent one from the other. The first, namely Development Environment, has been used to develop the components and run experiments. The second, dubbed Production Environment, will host the stable releases of EasyPIMS and will be used to design campaigns aiming at testing the whole platform. Differently, the development infrastructure is used to test functionalities in their early development phases, leaving developers and researcher the possibility to run tests and experiments in an actual working environment without the worry of impairing the service delivery and/or users' experience.

4.1.3.1 Resource requirements for Development Infrastructure

For the Development Environment, we will use a set of Virtual Machines for all PIMCITY technical partners. Following the criterion of on demand service, when necessary, additional Virtual Machines will be deployed. Virtual Machines are orchestrated using an OpenStack cluster manager, hosted in the POLITO premises, and managed by the POLITO IT staff.

The resources currently allocated are six virtual machines. The first virtual machine acts as master, and it provides centralized login for all developers and runs an Nginx[6] web server to centralize web access. The other virtual machines are used by developers to develop, test and run PDK components.

The VMs have the following specifications:

- Virtual CPUs: 8 cores
- RAM: 32 GB
- Disk size: 64 GB
- Image OS: Ubuntu Server 20.04
- Services Docker is installed, and login is handled remotely via LDAP

The resources are adequate to support the current developments and they will be updated properly according to the needs of the progression of the different components.

4.1.3.2 Development Environment realization

The Development infrastructure has been set-up on POLITO servers, and policies and procedures are being established and maintained in support of data security to include confidentiality, integrity, and availability across multiple system interfaces, jurisdictions, and business functions to prevent improper disclosure, alteration, or destruction.

The actions taken to put in place the security protocols and measures for this type of environment are:

- **Unique identification and authentication:** Internal corporate user account credential shall be restricted for ensuring appropriate identity, entitlement, and access management and in accordance with established policies and procedures:
  - One account per user on the infrastructure (VMs)
  - One account per user on Gitlab.com
- **Version Control Systems:** Git is a distributed Version Control System (VCS) where each local code repository maintains a complete copy of the history of changes to code. A PIMCity group on GitLab[7] has been created and is used to host the developed source code.
- **Package Repositories:** A different public repository for each component/application are set up within the PIMCity group on GitLab[8]. The technical partners have to provide a

[6] https://www.nginx.com/

[7] https://gitlab.com/pimcity

[8] If the partner's code should be kept private, the developers will have the possibility to the use the Gitlab repository for their compiled application, instead of the source code.

README.md file for describing application functionality and deployment instructions. It should be noted that each partner has to deploy its application.

- **Dashboards and administration:** the VMs are created on a large cluster hosted in the POLITO premises. The POLITO cloud is based on the OpenStack cloud manager and managed by the IT staff of the University. The infrastructure responsible for the PIMCity project has an account on the OpenStack[9] Dashboard, which is used to administrate and control the VMs. Below, it is attached a screenshot of the dashboard.
- **HTTP Routing:** as said above all services hosted in the Development environment are exposed to the web via an Nginx instance that is responsible to route incoming HTTP requests to the proper component. For the development infrastructure we opted to use a path-based routing so that components are mapped to specific URL paths. For instance: https://easypims.pimcity-h2020.eu/privacy-metrics/ is the URL used by the web app to contact Privacy Metrics backend service.

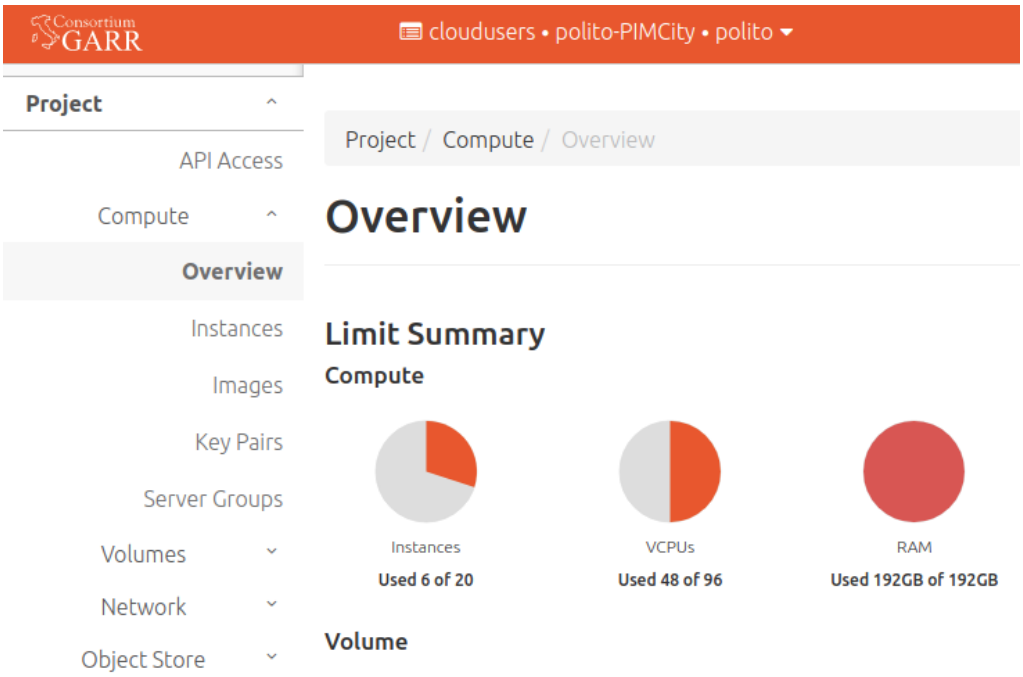


Figure 4 - OpenStack dashboard at POLITO's premises

[9] https://www.openstack.org/

4.1.3.3 Resource requirements for Production Infrastructure

For the Production environment we use a set of Virtual Machines for all PIMCITY technical partners. Like for the Development infrastructure, we follow the criterion of on demand service: when necessary, we deploy additional Virtual Machines. Virtual Machines are orchestrated using an VMWare hypervisor, hosted in the FASTWEB premises, and managed by the FASTWEB IT staff.

The resources currently allocated are 6 virtual machines. The first virtual machine acts as master, and it provides centralized login for users and runs an Nginx[10] web server to centralize web access. The other virtual machines are used by to deliver the backend services offered by each component.

The VMs have the following specifications:

- Virtual CPUs: 8 cores
- RAM: 16 GB
- Disk size: 16 GB
- Image OS: Ubuntu Server 20.04
- Services Docker is installed, and login is handled remotely via LDAP

The resources are adequate to support some tens of thousand users and they will be augmented according to the computational power or storage needs. The PIMCity project on the FASTWEB data center has at its disposal significant spare computing power that we can allocate to components requiring additional resources.

4.1.3.4 Production Environment realization

The production infrastructure has been set-up on FASTWEB's datacenter. Also in this case, we have established policies and procedures to guarantee data security, confidentiality, integrity and availability.
The actions taken to put in place the security protocols and measures for this environment are similar to those applied in Development Environment:

- **Unique identification and authentication:** Internal corporate user account credential shall be restricted for ensuring appropriate identity, entitlement, and access management and in accordance with established policies and procedures:
  - One account per user on the infrastructure (VMs)
  - One account per user FASTWEB VPN to access the private cloud. The access to the VPN is secured through two factor authentication.
- **Dashboards and administration:** the VMs composing the Production Environment are created on a large private virtual cluster hosted in the FASTWEB premises. The FASTWEB cloud is based on the VMWare cloud manager and managed by the IT staff of the company. The infrastructure responsible for the PIMCity project has an account on the VMWare[11] Dashboard, which is used to administrate and control the VMs. Below, it is attached a screenshot of the dashboard.
- **Authenticated Access:** in order to reach VMs via SSH, component providers must use the VPN made available by FASTWEB IT support. This VPN builds on FortiClient and allows only authorized personnel to access the machines. User who are not equipped with the tool and did not receive the necessary credentials cannot access the machines hosting the EasyPIMS backend. Physical access is allowed only to FASTWEB authorized staff and impossible for PIMCity personnel or any other person.

[10] https://www.nginx.com/

[11] https://www.vmware.com/

- **Backup and disaster recovery:** In order to prevent loss of data and limit service discontinuity, we designed a backup policy which generate full backups of all disks mounted on production machines On a daily basis, the orchestration system performs a complete backup of the state of the 6 virtual machines and stores the backup in a secure physical location different from the location where virtual machines are located. This policy has been implemented by FASTWEB IT support through the administration dashboard. Second, in order to prevent data loss caused by catastrophic events, we also programmed the DBs hosting users' data in production environment to generate periodic mirrored duplicates of the content of DBs, encrypt and copy them to POLITO infrastructure. By doing so, we are reasonably confident that no loss will occur also in case of catastrophic events involving FASTWEB datacenter. POLITO and LST's teams are responsible for restoring the environment within 48 hours in case of a failure.
- **HTTP Routing:** also in this case services are exposed to the web via an Nginx instance that is responsible to route incoming HTTP requests to the proper component. However, For the production infrastructure we opted to use a domain-based routing so that components are mapped to specific internal domains. For instance, for the production environment: https://privacy-metrics.easypims.eu is the URL used by the web app to contact Privacy Metrics backend service. The Nginx acts as an ingress proxy, routing the incoming requests to the corresponding backend server based on the URL domain.

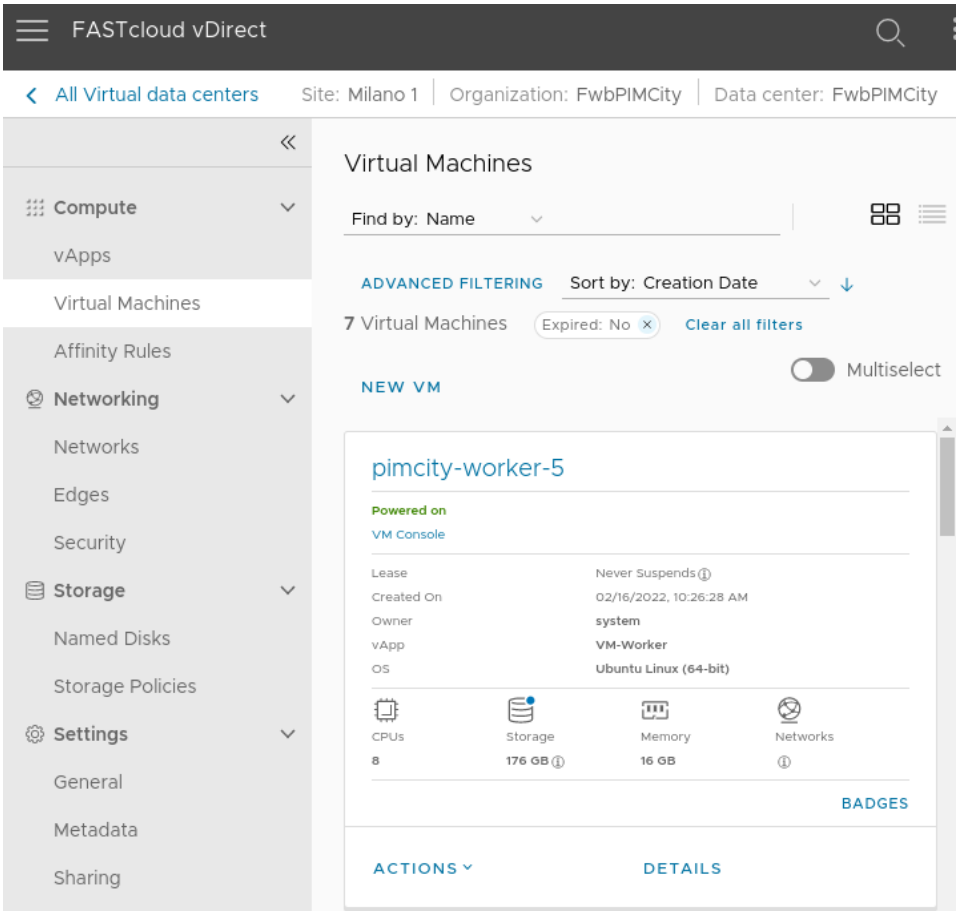


Figure 5 – vDirect Dashboard at Fastweb's premises

5 Design of the Cloud controller

The Cloud controller is the component responsible for orchestrating and coordinating all the interactions occurring among PDK and EasyPIMs components. Such orchestration will be implemented by defining authentication and authorization policies to enable users and components to access resources and perform communications, but preventing unauthorized operations, such as, e.g., a user to access data from other users, a component to read/write data out of its authorizations sphere. Moreover, the Cloud controller will provide end-user management, authentication and single sign-on to all of EasyPIMS and its software.

The Cloud controller defines the policies which will be made operative by the Open APIs implemented by each PDK components. Indeed, as the admin interacts with the Cloud controller to define the policies (e.g., authorization scopes, RBAC, etc.), their application is demanded to the Open APIs which are responsible for activate such policies and control they are respected and fully managed by, e.g., returning proper communications error anytime an unauthorized access occurs.

5.1 Design choices

There exist different models and standard to manage authentication, authorization, and access delegation. For instance, XACML (eXtensible Access Control Markup Language)[12] is an attribute-based access control authorization framework introduced by OASIS project in 2001. Version 3.0 has been released in 2013. Another popular standard is OAuth[13], whose development started in 2006 by Twitter. OAuth is massively deployed nowadays, with all tech giants (Facebook, Google, Twitter, Microsoft, etc.) implementing and deploying it for their users. Indeed, it allows admins and developers to combine its authorization capabilities with access tokens, which are commonly used in web applications. The second release of OAuth standard, namely OAuth 2.0, came in 2012, and provides specific authorization flows for web application, desktop applications, mobile phones and smart devices. Compared to XACML, OAuth approach is simpler and does not offer the same fine-grained levels of policy definitions. In fact, XACML is quite more flexible and customizable, but, for this, it is also more complex and difficult to implement, test and validate. For this reason, the PIMCity project has decided to use OAuth 2.0 authorization standard to define and deploy the authorization policies.

For what concerns the implementation, there is a considerable market of Authorization Providers in the web. Main players are Auth0[14], Okta[15], Amazon Cognito[16]. These services allow web designers and developers to focus on the design of web and mobile applications, but given their commercial nature, they introduce monetary costs, or they build their business on collected authorization data. For this, The PIMCity consortium opted for a on-premises solution, and in particular, the Cloud controller is an implementation of the open-source authentication and authorization provider Keycloak[17], run on EasyPIMS virtual machines and managed by PIMCity partners. This choice guarantees EasyPIMS does not depend on third parties for user management or authentication, which is crucial since many authentication providers are data-intensive companies (e.g., Google, Amazon) whose user management practices are not aligned with those of the PIMCity project.

Being free/libre open-source software, the choice also helps security, and it minimizes costs – e.g., by not incurring in licensing fees. More in detail, Keycloak provides these features:

- User Registration
- Single Sign-On/Sign-Off across all applications belonging to the same Realm (in our case, across all software components in the EasyPIMS platform)
- 2-factor authentication
- LDAP integration

[12] https://www.oasis-open.org/committees/xacml/

[13] https://oauth.net/

[14] https://auth0.com/

[15] https://www.okta.com/

[16] https://aws.amazon.com/cognito/

[17] https://www.keycloak.org/

- Kerberos broker
- Multitenancy
- Social login

The Cloud Controller has been built by FW and deployed on both EasyPIMS' Development and Production infrastructures by POLITO.

5.2 Authentication and authorization flows

In the following we report the authentication and authorization flows that component providers will adopt to properly secure the communications using the Cloud controller in combination with the Open APIs. More in detail, we foresee to manage two kinds of interactions: User to App, used to authenticate and authorize users willing to interact with a web application, and Machine to Machine, used to authorize components willing to interact with other components. Both approaches leverage access tokens, thus ease the development for both web and mobile applications.

5.2.1 User to App authorization flow

This flow will be used by EasyPIMS users to authenticate on the system. In a nutshell, the user performs the login through the Dashboard to gain access to her data and interact with the components. Under the hood of the Dashboard, in practice, the user authenticates her identity to the authentication provider, i.e., the Cloud Controller. This then generates the tokens and communicates the permissions associated to the user's role ("scopes" in OAuth 2.0 terminology) to the components cooperating to populate the information in the Dashboard. This model will be used in all EasyPIMS subsystems requiring the user to authenticate. These are the Dashboard and the Personal Data Avatar where the EasyPIMS end-user is required to login, and the Data Marketplace which provides a web interface for the data buyers.

The steps needed by a user to gain authorization using access tokens are the following. Using OAuth 2.0 terminology, this flow is named Authorization Code flow.

- The user clicks Login within the regular web application.
- The login page redirects the user to the OAuth Authorization Server (/authorize endpoint).
- The Authorization Server redirects the user to the login and authorization prompt.
- The user authenticates using one of the configured login options and may see a consent page listing the permissions OAuth will give to the regular web application.
- The OAuth Authorization Server redirects the user back to the application with an authorization code, which is good for one use.
- The login page sends this code to the OAuth Authorization Server (/oauth/token endpoint) along with the application's Client ID and Client Secret.
- The OAuth Authorization Server verifies the code, Client ID, and Client Secret.
- The OAuth Authorization Server responds with an ID Token and Access Token (and optionally, a Refresh Token).
- Your application can use the Access Token to call an API to access information about the user.
- The API responds with requested data.

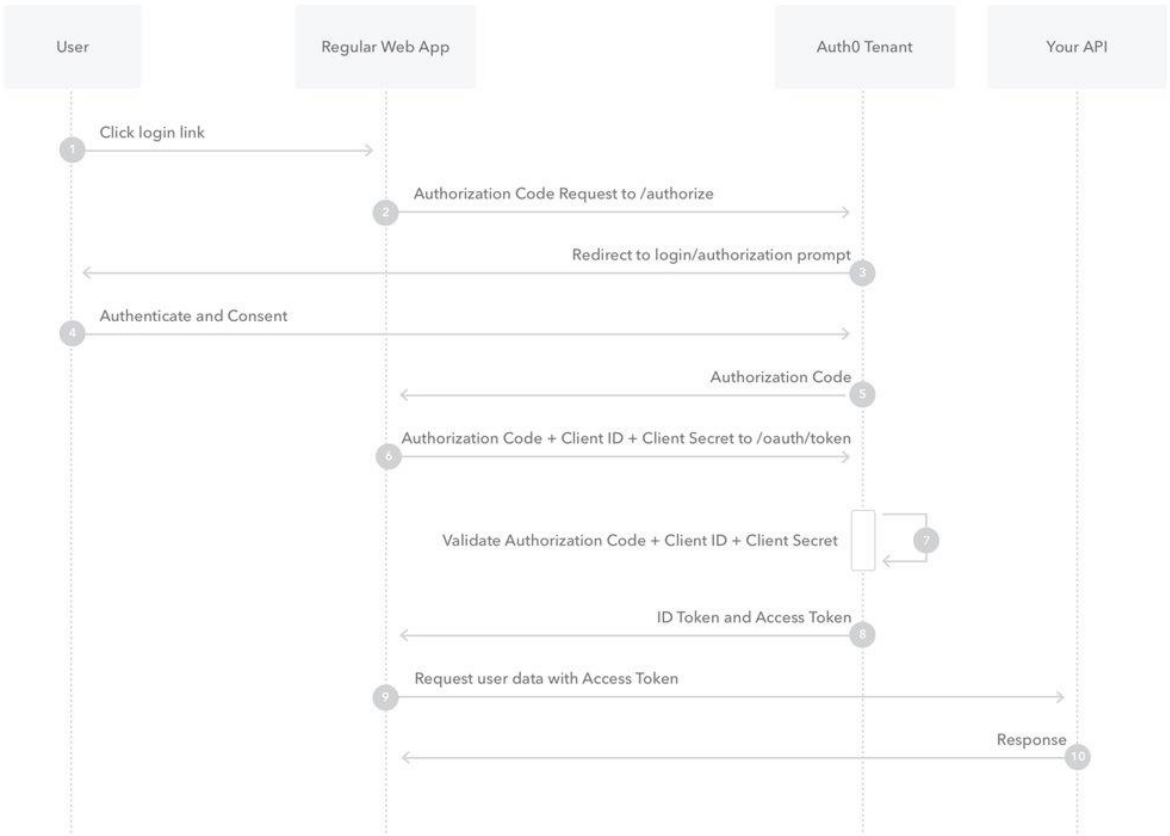


Figure 6 - OAuth 2.0 Authorization Code flow – courtesy of Auth0

A more detailed description of this flow is available on the official Auth0 documentation [2].

5.2.2 Machine to machine

This flow will be used by EasyPIMS components to authenticate on the system, retrieve the authorization configurations and enforce them on each API call. In a nutshell, the component, e.g., the Dashboard web app, has to authenticate to the authentication provider to retrieve the access tokens needed to contact each EasyPIMS components, such as the Privacy Metrics or the Personal Data Safe. Then, based on the tokens, each called component will apply the permission controls needed to verify that the calling component has the authorization to perform the call. For instance, the Dashboard must retrieve the authorization token needed to call the API exposed by Privacy Metrics in read-only mode. Similarly, the Data Marketplace must fetch the access token to allow the data buyer to interact with APIs exposed by Privacy Metrics component with read/write permissions.

There exist alternative approaches to implement M2M authentication such as, e.g., the classic flow based on API keys. However, the enforcement of permissions is more complicated in this case and does not integrate well in the OAuth 2.0 workflow.

The steps needed by an app to gain authorization to interact with another app are the following. Using OAuth 2.0 terminology, this flow is named Client Credentials flow.

- The app authenticates with the OAuth Authorization Server using its Client ID and Client Secret (/oauth/token endpoint).
- The OAuth Authorization Server validates the Client ID and Client Secret.
- The OAuth Authorization Server responds with an Access Token.
- The application can use the Access Token to call an API on behalf of itself.
- The API responds with requested data.

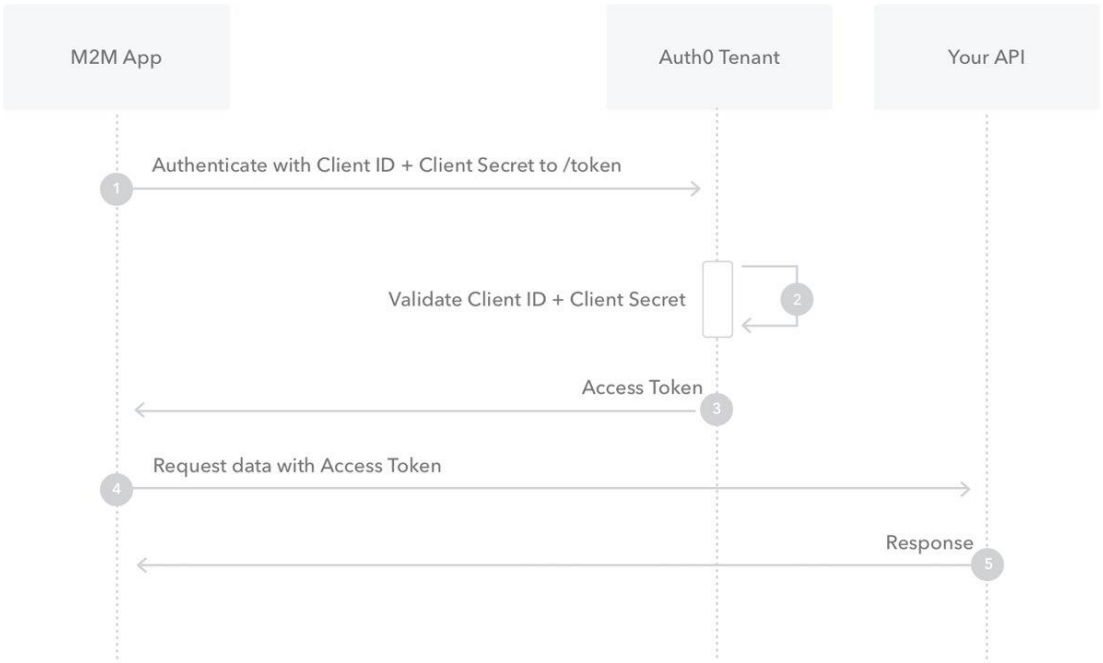


Figure 7 - OAuth2 Client Credentials flow – courtesy of Auth0

A more detailed description of this flow is available on the official Auth0 documentation [3].

6 Design of the Open APIs

Open APIs are a key part of the SDKs as they permit to seamlessly integrate PIMCity components for enabling their interoperability and re-usability. They represent a logic substratum of endpoints which will be designed and implemented uniformly across all components, so that the same security, privacy and scalability requirements will be met overall the deployment.

In this section we describe the principles and requirements that drove us in the design of the Open APIs for all components building PIMCity's PDK and EasyPIMS. Then, we will present the standards we adopted for the design and the tools chosen to accelerate the implementation. Finally, for each component designed in WP2, WP3 and WP4, we will report the summary of the preliminary design of its APIs, following a common, easy-to-understand presentation scheme. More in details, we provide the Open APIs designs for the Personal Consent Manager, the Personal Data Safe, the Privacy Metrics and Privacy-preserving Analytics, whose design and implementation are provided in Deliverables D2.1 [5] and D2.2 [6]. Then, for the Data Valuation Tools from User and Market Perspectives and the Data Trading Engine whose design and implementation are described in Deliverables D3.2 [7] and D3.3 [8]. Finally, for the Data Aggregation, the Data Portability and Control tool, for the Data Provenance and the User Profiling whose design and implementation are provided in Deliverables D4.1 [9] and D4.2 [10].

The first design of Open APIs has been described in Deliverable D5.1 [1], and their code implementation has been published in PIMCity's official Gitlab repository. Since the publishing of D5.1, the code has been updated to introduce improvements and fix bugs.

6.1 Design principles

The design of Open APIs has been driven with three fundamental objectives in mind: security, privacy and scalability. All these three aspects are key for the design of any modern application. Indeed, with the advent of modern programming languages and development tools, the effort of software engineers and developers has been profoundly eased and reduced, so that design and implementation efforts may be dedicated to other aspects as important as functional ones, i.e., security and privacy. On the other hand, the regulatory frameworks have evolved to require software vendors to put particular attention on security and privacy, introducing penalties and fees to punish bad design choices which go against these principles. The result of this evolution is that we observe a great growth of design approaches built on security- and privacy-by-design paradigms.

The software developed within PIMCity cannot be an exception. Instead, given the spirit of the project, we believe it should go a little beyond SOA practices, and we did our best to implement the most modern and innovative design solutions to guarantee security, privacy and transparency for all stakeholders involved without neglecting scalability and efficiency.

In this section we present the design principles that each component provider must follow for the design of the corresponding Open APIs. The authentication and authorization flows (OAuth2 Client Credentials and Authorization Code) that the consortium agreed to adopt are presented and detailed in Section 5.

In the following we report the set of design recommendations that have been considered for the design of PIMCity's Open APIs.

6.1.1 RESTful design

REST is acronym for **RE**presentational **S**tate **T**ransfer. It is an architectural style for design of distributed systems that was first presented by Roy Fielding in 2000 in his PhD dissertation. The key abstraction of information in REST approach is a resource. Any information such as a document, an image, a temporal service or a collection of objects is a resource, and all resources are associated with an identifier. REST uses such resource identifiers to identify specific resource involved in the interaction between components. The state of the resource is known as resource representation which consists of data, metadata describing the data and links which help the clients in the transition to the next desired state. The second most important item of the RESTful approach is the resource method which is used to perform the desired transition. The most important guideline to follow is the Surprisingly though, methods do not relate to HTTP methods, I.e., GET/POST/PUT/DELETE. Indeed, it is important that interfaces across all components are uniform. Nevertheless, the most common protocol for these requests and responses is HTTP, whose methods adapt very well to CRUD (Create, Read, Update, Delete) operations: GET -> Read, POST -> Create, PUT -> Update, DELETE -> Delete.

The REST approach builds on 5 fundamental principles:

- **Client-server**. The REST philosophy separates user interface from the data storage, so that we can improve the portability of user interface across multiple platforms and improve scalability by simplifying the server implementation
- **Stateless**. Each request from client to server must contain all the information necessary to understand the request and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.
- **Cacheable**. Cache constraints require that data in a response to a request has to be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.
- **Uniform interface**. By applying the engineering principle of generality to the component interface, the general architecture of the system is simplified, and the visibility of flows is improved. To obtain a uniform interface, multiple architectural constraints are needed to shape the behavior of components. REST is defined by four constraints on interfaces: identification of resources; manipulation of resources through representations; self-descriptive messages; and hypermedia as the engine of application state.
- **Layered system**. The layered system approach allows to compose the architecture with hierarchical layers by constraining component behavior so that components have no visibility beyond the layer with which they are interacting.

There is also a last principle, names “Code on demand” which would allow the client to extend these functionalities by downloading and executing code on demand from scripts. This may simplify clients by reducing the number of features to be implemented, but it also exposes the client and the overall system to severe security risks.

Despite being only an architectural style, RESTful became a popular alternative to actual protocols like SOAP (Simple Object Access Protocol), which deeply builds on XML messaging and has some built-in security compliance that makes it slower and heavier. In contrast, REST is a set of guidelines that can be implemented as needed, making REST APIs faster, more lightweight and scalable, especially for mobile app development and Internet of Things (IoT).

