
# “Building the Next Generation Personal Data Platforms”
# G.A. n. 871370

**DELIVERABLE D5.1**
**Design 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:** 10/06/2021
Deliverable due date: 31/05/2021

## Document Information

**Document Name: Deliverable 5.1 - Design 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 | 31/05/2021 |
| **Author** | Evangelos Kotsifakos | LSTECH | 31/05/2021 |
| **Author** | Xavi Olivares | LSTECH | 31/05/2021 |
| **Author** | Ruben Cueva Rumin | UC3M | 31/05/2021 |
| **Author** | Martino Trevisan | POLITO | 31/05/2021 |
| **Author** | Alvaro Garcia-Recuero | IMDEA | 31/05/2021 |
| **Author** | Santiago Azcoitia | UC3M | 31/05/2021 |
| **Author** | Kleomenis Katevas | TID | 31/05/2021 |
| **Author** | Daniel Fernandez | Wibson | 31/05/2021 |
| **Author** | Rodrigo Irarrazaval | Wibson | 31/05/2021 |
| **Author** | Guglielmo Bondioni | FW | 31/05/2021 |
| **Author** | Miguel Herranz | IAB | 31/05/2021 |
| **WP Leader** | Stefano Traverso | ERMES | 31/05/2021 |
| **Coordinator** | Marco Mellia | POLITO | 31/05/2021 |

## Document history

| Revision | Date | Modification |
|---|---|---|
| **Version 1** | 31/05/2021 | V1 |
| **Version 2** | **22/10/2021** | **Addressed reviewer suggestions. In details**<br>**1. We modified the presentation of results obtained from questionare about Transparency Tags design**<br>**2. We included in the appendix the raw responses of questionare about Transparency Tags design**<br>**3. We clarified that Data Buyers shall declare the data retention period/data deletion date, any time a new data transaction is created in the Data Marketplace.** |

## 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 |
| DA | Data Aggregation |
| DPC | Data Portability and Control |
| DKW | Data Knowledge Extraction |

## Table of Contents

## Disclaimer

# 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 first of WP5, aims at providing the first 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 describe the high-level design of WP5 components, how the modules satisfy the requirements defined in Deliverable 1.1 and motivate the choices from both technical and integration perspectives. As such, we defer the reader to Deliverable 1.1 for the full requirements for all EasyPIMS components.

Apart from defining the overall integration and system deployment, 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.

Design proposals represent the first fundamental step towards the integration process of WP5. Nevertheless, we expect such proposals to evolve during the development of the project based on the feedback we will obtain from developers, users and data buyers, and depending on possible modification of the technical requirements.

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 8 describes the design choices made for Transparency Tags. Finally, Section 9 describes the structure of the Data Marketplace, and Section 10 concludes the document.

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

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

# 2 Deliverable objectives

This deliverable is the first of WP5 whose goals are:

1. Designing and implementing the technical platform which will run EasyPIMS and make it easy for components to be integrated in the systems.
2. Designing, documenting and implementing Open APIs and SDKs to ease the integration among modules, developed within and outside PIMCity.
3. Designing and developing an orchestrator that allows the execution and interoperability of the building blocks in some existing computing platform.
4. Design and implement the components at the basis of EasyPIMS, I.e., Transparency Tags, Personal Data Avatar, and the Marketplace.

Specifically, this deliverable provides the first complete 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 **Dashboard**, the **Personal Data Avatar,** the **Transparency Tags** and the **Data Marketplace** with a special attention to their relations with the PDK components.

In this document we provide a first complete design for each of such component, providing insights on technical requirements that have been considered and preliminary tests conducted to validate design hypothesis and proposals.

# 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, so as 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 Dashboard, 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. 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.