6.1.1.1 Best practices for API design

RESTful API design is well documented and there exist many tutorials and books describing the best practices to follow for their implementation and deployment. We resume them in the following:

- **Use JSON**: REST APIs should handle JSON for requests and responses. JSON is indeed the standard for transferring data, and it is used by a number of popular technologies such as Javascript on the client side and NoSQL databases on the backend side. For this, HTTP Content-Type header should always be set to application/json

- **Use nouns and not verbs in paths**: endpoints in paths represent the resources or the entity that we aim to manipulate. Hence, it is a good practice to not use verbs. These are indeed represented by the HTTP method (or CRUD operation) that we want to execute. In general, the most common methods include the following:
    - GET retrieves resources
    - POST submits new data to the server
    - PUT updates existing data
    - DELETE removes data
    - To give a more practical example, the correct way to fetch the list of privacy metrics is “GET /privacy-metrics”.
- **Use logical nesting on endpoints**: when designing endpoints, it is good practice to group those that contain related information. For instance, if one object contains another object, the design should reflect that structure. For example, for the case of privacy metrics of a website, if we want an endpoint to get the scores for the same website, we should append the /scores path to the end of the /privacy-metrics path: /privacy-metrics/{id}/scores
- **Handle errors and return proper status codes**: it is important to eliminate confusion sources for users when an error occurs on APIs. For this, we should handle errors gracefully and return HTTP response codes indicating what kind of error happened. First, we must manage errors so that our system does not fail, and second, we must provide the user or component interfacing with the API enough information to understand what is going on and react properly. Common error HTTP status codes include:
    - 400 Bad Request – client-side input fails validation.
    - 401 Unauthorized – the user is not authorized to access a resource. It usually returns when the user is not authenticated
    - 403 Forbidden – the user is authenticated, but not authorized to access a resource.
    - 404 Not Found – that resource is not found.
    - 405 Method not allowed – this method is not allowed on the specific endpoint.
    - 406 Unacceptable – The client provided a content type in the Accept header which is not supported by the server.
    - 413 Payload too large – Request size exceeded a given limit
    - 415 Unsupported Media Type – Requested content is not supported by the server
    - 429 Too many requests– the caller has overcome the number of requests it can perform in a given amount of time.
    - 500 Internal server error – This is a generic server error that should not be thrown explicitly.
    - 502 Bad Gateway – This indicates an invalid response from an upstream server.
    - 503 Service Unavailable – This indicates that something unexpected happened on server side (It can be anything like server overload, some parts of the system failed, etc.).
- **Provide filtering, pagination and sorting**: As databases behind REST APIs might grow very large, it is important to expose functions to reduce the number of items contained in a single API response. In fact, the system may take too long to answer in case of the number of items in the response is too large, or in the worst case, it can break down. For this, filtering functionality helps the API caller to reduce the number of expected items. Second, it is important to limit the number of items returned in the single response systematically: for this, we can introduce pagination option. Pagination should be made available by default, and API requests with no pagination parameters should be blocked to prevent the system to collapse. Another, more optional, feature to provide is sorting to help the API caller to iterate easily on the set of items returned in a response.

- **Caching**: use a local memory cache instead of querying the database to get the data every time we want to retrieve some data that users request. This leads to a win-win, so that, on the one hand, users can get data faster, while we also reduce the workload on the database. However, it is important to add features to check that data that users get is not outdated. Etags may help here.

6.1.1.2 API Security and Privacy

We now focus on the design part addressing security and privacy in RESTful APIs. We treat these aspects in a separate subsection as these are fundamental in the context of this project, especially because we will treat users' personal data, by far the most valuable asset in the systems developed in this project. In the following we list the must-have requirements we must address in the implementation of the APIs of PDK components and deployment of EasyPIMS.

- **Encryption -** All communications between client and server must be private since we often send and receive private information. Therefore, we will use TLS to encrypt communication channels carrying API messages. With the advent of free Root CAs such as Let's Encrypt, certificates have become very cheap to get, deploy and maintain.
- **Authentication –** The process of verifying that the API caller is whom it claims to be. This aspect is necessary to identify caller's role and authorization scopes. This is commonly performed by submitting username (in case of human users) or ID (in case of machine-to-machine interactions), and one or more piece of information which only the caller should know (e.g., a multi-factor token). Authentication must be centralized in an Identity Provider (IdP), or an Authentication Provider (IdP which performs shallow forms of identification).
- **Access control and Authorization –** API callers must not be able to access more information that they requested and access control must be performed at each API endpoint. For example, a user must be able to access its own APIs, but must not be able to access information of another user, or even worse of admins, or any other role considered in the service. This is the so-called *least privilege* principle. To enforce it, we must apply strict policies based on Role-Based Access Control (RBAC in short), for which roles must be defined with fine (ad-hoc role for single user) or coarse (one role for a whole group of users) granularity. If we choose to group users into a few roles, then the roles must provide the permissions that cover all they need and no more. If we set more granular permissions for each feature that users/groups have access to, then we have to make sure that admins can add and remove those features from each user accordingly. Then, authorization policies define the permissions associated to each caller-enpoint couple. For instance, we can allow the user to read a piece of information (allow GET), but not to modify that same information (block PUT, PATCH and DELETE). In general, we must ensure the "block all/allow some" policy is adopted by design, so that all permissions are disabled by default, with just a few exceptions. The access control decision is applied locally by REST endpoints, but their definition might be provided at the authentication provider. This is the case, e.g., of OAuth2.0 implementation.
- **Session management –** This is the process required by the server to remember state of the client with which it is interacting. Sessions are maintained on the server using an identifier which can be passed back and forward between the client and server when transmitting and receiving requests. Session identifiers must be unique per user and computationally very difficult to predict. JSON Web Tokens (JWT) are currently widely adopted to handle session management in the format for security tokens. JWTs are JSON data structures containing a set of claims that can be used also for access control decisions. A cryptographic signature or message authentication code (MAC) can be used to protect the integrity of the JWT.
- **Rate limiting -** The server API must implement rate limiting policies to prevent the caller to generate too many requests in a given amount of time. Limiting may vary depending on the load required by the API to return a result, so that more CPU-intensive APIs should be called with lower frequency. In the extreme case, exceeding rate limiting may lead to the ban of the JWT, or the IP address from which requests have been generated.

- **Input validation –** Each endpoint must implement input validation to prevent the caller to introduce in the payload pieces of information which do not respect the expected ones. In general, we cannot trust any input parameter, and we must validate its length, range, format and type. Implicit input validation is achieved by using strong types like numbers, booleans, dates, times, etc. When possible, we must constraint inputs using regular expressions. All content which does not look like expected must be rejected. Secure parsers are very useful for input validation, especially when dealing with complex data structures. In this case indeed, we must define proper models or schemas and match them against input to block corresponding request.
- **Validate content types –** Similarly as above, request content types must be validated and requests with unexpected or missing content type headers must be rejected.
- **Audit logs –** PIMCity's systems must be equipped with audit logs to allow admins to have enough information to perform analysis upon security incidents. This is indeed the first source of data for investigation. Audit logs must report at least the following information: an identifier of the user performing the action, the timestamp, the action performed, the endpoint involved, the user agent and the IP address from which the connection was established.
- **Security Headers -** There are a number of security related headers that can be returned in the HTTP responses to instruct browsers to act in specific ways. However, some of them are intended to be used with HTML responses only, and may provide little or no security advantages on an API not returning HTML.
- **Schema and Database structuring -** In general, it is advisable to avoid mirroring the database structure in the endpoints to avoid giving attackers unnecessary information. This comes quite natural when dealing with relational databases, where data can be fetched from different tables, and APIs can follow quite different design. On the other hand, when using document-oriented non-relational databases one tends to create collections of objects with nested structure. In this case, it is easier that the API structure will reflect how data is actually structured in the database. For this, we recommend to build schemas of nested well-defined objects, so that access controls, permissions and authorizations can be define specifically per object with fine granularity.

In the following we present the standards and tools we decided to use all over the project for the design and implementation of Open APIs.

6.2 Design Standards, Specifications and Tools

6.2.1 Standards and Specifications

In this project we decided to adopt technologies that will allow us to speed up design and implementation of APIs, while still being standard-de-facto solutions widely adopted in the community of developers and software engineers. Moreover, the choice has been driven by the expertise of some of the partners involved in the project.

Specifically, for the development of RESTful APIs, the choice fell on the standard-de-facto OpenAPI Specification[18], originally known as the Swagger Specification. It has been introduced to define machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services. OpenAPI was originally part of the Swagger framework, and became a separate project in 2016, overseen by the OpenAPI Initiative, an open-source collaboration project of the Linux Foundation. The OpenAPI specification has reached its third edition and version 3.1.0 was released in February 2021.

The OpenAPI has reached this popularity for a number of features that made it welcome in the community of software developers. In fact, the OpenAPI interface files can automatically generate documentation of methods, parameters and models, considerably reducing the efforts to generate and maintain the documentation, client libraries, and source code in sync. Moreover, the OpenAPI specification is language agnostic, so that clients can understand and use services without the need of knowing the server implementation or its source code.

6.2.2 Tools

The easiest way to design and implement RESTful APIs based on OpenAPI specification is using the Swagger Framework[19]. Swagger is an Interface Description Language for describing RESTful APIs using JSON or YAML languages. It comes with a set of open-source tools which deeply reduce the costs and efforts to design, build, document and use web services APIs based on RESTful approach. Indeed, Swagger includes automated documentation, code generation (in many programming languages, for both client and server) and test-case generation.

The Swagger Codegen[20] project supports over 50 different languages and formats for client and server SDK generation. Specifically for this project, we use Node.js and Python-Flask server implementations. For what concerns the client, we use HTML v1 generation of client SDK to create the documentation which will be provided in the next pages.

6.2.2.1 Usage

Swagger's open-source tooling usage can be divided into different phases: development, interaction with APIs, and documentation:

- **Developing APIs:** When creating APIs, Swagger tooling may be used to automatically generate an OpenAPI document based on the code itself. This embeds the API description in the source code of a project and is informally called code-first or bottom-up API development. Alternatively, developers can rely on Swagger Codegen to decouple the source code from the OpenAPI document, and then generate client and server code directly from the design. This makes it possible to defer the coding aspect. The API development can be conducted using standard IDEs for software development. For instance, Visual Studio Code includes a plugin to automatically build debug and build previews of APIs using Swagger UI format. Similarly, the online tool Swagger Editor[21] implements the same features and integrates the export functionalities provided by Swagger Codegen.
- **Interacting with APIs:** Swagger Codegen generates client and server SDKs directly from the Open API document (in YAML or JSON format), reducing the amount for human-generated code. This allows the developer to focus on the logic behind the APIs, notably reducing the development and testing efforts. For instance, the server SDK allows to access pre-compiled HTML-based UI to check and test APIs, thus easing and accelerating the development process. When described by an OpenAPI document, Swagger open-source tooling may be used to interact directly with the API through the Swagger UI[22]. This project allows connections directly to live APIs through an interactive, HTML-based user interface. Requests can be made directly from the UI and the options explored by the user of the interface.
- **Documenting APIs:** The documentation needed to describe the APIs is automatically generated again using Swagger Codegen. Also in this case, both the standalone command-line tool and the online tool Swagger Editor provide this functionality.

In the following we report the preliminary design of the Open APIs for each PDK component.

6.3 Design of APIs components

6.3.1 WP2

6.3.1.1 Personal Consent Manager

[18] https://www.openapis.org/

[19] https://swagger.io/

[20] https://swagger.io/tools/swagger-codegen/

[21] https://editor.swagger.io

[22] https://github.com/swagger-api/swagger-ui

The Personal Consent Manager (P-CM) is the means to define all the user's privacy preferences. It defines which data a service is allowed to collect, process, or which can be shared with third parties by managing explicit consent. Users' settings are imposed on all participating systems. The P-CM is described in Section 3 of Deliverable 2.2 [5] and is available online as an open-source project[23]. The Open API implementation of Personal-Consent Manager component is available at official PIMCity's Gitlab code repository, under WP5 project[24].
The Open API of the P-CM offers data subjects and components on behalf of them a way to store all the up-to-date users' consents. Next, we provide an overview of the APIs.

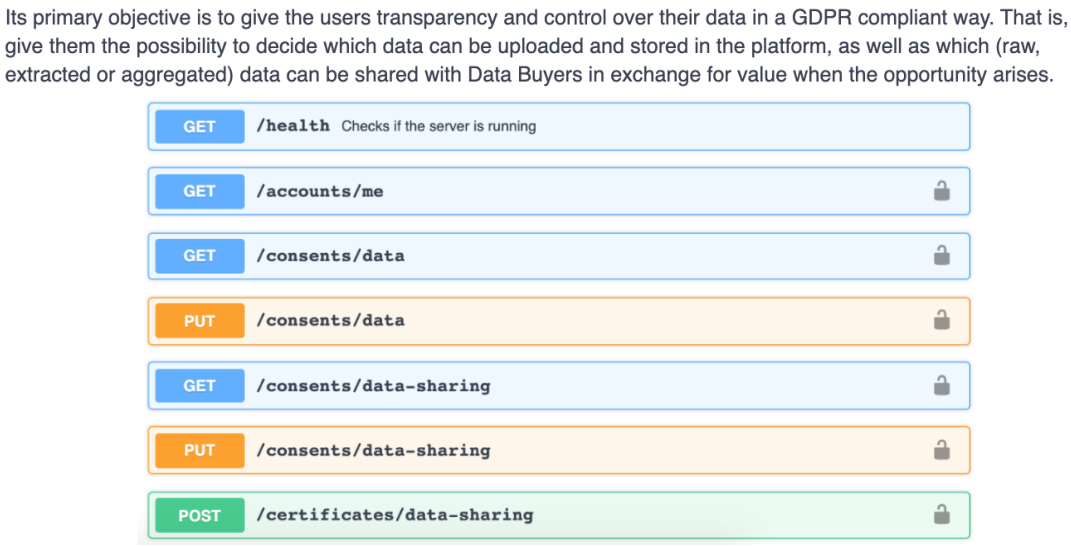


Figure 8 - Open APIs for Personal Consent Manager

6.3.1.2 Personal Data Safe

The Personal Data Safe (P-DS) is the means to store personal data in a controlled form. It implements a secure repository for the user's personal information like navigation history, contacts, preferences, personal information, etc. It is described in Section 6 of Deliverable 2.2 [5] and is available online as an open-source project[25]. The Open API implementation of Personal Data Safe component is available at official PIMCity's Gitlab code repository, under WP5 project[26].

The OpenAPI of the P-DS offer data subjects and components on behalf of them a way to store, update and manage their personal information stored on the system. It also offers Data Buyers a means to access users' personal information upon receiving consent though the PIMCity Personal Consent Manager. Next, we provide an overview of the APIs. The PDS can hold local account or authenticate users using the OAuth mechanism described in Section 5.2.

With respect to the specifications hold in the previous deliverable, we improved the management of remote OAuth authentication. Moreover, we introduced new endpoints to better integrate with other components (e.g., an HTTP endpoint to perform bulk insert of PI), and we integrated the PDS and the Task Manager components to communicate with each other.

[23] https://gitlab.com/pimcity/wp2/personal-consent-manager
[24] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP2/personal-consent-manager.yml
[25] https://gitlab.com/pimcity/wp2/personal-data-safe
[26] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP2/personal-data-safe.yml

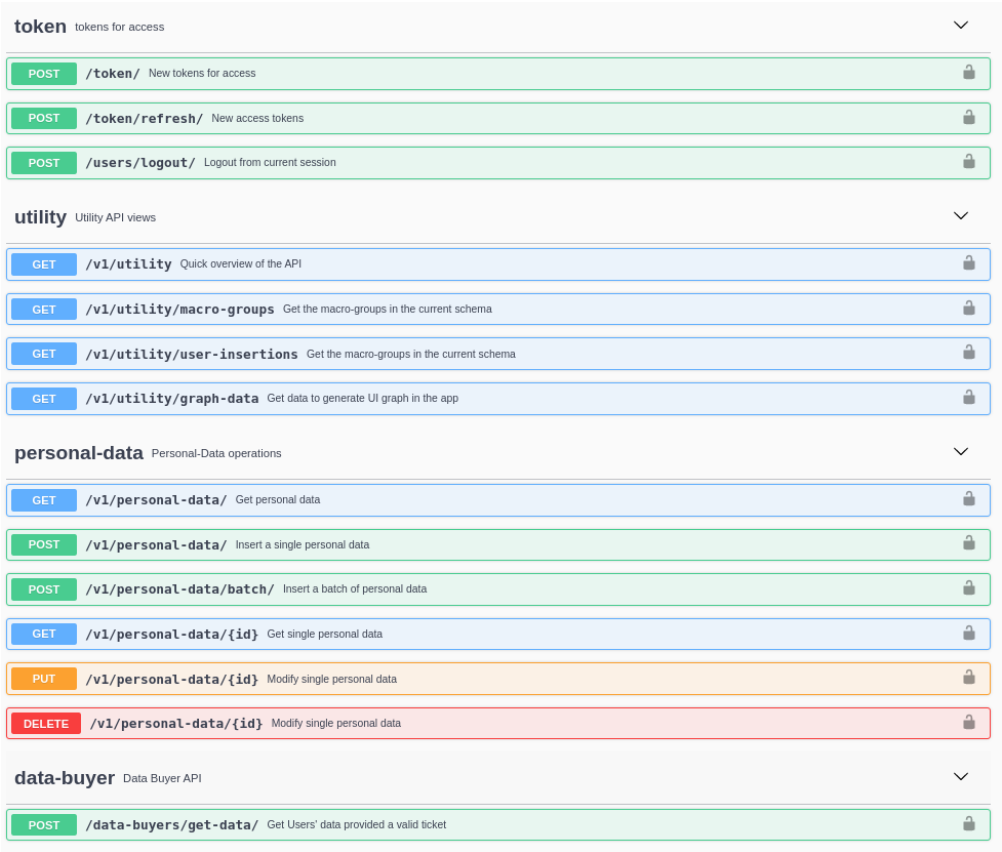


Figure 9 - Open APIs for Personal Data Safe

6.3.1.3 Privacy Preserving Analytics

The Personal Privacy Preserving Analytics (P-PPA) module has the goal of allowing data analysts and stakeholders to retrieve useful information from the data, while preserving the privacy of the users whose data are in the studied datasets. It leverages concepts like Differential Privacy and K-Anonymity so that data can be exchanged among different systems while preserving the actual information as private. It is described in Section 5 of Deliverable 2.2 [5] and is available online as an open-source project[27]. The Open API implementation of Personal-Privacy Preserving Analytics component is available at official PIMCity's Gitlab code repository, under WP5 project[28].

The Open APIs of the P-PPA offer data buyers and any software component a mean to anonymize data, according to a set of predefined algorithms. Next, we provide an overview of the APIs.

[27] https://gitlab.com/pimcity/wp2/personal-privacy-preserving-analytics

[28] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP2/privacy-preserving-analytics.yml

The PIMCity P-PPA, it's a tool to allow data analysts and stakeholders to retrieve useful information from the data, while preserving the privacy of the users whose data are in the studied datasets.
It follows some query example.

As we can see, this is an example query to k-anonymize a dataset taken from a postgreSQL database.
Please note that in this case the parameter **"csv_path"** is useless and will be ignored.

- base_address + `"/kanon"` , {**"token"**:*"tok2"*, **"data_source"**:*"postgre"*, **"csv_path"**:*"../path_to_csv_data"*, **"host_data"**:*"localhost"*, **"port_data"**:*5432*, **"server_data"**:*"postgres"*, **"user_data"**:*"postgres"*, **"pass_data"**:*"provapost"*, **"db_data"**:*"adut_data"*, **"table_coll_data"**:*"adult_table"*, **"k"**:*3*, **"qi_indexes"**:[*3,5,6,9,10*]}

Exploiting the following example query it is possible to retrieve differentially private sum statistic from a mongoDB database.
Please note that

1. the parameters **"user_data"** and **"pass_data"** are not necessary if username and password for the database are not set; or it is possible to set as an empty string as in the example.
2. the parameter **"keepdims"** accepts 0 for False, and 1 for True, in a pythonic flavour.
3. the parameter **"dtype"** accepts the straight type or a string containing the desired type.
4. the **"bounds"** parameter can be also in the form ((1),(100)) or [[1],[100]], but in any case it will be transformed in [1,100]. The first list represents al the minimum bounds, the latter the maximum ones. Each tuple can contain n values, according to the requested query, for instance [min1,min2,min3,max1,max2,max3]
5. the **"axis"** parameter can be a tuple like in a Numpy operation, but can be also a generic array like in the example.

- base_address + `"/diffpriv"` , {**"token"**:*"tok2"*, **"data_source"**:*"mongodb"*, **"host_data"**:*"localhost"*, **"port_data"**:*27017*, **"user_data"**:*""*, **"pass_data"**:*""*, **"db_data"**:*"adult"*, **"table_coll_data"**:*"adult_collection"*, **"column_indexes"**:[*0,2,4*], **"query"**:*"sum"*, **"epsilon"**:*0.6*, **"keepdims"**:*1*, **"dtype"**:*"float"*, **"bounds"**:[*1,100*], **"axis"**:[*0,1*]}

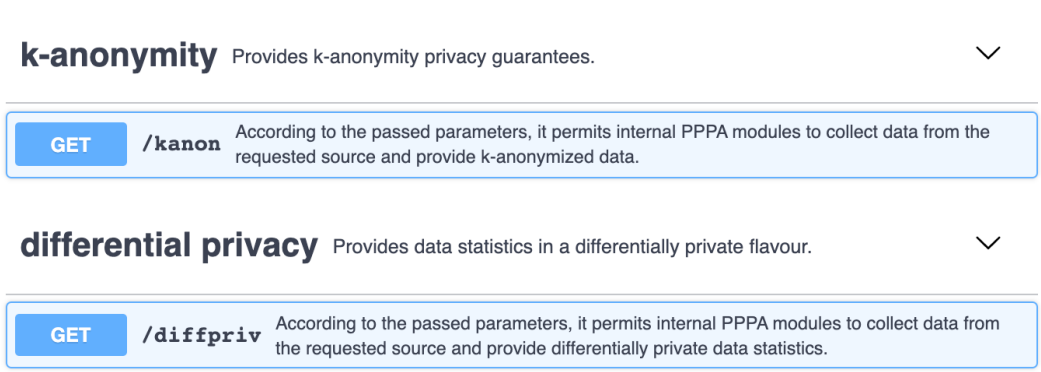


Figure 10 - Open APIs for Personal Privacy Preserving Analytics

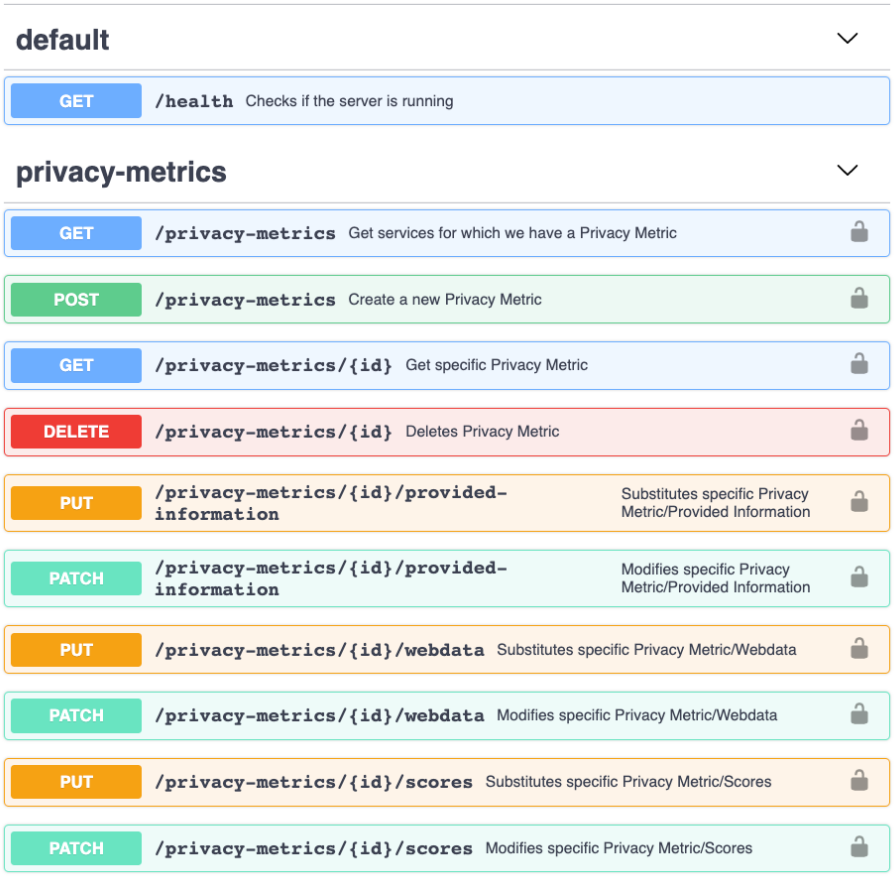
6.3.1.4 Privacy Metrics

Privacy Metrics represent the means to increase the user's awareness. This component collects, computes and shares easy-to-understand data to allow users know how a service (e.g., a data buyer) stores and manages the data, if it shares it with third parties, etc. These are all fundamental pieces of information for a user to know to take informed decisions. The PM computes and offers this information via a standard interface, offering an open knowledge information system which can be queried using an open and standard platform. PMs combine information from supervised machine learning analytics, services themselves and domain experts, volunteers, and contributors.

The Open API implementation of Privacy Metrics component is available at official PIMCity's Gitlab code repository, under WP5 project[29].

In the following we report the design draft of the APIs for the Privacy Metrics as exported by Swagger UI. As shown, for each API (defined by method and endpoint), we report the stakeholders involved, and thus authorized, to use such API. In particular, for the Privacy Metrics component, only three stakeholders are involved: Analytics app generating the Privacy Metrics, Data Buyers updating them with their data, and all stakeholders with access PIMCity, especially Users, who can get data in read-only mode.

- `GET /privacy-metrics` allows **all stakeholders** to obtain the list of services for which Privacy Metrics are available.
- `POST /privacy-metrics` allows **the analytics** to create a new Privacy Metric.
- `GET /privacy-metrics/{id}` allows **all stakeholders** to obtain the Privacy Metric corresponding to `id`
- `DELETE /privacy-metrics/{id}` allows **the analytics** to delete the Privacy Metric corresponding to `id`
- `PUT|PATCH /privacy-metrics/{id}/provided-information` allows **the Data Buyer** to substitutes|modify its Provided Information
- `PUT|PATCH /privacy-metrics/{id}/webdata` allows **the analytics** to substitutes|modify webdata corresponding to Privacy Metric `id`
- `PUT|PATCH /privacy-metrics/{id}/scores` allows **the analytics** to substitutes|modify scores corresponding to Privacy Metric `id`



[29] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP2/privacy-metrics.yml

Figure 11 - Open APIs for Privacy Metrics

The APIs for Privacy Metrics have been slightly modified to accommodate some functional requirements emerged during the development of EasyPIMS platform. In detail, both PDA and Data Marketplace interfaces required some modifications. First, the list of services returned by GET /privacy-metrics has been enriched to return some basic information beyond the mere service domain. These pieces of information (in detail, the scores, the category, and the company owning the domain) have been added to enhance the presentation of Transparency Tags in PDA (more details on this in Section 8.6).

Second, ERMES has introduced some protection mechanisms to protect the Privacy Metrics of a company from unwanted modification. With this protection enabled, only the owner of the Privacy Metrics will be allowed to modify it. The owner is, for instance, the data buyer who creates the Privacy Metric for its own company using the Data Marketplace perspective.

Finally, the server delivering the APIs has been updated to fix minor integration issues such as, for instance, CORS errors occurring in the production environment and the need of having authorization server URL as a configurable parameter.

All these changes have been tracked in the Privacy Metrics code repository hosted on PIMCity's official Gitlab.

6.3.2 WP3

6.3.2.1 Data Valuation Tools – User Perspective

The objective of the Data Valuation Tools from an End-User Perspective (DVTUP) module is to provide estimated valuations of end-users' data for the dataset they are selling through the marketplace as bulk data. DVTUP is an internal module that provides tools for the TE to:

i. Provide buyers with a hint of how valuable a piece of data is for a certain type of model or even for a specific AI/ML task.

ii. Calculate a fair breakdown of data transaction charges by seller, looking forward to rewarding each user proportionally to the value that each piece of data from different sellers brings to the buyer for a specific AI/ML task. Different methods are provided to balance precision and processing time in this valuation task.

The following figure summarizes the calls to the endpoints of the DVTUP that are required to carry out the former valuation processes

Figure 12 – DVTUP endopint usage example

The complete code of the module is available in the project Gitlab[30], and includes sample test models, demand prediction models and proximity detection models based on location data. This module communicates with the Trading Engine using OpenAPI protocol and through two endpoints providing the functions above. The TE must provide the set of users that were part of a transaction and the AI/ML task for which their data is intended to be used to get the value that each user (or group of users) will be providing to the buyer for that specific task. The OpenAPI documentation is available in WP5 code repository of the project[31].

The figure below shows a report from the Module API draft documented by Swagger UI.

[30] https://gitlab.com/pimcity/wp3/dvtup

[31] https://gitlab.com/pimcity/wp3/-/blob/master/DVTUP/DataValuationTool_UserPerspective.yaml

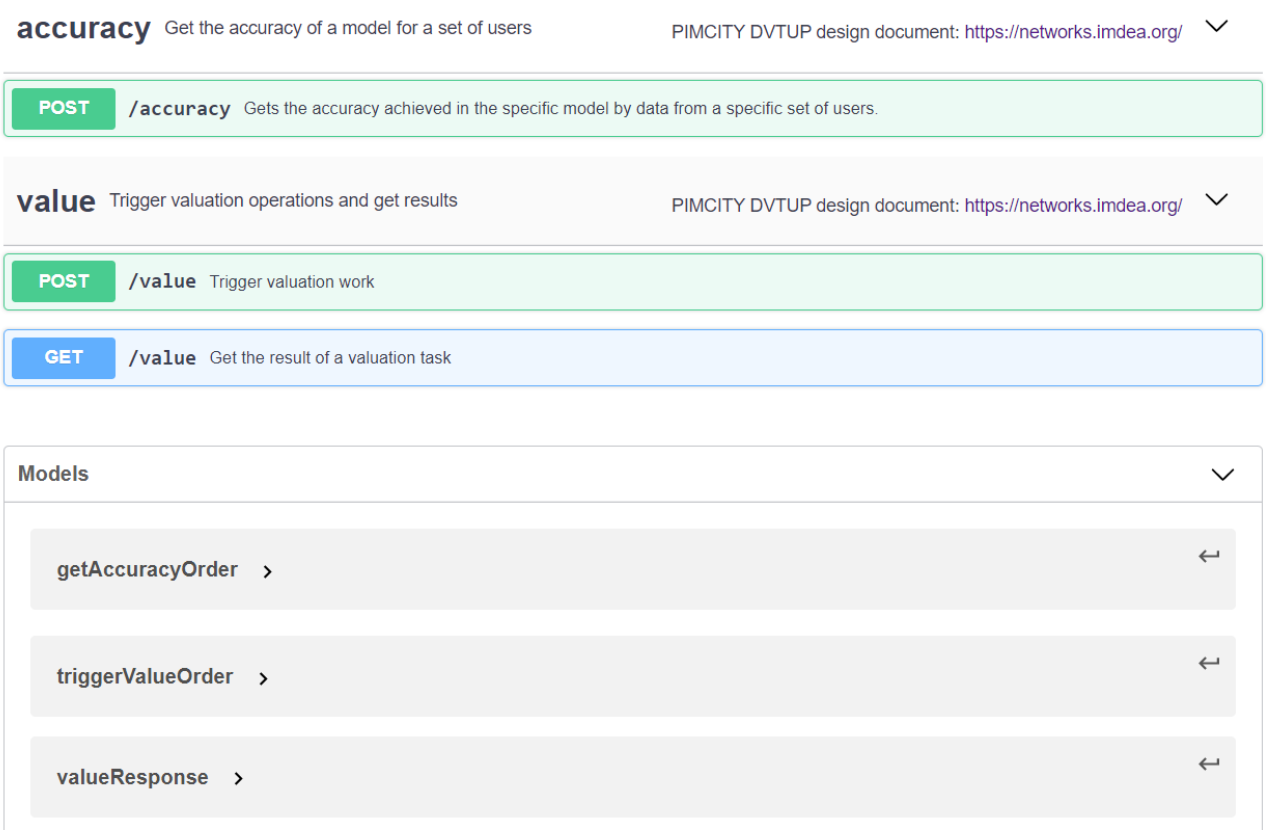


Figure 13 - Design Open APIs for Data valuation tool user perspective

6.3.2.2 Data Valuation Tools – Market Perspective

The Data Valuation Tools from the market perspective (DVTMP) module developed in PIMCity will leverage some of the most popular existing online advertising platforms to estimate the value of hundreds to thousands of audiences. The DVTMP module aims to provide the monetary value of audiences traded on the main online advertising platforms. This will serve any PIM deciding to implement the DVTMP module to have a realistic estimation of audiences' value to be traded. Since the information about values collected from these advertising platforms is based on aggregated historical pricing data, we can assert that DVTMP provides full-privacy guarantees. Moreover, a given audience's value can be obtained in real-time from the referred online advertising platform. In particular, the design of the DVTMP pursues the following objectives:

1. Crawling data value of audiences from Facebook, Instagram, and LinkedIn
2. Process, clean, and curate the collected data
3. Store processed data
4. Provide access to the data through an API

The complete code of the module is available in the project Gitlab[32]. This module communicates with the Trading Engine as an endpoint using OpenAPI protocol. The OpenAPI documentation is available in the WP5 code repository of the project[33]. Below you can find a report from the Module API draft documented by Swagger UI. As can be observed, the only endpoint of the communication for the API is Data Trading Engine. Data Trading Engine sends a POST request to the module server with the specified audience in the body. The module gets the audience value and stores it in a local database.

The DVTMP in its latest update added the authentication protocol and the endpoint to estimate audience values using Instagram and LinkedIn marketing platforms.

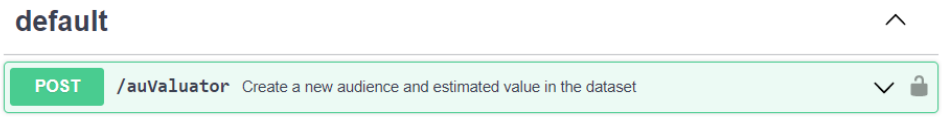


Figure 14 - Design of Open APIs for Data valuation tool Market perspective

6.3.2.3 Data Trading Engine

The primary objective for the Data Trading Engine is to execute all transactions within the platform to exchange data for value in a secure, transparent, and fair-for-all way. Moreover, its key requirement is to be fully GDPR compliant. It must be able to receive Data Offers from Data Buyers and fulfil them with users from the PIMCity platform that fits within the target data sought and have proactively consented to share that data with that company for a specific purpose. The Trading Engine is presented as a REST API exposed to external Data Buyers and internal components from the PIMCity platform. Data Monetization over the internet is still brand new as regulations define new rights and responsibilities, opening new opportunities to include individuals in the value proposition. Therefore, there is no standard way of performing these transactions. With PIMCity's Trading Engine, we aim at:

- Defining a standard interface to carry out transactions where data is exchanged for a value between Data Buyers (i.e., companies and organizations) and End Users.
- Providing a REST API as an implementation of such a standard interface.
- Implementing state-of-the-art techniques using standard community-approved software libraries.
- Putting an integration flow which eases the deployment and allows for quick upgrades to the production environment tool.

[32] https://gitlab.com/pimcity/wp3/-/tree/master/DVTMP

[33] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP3/DVTMP.yaml

- Making the component efficient, fast, and scalable to accommodate a huge number of transactions for a very large set of users and Data Buyers.
- Bookkeeping points earned by end users for sharing data, completing tasks and more.
- Bookkeeping each transaction performed by end users for sharing data, completing tasks and more.
- Bookkeeping credit balances for data buyers.

The complete code of the module is available in the project Gitlab[34]. The Open API documentation is available in the WP5 code repository of the project[35]. Below we report the module API documented by Swagger UI.

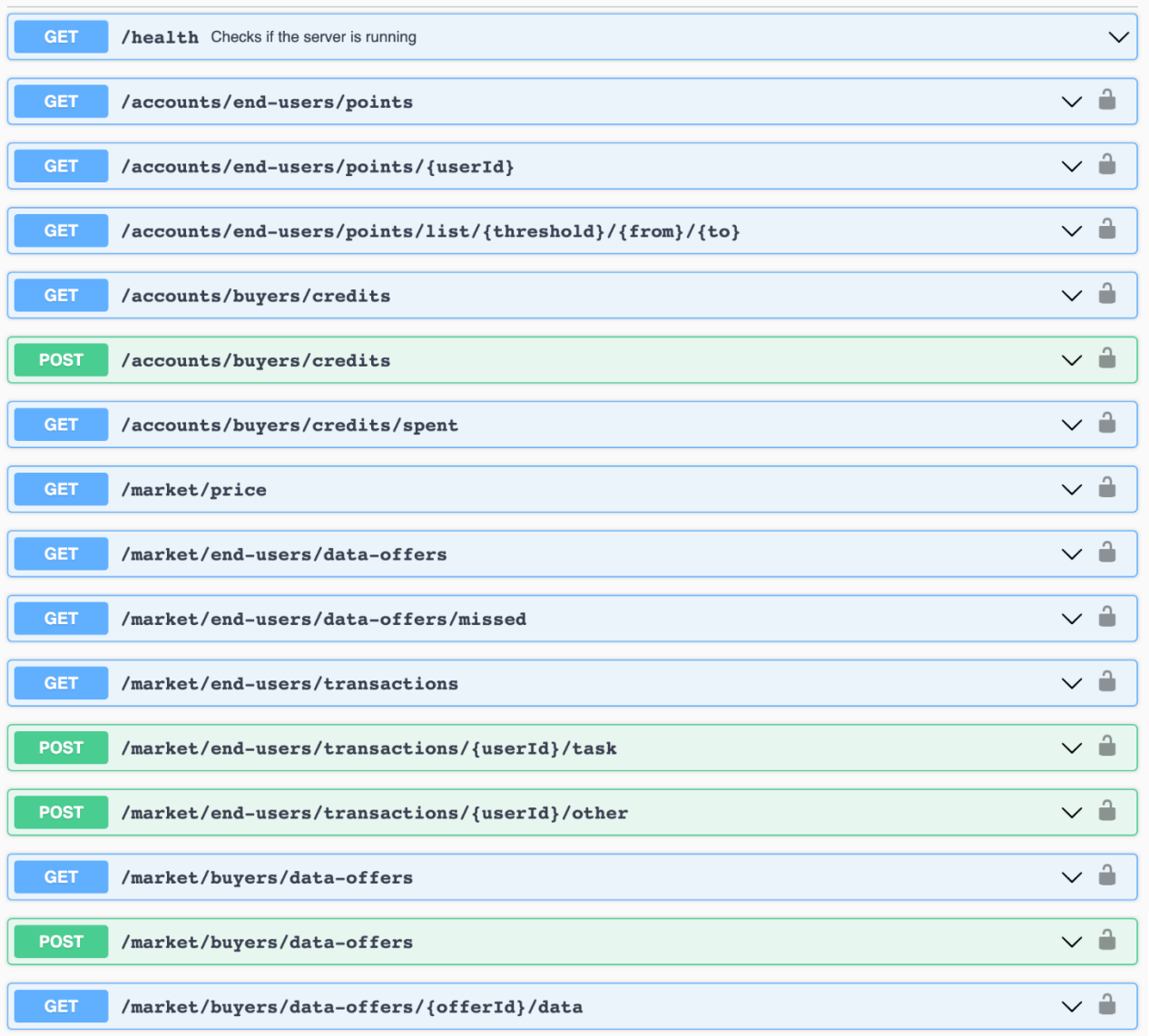


Figure 15 - Design of Open APIs for Trading Engine

[34] https://gitlab.com/pimcity/wp3/-/tree/master/DataTradingEngine

[35] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP3/data-trading-engine.yml

6.3.3 WP4

6.3.3.1 Data Aggregation

The Data Aggregation (DA) tool enables data owners (for example an Internet Service Provider -ISP) that hold a bulk of their users' data to perform two important processes on their data: aggregation and anonymization. This allows them to share these data in a privacy-preserving way. This module resides on the data owner's side. The input to the module is the raw data that is available through the initial sources (telco data, sensor data, etc.) and it is transformed in a predefined schema / metadata model. The user (data owner) is responsible to prepare the data for processing (export from their initial source (internal database), clean them if needed, etc.). Afterwards, through the module, the user is able to choose the subset of the data to be aggregated / anonymized and set the related algorithmic parameters (for aggregation and anonymization). The output is the processed (aggregated / anonymized) data that can be exported to the PIMCity marketplace through an API that the module provides. The data resides on the data owner side and the interested party is able to retrieve them through this API.

This is achieved through designing and implementing the data aggregation and anonymization pipelines in a generic way that allows different types of data to be processed. The tool has been tested on real datasets provided by TID and are mainly related to the end-user's mobility (location, travelling distances etc.). The user (the data owner) is able to select a number of parameters and apply aggregation and anonymization methods and check their effectiveness (achieved anonymization) in a simple user interface. This interface will be a simple web interface with options to visualize the processed data, in the form of tables and graphs using also basic controls over them.

The REST API is based on two public endpoints accessible from outer components and one endpoint only accessible from the Data Aggregation UI. The project is hosted on PIMCity's code repository under WP4 folder. [36]

[36] https://gitlab.com/pimcity/wp4/data-aggregation-api

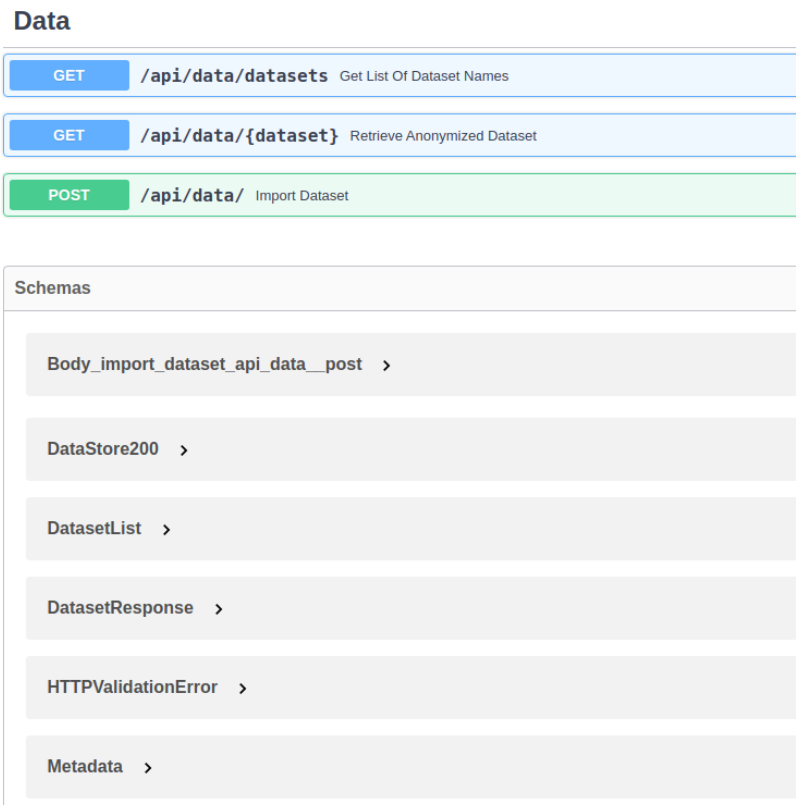


Figure 16 - Design Open APIs for Data Aggregation tool

6.3.3.2 Data Portability Control Tool

The purpose of the Data Portability and Control (DPC) tool is to allow individual users to migrate their data to new platforms, in a privacy-preserving fashion. More specifically, it provides methods for extracting data from one PIMS (e.g. Facebook, bank, mobile phone), process it by filtering out sensitive information such as platform-inferred data (e.g., filter out location from Facebook entries) or user-inputted data (e.g., remove login credentials or debit card numbers), and outport it into the PDK module, a new PIMS (e.g., EasyPIMS) or an exported file in a common data interchange format (e.g., JavaScript Object Notation (JSON)). The Open API implementation of DPC Tool component is available at official PIMCity's Gitlab code repository, under WP4 folder[37].

In the following we report the design of the APIs for the DPC tool as exported by Swagger UI. The API consists of three main sections (i.e., data-sources, transformations, exports). The Data Sources are drivers for supporting specific platforms (e.g., Facebook, TrueLayer Open Banking, Mobile Phone Use from a smartphone). The Data Transformations are modules responsible for the anonymization process of the tool. Finally, the Data Exports are responsible for outporting the data to supported platforms. For more information about the DPC Tool, please refer to Section 4 of the Deliverable D4.2.

[37] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP4/data-portability-control-tool.yaml

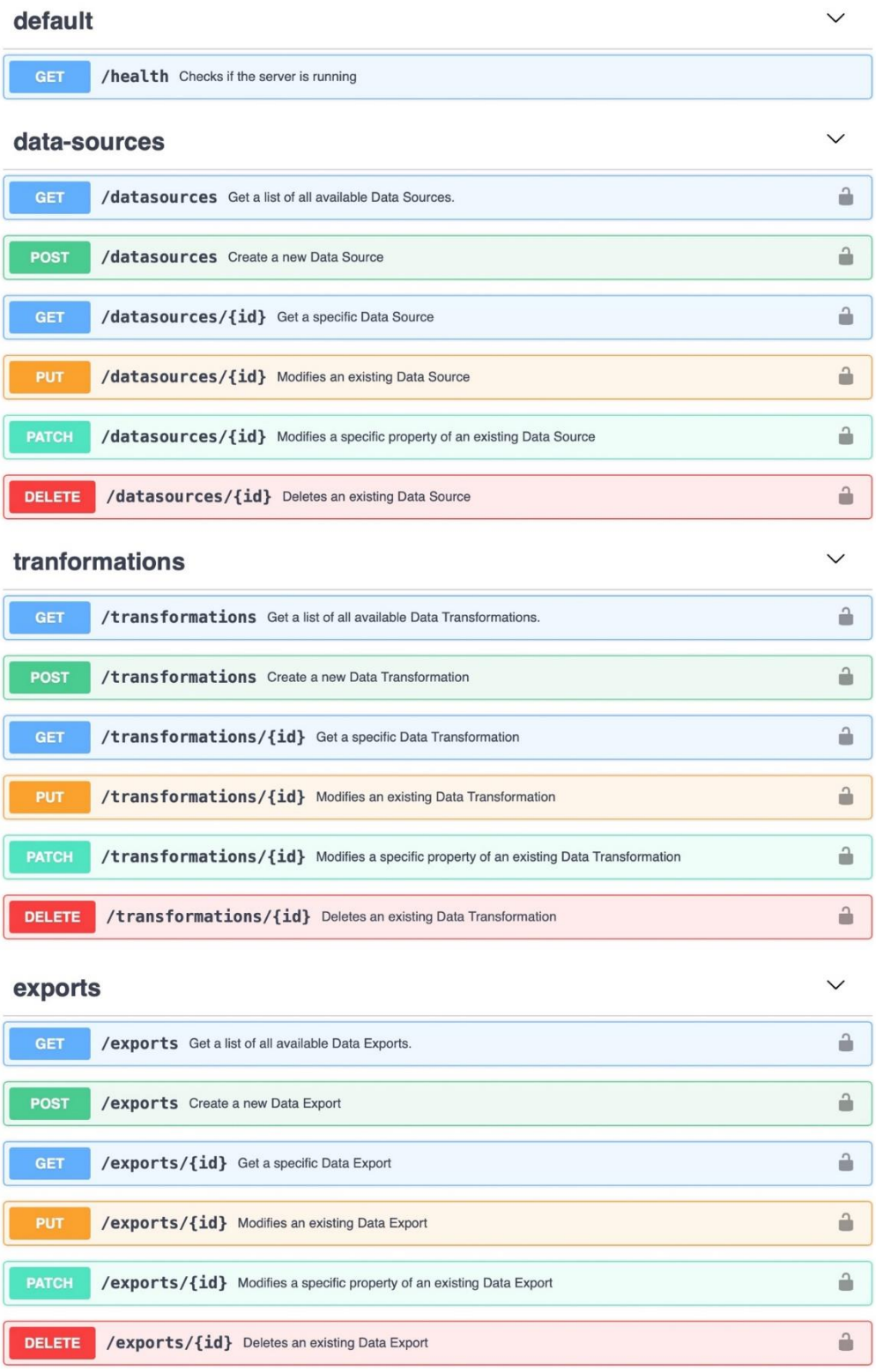


Figure 17 - Design draft of Open APIs for the Data Portability Control Tool

6.3.3.3 Data Provenance

The Data Provenance module OpenAPI allows developers to insert watermarks of ownership in the datasets they share in the PIMCity marketplace eventually. In general, this component is used internally by the PDK and developers that are in need of controlling data ownership even after a dataset has left the PIMCity platform. This is done by embedding difficult to remove watermarks int the datasets. Initially, we provide a simple algorithm, but we are developing more complex algorithms that can overcome the type of attacks we presented in D4.1 [9]. For this development, we have created a series of endpoints for our API as seen in Figure 16, out of which the most relevant for the Data Trading Engine (which interacts with the Data Provenance here) are the two we list below.

POST	*/dp/dataset/{queryId}*	Send id of dataset hash in IPFS to our DP module for watermarking it.
GET	*/dp/wm/job={datasetId}*	Get a watermarked dataset by its id.

Table 1- API endpoints for getting back Watermarked datasets

The above are the main endpoints for watermarking a dataset (POST) and for verifying the watermarking inserted in the dataset (GET). They are used and compatible with requests are coming from the DTE in WP5. In addition, we have added OAuth2.0 to the endpoints that require authorization for security purposes in the flow of data among PIMCity components.

We have tested and processed/retrieved data using our endpoints as said with a large sample dataset of URLs to ensure smooth future integration. The end-users will mainly use two endpoints listed in the following Table 6, which are self-explanatory.

The Open API implementation of our Data Provenance component is available, including any implemented and deployed updates in production, to the official public developer repository of PIMCity for in WP5. [38]

[38] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP4/data-provenance-api-docs.yaml.

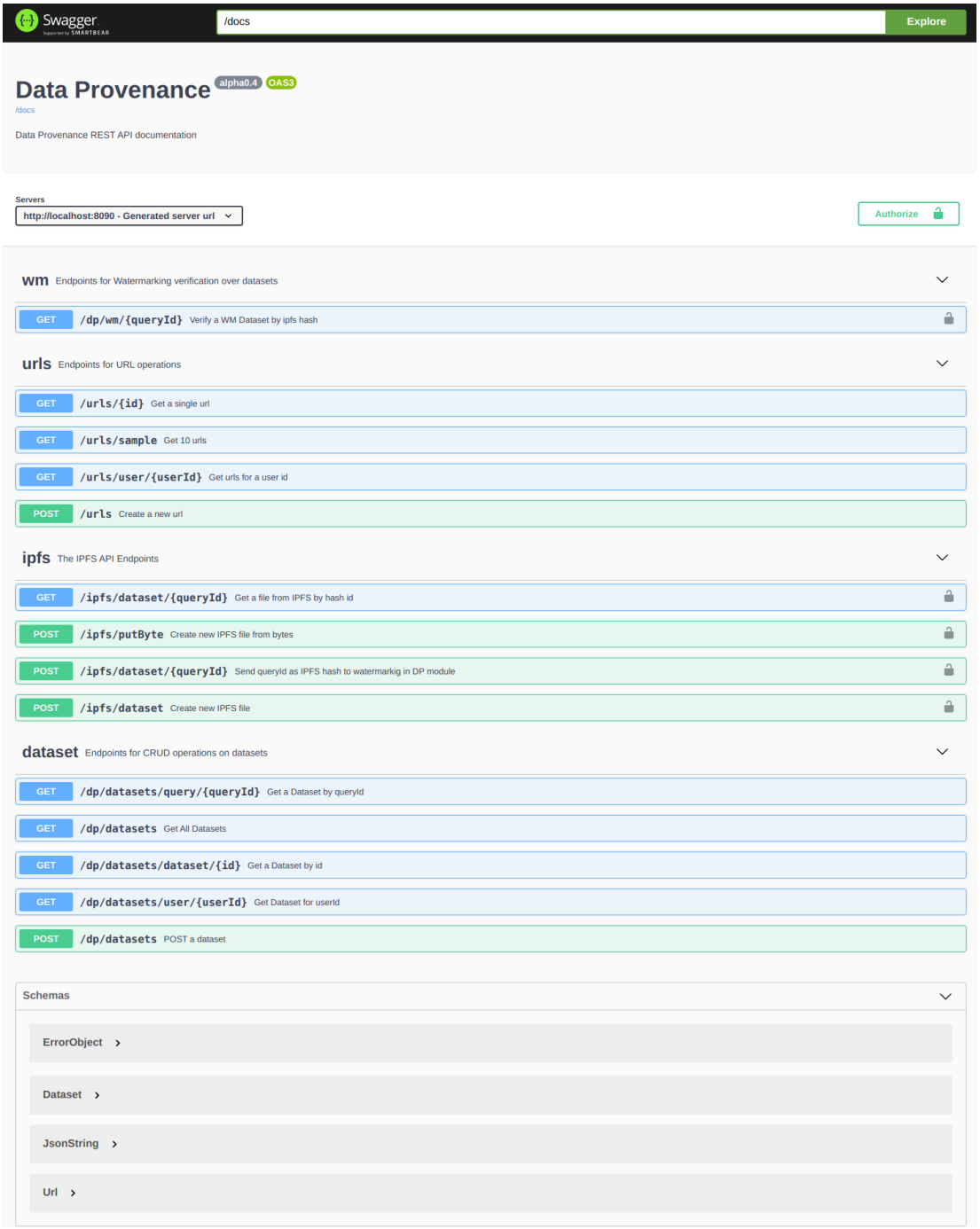


Figure 18 – Design of the Open APIs for Data Provenance tool

6.3.3.4 User Profiling

The user profiling system is a module developed under the umbrella of the Data Knowledge Extraction (DKE) component. This component is the means to extract knowledge from the raw data. One of the biggest challenges in big data and machine learning is the creation of value out of the raw data. When dealing with personal data, this must be coupled with privacy preserving approaches, so that only the necessary data are disclosed, and the data owner keeps the control on them. The DKE consists of machine learning approaches to aggregate data, abstract models to predict future data (e.g., predict user's interests in recommendation systems), fuse data coming from different sources to derive generic suggestions (e.g., to support decision by users, providing suggestions based on decisions taken by users with similar interests).

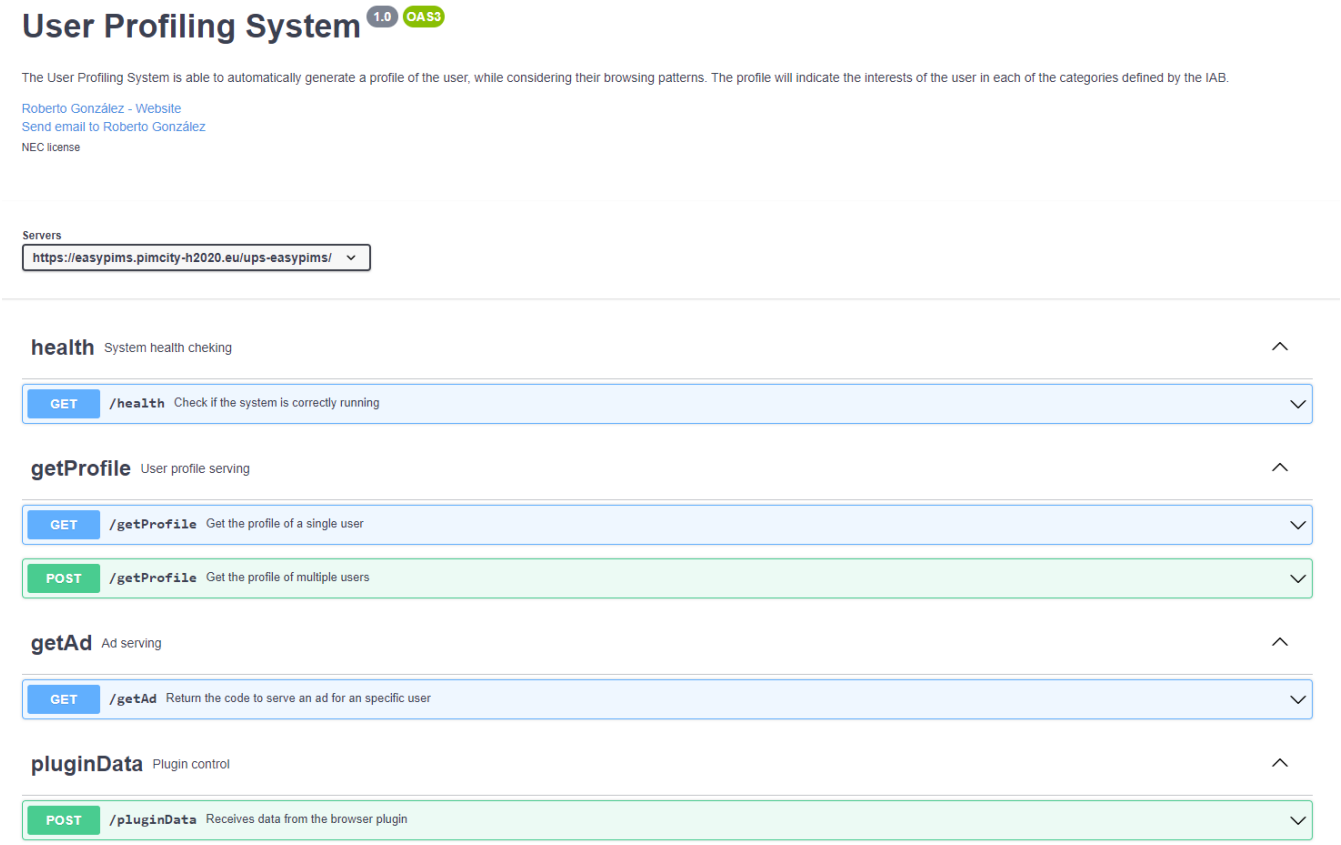


Figure 19 - Design of Open APIs for User Profiling component

In the case of the User Profiling system, the final goal is to create meaningful user profiles that can be used for companies involved in the online advertising ecosystem. To this end, the profiles will be generated using the taxonomy defined by the IAB[39]. Moreover, the system will be able to incorporate different data sources to create a profile as comprehensive and representative of the user as possible.

The system is designed to work in a wide variety of scenarios, allowing the development of multiple use cases (by adding different analysis algorithms) and the addition of custom data sources. However, during the PIMCity project, we will develop those connectors and algorithms that use the network browsing history of the users as input to generate a profile for the online advertising ecosystem. This component will be a key piece of the Personal Data Avatar and will be integrated into the EasyPIMS platform.

The Open API implementation of User Profiling component is available at official PIMCity's Gitlab code repository under WP4 folder.[40]

For the final version of EasyPIMS we have simplified the original API, at the time that we have added support to novel functionalities. Right now, the User Profiling system allows to:

- Obtain a profile for a user or a list of users: the user profiling system will create the profiles on demand by obtaining the user data from the PDS;

- Obtain an Add for a specific user: The user profiling system will first generate a profile for the user, and then will interact with the TapTap Sonata DSP to select the best add;
- Handle the data obtained with the plugin: The server will receive the information from Chrome Extension and will store it in the PDS.

[39] https://www.iab.com/guidelines/content-taxonomy/

[40] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP4/user-profiling.yaml

7 Personal Data Avatar (PDA) and User Dashboard

The PDA is a digital projection of data stored in the P-DS, under full control of the user. PDA contains a set of information synthesized by ad hoc analytics fed with the data made available by the user in the P-DS and the settings specified in the CM. In other words, the PDA is a user-controlled privacy-preserving P-DS synthesis. The PDA is the interface between the user and the services. Thanks to PDA, the user becomes the only owner of her data and acquires the freedom and power of deciding which data to share with which service.

Inside the EayPIMS platform, the PDA is the side of the platform dedicated to interacting with the users. It serves two main purposes, data presentation and data control:

- Data presentation: The PDA allows the user to check both the raw data stored in the P-DS and the different data analysis elaborated by the platform with a special emphasis on the User Profiles and the quantified-self dashboards that provide users an additional value, even if they do not want to trade with their data. Moreover, the PDA provides users with information about the different data transactions involving the user data and allow them to check their "monetization overview".
- Data control: The other important purpose of the PDA is to allow users to control their data. From the PDA, users can exercise their rights to modify, delete or export the data they have stored into the platform. User can also import new data sources. However, that is not the main function that allow the control of their data. Instead, a complete consent manager allows the user to decide how the data of the user is processed and traded.

In practice, the PDA is a responsive web platform that allows users to observe and control the data stored into the EasyPIMS platform.

The PDA is hosted in: https://dashboard.easypims.eu/

7.1 General Design

As for the rest of the components in the PIMCity project, the PDA will be designed in a modular way. In particular, we will use a MVC design to ensure the web views are independent from the PDK components generating the information. It will allow the easy adaptation of the whole platform to different branding options (I.e., if after the project both Telefonica and Fastweb decide to use the platforms with their own colors and web image).

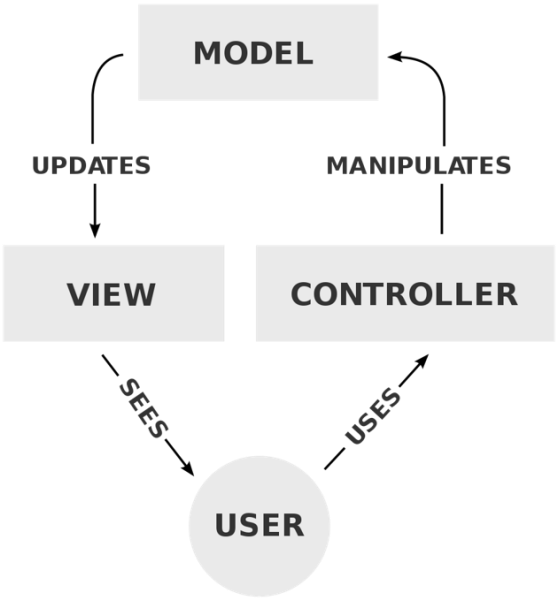


Figure 20 - MVC design of PDA

In this design, the different web views simply show the information to the user and provide an interface acting as the front-end of the different components executing in the platform. That way, the view is presented on the website, the model is stored and managed by the different PDK modules and the Open APIs are used to interact with the different components.