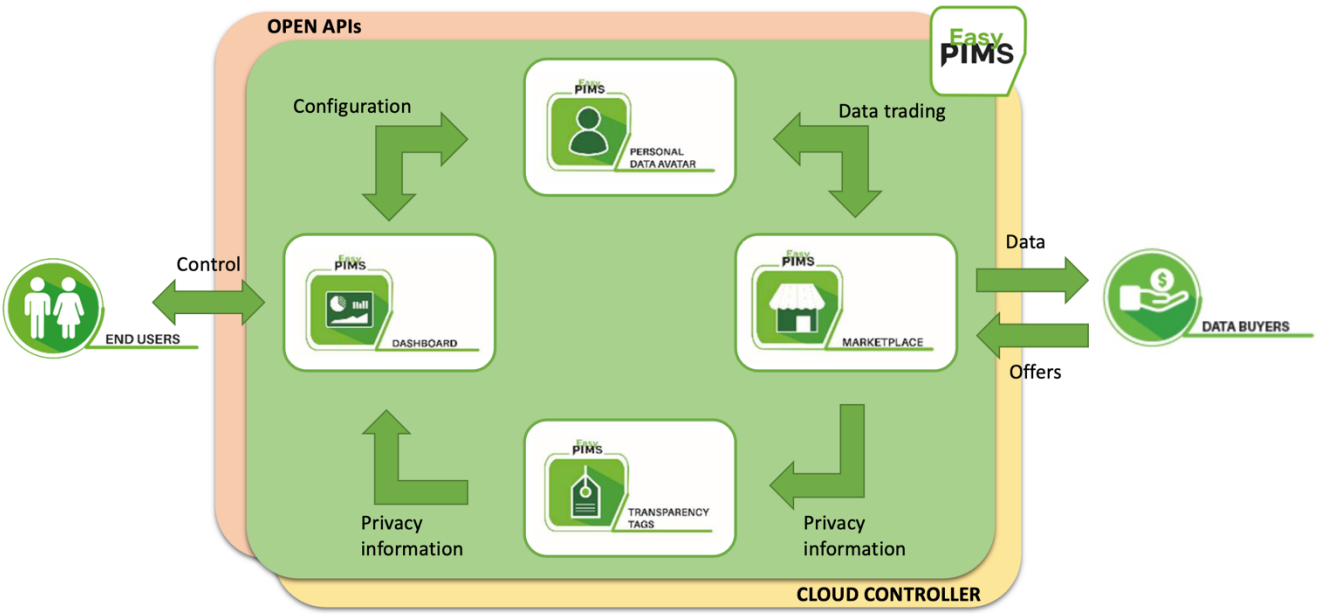



*Figure 1 - High-level overview of EasyPIMS platform*

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.

- **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 8).
- **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 8.
- **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.

# 4 Physical platform, environments and deployment

## 4.1 Infrastructure overview

In this section we describe the development environment of the cloud platform that we use for developing the PDK components of PIMCITY.

The development platform for PIMCITY should 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. To allow rapid yet safe evolution of systems, we will provide an automated build, test and deploy process. DevOps tools will allow the developers to build and test their applications directly on the cloud.
- **Security:** The environment should be secure with restricted access and no connection to the external world. If it is necessary for applications to connect to external applications, this will be done through secure procedures, such as proper firewall exceptions.
- **Extensibility:** The environment should be easily extensible. New applications should be easily integrated.
- **Interoperability:** The environment should allow connections to external services when needed.
- **Administration and manageability:** The environment should allow easy administration and management. The infrastructure must be a dynamic, powerful, and robust centralized environment, able to provide PIMCITY members a system to access and share their different components without affecting each other, within their own workspace. To this end, each technical partner will have at least one user to manage their workspace, with the proper access rights.

For the development of the PIMCITY platform we rely on Dockerized environment for most of the components. This will allow ease of implementation, extensibility, portability, and security.

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[3] 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.

Container 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 its environment and ensure that it works uniformly despite differences for instance between development and staging.

[3] 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.