Following, we present a webmap with the different navigation options of the user in the platform:

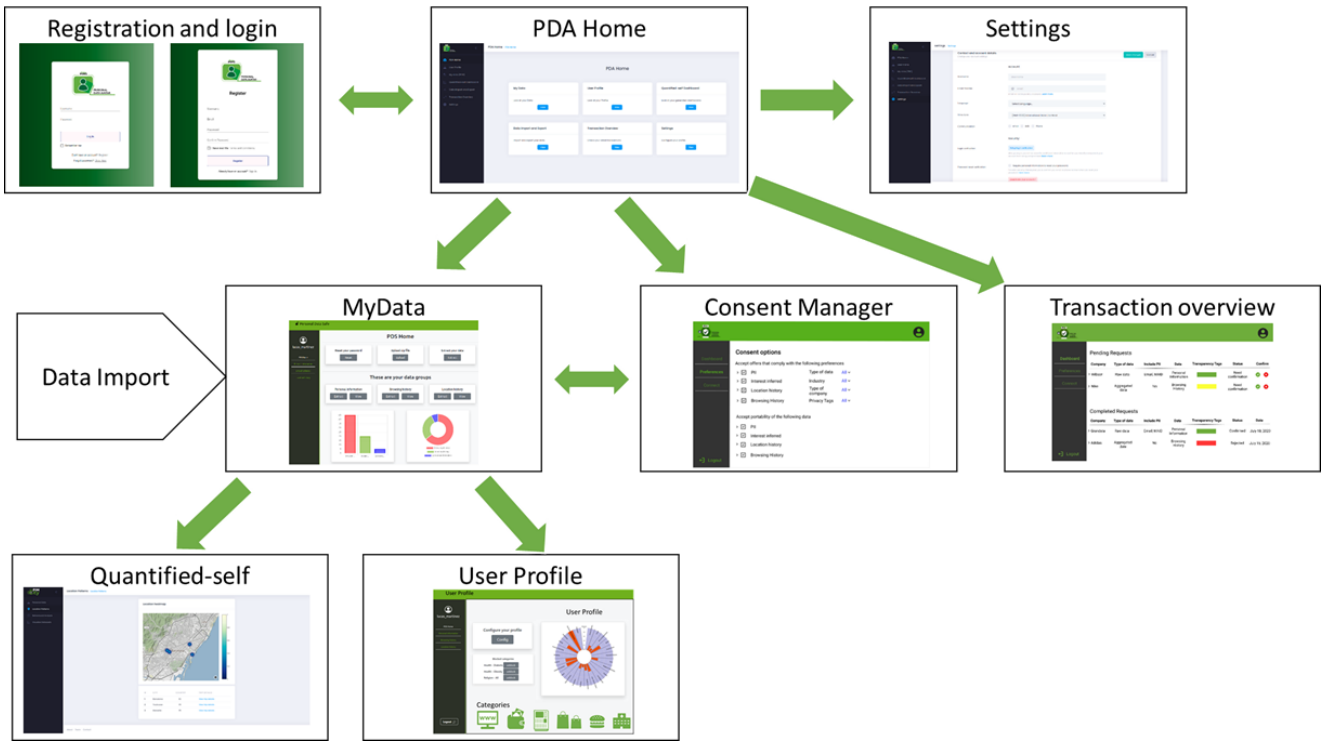


Figure 21 - Webmap with navigation options

As represented above, the external users are provided with a screen for the registration and login into the platform. The first internal screen is the PDA home that presents a summary of the different information present in the platform, it gives access to the rest of the components, and settings such as the password change or the deletion of the account.

Then, a first level of screen composed by the MyData, the Consent Manager and the Transaction overview presents the user the information contained about him in the platform.

Finally, a second level includes the quantified-self dashboards and the profile automatically generated for the user.

The reader can find below a complete description of the different components.

7.2 PDA components

Following the spirit of the whole PIMCity project and the EsyPIMS platform, the PDA is designed in modular way to allow the easy extension and modification of the platform. The PDA makes use of the whole power of the PDK components developed within WP2, WP3 and WP4 in the backend and presents an intuitive web interface to the users.

Following we describe the content of the different screen.

Information screen

Before a user logs into the PDA they can see information about the project in an informative screen.

Functionalities

The information screen contains information about the PIMCity Project and the EasyPIMS platform. It informs the users about the prizes they can win by using the platform and provides an easy way to access the sweepstakes results.
This part of the website is translated to the 7 languages that are more common on the EU: English, Spanish, Italian, German, French, Portuguese, and Dutch.

Design

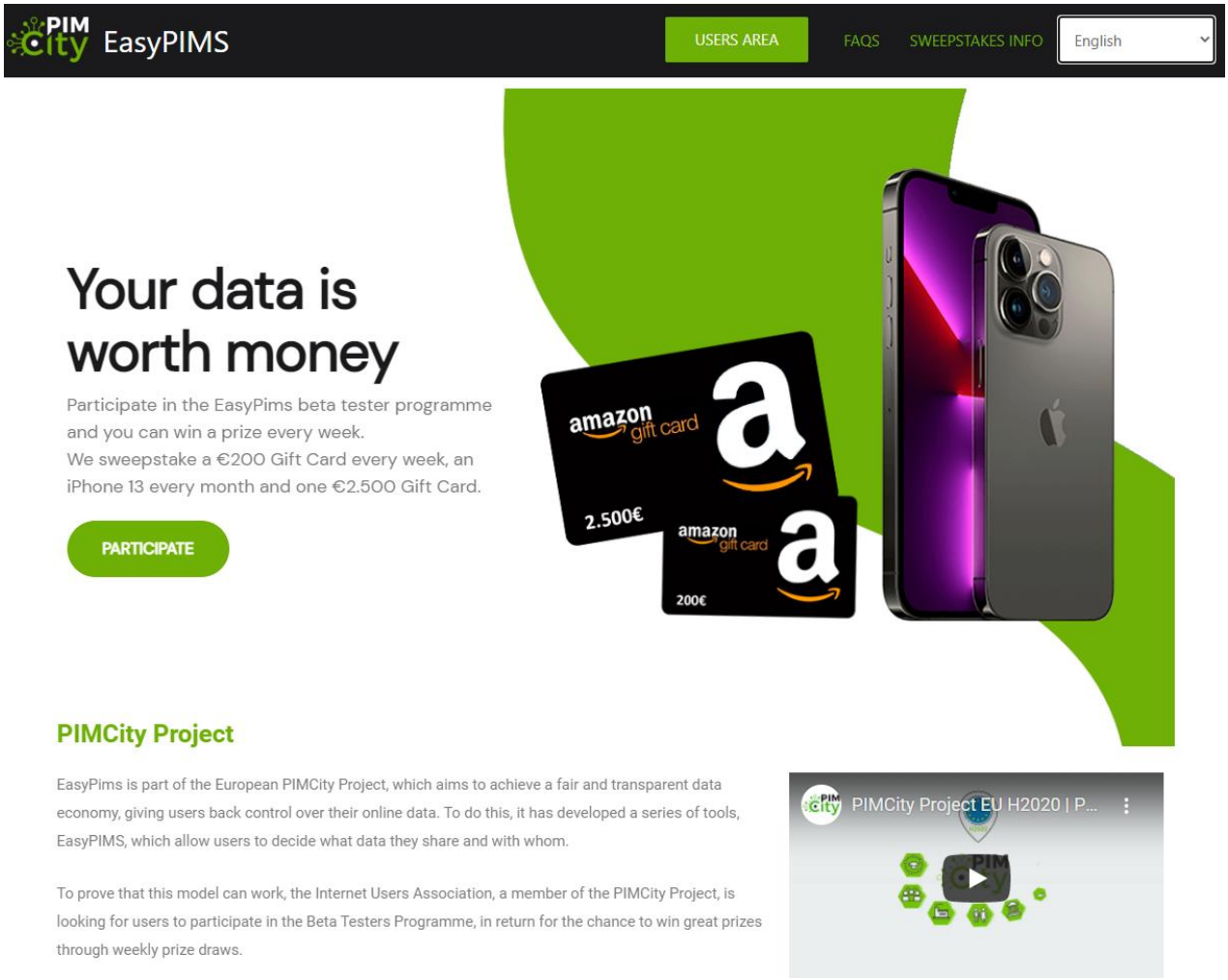


Figure 22 – EasyPIMS' informative website.

Interactions

This website contains static information that is only updated after a sweepstake has finished. Thus, it does not need interactions with any PDK component.

7.2.1 Registration and Login

A user can register to the PDA, accept the EasyPIMS privacy policy, add the main personal information and verify their email. The registration and login process follow all the current security standards to ensure privacy and avoid attacks.

Functionalities

A set of personal information fields is available for the user registration form and for the login form. These includes at least a name, a username and email.

Design

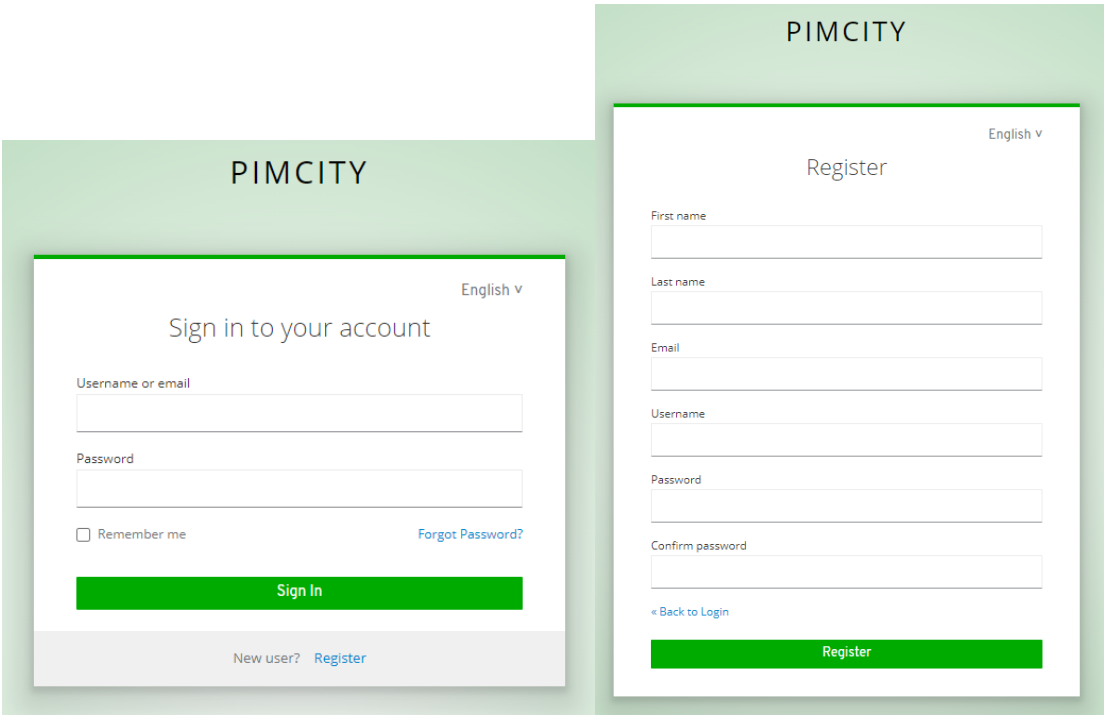


Figure 23 - PDA's Login and Registration screen mockups

Interactions

This component is the entry point of the user to the PDA. After a successful authentication the user is introduced to the main screen (see below). The login part is handled by the KeyCloak component of the Cloud Controller.

7.2.2 Main Screen

The “Home” or main screen is shown to the user after they log in to the platform. It contains general information about their usage of the platform and their position to win some of the prizes.

Functionalities

The main screen is designed to do not provide users information about their data, but about their status to obtain prizes from the platform. It contains four different subsections:

Pending Tasks: It provides information of the periodic tasks the user has to do in order to obtain points. Moreover, it provides a list of the points that were obtained in the past months.

Extra Points: It provides information about the actions that can be done to obtain points.

Total Points: The total number of points.

Next Sweepstakes: The calendar of sweepstakes with the information of the actions required to participate on them.

Design

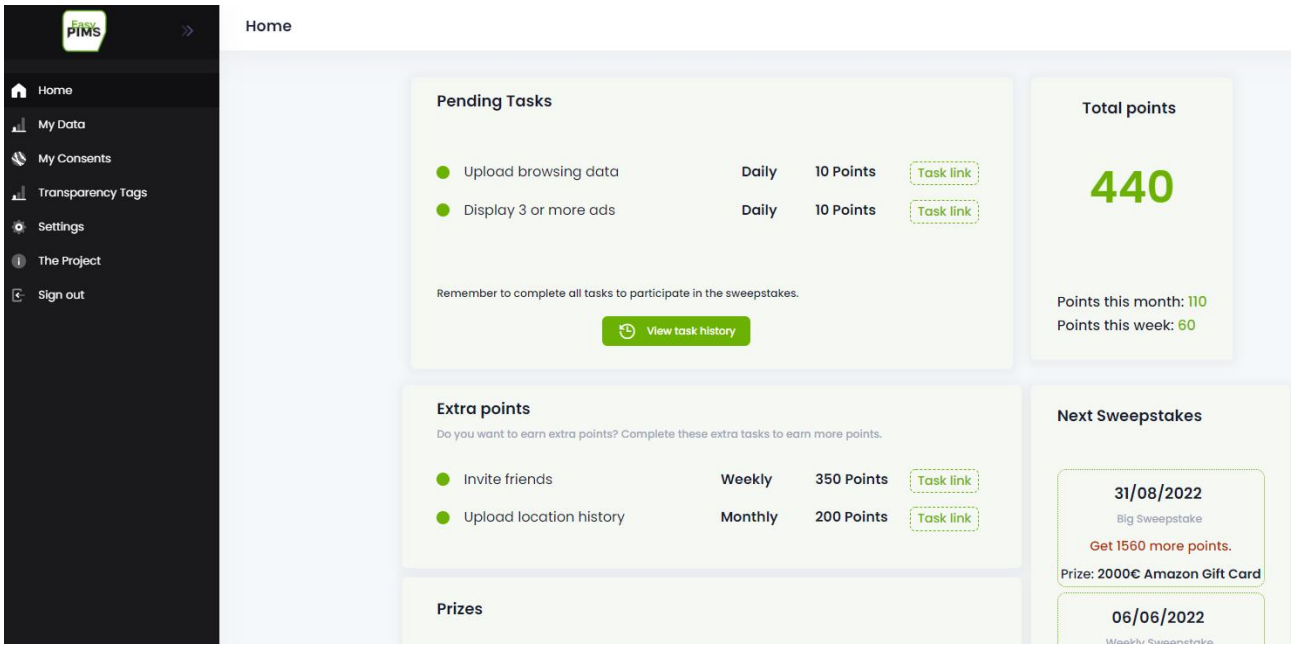


Figure 24 – Home screen of PDA

Interactions

The information of past actions completed and point earnt are collected from the trading engine. Other information about tasks and sweepstakes is collected from the task manager.

7.2.3 My Data

The My Data section allows the users to visualize and control their data.

Functionalities

This screen allows users to manage (insert, amend and delete) their personal data.

A first section is devoted to the direct input of personal data from the user. Moreover, the users have the possibility to observe their browsing history (that is uploaded using a Chrome extension that can be downloaded in the Chrome Store). And to upload their location history following a tutorial to obtain their data from Google.

Design

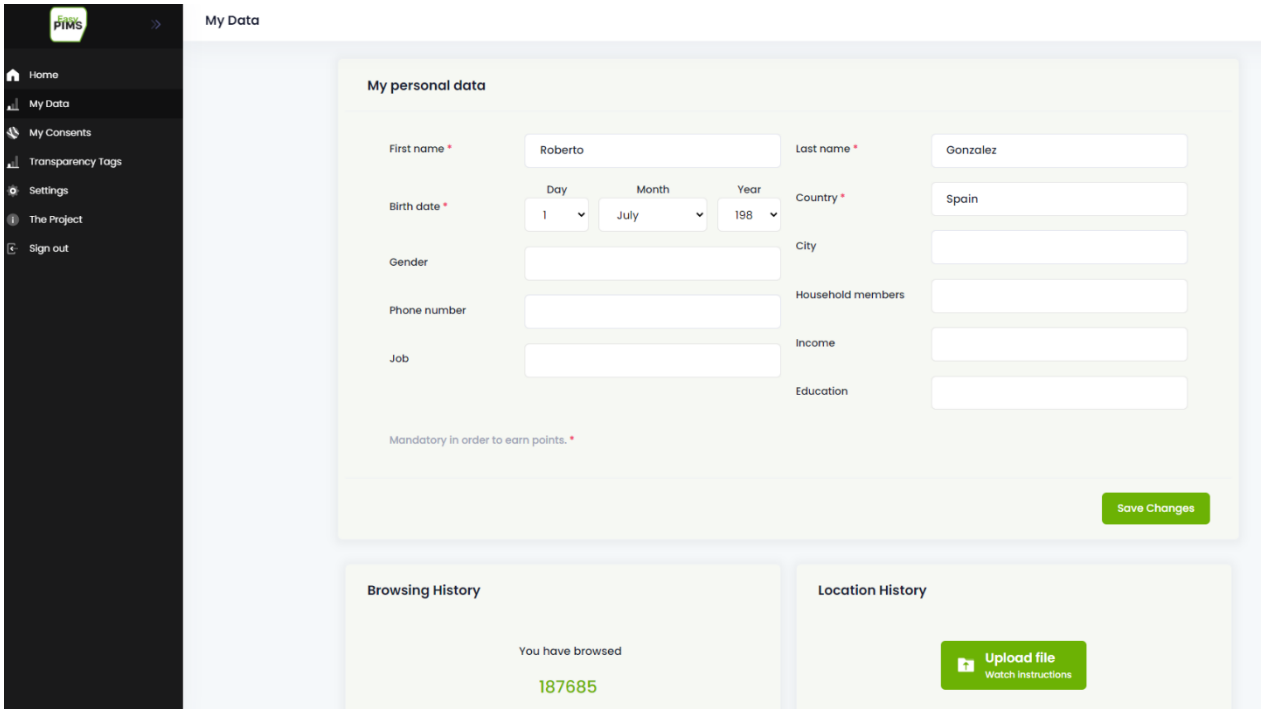


Figure 25 - My Data page in PDA

Interactions

This screen mainly interacts with the PDS. It stores all the data and shows it. However, it also interacts with the data portability and control module for the upload of the location history. Finally, the browsing history allows the user to observe the transparency tags for the websites they have visited.

7.2.4 My Consents

The “My Consents” section allows the user to configure their consents for data transfer to third parties.

Functionalities

The user can select the kind of data they want to share with third parties, if any, and the purposes for which that data can be used.

Moreover, a profile of the user will be automatically shown to the user if they are uploading their browsing history. This allows the user to filter the parts of their profile that can be used for both, data sharing and advertising.

Design

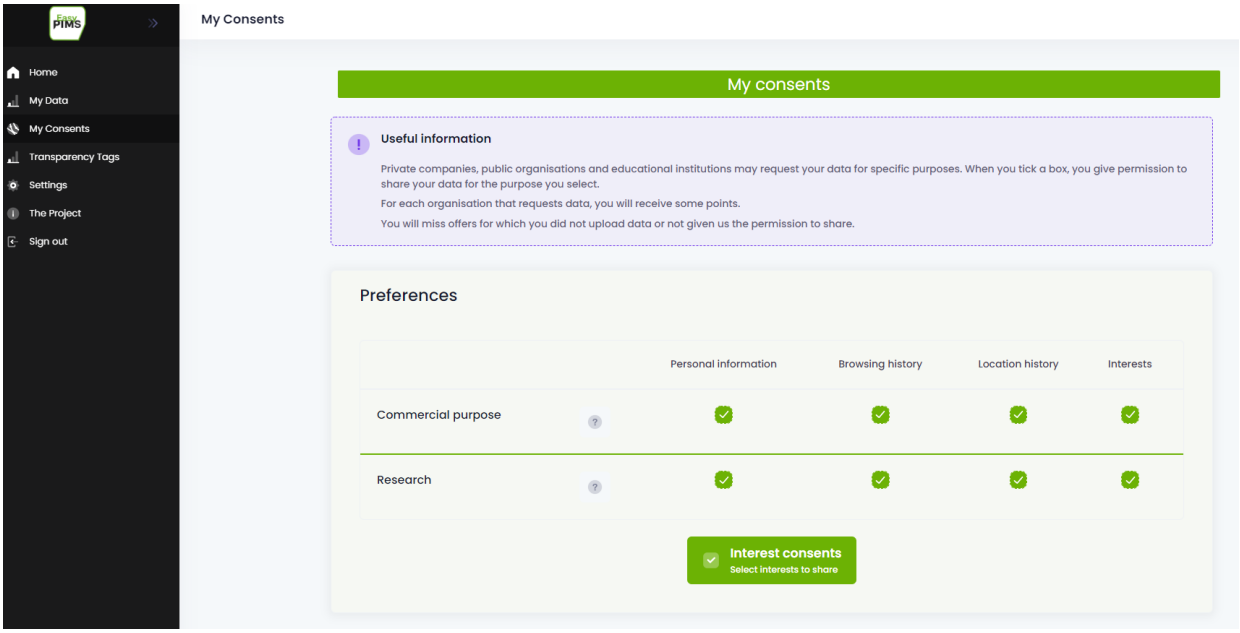


Figure 26 - My Consents page in PDA

Interactions

This section mainly interacts with the consent manager. Moreover, it obtains the user profile from the User Profiling System.

7.2.5 Settings

The final section of the PDA corresponds with the settings for the user account.

Functionalities

This section allows the user to download their data, change the password or delete the account. Moreover, it provides the information required to invite friends.

Design

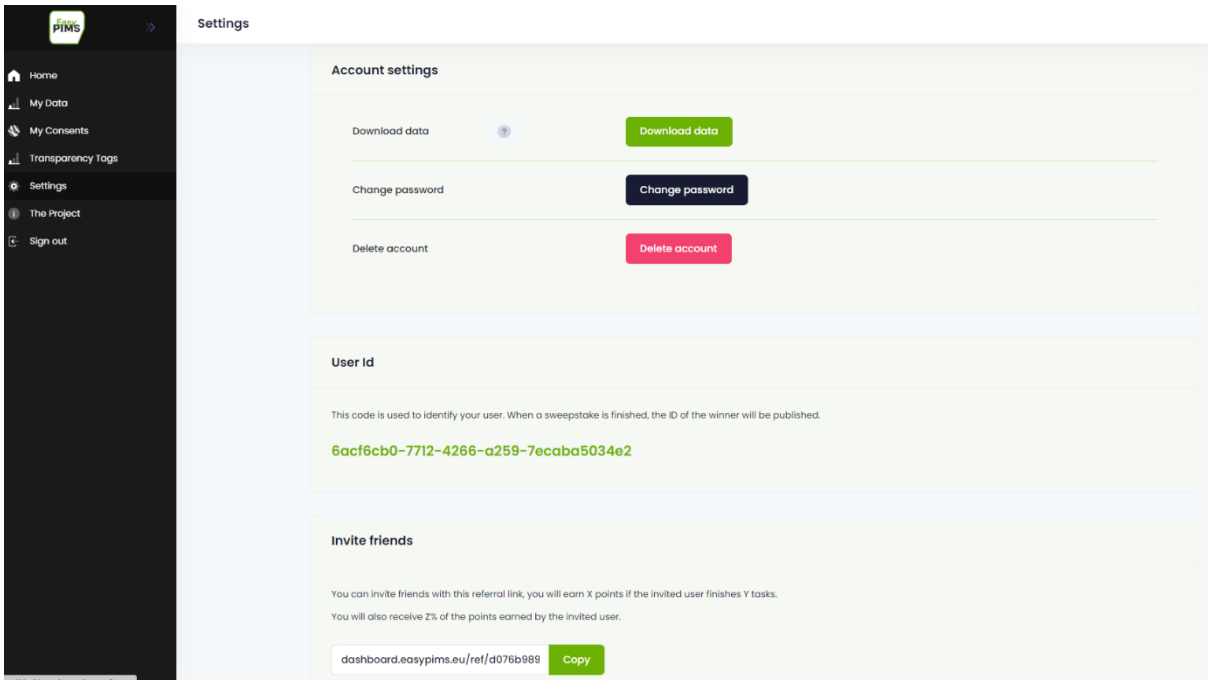


Figure 27 - Settings page in PDA

Interactions

This section interacts with the KeyCloak component of the Cloud controller, the Data Trading Engine and the PDS.

7.3 Deployed Implementations

The PDA is deployed in the development environment for testing purposes at: https://easypims.pimcity-h2020.eu/avatar. While the production version is deployed at: https://dashboard.easypims.eu/

8 Design of Transparency Tags (TT)

We envisioned the Transparency Tag as a tool similar to a Nutrition Label for food which provides the information about the ingredients, their provenience, intolerance risks, etc. of food.

The TT has been introduced as an easy-to-understand way to provide the user with information about the nature of the web services contacted. For each web service (e.g., a website, a mobile app, a data buyer) EasyPIMS exposes the information contained in the corresponding Privacy Metric, such as its owner, its purpose, the personal data it collects, etc. Apart from providing all the details, the TT also summarizes such information in scores - automatically computed by the analytics behind the Privacy Metrics - revealing the potential security and privacy risks associated to that service. TT also attempts to describe how transparent the service under exam is towards the user, i.e., whether crucial information such as data processing purposes, data controllers, contacts, etc. is provided publicly.

We remark that the Transparency Tags is meant to be the presentation module responsible for presenting the information contained in Privacy Metrics. As such, it is tightly coupled with the content of Privacy Metrics, and any modification to the information contained in the Privacy Metrics will inevitably impact the UI in the Transparency Tags. For this, the purpose of this design phase is to define the best UI tools to deliver the information in general and define overall design guidelines without focusing on single items specifically. Nevertheless, there exists items in TTs which will persist and for which we do not expect future modification (e.g., scores).

In this section, we first describe what is the state of the art about UIs for delivery of privacy-related information, then we describe the UI requirements that have emerged in research conducted so far and finally we explain how the design of TT has evolved since its first draft presented at the beginning of the project. In particular, we will describe which stakeholders have been contacted to collect feedback and which requirements have emerged from this phase. Then, we will present the mockups that have been circulated, the result of survey campaigns, and we conclude describing the final draft taking into account all the inputs received so far.

8.1 State of the art

Unfortunately, the amount of work available in the literature which specifically focus on solving the problem of delivering privacy-related information to users is little. In this area, the most prominent project to mention is Privacy Nutrition Labels[41] carried on since 2010 by the CUPS group (Cylab Usable Privacy and Security Laboratory) of Carnegie-Mellon University. In particular, researchers at CUPS focused on making the privacy policies publicly available by web services easy to understand and compare. CUPS researchers published a set of papers to extrapolate, condensate and present the information contained in privacy policies to make them easy to understand to non-expert users. They used a user-centered design process to identify the best privacy policy format. The key findings of their work are summarized in [8]. They conclude that standardized formats are significantly better that full-text and layered-text policies that are available on most of websites. In particular, textual format increases the amount of time and effort for the user to read and understand the privacy policy, even when this is minimized or simplified. Moreover, they observe that more complex questions about data practices require the reading multiple sections of policies. This task deeply increases the amount of time needed to read the policy. Interestingly, the paper explains that it is key to provide a standardized format to inform the user. This approach increases the possibility to compare different policies, and this is more important than how information is presented in practice (short text, scores, tables). Some users involved in the experiments suggested the use of charts and figures, which visually allow to check and compare information. However, synthetizing text content in visually presentable metrics is hardly achievable especially if using automatic processes. Finally, researchers observe that standardized text-based policies do not scale well as table-based policies.

[41] https://cups.cs.cmu.edu/privacyLabel/

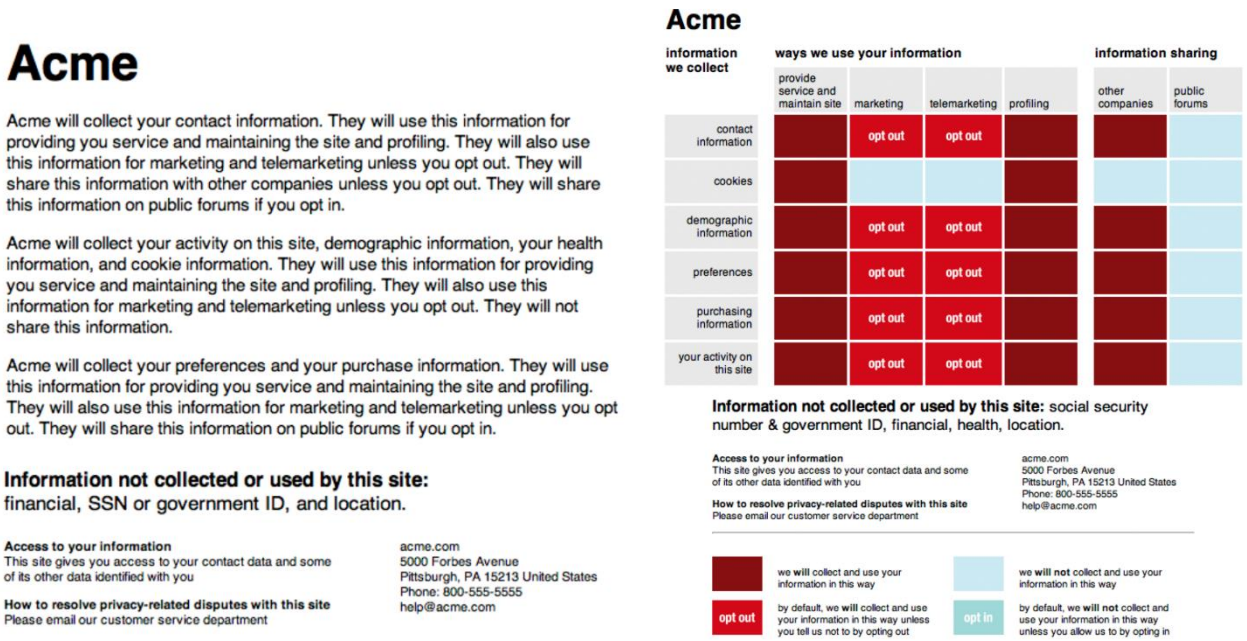


Figure 28 - Examples of formats for Privacy Nutrition Labels developed by CUPS researchers. Standardized short-text (left) and standardized table (right)

Another important, much more recent initiative with the objective of presenting privacy-based classification to the user are Apple's Privacy Labels. These have been introduced recently by Apple to classify apps downloadable from the App Store. Their objective is to give users more information about privacy by letting them know which personal data is collected and how by app providers. This initiative spurred a great noise in the media sphere, and most of users with tech background acknowledge Apple reached the goal for which labels where introduced[42]. However, some of them criticized Apple for not advertising them enough, for placing labels in areas too difficult to find (privacy labels are placed almost at the bottom of the app page), and labels are informative up to a point, as they do not provide any information about how personal data is being used for. From a design perspective they satisfy all basic UI requirements: they are indeed usable, easy to understand and visually enjoyable.

One of the most interesting features of Apple Privacy Labels is they specify whether collected data is connected to the identity of the user or not. In other words, whether data is anonymized or not. However, we have to highlight that information provided in Apple's Privacy Labels are provided by app providers, and there is no real check that what they claim corresponds to the actual data collection practice. In this sense, Apple claims it will penalize or ban developers in case inconsistencies will emerge.

[42] https://www.vox.com/recode/22285020/apple-privacy-nutrition-labels-ios-14

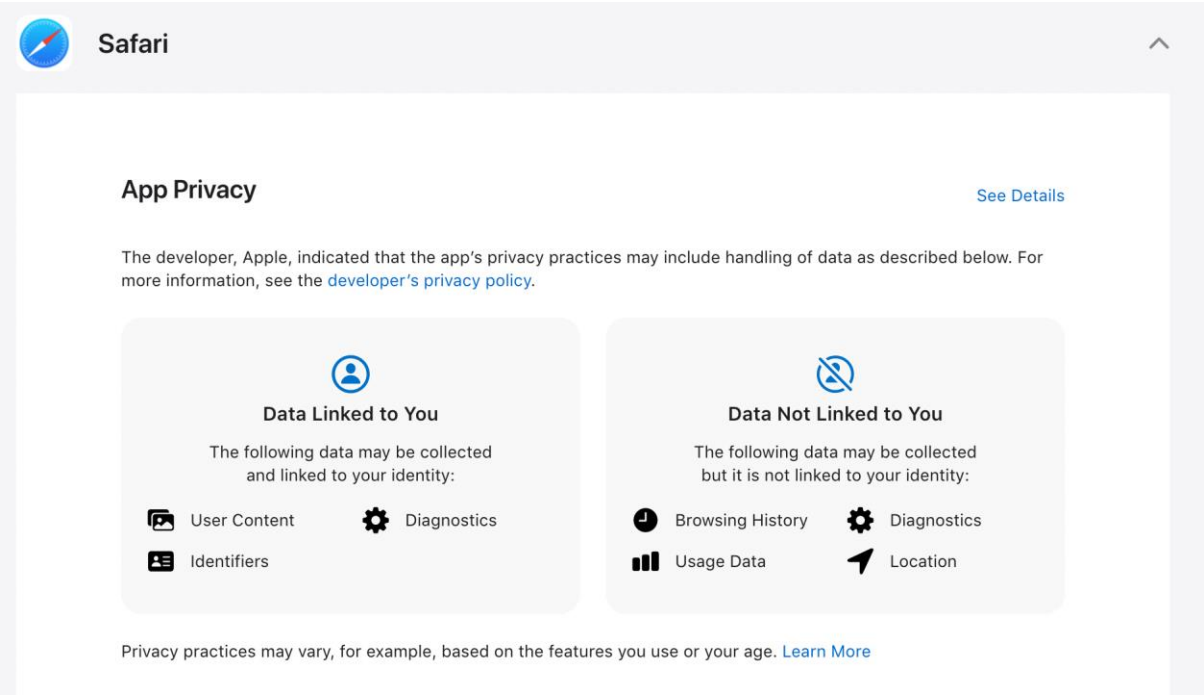


Figure 29 - Example of Apple's Privacy Label for Safari browser.

It is important to highlight that the advertisers' industry itself has introduced some novel tools to increase transparency in the data market. This is the case of the Transparency and Consent Framework[43] (TFC) which has been promoted by IAB Europe. The TCF's objective is to assist all parties in the digital advertising industry in the compliance process for EU's GDPR and ePrivacy Directive. TFC basically designs guidelines to process personal data or accessing and/or storing information on a user's device, such as cookies, advertising identifiers, device identifiers and other tracking technologies. The TCF creates an environment where website publishers can tell visitors what data is being collected and how, and which other parties they partner with intend to use such data. The TCF gives the publishing and advertising industries a common language with which to communicate consumer consent for the delivery of relevant online advertising and content. TFC initiative has distributed its second version (TCF v2.0) in August 2020.
Finally, even if not directly related, we must mention the General Data Protection Regulation[44] (GDPR). In fact, the GDPR does not explicitly provides rules or guidelines to design UIs, but it does rule which information about personal data processing services must provide to users. Apart from greatly increasing transparency, the GDPR has implicitly compelled lawyers who write privacy policies to somehow schematize their content. Unfortunately, there exist no standard format yet, but surely, GDPR has introduced a list of points "to touch". We considered also this aspect in the development of TT prototypes.

8.2 Requirements

As most of UI designs, also the UI at the core of TTs bring some challenges which we must face. In particular, we have to identify the best solutions to deliver, important and complex information such as those required for GDPR compliance in a quick and intuitive manner. Unfortunately, though, most of information contained in Privacy Metrics are textual and descriptive. This complicates the possibility of using graphical elements which are extremely useful to deliver messages quickly and intuitively.

Initiatives from CUPS researchers and Apple greatly guided us in the definition of Privacy Metrics and Transparency Tags, but given the spirit of the project, we also aim at going a little beyond state-of-the-art solutions. In fact, we believe both CUPS' and Apple's labels are flat and do not respond all

[43] https://iabeurope.eu/tcf-2-0/

[44] https://gdpr-info.eu/

the questions users may have. Moreover, we must find a compact solution to pack all information required by GDPR and make them accessible with few user interactions.

For all reasons above, TT's UI is quite specific from the perspective of design requirements, and since user interaction might be needed, we must address requirements typical of Human to Computer Interaction, which stands at the base of software interface and web development:

- **Usability:** this represents the main requirements in all UI and software in general. It defines the effectiveness with which users can achieve their objective and has been widely explored in Human-Computer Interaction sciences.
- **Visibility:** This is the ability for the user to easily find controls and tools that are meant to be interacted with. For instance, we must pay attention to element disposition and make clear what is the state associated to each element.
- **Feedback:** The user has to clearly understand what the response from the control/tool is (e.g., buttons) in any phase of the interaction (before, during and after)
- **Affordance:** This is a physical property of an UI element that indicates how it is to be used. For instance, 3D corner bars in panels allow the user to understand these can be resized.

8.3 Preliminary mockups

Based on requirements defined above, we built two different mockups for Transparency Tags. The first version consists of a single page where the three main data blocks of Privacy Metrics are presented in tabular manner, in three separate tables. We report this first mockup in the following[45].

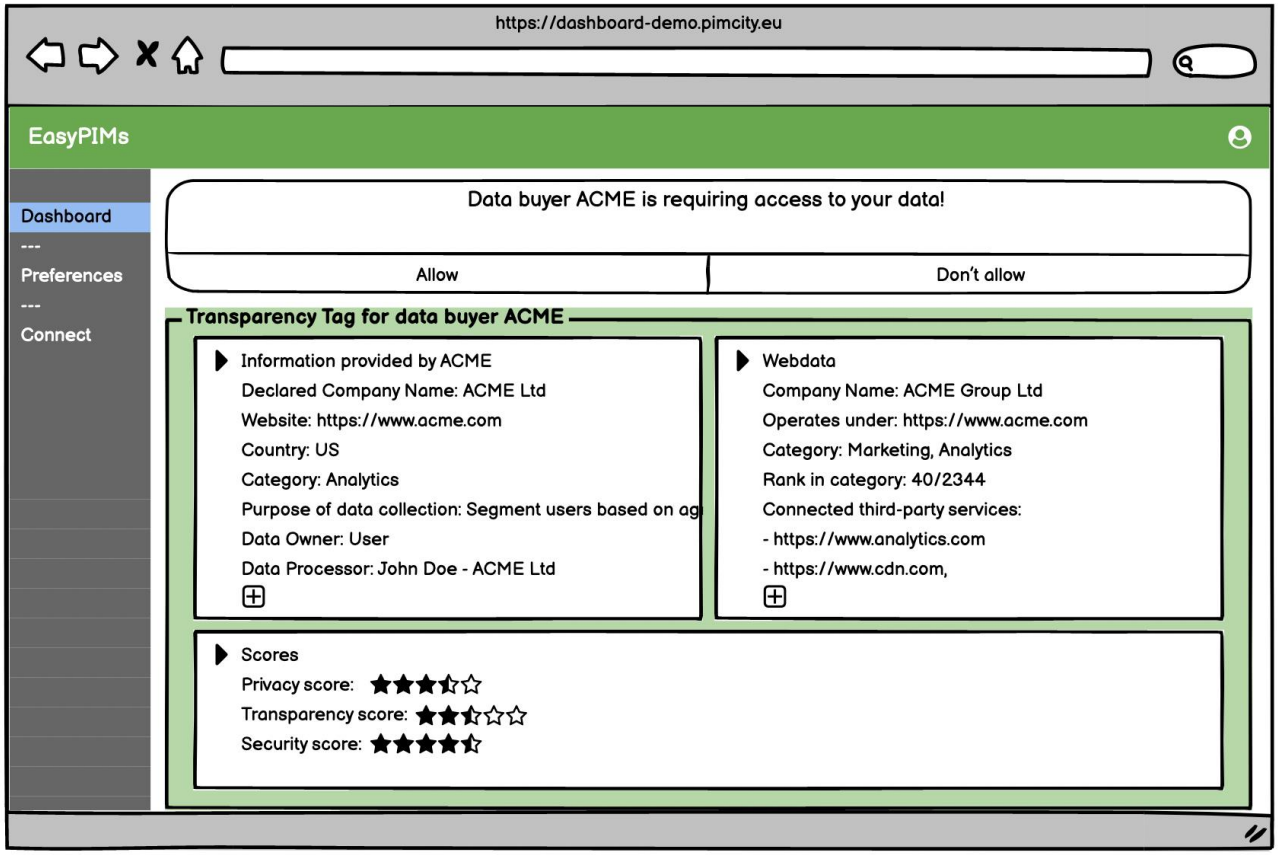
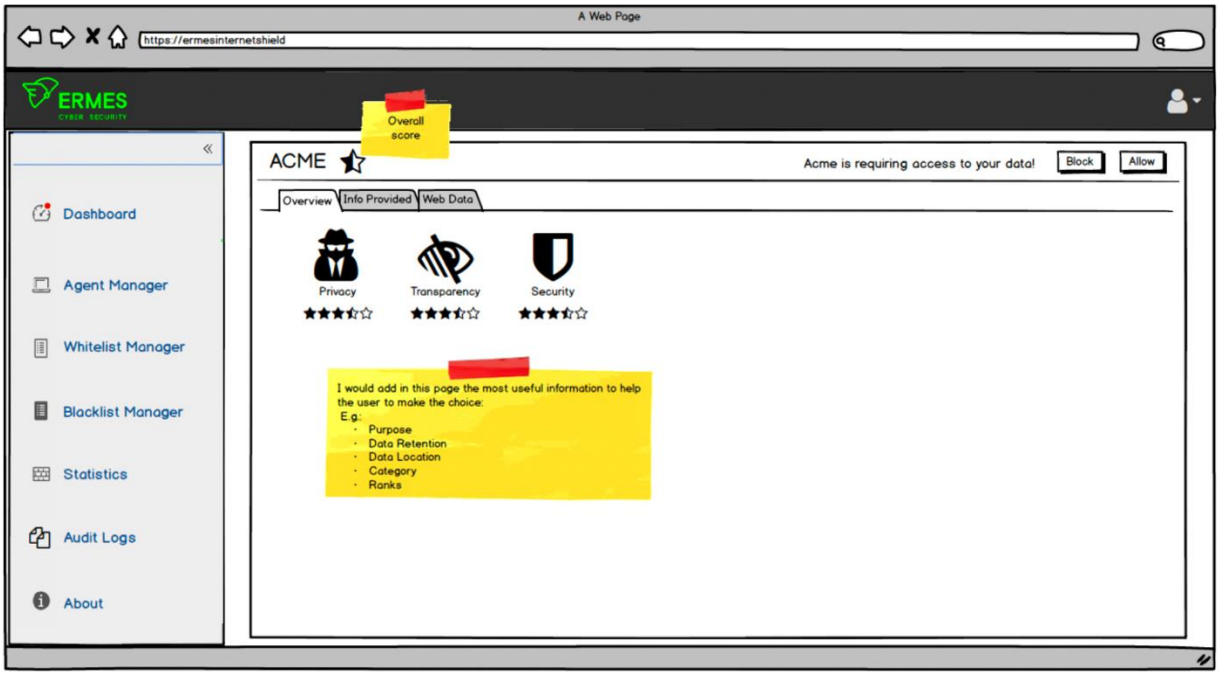
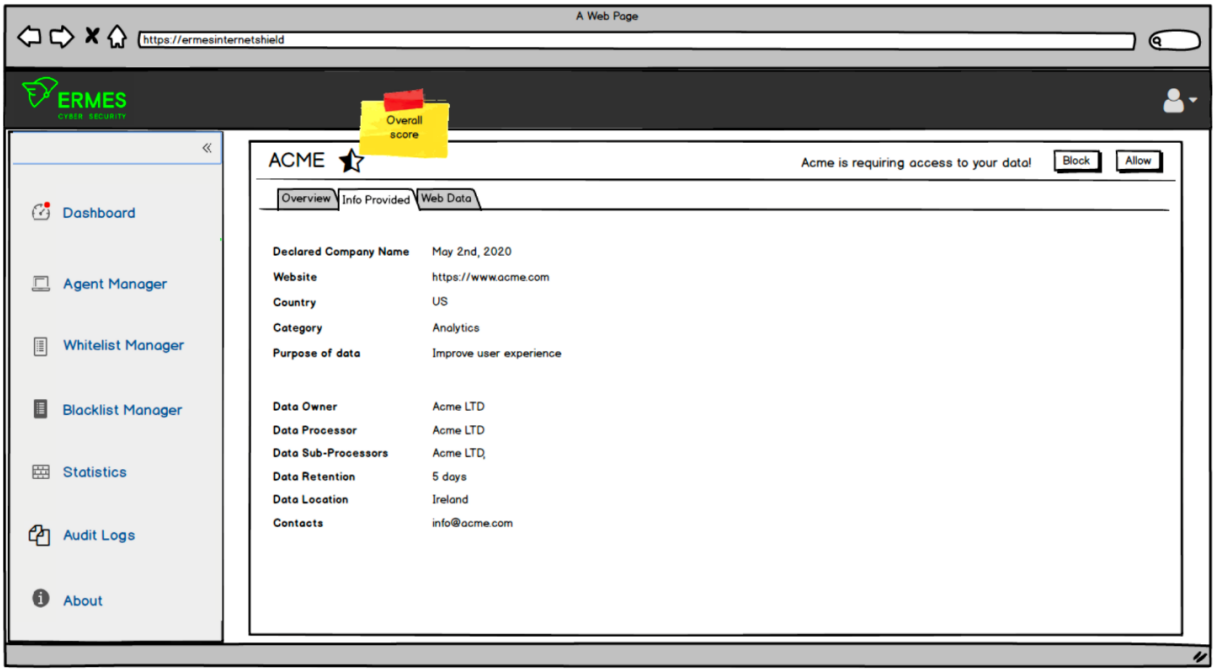


Figure 30 - Single-page TT mockup for desktop

We also developed another proposal in which the same information is distributed across multiple tabs. In this case the overall presentation is less dense, but users are required to perform more interactions (clicks) to browse the information. We report this second version in the following.

[45] All mockups include a UI interaction in which the user is required to authorize the service to access her personal data.


A Web Page
https://ermesinternetshield
ERMES
CYBER SECURITY
Overall score
ACME
Acme is requiring access to your data!
Block
Allow
Overview
Info Provided
Web Data
Privacy
Transparency
Security
I would add in this page the most useful information to help the user to make the choice:
E.g.:
Purpose
Data Retention
Data Location
Category
Ranks
Dashboard
Agent Manager
Whitelist Manager
Blacklist Manager
Statistics
Audit Logs
About


A Web Page
https://ermesinternetshield
ERMES
CYBER SECURITY
Overall score
ACME
Acme is requiring access to your data!
Block
Allow
Overview
Info Provided
Web Data
Declared Company Name
May 2nd, 2020
Website
https://www.acme.com
Country
US
Category
Analytics
Purpose of data
Improve user experience
Data Owner
Acme LTD
Data Processor
Acme LTD
Data Sub-Processors
Acme LTD,
Data Retention
5 days
Data Location
Ireland
Contacts
info@acme.com
Dashboard
Agent Manager
Whitelist Manager
Blacklist Manager
Statistics
Audit Logs
About

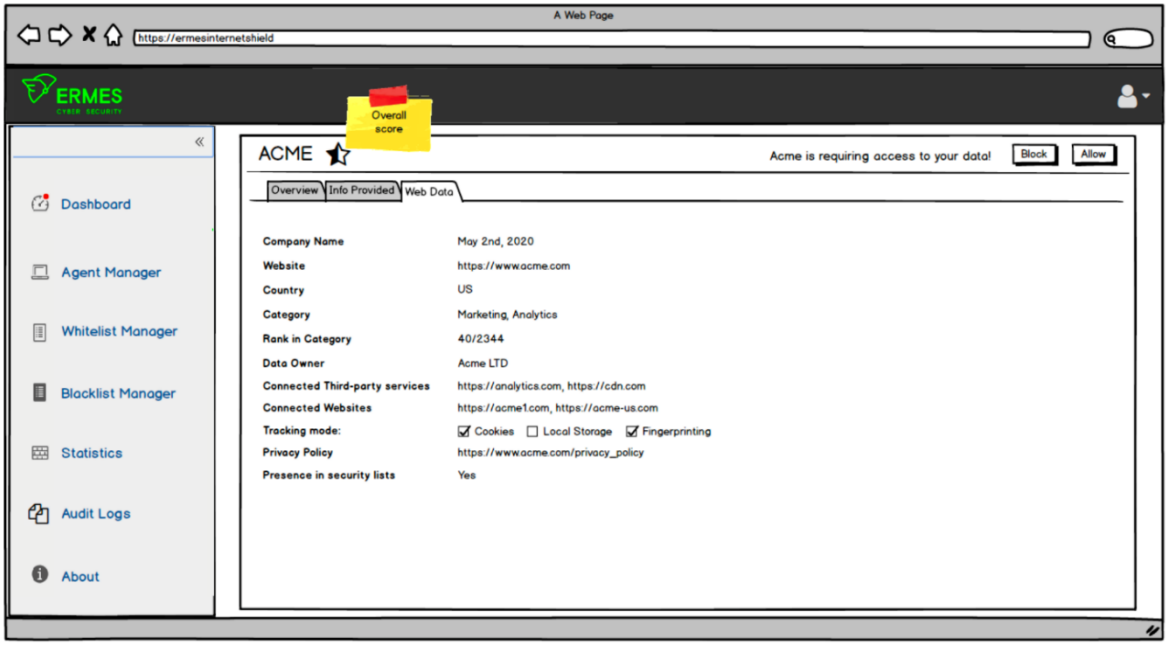


Figure 31 - Multi-tab TT mockup for desktop

For the sake of completeness, we also developed a second multi-tab mockup for the mobile case.

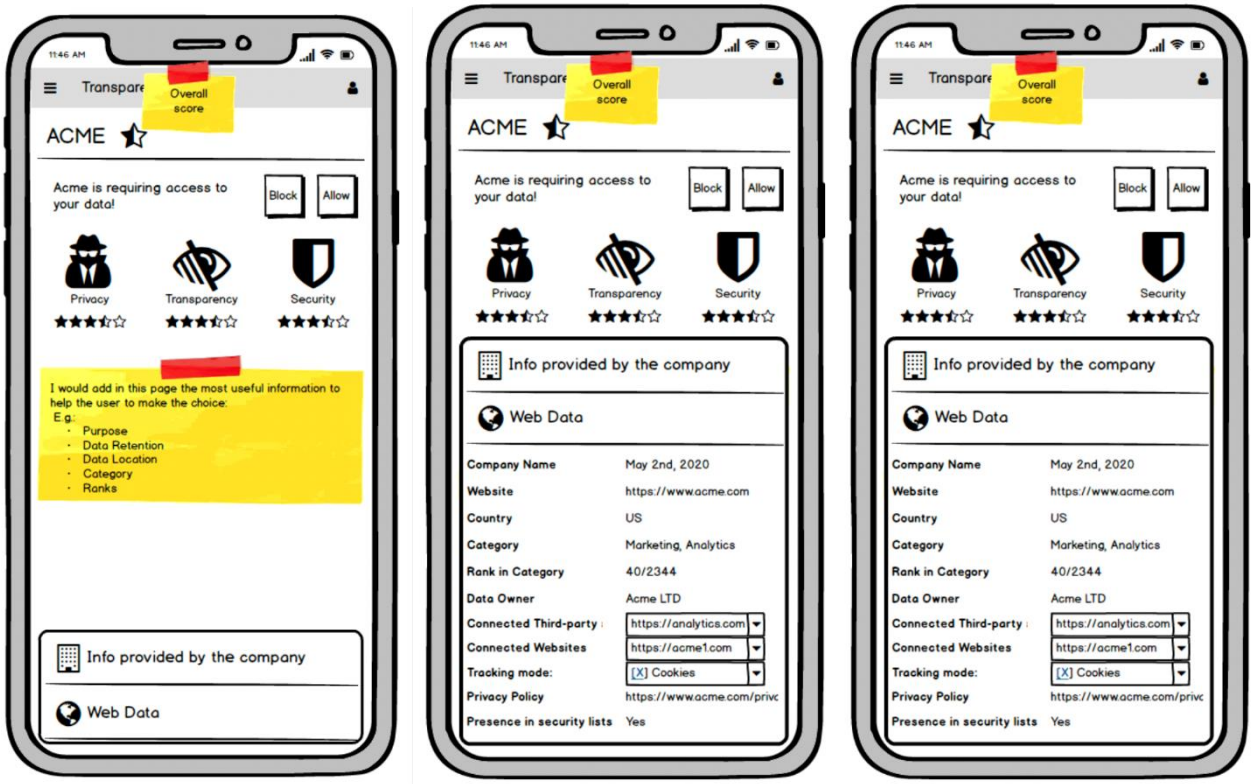


Figure 32 - Multi-tab TT mockup for mobile

8.4 Feedback campaigns

Mockups presented in the section above have been used to run surveys and collect feedback from different communities and stakeholders. For each campaign we describe how it has been conducted and what feedback we received.

8.4.1 Feedback from focus groups

First, we organized a series of focus groups to get detailed feedback from several user communities. For this, ERMES has been assisted by Wibson and AUI. Wibson organized a focus group involving users from their co-working building. In total they involved 11 people. AUI has organized multiple focus groups, and results will be considered and presented by the end of the project.

All people in Wibson's group have been shown the mockups. 8 of them preferred the single-page and 3 liked the multi-page mockup the most. Of the 8 users who preferred the single page, most of them said that they would like to see it in a simple way. In particular, they think the categories should be presented first, and the details could be summarized more. They also believe that details should be made available on user's demand, with some interaction (e.g., a click).

Users also liked the scores for privacy, security and transparency. They believe that scores simplify the decision making. Users also liked the idea of having a single general score that summarizes all the others (the large star in the mobile mockup).

Users also suggested to introduce a fourth score describing company's reputation in the PIMS system. This score would build on its purchase history: If a company bought data and never had problems with payments, data breaches, etc., the score is higher. Similar score is provided by Uber or Ebay to give users an idea of drivers' and sellers' reliability.

8.4.2 Feedback from Wibson users

In March 2021 Wibson has circulated an online questionnaire to its community of users to get feedback about current Transparency Tags mockups. In total, 52 people answered the questionnaire, with these being interested in data monetization matter. This number is by no mean statistically representative, but the results we obtain are however meaningful for a qualitative perspective as respondents belong to a community deeply interested in solutions that could help them monetizing their data.

In the following we report two plots describing the demographics of the sample that has been considered. As shown, the age distribution is slightly polarized towards young users. Differently, we observe a stronger polarization for what concerns to the gender of respondents, with about 80% of users being male.

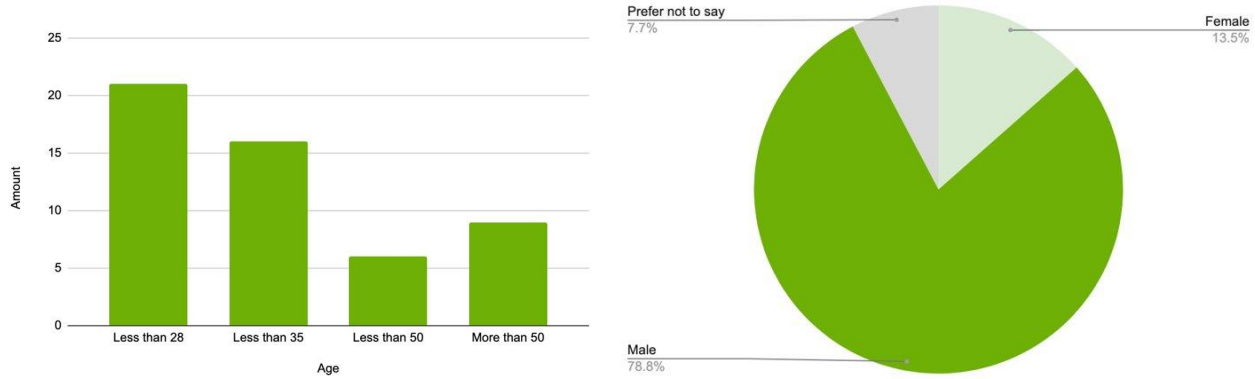
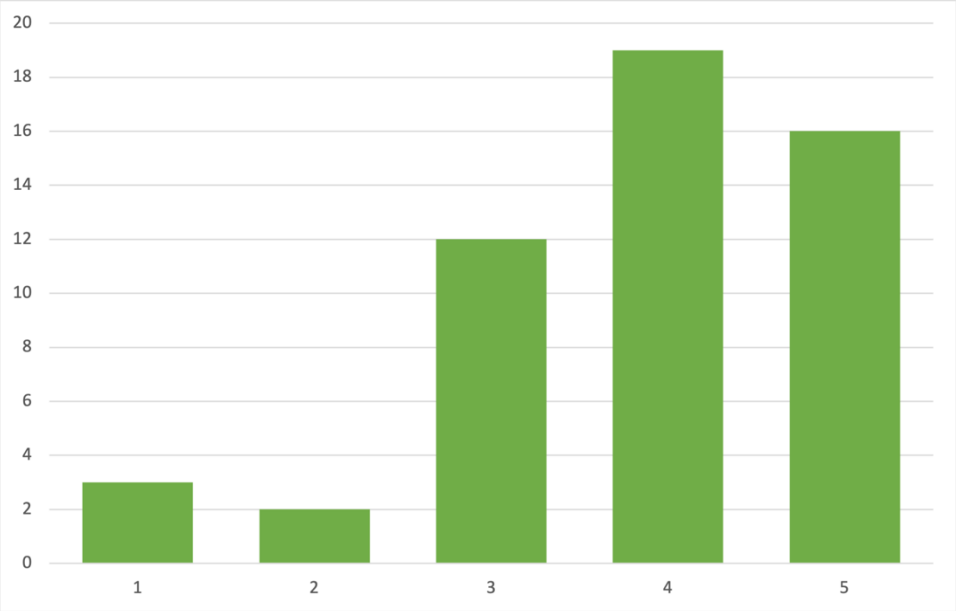


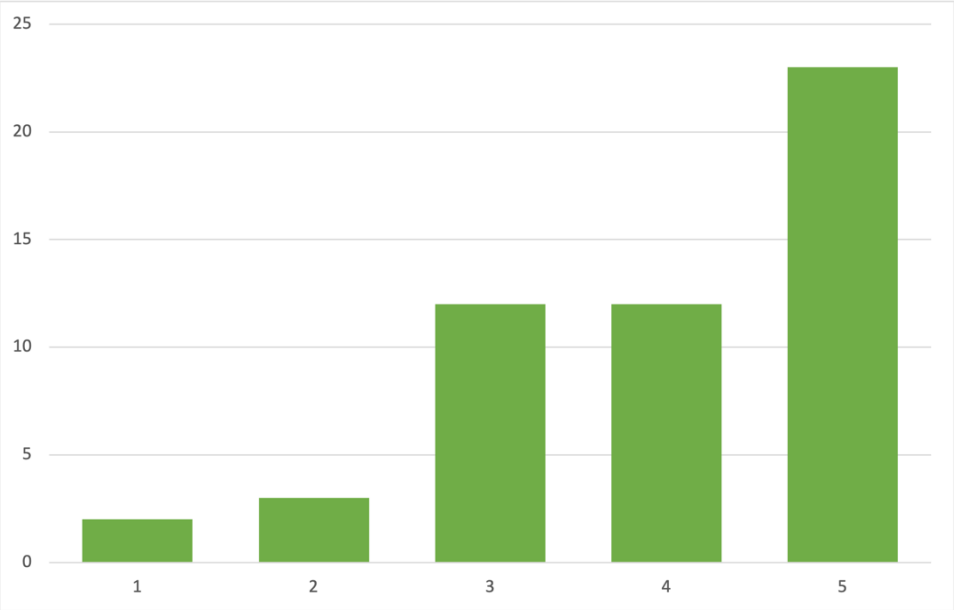
Figure 33 - Demographics of end users who participated the online survey

The questionnaire has been organized in closed-answer questions. In particular, for each information present in the Transparency Tag mockup, users have been asked to rate the importance based on their personal opinion.