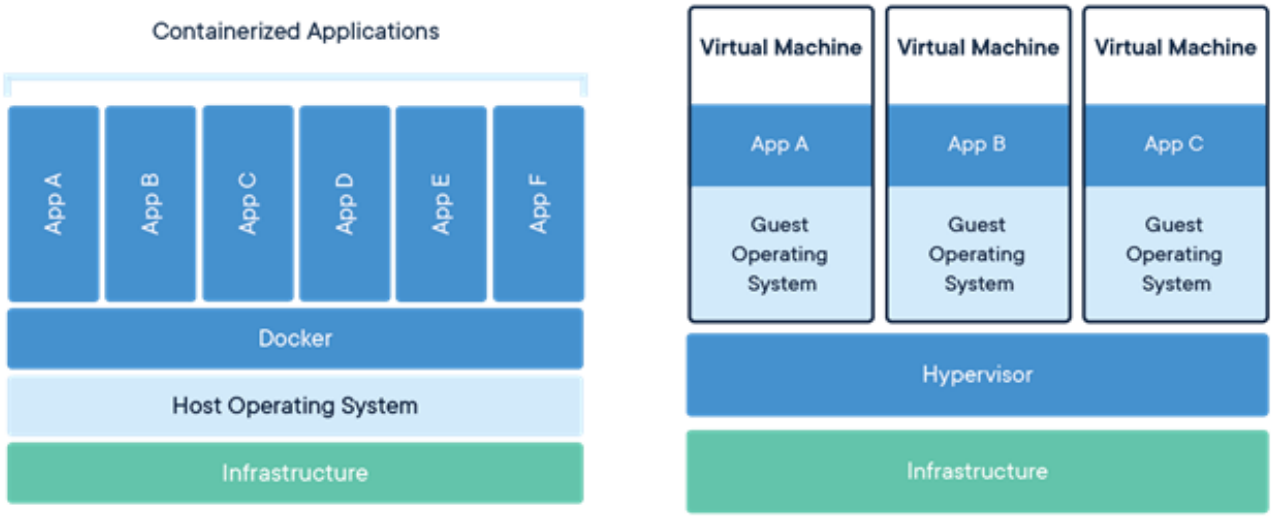



*Figure 2 - Virtualization architectures – courtesy of Docker*

Each PIMCITY partner will create 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 will consider running it natively on the infrastructure machines. The following figure illustrates this setup.