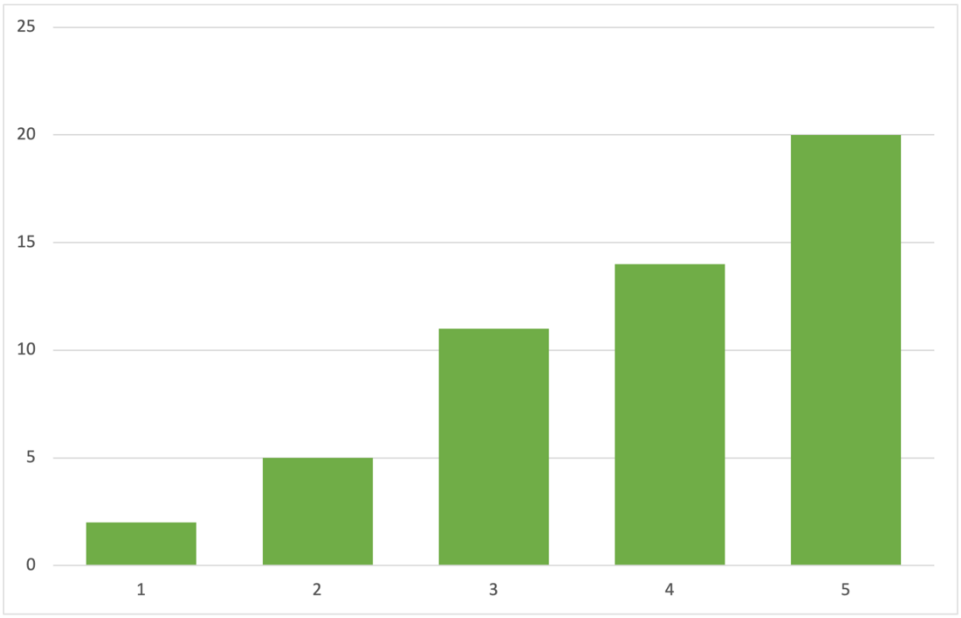
How important is to know the category of the company buying your data?
As shown below, users greatly appreciate to know the category of the company asking to collect and process their personal data. 47 (90%) users rate this information greater or equal than 3.



How important is to know the purpose of the company buying your data?
Also in this case users want to know for what purpose the company is asking to collect and process their personal data. 47 (90%) users rate this information greater or equal than 3.

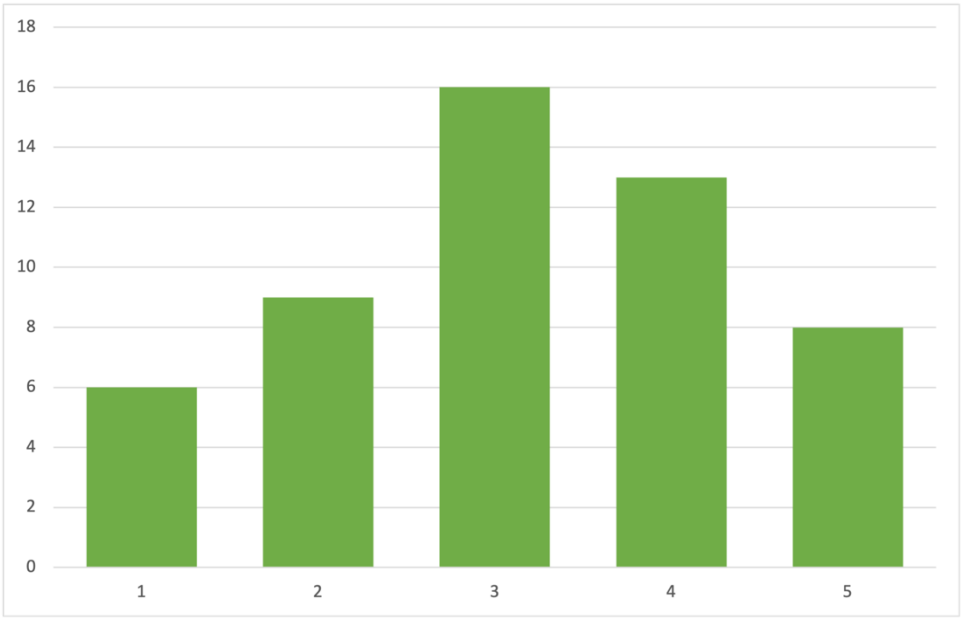


How important is to know the security and privacy rating of the company buying your data?
Yet users want to get information about the reliability of company asking to collect and process their personal data. Also in this case 47 (90%) users rate this information greater or equal than 3.



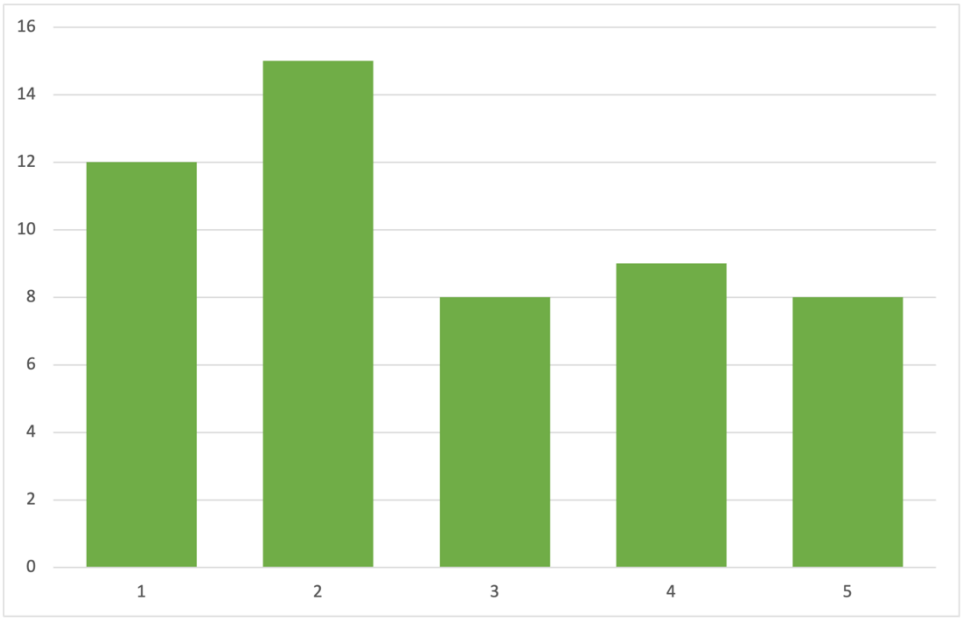
How important is to have a description of the company asking to buy your data?

In this case users welcome the idea of having a short text describing the company business, but this appears to be less relevant. In fact, 36 users rate this information greater or equal than 3, 16 think this is poorly or not relevant.



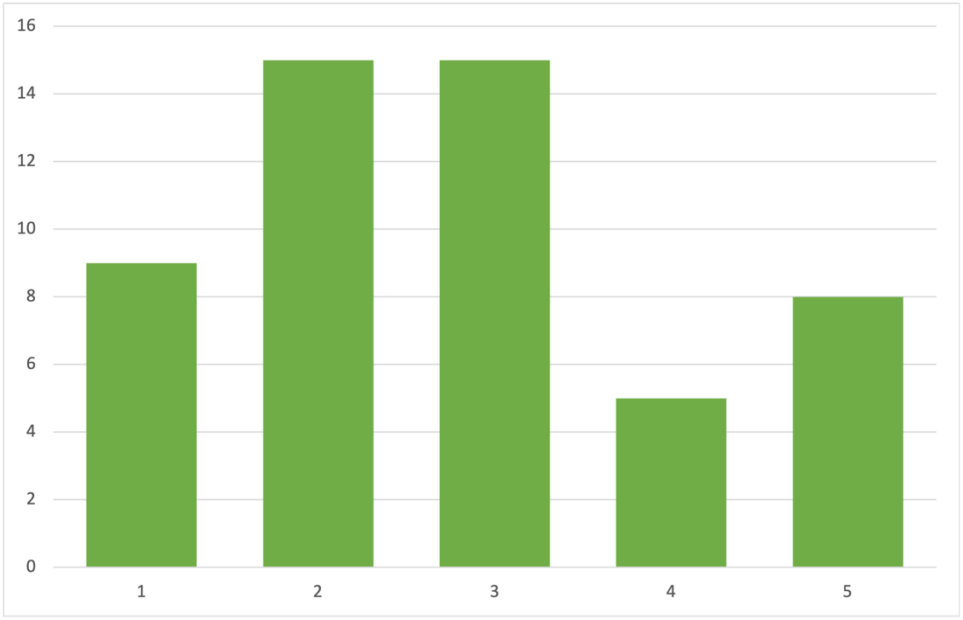
How important is to get the link to the company's privacy policy?

In this case it seems that most of users do not consider the company privacy policy a critical piece of information. We speculate that users tend not to read long texts as those contained in privacy policies.



How important is to know the company's country?

Also in this case users tend to disagree on the need of knowing the country from which the company is from. The rationale behind the question was to probe users' need to know whether the company (or the service it delivers) is based in a country with well-defined regulatory frameworks for online privacy and data protection. From this result we can speculate that users are not aware that some regulatory frameworks are stricter than others (e.g., California Privacy Act, EU's GDPR), and the country where the company runs its business plays an important role.



Free comments

We also let users provide free comments and feedback. Even in this case some interesting points have emerged. We resume them in the following:

- Users would like to know how many people are selling their data to the company. They think this is important to evaluate the reputation of the company
- Some users expressed the desire to know how long the company will use their data, I.e., the so-called data retention period.
- Users would like to know the monetary gain the company will obtain by selling their data.
- Users are interested in knowing whether data will be re-sold or shared and to whom.
- Users are particularly worried to know whether they will be able to delete their data replica for that specific company.

Finally, it emerged from the survey that people want to have a metric that summarizes all information presented in the Transparency Tags to take decision on whether is a good idea to share their data or not. Indeed, they welcome the privacy, security and transparency scores contained in TTs.

8.4.3 Feedback from data buyers

To validate the mockup proposals provided by ERMES, it is important to collect feedback from advertisers and data buyers too. Indeed, we aim to develop a new tool capable of helping users building awareness about personal data, but at the same time, useful for advertising industry to increase users' will to share their data. For this, data buyers' feedback is key to develop a model of TTs which will be welcome by the industry as a novel tool to increase transparency and develop brand reputation, value and reliability.

For this purpose, IAB Spain is in the process of interviewing a number of advertisers and data buyers to show and explain the content of Privacy Metrics, and present TT mockups. So far, we have not received a number of comments sufficient to drive any conclusion yet. Nevertheless, preliminary comments from both IAB and contacted advertisers suggest considering the Transparency and Consent Framework (TFC) described above. Even if slightly different, Transparency Tags share many common points with the Transparency and Consent Framework. We will carefully address this framework in the development of the task and find the best synthesis to present in Transparency Tags.

Further preliminary feedback received from data buyers was it would be useful for the user to know if they could import and display to the user the different certificates concerning, e.g., respectful advertising or security certificates (ISO 27000, ISO 27001 and more).

8.4.4 Feedback from legal partners

ERMES collected some technical feedback from KUL. In this case, we focused more on the legal aspects connected to GDPR compliance. ERMES and KUL agreed that, given the very privacy-focused spirit of the project it would be better to ask data buyers to provide more detailed information about the processing they will perform using users' personal data. In particular, some new fields have been added to the Privacy Metric, and consequently also in the Transparency Tags. This new GDPR-oriented fields are described in Deliverable D2.2 [6].

8.4.5 Feedback from technical partners

Finally, we also collected free feedback from users in the consortium. We resume these in the following:

- Some users would like to understand the motivations that lead a data buyer to get a low score.
- Some users suggest substituting scores with environmental-like scores, with low scores in red and higher scores in green. This looks more intuitive for them.
- Some users would like to know the ranking of the data buyer based on its scores. Some others see this as somewhat useful, but they notice it would be difficult to implement and explain.
- Other users would like to provide reviews and comments as it happens in online stores.

- Some users suggest renaming some fields: for instance, they suggest replacing “Web Data” with “Publicly available information”.
- Finally, some users would simplify the content to make it easier to understand for the non-tech-savvy users.

8.4.6 Feedback from B2B market

The Transparency Tags design proposals presented in this section hold for both B2C and B2B scenarios, but especially for the second one changes may occur depending on implementation and product integration requirements. Indeed, content proposed by ERMES for this deliverable has been validated by few of its clients via direct chats or aside of meetings. For privacy reasons, ERMES is not authorized to share the clients’ name. In general, clients nicely welcomed the initiative, with some of them asking to have the feature implemented as soon as possible. However, ERMES registered no clear preference for a TT design proposal.

8.5 Conclusions on design

All in all, the feedback we received about the design of Transparency Tags is quite positive. Users we contacted welcome the information and the presentation contained in Transparency Tags, but they also suggested modifications and improvements that we are evaluating. In general, we registered a slightly wider preference for the single-page TT design, and some of the feedback we received have been approved for integration.

Considering the other communities which have been contacted, i.e., advertisers, legal and technical partners, also for these cases, the feedback we received validated the design choices taken so far. We keep iterating over such choices as new feedback arrives.

Finally, for what concerns TT proposals in B2B solutions, ERMES has kept in consideration feedback collected from user communities and find the best solutions to integrate TTs in its products. ERMES has identified the best implementation trade-offs to deliver TT functionalities while minimizing the development efforts.

8.6 Implementation in EasyPIMS

Based on the design proposed in Section 8.3 and feedback resumed in Section 8.4, we implemented TTs as an Angular-based webpage included in the PDA. The implementation considers all the feedback collected so far, but it also introduces new features that addresses issues emerged during the building and deploying of the code.

First, we introduced a dedicated section in the PDA that presents the list of service domains for which we have generated a Transparency Tag. Such list is obtained based on the services which have been extracted from the browsing history of the user. In short, for each website or service that has been visited by the user while browsing the web, we extract those whose domain is contained in the database of Transparency Tags. This way, the user is provided with TTs about domains with which she has contacted. Moreover, we also present TTs obtained from data buyers participating EasyPIMS who showed some interest in user’s data or bought it.

These choices allow to provide a more personalized experience to the user and present Transparency Tags of services with which there had been some interaction, thus filtering off services uninteresting to the user.

The figure below depicts the Transparency Tags section which has been included in the PDA. As shown, TTs are presented in a tabular manner, with each row reporting a quick summary obtained from information contained in Privacy Metrics. We also add a tag describing the kind of interaction the user has established with the service: visited website (“Browsing” tag) or data buyer (“Data buyer” tag).

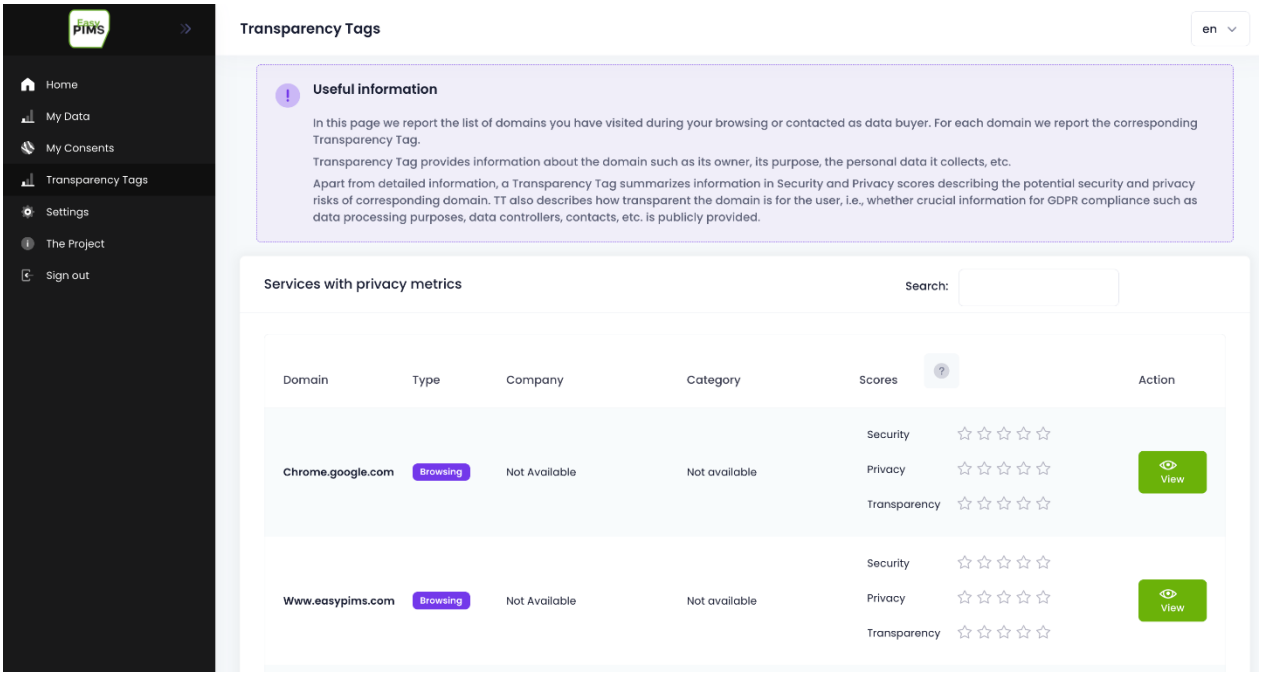


Figure 34 - Transparency Tags page in PDA

To allow the dashboard to retrieve the information to present in the TT table, the APIs of Privacy Metrics component have been modified to return information “Company”, “Category” and “Scores”. This allowed to simplify the design of the PDA and avoid to fire too many requests to obtain such pieces of information.

Next, we present the content of the actual TT page. By clicking on button “View”, associated to each service in the table, the user lands on the page reporting all the data available for the corresponding Transparency Tag. We report an example of TT page in the figure below.

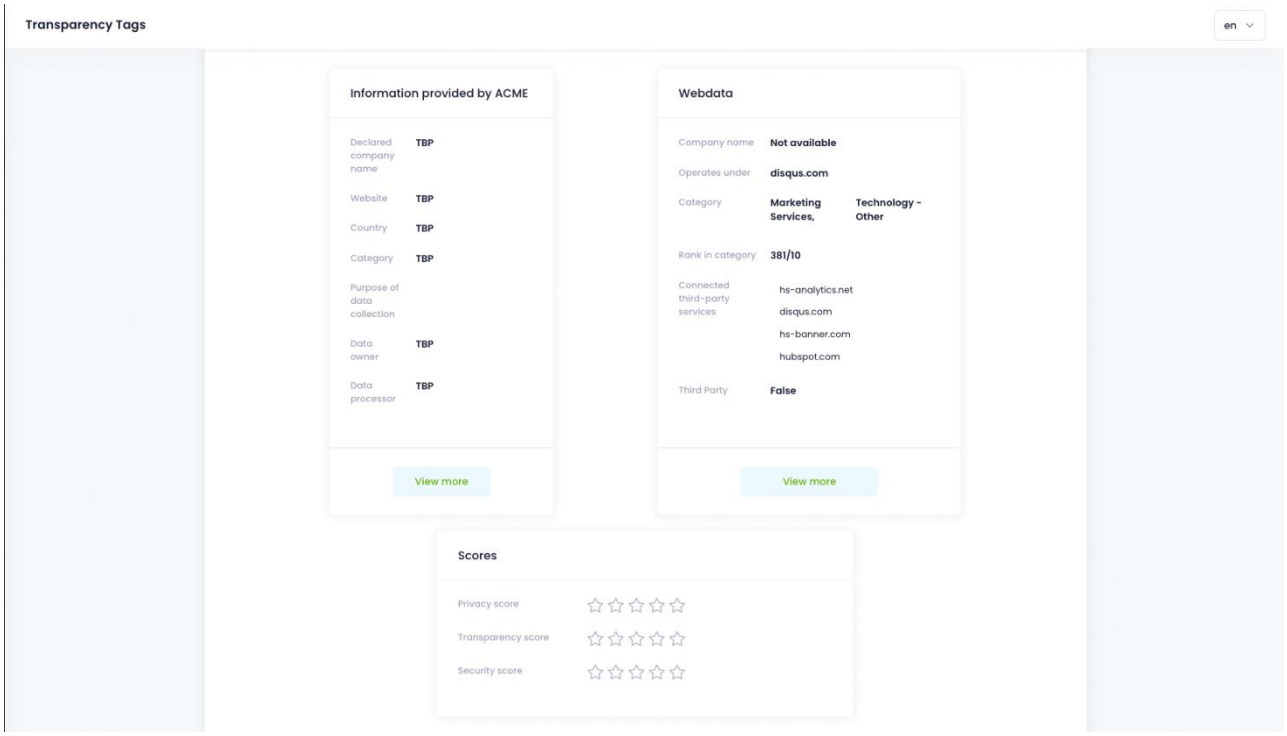


Figure 35 - Transparency Tags page example

As shown, the webpage has been developed based on the single-page design proposed in Figure 30. The pieces of information reported in the three sections composing the TT are those constituting the Privacy Metrics and described in detail in Deliverable D2.2 [7].

8.7 Implementation in Ermes For Enterprise

The design presented in Section 8.3 has been inspirational for the Transparency Tags which have been implemented by ERMES and integrated in the administration dashboard of its web protection product, Ermes for Enterprise. Also in this case, the implementation has been developed using Angular, but using design templates which ERMES has customized for the development of its products.
For the case of Ermes for Enterprise, Transparency Tags have the objective of providing users with a security-oriented piece of information to use to build security and privacy awareness and perform intelligence tasks. For instance, security administrators (these are the actual users of Ermes for Enterprise dashboard) can use Transparency Tags to understand why a connection to a service was blocked and retrieve reputational information about the involved services. Based on this assumption, ERMES integrated the Transparency Tags as pages directly connected to blocked hostnames.

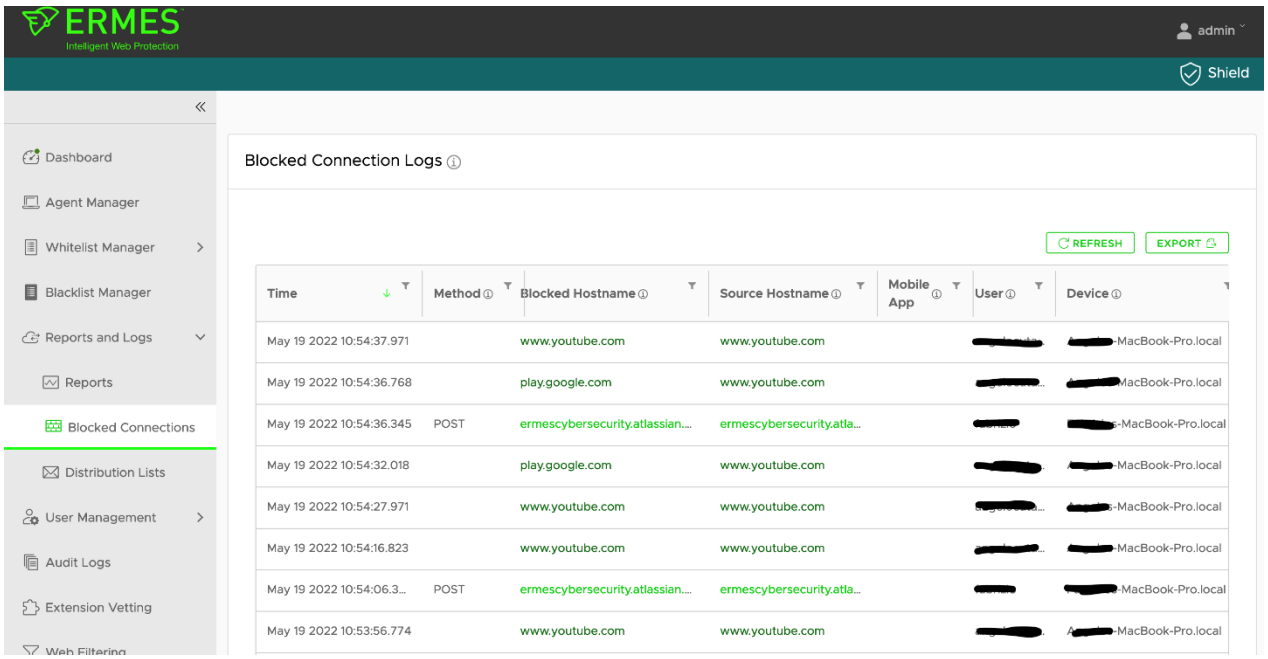


Figure 36 - Report of blocked connections in Ermes for Enterprise (IDs have been obfuscated to preserve the privacy of involved users).

The figure above shows an example of report of connections which have been blocked by Ermes for Enterprise agents. Each row corresponds to a blocked connection and reports the time at which the connection was blocked, the HTTP method used in the connection, the hostname of the web service which has been blocked, the hostname of the website from which the connection was originated, the mobile app generating the request (only for mobile devices), the IDs of user and device involved.
By clicking the hostnames presented in Blocked Hostnames and Source Hostnames columns, the administrator lands on the corresponding Transparency Tag page. In this case we identify two different kinds of hostnames: hostnames corresponding to first-party services such as websites intentionally visited by users (Source Hostnames) and hostnames typically corresponding to third-party services (Blocked Hostnames). Therefore, given a hostname such as *www.youtube.com*, by clicking on the corresponding row under Blocked Connections, the administrator will land on the Transparency Tags of *www.youtube.com* when this service is contacted (and blocked) as a third party, and more specifically as web tracker.

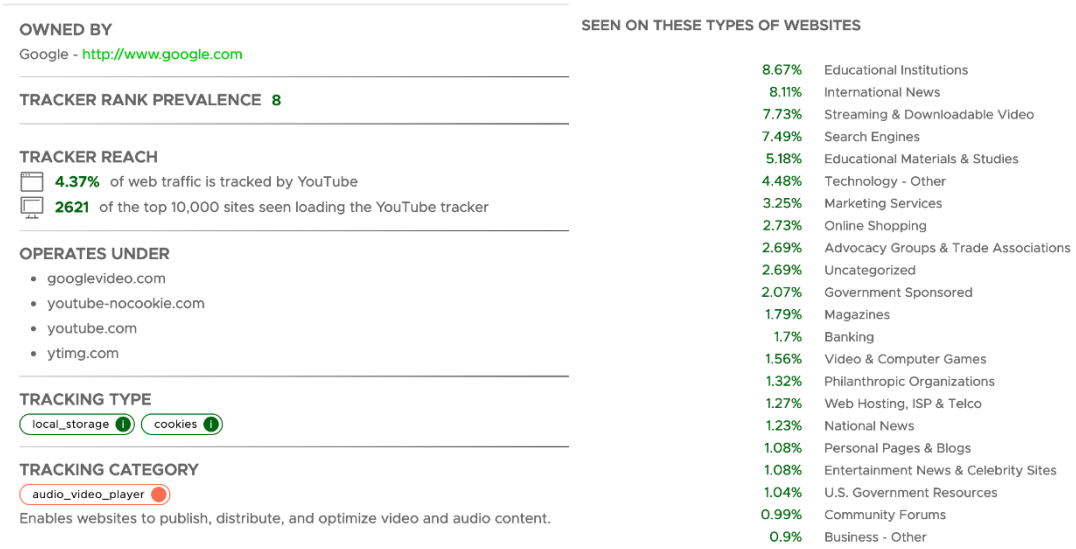


Figure 37 - Ermes For Enterprise Transparency Tags for www.youtube.com when this operates as a third-party tracker.

As shown in the figure above, the TT reports information which help the administrator of the system understand why the service was blocked (it is a tracker), how pervasive it is (prevalence and reach), which company runs it (Google), what other hostnames are connected to this tracker (googlevideo.com, youtube-nocookie.com, etc.), which kind of tracking category it belongs to (audio_video_player) and which kind of technical approach it uses to track users (local storage and cookies). The right-most section shows the web categories the tracker monitors in general.
By clicking on hostnames in Source Hostname the administrator lands on a webpage providing the Transparency Tag of the corresponding service, I.e., a first party such as a website or a portal. In the figure below, we report the Transparency Tag for of *www.youtube.com* as a first party, I.e., when it is intentionally visited by the user.

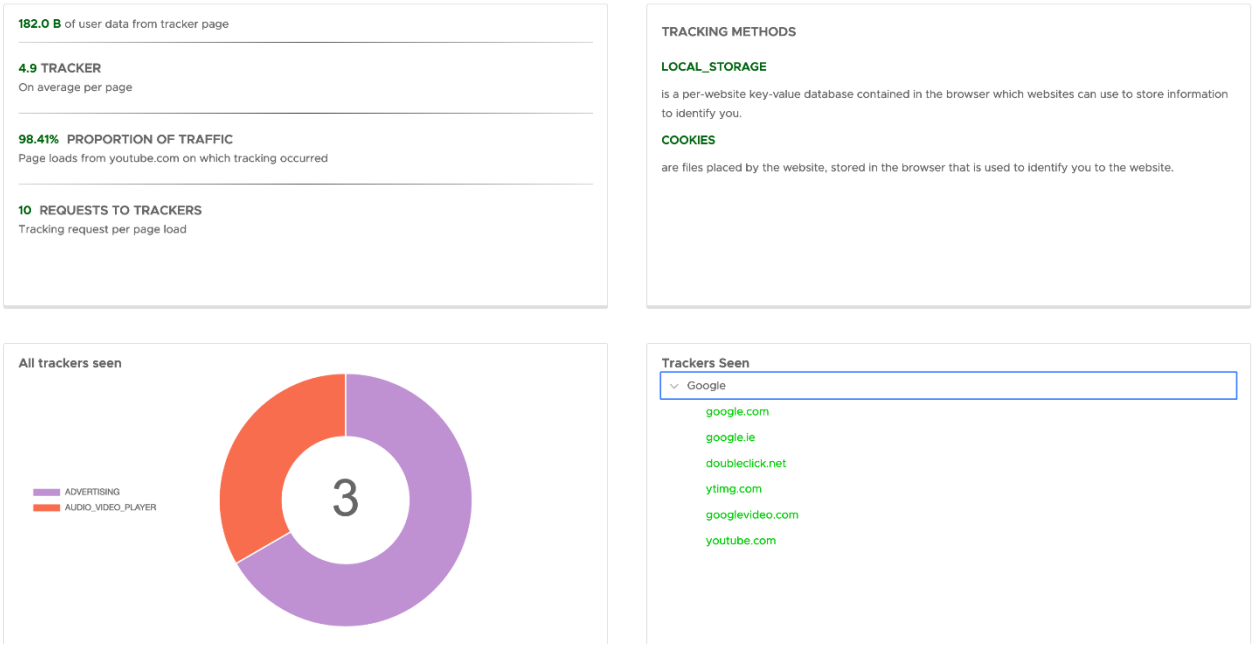


Figure 38 - Ermes for Enterprise Transparency Tag for www.youtube.com when this operates as a first-party website.

As shown in the figure above, in this case, the TT reports information which help the administrator of the system understand how the visited website is behaving from the privacy and security perspectives. The TT in this case reports how much traffic is sent to trackers, how many trackers on average are contacted starting from this website, the fraction of subpages performing some tracking and the average number of connections established with trackers on average. Then, the TT reports the tracking methods used on the websites, the categories of tracking observed on the website and, finally, the tracking hostnames connected to the first party.

8.8 Conclusions on implementation

The code built for the EasyPIMS version of TTs is available on PIMCity's official Gitlab repository, within the PDA implementation.[46] ERMES does not plan to release its software as open source in the short term, but this is an option which is not excluded for the future.
The Transparency Tags interface developed for EasyPIMS and Ermes for Enterprise will be thoroughly used in the demonstrator phase of the project to validate whether they can be successful at creating knowledge and awareness useful for users in both private consumer and business segments. The results of the demonstrators involving actual users will be presented in the next WP1 deliverables.
Finally, we remark that further minor adjustments to code and design might occur in the next months. We will keep track of these in the official Gitlab repository and we will describe them in next deliverables, if needed.

8.9 Deployed Implementations

Transparency Tags are deployed in the development environment for testing purposes at:
https://easypims.pimcity-h2020.eu/privacy-metrics/.
While the production version is deployed at: https://privacy-metrics.easypims.eu/

[46] https://gitlab.com/pimcity/wp5/easypims-ui

9 Design of the Data Marketplace

The Data Marketplace is a platform where companies can participate as Data Buyers in a huge data ecosystem interacting directly with end users, that is, individuals, without intermediaries.

The Data Marketplace is an EasyPIM that integrates with PIMCity's Trading Engine, Consent Manager, Data Valuation Tools and many other PDKs to provide a state-of-the-art service to companies by creating the most powerful and transparent data platform in the industry.

Through its simple and user-friendly interface, Buyers can create Data Offers that are sent to the Trading Engine to collect data points only from users that have given the corresponding consent to share that validated data. This enables companies to acquire either raw, extracted or aggregated data of excellent quality in a GDPR compliant way.

The Data Marketplace has two main ways of buying data:

- Audiences: Data Offers seeking for audiences set up filters that shape the type of consumer profile that the company is looking for. Once the offer is executed, the Buyer obtains an audience with which it can interact through ads and other use cases.
- Raw, extracted or aggregated data: Data Offers seeking actual data set up consumer filters as well, but in the end the company obtains records over which it can run machine learning models or use for many business intelligence use cases.

9.1 General Design

The system is designed in layers where:

1. Layer Zero contains all the databases required to persist the platform's state
2. Layer One contains PIMCity PDKs and microservices which encapsulate the business rules.
3. Layer Two contains the Data Marketplace EasyPIM and the Cloud Controller for authenticated access
4. Layer Three is where the user resides and therefore where all the actions are generated.

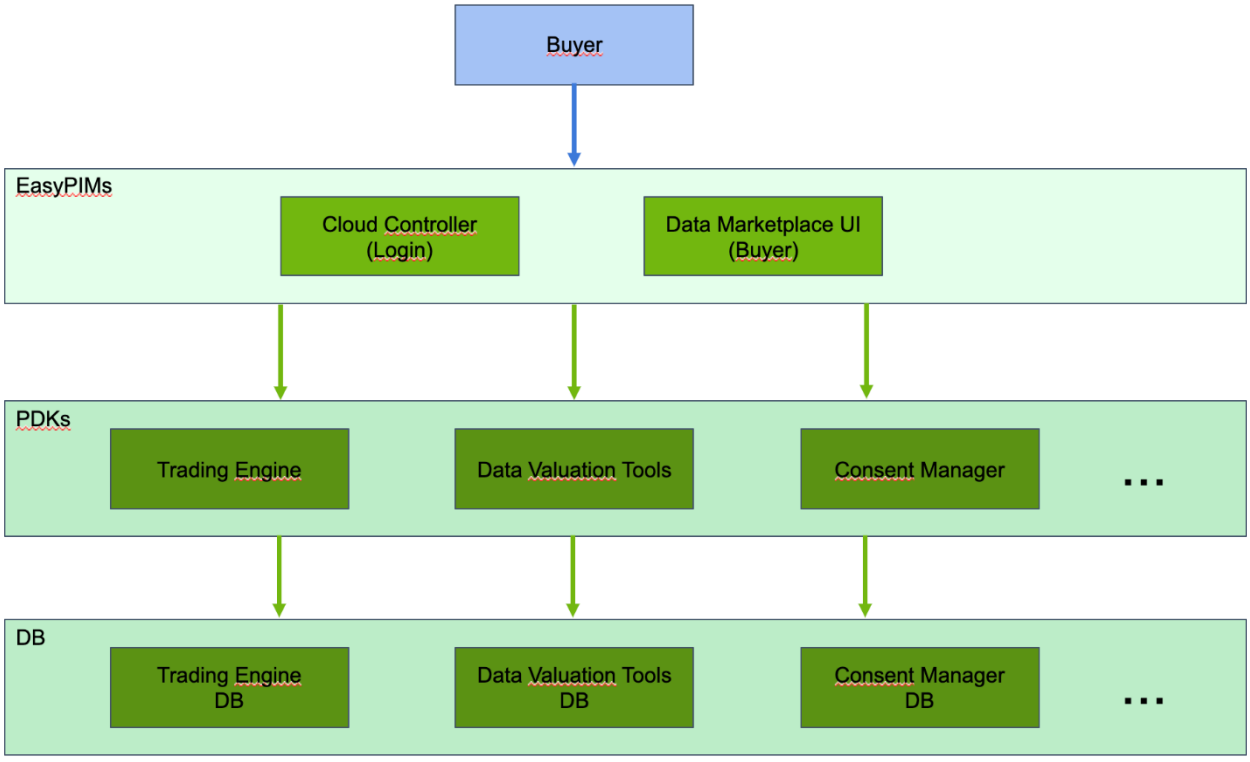


Figure 39 - Data Marketplace webmap

9.1.1 Data Marketplace components

9.1.1.1 Registration and Login

Entrance points for all the Data Buyers to the platform. On these screens the visitor can register on its first-time visit to create an account and become a Data Buyer. Afterwards, the user can log in to manage his account, with his balance and Data Transactions.

When registering, a basic form needs to be completed, including details such as Company Name, Job Title, user email, and so on.

In this part of the platform, the user can also choose to reset her password in case the password was lost, forgotten or even compromised.

Functionalities

First, it provides the Data Buyer the webpage to register on the platform by submitting the following information and accepting PIMCity's Privacy Policy and Terms of service:

- Name
- Last Name
- Corporate email
- Company

This screen also provides the possibility to pick a password, and access the Privacy Policy.

Alternatively, the user can log in to the platform using email and password or reset her password.

Implementation

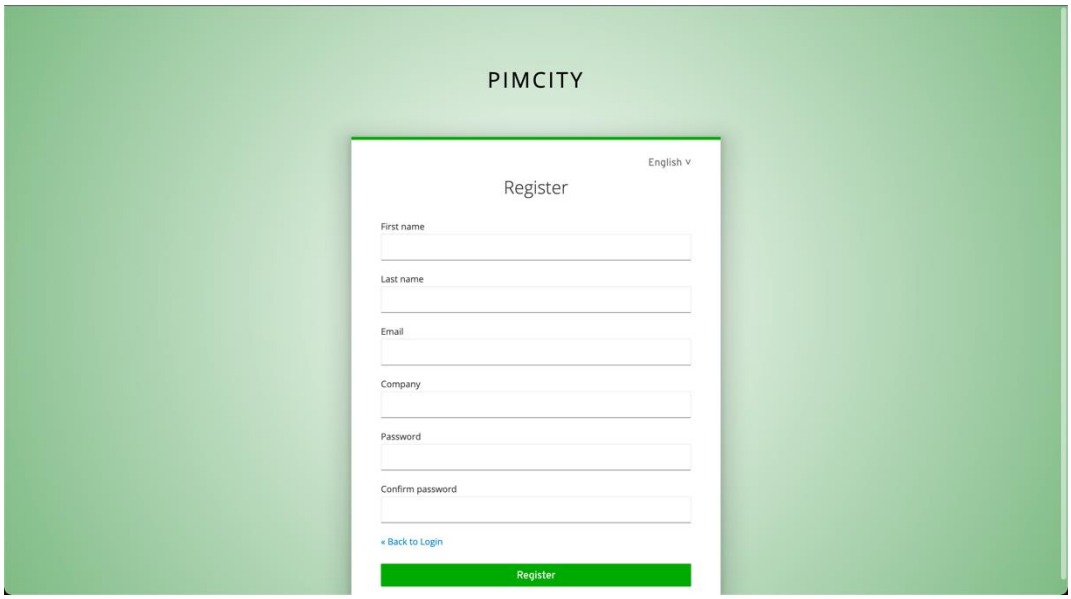


Figure 40 – Registration screen implementation for Data Marketplace

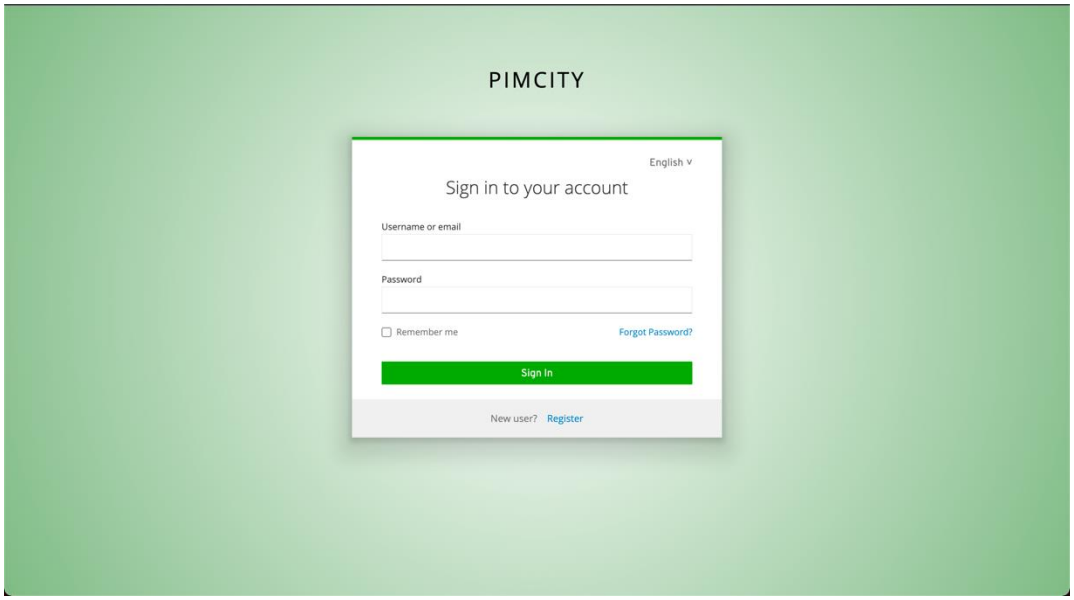


Figure 41 - Data Marketplace's login screen implementation

Interactions

This screen interacts mainly with the Cloud Controller to generate the account, reset passwords and log in. However, it also interacts with the Trading Engine to create the Data Buyer entity with all the company information and enable the creation of Data Transactions within the platform.

9.1.1.2 Main Screen

The main screen is the place where Data Buyers go after registering or logging in. It is a dashboard where the buyer can see a short list of the last five Data Transactions done, as well as the last (or most popular) five Data Transactions placed by other Data Buyers. The latter works as a way of showing trends within the market, without disclosing the specific companies that are creating those transactions.

Last but not least, the current balance in credits is shown. This is how the Data Buyer can do budget planning for the following Data Transactions to be done. For instance, use 30% of the budget to obtain a certain audience, while using the other 70% to obtain data from another cohort.

Functionalities

This screen provides the following functionalities:

- Show a short list of the last five opened and closed data transactions
- Show the current balance in credits
- Show the number of credits already spent

Implementation

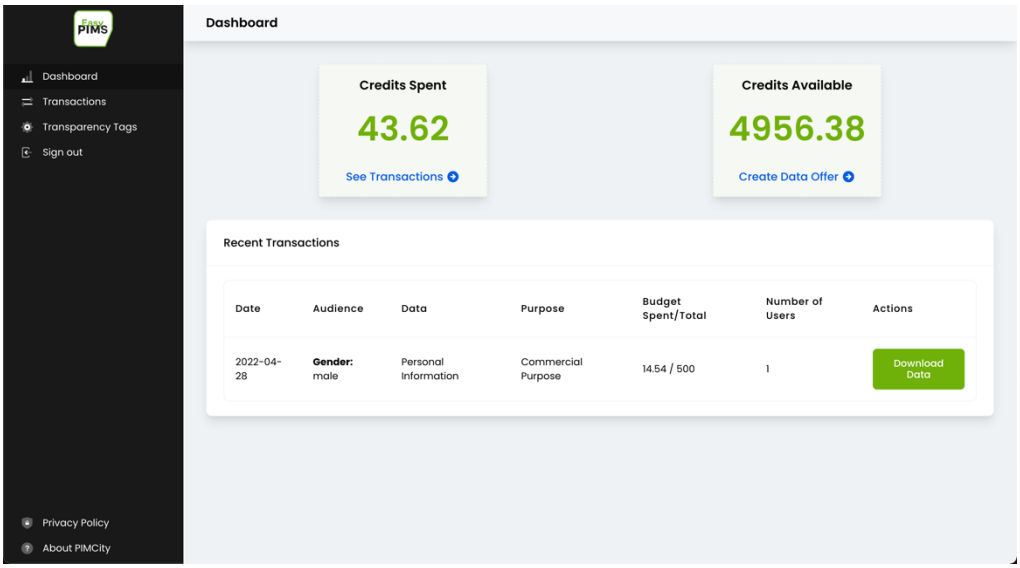


Figure 42 - Implementation of Data Marketplace's main screen

Interactions

This screen interacts directly with the Trading Engine to extract the Data Buyer's credits balance, as well as a short list of past and most popular Data Transactions. As the PDK service requires authentication, the JWT obtained from the Cloud Controller when the buyer logged in is used.

9.1.1.3 Transaction overview

The Transaction Overview presents a list of all past Data Transactions made by the data buyer. It works as a log of all the data acquired or given access to, and for which purpose. In the same list, the buyer can also see the credits spent in the transactions.

Functionalities

The Transaction Overview screen provides the following functionalities:

- List all Data Transactions made by the buyer.
- Show all details of all Data Transactions listed: data acquired, credits spent, privacy policy used, dates, and so on.

With these functionalities, the buyer can have and read the log of all the transactions involving the data obtained.

Implementation

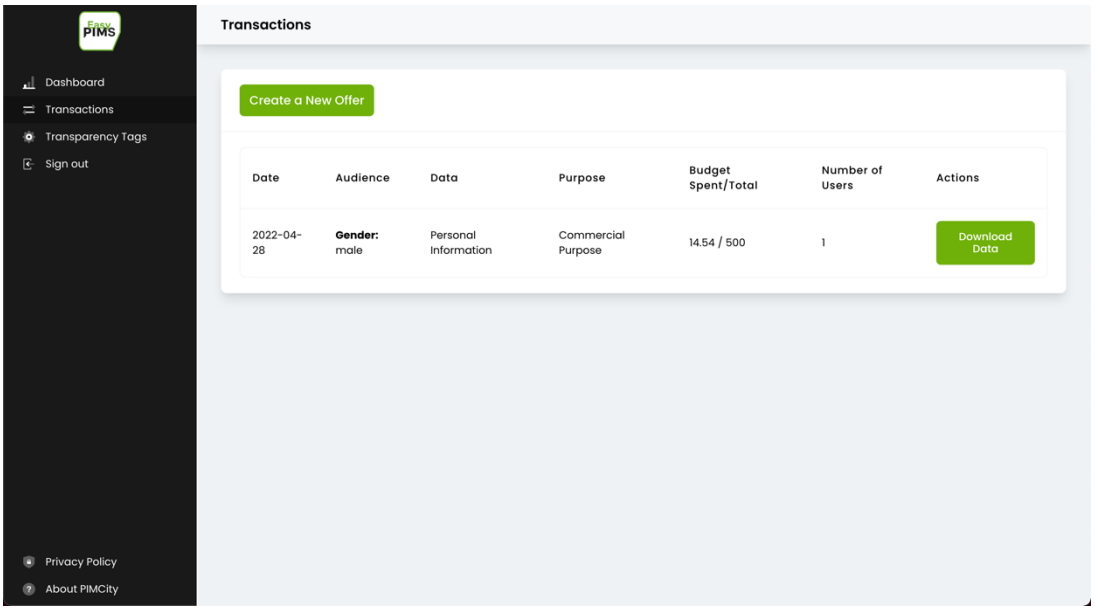


Figure 43 - Implementation of Data Marketplace's transactions overview screen

Interactions

This screen interacts directly with the Trading Engine to extract the list of past Data Transactions and all their details. As the PDK service requires authentication, the JWT obtained from the Cloud Controller when the buyer logged in is used.

9.1.1.4 Creation of a Data Transaction

The Creation of a Data Transaction presents the Data Buyer with the possibility to place a Data Offer in the Data Marketplace available to the users. When creating an offer, the Data Buyer can specify the audience, the type of data, the privacy policy and the budget to be spent, among other information.

Functionalities

This screen allows the buyer to create a new Data Transaction. The buyer is able to define the audience sought, the privacy policy to be used, the purpose for which the data is being obtained, which data is requested, the budget to be spent in the offer, how long that will be used and when it will be deleted, and so on. When this Data Transaction is fulfilled, the Data Buyer can download the data.

Implementation

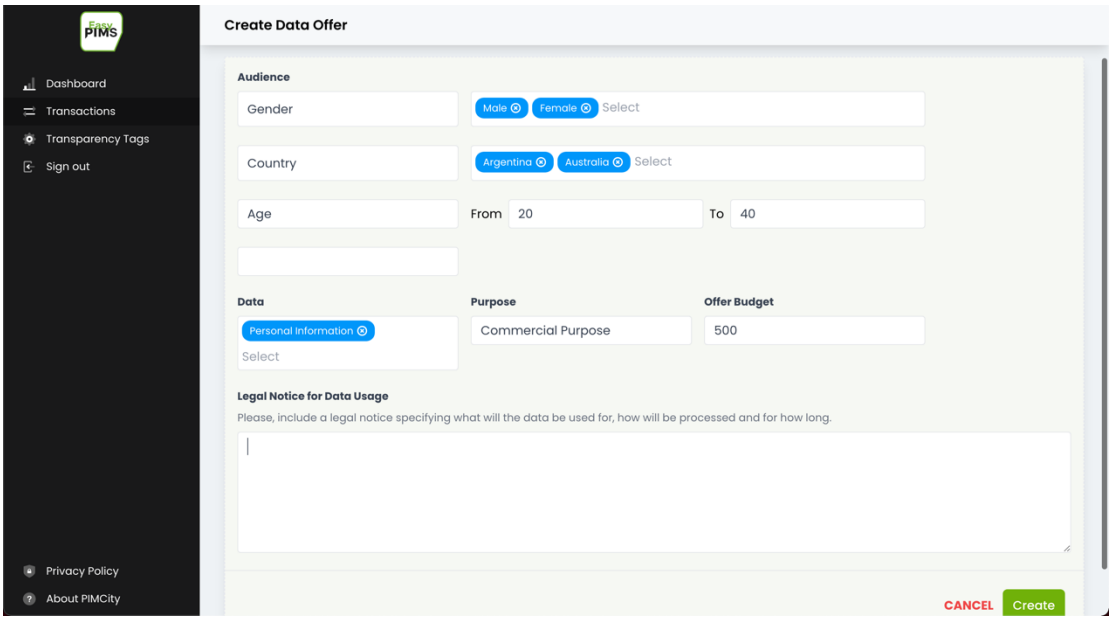


Figure 44 – Implementation of Data Marketplace's screen to create new data offer

Interactions

This screen interacts directly with the Trading Engine to create a new Data Transaction. As the PDK service requires authentication, the JWT obtained from the Cloud Controller when the buyer logged in is used.

9.1.1.5 Transparency Tags

This screen allows the buyer to read all the account information and company details, as well as to edit the transparency tags of the account.

Functionalities

This screen allows the buyer to perform the following actions:

- Read all the company details submitted during the registration phase
- Edit Transparency tags

Implementation

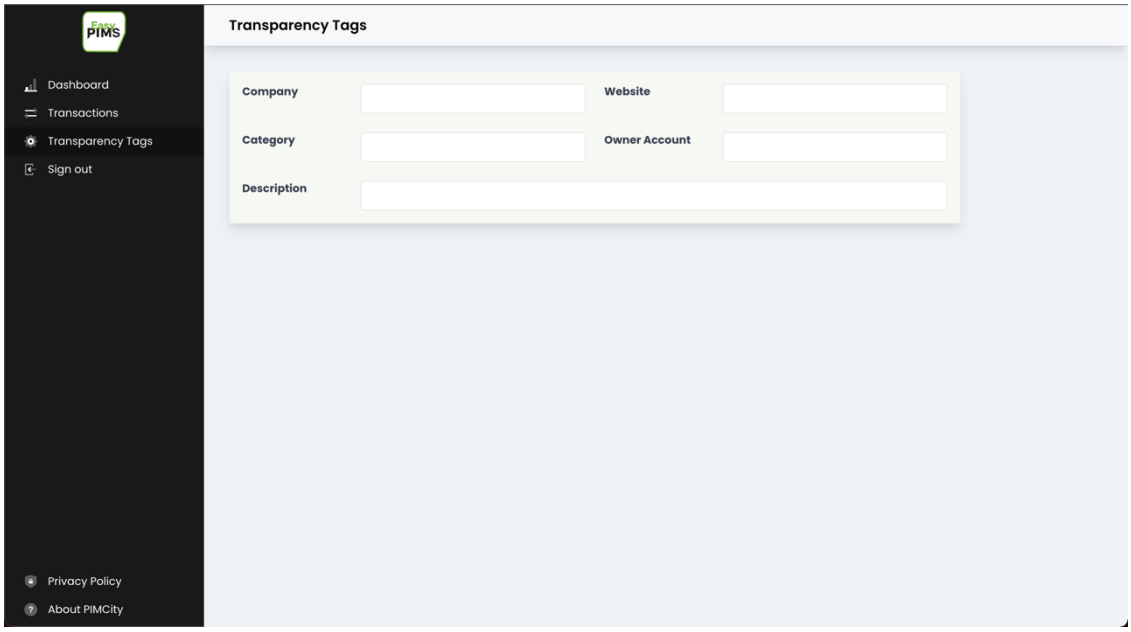


Figure 45 - Implementation of Data Marketplace's settings screen

Interactions

This screen interacts directly with the P-PM PDK to extract all the information about the company and apply the Transparency Tags to it. As P-PM PDK services require authentication, the JWT obtained from the Cloud Controller when the buyer logged in is used.

Also, the screen interacts with the Cloud controller to change the password or delete the buyer account.

9.2 Deployed Implementations

The Data Marketplace is deployed in the development environment for testing purposes at: https://easypims.pimcity-h2020.eu/data-marketplace. While the production version is deployed at: https://marketplace.easypims.eu/

10 Design of the Task Manager

The Task Manager is a novel component that has been introduced to allow user engagement. Indeed, users can get rewards upon completing tasks and small challenges within the system. With the Task Manager end-users can:

- See which are the available tasks in the platform to be done.
- Earn points by completing such tasks.
- Participate in sweepstakes when reaching certain number of points.
- Invite friends to the platform with a referral code and earn more points.

And where PIMCity administrators can:

- See the list of active sweepstakes.
- View the list of qualified participants for each sweepstake.
- Set a winner for a specific execution of each sweepstake.
- View past winners.
- Export the list of qualified participants in PDF format.

The Task Manager is an EasyPIMS component that interacts with PIMCity's Trading Engine and Cloud Controller to provide a state-of-the-art service that raises the level of engagement in PIMCity end-users by introducing the gamification concept. Through its simple and user-friendly API and back-office UI, administrators and end-users (using the PDA) can interact with these features without doing any heavy-lifting step. The expected outcome of including the Task Manager in PIMCity is to see users being incentivized to upload data, enable consents, and share it with Data Buyers.

10.1 General Design

The system is designed in layers where:

1. Layer Zero contains the Task Manager DB required to persist the platform's state and the Trading Engine PDK to bookkeep the end-users' points.
2. Layer One contains the Task Manager API and the Cloud Controller for authenticated access.
3. Layer Two (optional) contains the Sweepstakes back-office UI to administer the sweepstakes in a user-friendly way.
4. Layer Three is where the end-users, administrators and other internal components reside and therefore where all the actions are generated.

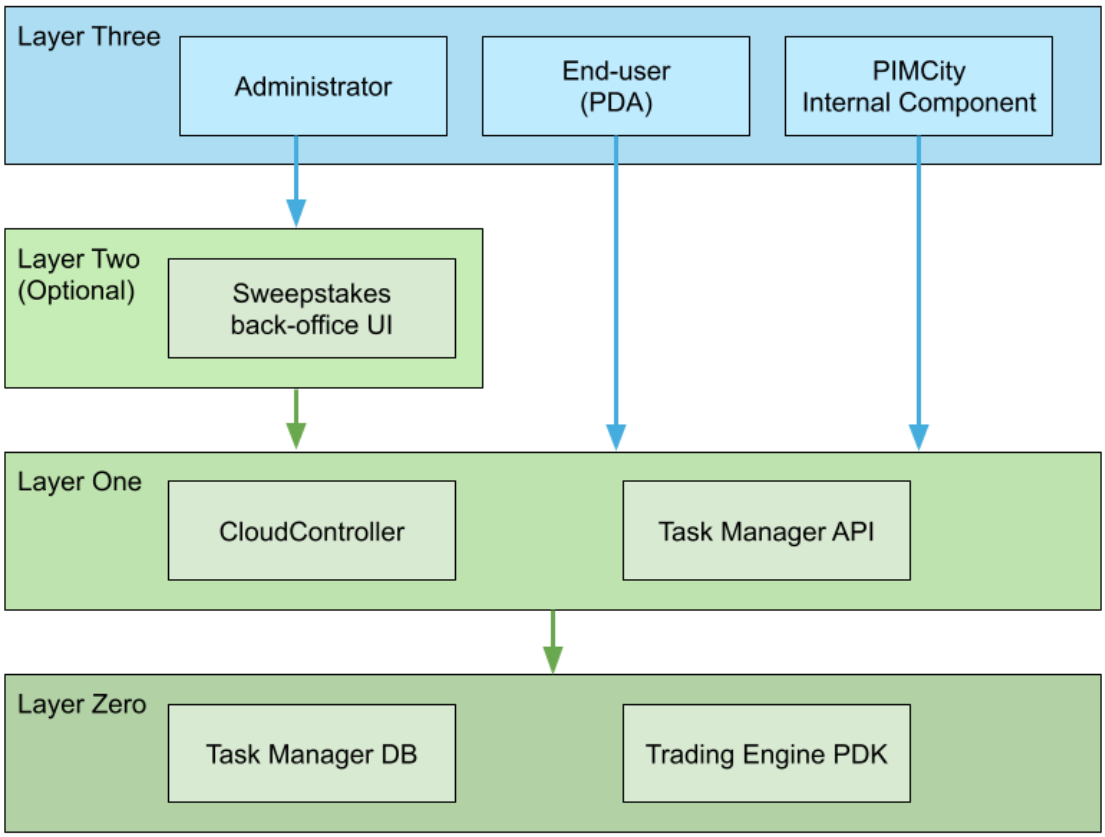


Figure 46 - Task Manager layered design

10.2 Components

The Task Manager is composed of two main components:

- Task Manager API (with its DB)
- Sweepstakes back-office UI

Between these two parts of the Task Manager, the user can earn points, invite friends to the platform and qualify for sweepstakes when surpassing points thresholds.

10.2.1 API

The Task Manager API is divided in three main features:

- Tasks
- Sweepstakes
- Referrals

These features are used by end-users through the PDA, internal components using API Keys and PIMCity administrators to execute sweepstakes. This API is implemented using node.js with express.js, the same framework used to create the Personal Consent Manager and Trading Engine.

10.2.1.1 Tasks

Users can earn points in three different ways:

- By sharing data
- By grant from the administrators
- By completing tasks

The latter is the core feature of the Task Manager. Users get a list of the available tasks to be completed to earn points. Each task is defined with the following information:

- TaskId: An ID that uniquely defines the task
- Points: The amount of points the user earns when completing the task
- Periodicity: The periodicity in which the user can complete the task (one-time, daily, weekly, monthly, yearly)

- Max. Repetitions: The times the user can repeat the task to earn points within the given periodicity.

The Task Manager API allows the user to get the list of the available tasks, along with its completion status. It also has an endpoint for internal PIMCity components to mark any task as completed (e.g., the Personal Consent Manager sends a request to the Task Manager to mark the task "Set Consent" as completed when the user enables sharing the Personal Information for Commercial Purposes).

API definitions

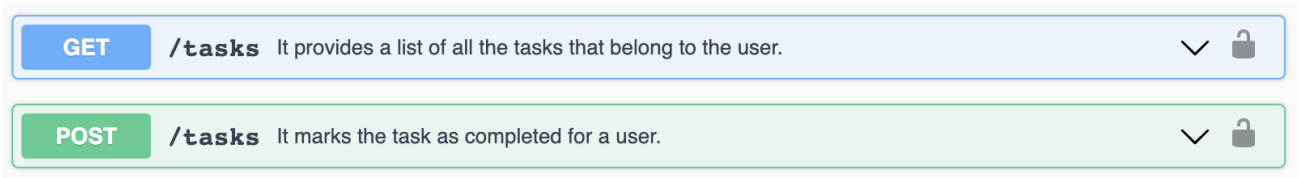


Figure 47 – Tasks API endpoints design

Interactions

The Task Manager API interacts with the Trading Engine to update the end user's points whenever a task is marked as completed. It also requires the corresponding JWT from the Cloud Controller.

10.2.1.2 Sweepstakes

Points earned by end users in the platform are used to qualify in different sweepstakes. All points serve for such purpose, not matter their origin, i.e. points earned by completing tasks, sharing data and/or any other interaction rewarded with points are added up to automatically register the user in a particular sweepstake.