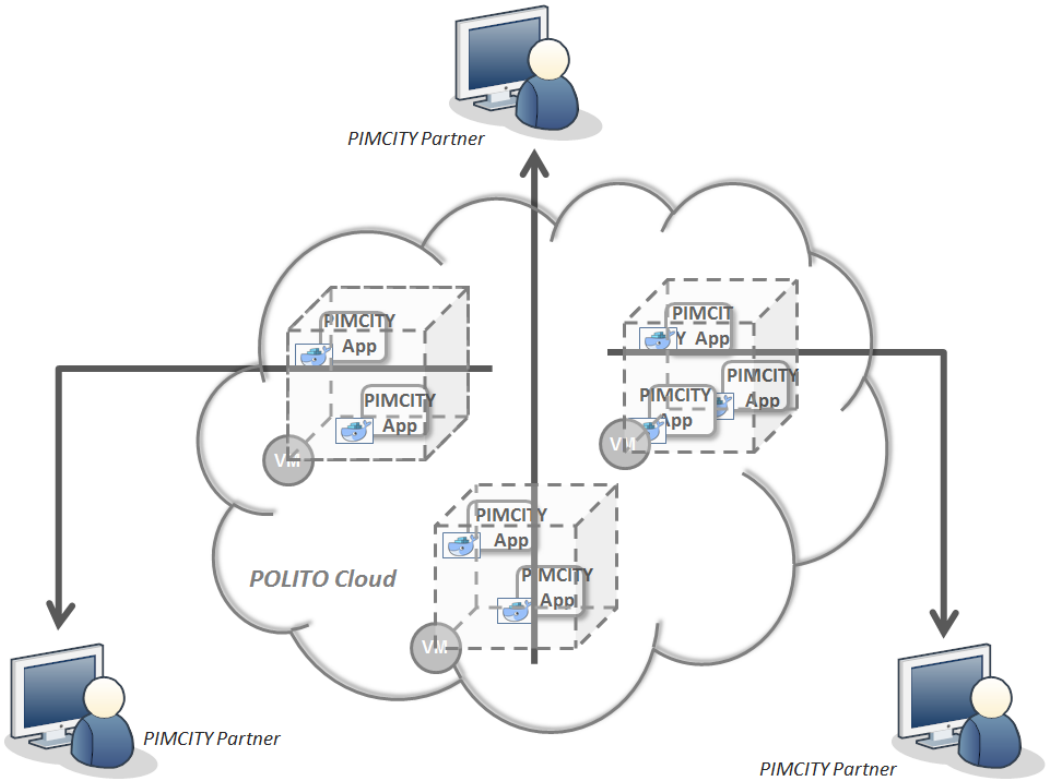



*Figure 3 - PIMCity physical and virtual infrastructure setup*

According to the size progression of the project, new tools will be used in order to have better control of the infrastructure. We will use the Ansible automation tool to manage the infrastructure as the number of Virtual Machines keep growing.

### 4.1.1 Security and access control

The environment is 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, previously the developers have 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)

On a daily basis, the infrastructure is monitored to identify possible inconsistencies, conflicts or security issues based on system, application and access logs.

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

Backups are taken periodically, once every week in the beginning of the project. As the project advances more frequent backups will be implemented. POLITO and LST's teams will be responsible for restoring the environment in case of a failure within 48 hours.

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 are needed between the partners to ensure secure access to the developed software and to disclosure of information or data, these will be drafted and signed according to the counselling of the project's legal partners, and we will implement the appropriate measures.

The technical partners may also provide 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 allows 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 Resource requirements

Initially, 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[4] 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 Environment realization

The 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[5] 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[6]. The technical partners have to provide a 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[7] Dashboard, which is used to administrate and control the VMs. Below, it is attached a screenshot of the dashboard.

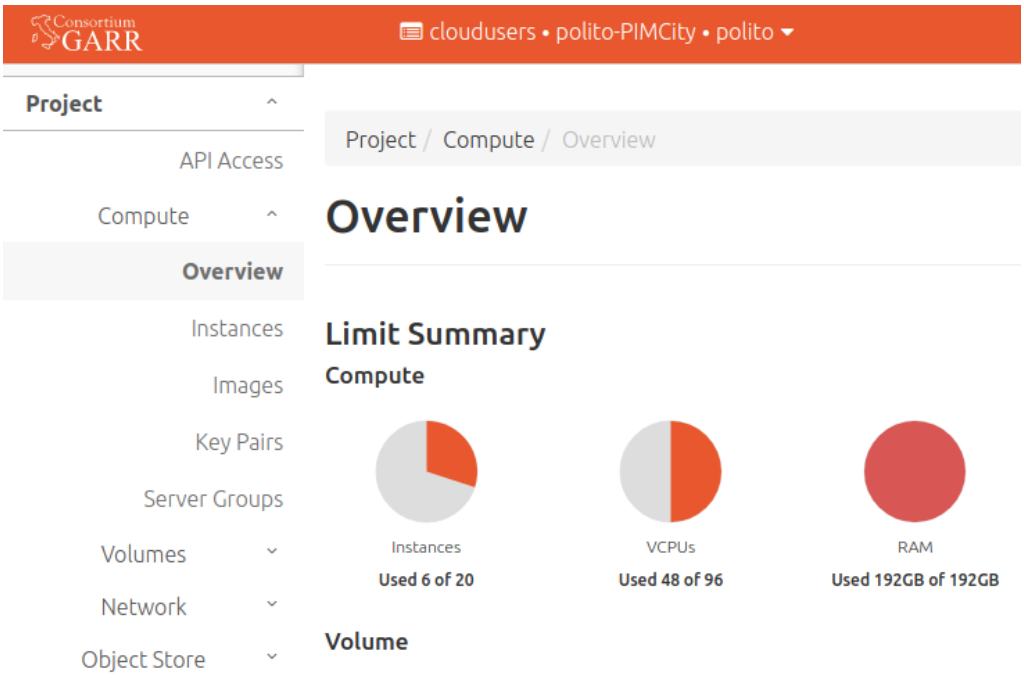



*Figure 4 - OpenStack dashboard at POLITO's premises*

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

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

[6] 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.