The only restriction is that points only count toward a sweepstake if they were earned within a particular time interval. For example, weekly sweepstakes only count points earned within the previous week. Monthly sweepstakes count points earned within the previous month. "One-time" sweepstakes count points from the launch date of the platform until the execution date of such sweepstake. The Task Manager API allows the user to get the list of active sweepstakes. It also has the back-office endpoints for the sweepstake administration.

API definitions

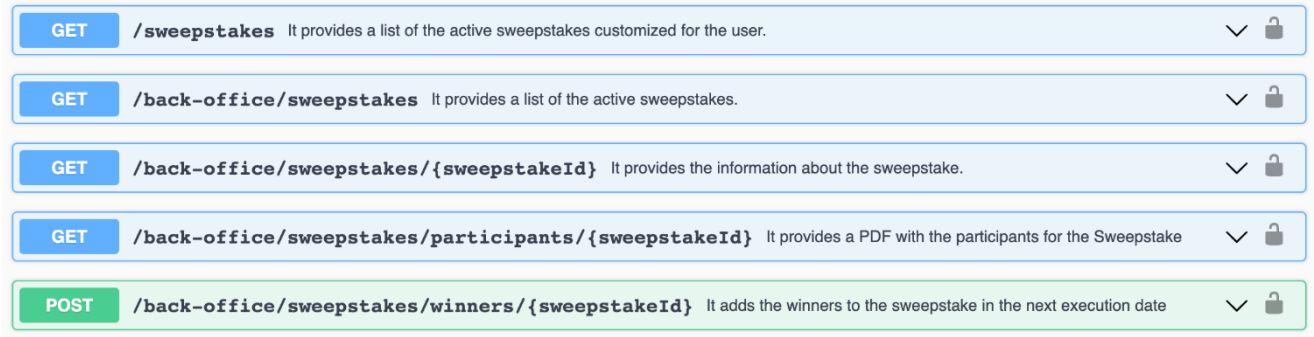


Figure 48 – Sweepstakes API endpoints design

Interactions

The Task Manager API interacts with the Trading Engine to get the end user's points and tailor the list of active sweepstakes by modifying the points needed to qualify.

It also requires the corresponding JWT from the Cloud Controller.

10.2.1.3 Referrals

Any end user in the platform can earn points by inviting other people to join PIMCity. In order for the referrer to earn those points, the referee must first register and earn at least 100 points. Only the first five successful referrals per month enable the referrer to earn points. Afterwards, the user can keep inviting people to the platform, but no points will be earned.
The Task Manager API defines three main endpoints for this feature:

- Get or create a referrer code
- Use a referral code when signing up
- Check if a referee has already surpassed the 100 points threshold and mark the “Invite Friends” task as completed in the referrer account.

The referrer code is shown in the Settings section of the PDA for the user to copy it and share it with friends.

API definitions

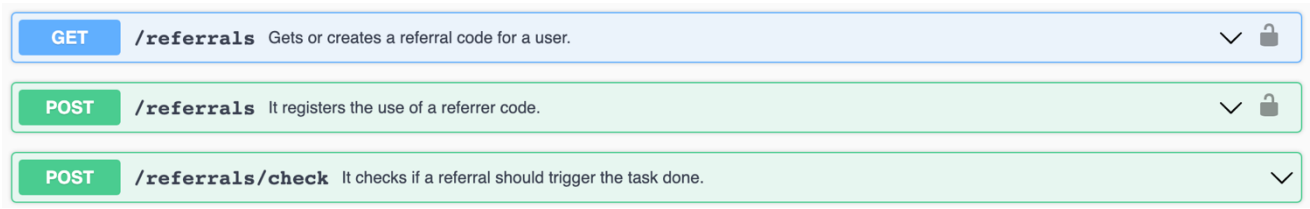


Figure 49 - Referral API endpoints design

Interactions

This feature is used by the PDA to get or create the referrer code and use a referral code. Also, the Trading Engine uses the endpoint to check if a referee reached the 100 points threshold every time a user registers a new transaction. Authentication using the Cloud Controller is needed to get, create, or use a referrer code.

10.2.2 Sweepstakes back-office UI

In order to administer the sweepstakes within the platform, a back-office frontend was defined, implemented and deployed. With this tool, any administrator can view the list of existing sweepstakes and keep updated each one of them.
For each sweepstake, the administrator can:

- Export the list of qualified participants in PDF format
- Set a winner
- View the list of past winners

Every time a winner is set, the execution date of the sweepstake is updated, and the winner is communicated within the platform.
This back-office frontend is implemented using React.js, the same tool used to create the Data Marketplace.

10.2.3 Login

The Sweepstake back-office requires authentication to access its features. Only the users with the role “task-manager-admin” within the Cloud Controller can log in. This role is manually set within the KeyCloak Administration Console by a Cloud Controller administrator.

Functionalities

By writing the username and password, the user can log in (if it has the proper role set) and access the Sweepstake back-office. Standard features like “Forgot Password?” are also offered.

Implementation

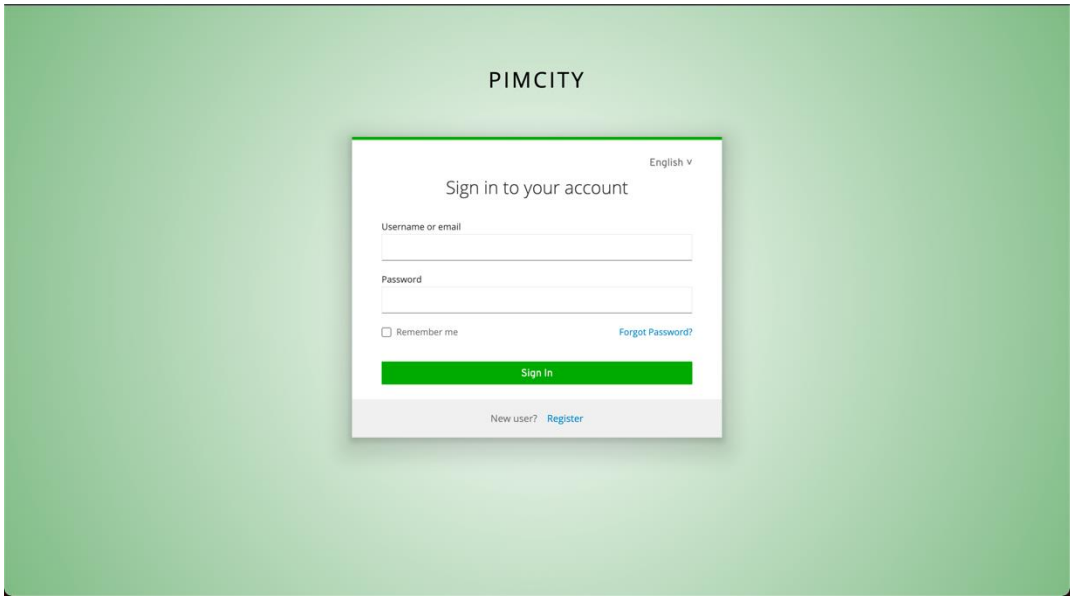


Figure 50 - Sweepstake back-office login screen

Interactions

This screen interacts mainly with the Cloud Controller to log in and reset passwords. The resulting JWT is then used in the rest of the screens when sending a request to the backend.

10.2.3.1 Viewing the list of Sweepstakes

Viewing the list of Sweepstakes gives the administrator of the back-office the possibility to see the currently active sweepstakes within the platform. Whatever is shown here, is shown to the user in the PDA home.

Functionalities

Each sweepstake shows its name, prize, points needed to qualify, next execution date and action buttons for the administrator.

Implementation

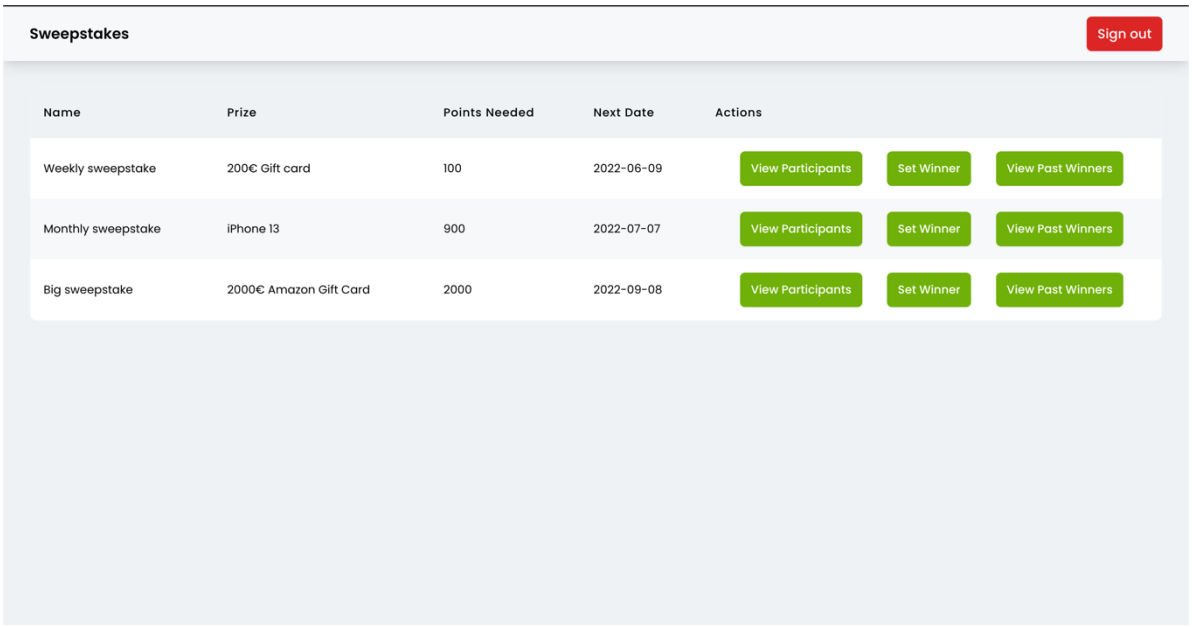


Figure 51 - Implementation of page showing the list of currently active sweepstakes.

Interactions

This feature interacts with the Task Manager API to get the list of sweepstakes, authenticating with the JWT obtained in the login of the tool.

10.2.3.2 Exporting qualified participants

Exporting qualified participants gives the administrator of the back-office the possibility to download a PDF file with the list of users that will participate in a specific sweepstake. The list assigns incremental numbers to the participants that represent the lottery ticket to be used in the sweepstake.

Functionalities

This feature allows the administrator to export the list of participants of a sweepstake in PDF format when clicking on the “View Participants” button. The administrator can then download this PDF and publish it in the EasyPIMS website to the public.

Each PDF shows the date interval for the sweepstake, the points needed to qualify, the prize and the list of user IDs along with their assigned lottery number.

Implementation

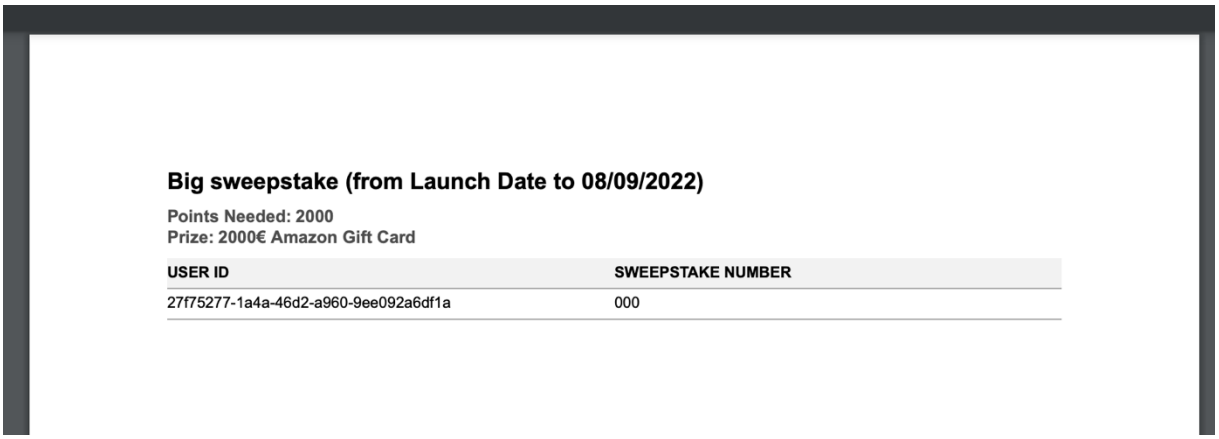


Figure 52 - PDF exported by the back-office to publish to the participants.

Interactions

This feature interacts with the Task Manager API to get the PDF export, authenticating with the JWT obtained in the login of the tool.

10.2.3.3 Setting a winner

Setting a winner gives the administrator of the back-office the possibility to define which user won a specific sweepstake on a specific date. This is logged within the platform and then end users can check in the PDA whether they won or not.

Functionalities

This screen allows the administrator to set the winner of a sweepstake rendered in a modal message (i.e. pop-up) when clicking on the “Set Winner” button.

Implementation

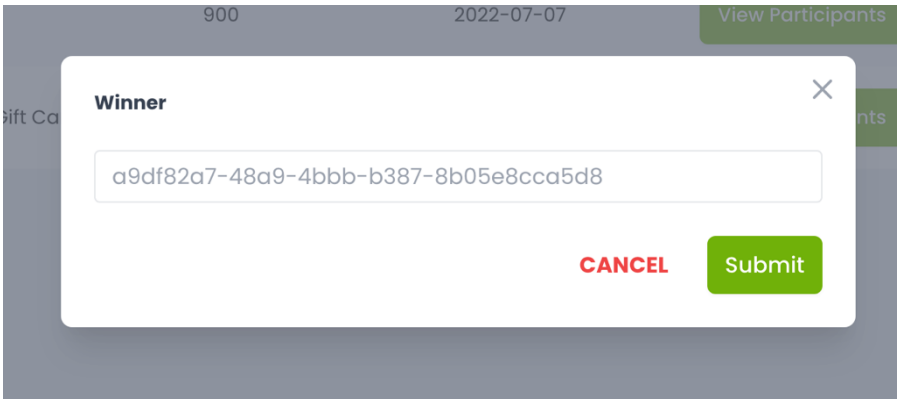


Figure 53 - Implementation of Set Winner screen.

Interactions

This pop-up message interacts with the Task Manager API to set the winner of a sweepstake, authenticating with the JWT obtained in the login of the tool.

10.2.3.4 Viewing past winners

Viewing past winners gives the administrator of the back-office the possibility to read the record of users that have already qualified and won a specific sweepstake since the launch of the platform.

Functionalities

This screen allows the administrator to see the list of past winners of a sweepstake rendered in a modal message (i.e., pop-up) when clicking on the “View Past Winners” button.

Implementation

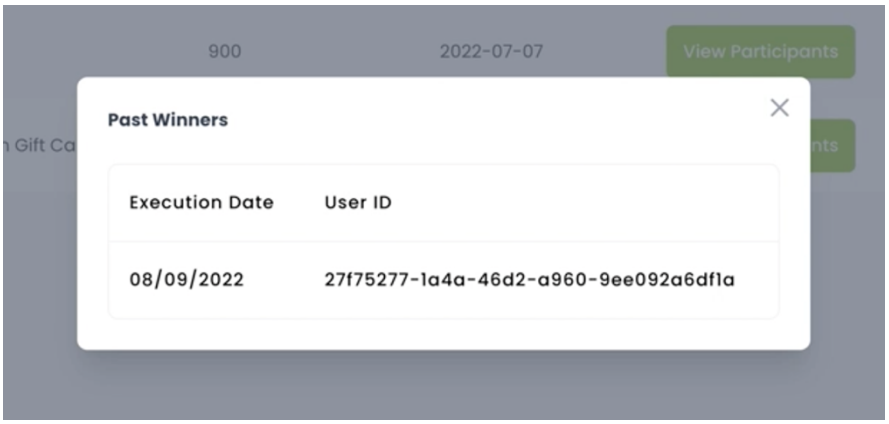


Figure 54 - Implementation of View Past Winner screen.

Interactions

This pop-up message interacts with the Task Manager API to get the list of winners of a sweepstake, authenticating with the JWT obtained in the login of the tool.

10.3 Deployed Implementations

The Task Manager is deployed in the development environment for testing purposes at:

- API: https://easypims.pimcity-h2020.eu/task-manager-api
- Sweepstakes back-office: https://easypims.pimcity-h2020.eu/task-manager

While the production version is deployed at:

- API: https://tasks.easypims.eu
- Sweepstakes back-office: https://sweepstakes.easypims.eu

11 Conclusions

This deliverable completes the design proposals presented in the first deliverable of WP5, Deliverable 5.1. In this deliverable we presented the final design of EasyPIMS' fundamental components and their corresponding implementations. We described the final design for the Cloud Controller, the Open APIs, the Personal Data Avatar, the Transparency Tags and the Data Marketplace. Moreover, we described the design of the Task Manager, a novel component introduced to allow to engage users with tasks and interactions. For each component we described the technical requirements, the design choices, and when possible, the preliminary feedback we collected. We also described how we implemented each of them and made available the code in PIMCity's official Gitlab repository. This way, any user interested in our solutions has the possibility to understand the design choices and check and use the corresponding implementation.

This document also provided the description of the platforms on which EasyPIMS has been built and deployed. We indeed provided an overview of the overall logic platform, as well as the detailed description of how the development and production platforms have been built. Furthermore, this deliverable has described the tools, frameworks and environment that has been used for the development.

12 Bibliography

[1] PIMCity, *Deliverable D1.1 - PIMCity Requirements and Specifications,* 2020.

[2] PIMCity, *Deliverable D7.1 - Data Management Plan,* 2020.

[3] Auth0, "Authorization Code authentication flow," [Online]. Available: https://auth0.com/docs/flows/authorization-code-flow.

[4] Auth0, "Client Credentials authentication flow.," [Online]. Available: https://auth0.com/docs/flows/client-credentials-flow.

[5] PIMCity, *Deliverable D2.1 - Design of tools to improve user's privacy,* 2020.

[6] PIMCity, *Deliverable D2.2 - Design of tools to improve user's privacy,* 2020.

[7] PIMCity, *Deliverable D3.2 - Design of data valuation tools and trading engine,* 2020.

[8] PIMCity, *Deliverable D3.3 - Final design and preliminary version of the data valuation tools and trading engine,* 2021.

[9] PIMCity, *Deliverable D4.1 - Design of tools for improved data management,* 2020.

[10] PIMCity, *Deliverable D4.2 - Final design and preliminary version of the tools for improved data management,* 2021.

[11] P. G. Kelley, L. Cesca, J. Bresee and L. F. Cranor, "Standardizing Privacy Notices:An Online Study of the Nutrition Label Approach," 2010.

13 – Appendix – A1

In this section we report the raw responses to the questionnaires about Transparency Tags design we circulated during a campaign of feedback collection. In total we collected data from 52 respondents. The aggregated results, together with the conclusions we drove out of them are presented in Chapter 8.

ID	Age	Gender	Category of the company that is buying your data	Purpose on the use of your data: For what purpose is the company acquiring your data	Short description of the company	Company security and privacy rating	Link to privacy policy of the company	Country (headquarter) of the company	Other information that you would like to know about the company (data buyer)
1	Less than 28	Male	5	3	4	5	5	3	No
2	Less than 28	Male	3	3	3	5	4	2	.
3	Less than 28	Male	5	5	4	4	5	4	Not really
4	Less than 28	Male	4	5	5	5	5	4	Prestige
5	Less than 28	Male	5	3	2	3	1	3	No
6	Less than 35	Male	3	2	3	5	3	5	Buisness
7	Less than 28	Male	3	5	2	4	2	3	Cantidad de usuarios a los que le solicita info.
8	Less than 28	Male	1	2	3	2	3	4	Nada
9	Less than 35	Male	3	4	1	1	1	2	.
10	Less than 28	Male	5	3	1	3	1	3	No
11	Less than 35	Male	5	5	3	4	4	5	No
12	More than 50	Male	4	5	5	3	2	2	How I can delete my data without contacting the company

13	Less than 35	Male	4	5	4	5	2	2	Number of people that are selling their data
14	Less than 28	Male	4	5	3	5	2	5	Company name
15	Less than 35	Femal e	4	4	3	4	2	5	Ceo
16	More than 50	Male	4	3	2	2	4	3	Adress
17	Less than 28	Male	4	5	4	5	1	1	Time of data used
18	Less than 28	Male	4	4	4	4	5	5	Na
19	Less than 28	Male	4	4	3	4	3	4	J
20	Less than 35	Male	5	5	5	5	3	3	No
21	Less than 28	Male	5	5	3	4	2	3	Fame
22	Less than 50	Male	4	5	2	3	4	3	A metric that shows how many pieces of data this buyer bought recently would make me feel confident like "hey people are selling their data to this company". Also, how many times I sold data to this company could help
23	Less than 35	Male	5	5	3	3	2	1	No
24	Less than 28	Male	4	4	4	5	5	3	I
25	Less than 28	Male	5	3	4	5	5	3	No
26	Less than 35	Male	4	4	2	4	3	1	What data is obtaining from me

27	Less than 28	Prefer not to say	1	1	1	2	1	2	How much is paying for the data
28	Less than 50	Male	4	5	5	4	1	5	How much they are making with my data
29	More than 50	Male	5	5	5	5	5	5	How much time are they going to use my data
30	Less than 28	Prefer not to say	3	3	2	3	1	2	No
31	Less than 50	Male	3	3	2	3	4	2	How much are they paying
32	Less than 35	Female	3	3	2	4	1	1	What data am i giving
33	Less than 50	Prefer not to say	1	4	4	4	3	3	Will they re-sell my data?
34	More than 50	Male	5	5	5	5	4	4	Can they share it with third-parties?
35	Less than 35	Female	4	4	3	4	5	2	How much are they paying me for the data?
36	Less than 28	Male	2	2	1	2	1	2	What do they give me?
37	Less than 35	Male	3	3	4	3	4	3	No
38	More than 50	Female	5	5	4	3	2	3	How long are they using my data?
39	Less than 28	Male	5	1	1	1	3	5	No
40	Less than 35	Male	5	5	1	4	1	3	Na

41	More than 50	Male	4	3	2	2	4	3	Adress
42	Less than 35	Male	5	5	3	3	2	1	No
43	Less than 28	Male	2	5	4	5	2	1	Will I be able to eliminate my data?
44	Less than 28	Male	3	3	3	5	4	2	.
45	More than 50	Male	4	5	5	3	2	2	How I can delete my data without contacting the company
46	Less than 35	Female	3	5	3	5	2	2	history record
47	Less than 50	Female	4	5	3	4	2	1	no
48	More than 50	Male	4	4	5	5	1	1	no
49	Less than 50	Female	3	4	3	5	2	2	price
50	Less than 35	Prefer not to say	5	4	4	5	3	2	privacy contact
51	Less than 35	Male	4	4	4	5	2	2	No
52	More than 50	Male	3	5	3	5	1	1	Cuanto tiempo la van a usar

14 – Appendix – A2

In this section we report the screenshot of PIMCity's official Gitlab repository containing all the code which has been developed for WP5 EasyPIMS components.

Main WP5 code page

PIMCity platform design and integration (WP5)
Group ID: 11384933

Search by name

D Data Marketplace

O open-api
This repo contains APIs defined using OpenAPI V3.0 for each of PIMCity's components

Personal Data Avatar

T Task Manager

Open APIs code page

Changes to free tier open source projects

Before July 1, 2022, all free tier public open source projects must enroll in the GitLab for Open Source Program to continue to receive GitLab Ultimate benefits.

For more information, see the FAQ.

O open-api
Project ID: 25274233

This repo contains APIs defined using OpenAPI V3.0 for each of PIMCity's components

minor
Martino Trevisan authored 3 months ago

Name	Last commit	Last update
WP2	minor	3 months ago
WP3	Update WP3/DVTMP.yaml	4 months ago
WP4	edits	7 months ago
README.md	Adds README file	10 months ago

README.md

EasyPIMS Open APIs

Open APIs are a key part of SDKs as they permit to seamlessly integrate PIMCity components for enabling their interoperability and re-usability. They represent a logic substratum of endpoints which will be designed and implemented uniformly across all components, so that the same security, privacy and scalability requirements will be met overall the deployment. In this deliverable D5.1 we report the principles and requirements that drove us in the design of the Open APIs for all components building PIMCity's PDK and EasyPIMS. In D5.1, we also present the standards we adopted for the design and the tools chosen to accelerate the implementation.

In this repository, for each component designed in WP2, WP3 and WP4, we provide the design and implementation of its APIs.

Personal Data Avatar code page

Changes to free tier open source projects

Before July 1, 2022, all free tier public open source projects must enroll in the GitLab for Open Source Program to continue to receive GitLab Ultimate benefits.

For more information, see the FAQ.

Personal Data Avatar
Project ID: 29605496

Header: Added hamburguer icon when smartphone mode
Xavi Olivares authored 3 weeks ago

Name	Last commit	Last update
i18n	Overall: Corrected typos	3 weeks ago
scripts	Docker: Added prod environment and scri…	2 months ago
src	Header: Added hamburguer icon when s…	3 weeks ago
.browserslistrc	Initial version	8 months ago
.dockerignore	Docker: Added prod environment and scri…	2 months ago
.editorconfig	Initial version	8 months ago
.gitignore	Initial version	8 months ago
Dockerfile	Fixed Docker dev env	5 months ago
Dockerfile-prod	Docker: Added prod environment and scri…	2 months ago
README.md	GREEN: My personal data form.	3 months ago
angular.json	Updated spanish translation	4 weeks ago
docker-compose-prod.yml	Docker: Added prod environment and scri…	2 months ago
docker-compose.yml	Fixed Docker dev env	5 months ago
karma.conf.js	Initial version	8 months ago
ngcc.config.js	Initial version	8 months ago
nginx.conf	Header: Added hamburguer icon when s…	3 weeks ago
package-lock.json	SCSS: Global deletion of comments due t…	1 month ago
package.json	Updated spanish translation	4 weeks ago
restart.bat	Changed UI colors	4 months ago
translation_conf.json	Added translations	3 months ago
tsconfig.app.json	Initial version	8 months ago
tsconfig.json	GREEN: My personal data form.	3 months ago
tsconfig.spec.json	Initial version	8 months ago

Name	Last commit	Last update
yarn.lock	Data portability control first version	8 months ago

README.md

EasyPims UI

This project was generated with Angular version 12.1.2.

Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The app will automatically reload if you change any of the source files.

Code scaffolding

Run `ng generate component component-name` to generate a new component. You can also use `ng generate directive|pipe|service|class|guard|interface|enum|module`.

Build

Run `ng build` to build the project. The build artifacts will be stored in the `dist/` directory.

Translations

In order to add translations, install this i18n helper library.

To generate translations, add "i18n" within HTML tags or use the $localize'Stuff to translate' within the component.ts. Then run this command to generate the new master translation file:

```
ng xi18n --output-path i18n
```

And then, to update the language files run:

```
xliffmerge --profile .\translation_conf.json
```

Running unit tests

Run `ng test` to execute the unit tests via Karma.

Running end-to-end tests

Run `ng e2e` to execute the end-to-end tests via a platform of your choice. To use this command, you need to first add a package that implements end-to-end testing capabilities.

Further help

To get more help on the Angular CLI use `ng help` or go check out the Angular CLI Overview and Command Reference page.

Personal Data Avatar code page

Changes to free tier open source projects

Before July 1, 2022, all free tier public open source projects must enroll in the GitLab for Open Source Program to continue to receive GitLab Ultimate benefits.

For more information, see the FAQ.

D Data Marketplace
Project ID: 33055911

restart script added
danielfx90 authored 1 month ago

Name	Last commit	Last update
.vscode	Initial commit	4 months ago
public	New UI: left menu and dashboard	3 months ago
src	Budget spent added to table	1 month ago
.env.development	Configs updated	1 month ago
.env.production	Config fixed	1 month ago
.eslintignore	Initial commit	4 months ago
.eslintrc.js	Initial commit	4 months ago
.gitignore	Initial commit	4 months ago
.yarnrc	Initial commit	4 months ago
Dockerfile	App dockerized	3 months ago
craco.config.js	Initial commit	4 months ago
docker-compose.yml	Configs updated	1 month ago
docker-entrypoint.sh	App dockerized	3 months ago
package.json	Data Offers module done	2 months ago
restart	restart script added	1 month ago
tailwind.config.js	New UI: left menu and dashboard	3 months ago
tsconfig.json	Initial commit	4 months ago
yarn.lock	Data Offers module done	2 months ago

Task Manager code page

Changes to free tier open source projects

Before July 1, 2022, all free tier public open source projects must enroll in the GitLab for Open Source Program to continue to receive GitLab Ultimate benefits.

For more information, see the FAQ.

T **Task Manager**
Project ID: 33713036

Sweepstake last winner deletion added to back-office frontend
danielfx90 authored 3 days ago

Name	Last commit	Last update
backend	Sweepstake last winners deletion and use...	3 days ago
frontend	Sweepstake last winner deletion added to...	3 days ago
.DS_Store	Back-office created	3 weeks ago
LICENSE	Task Manager v1	3 months ago
README.md	Task Manager v1	3 months ago
docker-compose.yml	Back-office created	3 weeks ago
docker-entrypoint.sh	Task Manager v1	3 months ago
restart	restart script added	1 month ago

README.md

Task Manager (TM)

Introduction

TODO

Installation

The documentation and the following instructions refer to a Linux environment, running with **Docker Engine v20.10.x** and **Docker Compose: v1.27.x**. The TM project has been cloned from this GitLab repository.

Follow accurately the next steps to quickly set-up the TM backbone on your server. All relevant steps are designed for a Linux machine, perform the equivalent procedure with other environments.

1. Prepare the environment:

```
> sudo apt-get update
> sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-
```

```
> echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubunt
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
> sudo apt-get update
> sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. Import the project from the GIT repository:

```
> git clone https://gitlab.com/pimcity/wp5/task-manager.git
```

Usage

Execution

To run the TM service, the following command should be executed at the root of the repository:

```
> docker-compose up
```

Configuration

The TM can be configured in three similar ways:

1. Using environment variables.
2. Defining those environment variables in a file called ".env" at the root directory of the folder with the PDK deployed, declaring them in the same way a variable is defined in the UNIX shell. You can check this example.
3. Modifying docker-compose.yml file.

Usage Examples

You can check here the Swagger OpenAPI definition to see how the API is defined and used. Examples are provided as well.

Authentication

KeyCloak service is used to carry out the authentication process. The user will not have to log in directly to the TM, but provide a JWT obtained from the authentication service or an EasyPIMS component, such as the PDA.

License

The TM is distributed under AGPL-3.0-only, see the LICENSE file.