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

### 4.1.4 Deployment

**User management and roles:** 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 Campus-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 via SSH to their machines.

**Gitlab project access and users:** Our setup is the following**:**

- The Group PIMCITY Platform has been created at https://gitlab.com/pimcity
- Each project member will be invited, and they will create their code repository for version control.

**Suggestion for deployment:** The easiest way would be to have all the necessary files in the Git repository (including a Docker file and Docker-compose), and to pull it in the VM when there are changes. If Docker Compose is used, partners can then easily run "Docker-compose restart" to apply the changes.

**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 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.

# 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 actually 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)[8] 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[9], 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[10], Okta[11], Amazon Cognito[12]. 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[13], 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
- Kerberos broker
- Multitenancy
- Social login

The Cloud Controller will be built by FW and deployed within the EasyPIMS platform hosted by POLITO.

[8] https://www.oasis-open.org/committees/xacml/
[9] https://oauth.net/
[10] https://auth0.com/
[11] https://www.okta.com/
[12] https://aws.amazon.com/cognito/
[13] https://www.keycloak.org/

## 5.2 Authentication and authorization flows

In the following we report the authentication and authorization flows that component providers will adopt in order 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.

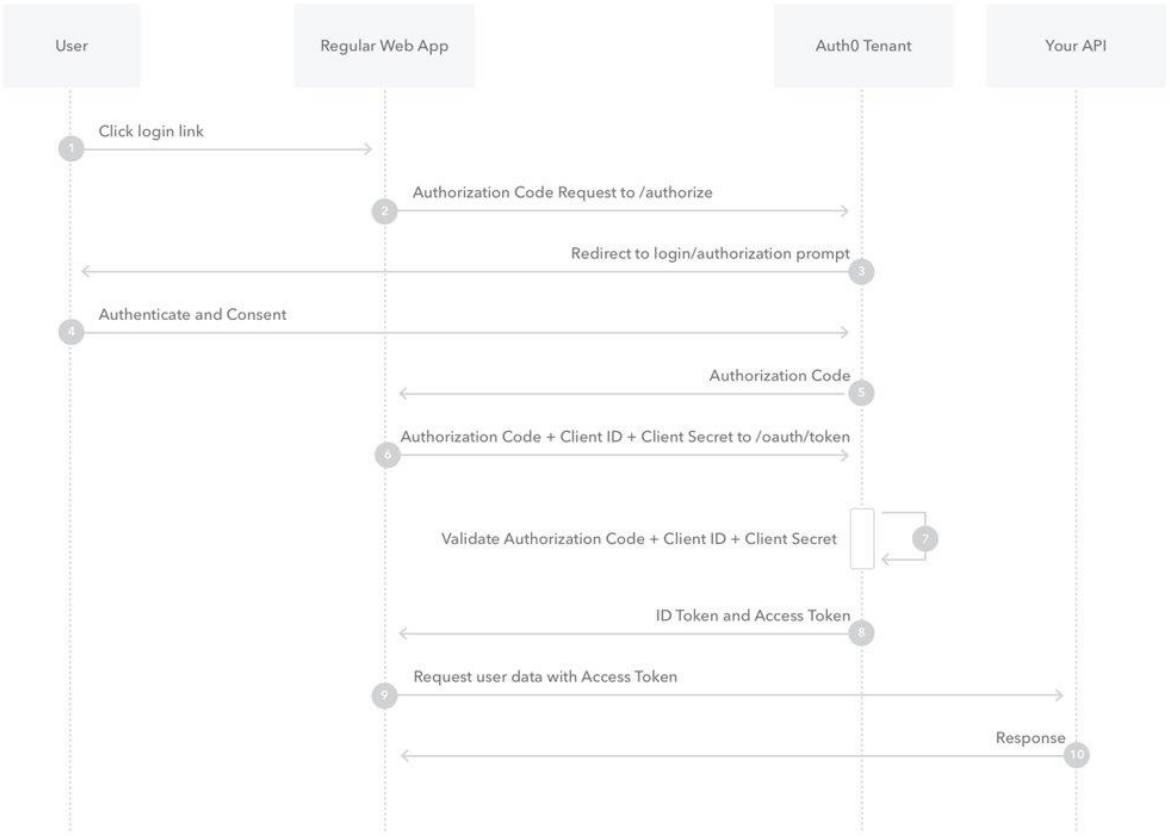



*Figure 5 - OAuth 2.0 Authorization Code flow – courtesy of Auth0*

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.

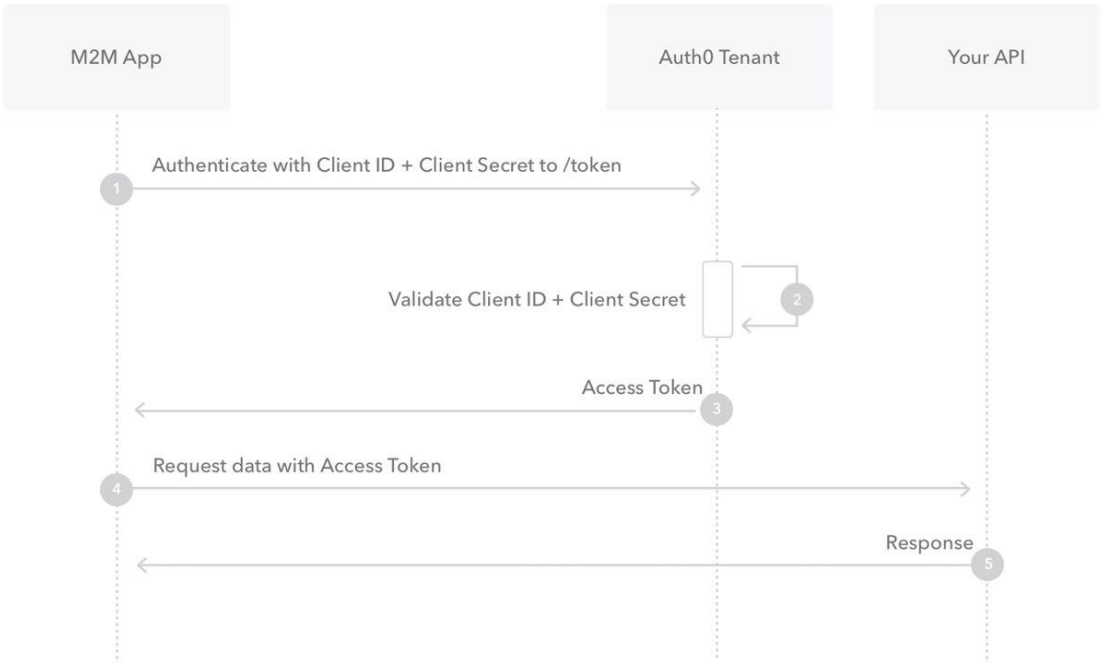



*Figure 6 - OAuth2 Client Credentials flow – courtesy of Auth0*

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].

## 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[14], 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

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

The easiest way to design and implement RESTful APIs based on OpenAPI specification is using the Swagger Framework[15]. 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[16] 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[17] 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[18]. 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

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[19]. The

[15] https://swagger.io/

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

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

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

[19] https://gitlab.com/pimcity/wp2/personal-consent-manager

Open API implementation of Personal-Consent Manager component is available at official PIMCity's Gitlab code repository, under WP5 project[20].
The Open API of the P-CM offer 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.

[20] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP2/personal-consent-manager.yml

Its primary objective is to give the users transparency and control over their data in a GDPR compliant way. That is, give them the possibility to decide which data can be uploaded and stored in the platform, as well as which (raw, extracted or aggregated) data can be shared with Data Buyers in exchange for value when the opportunity arises.

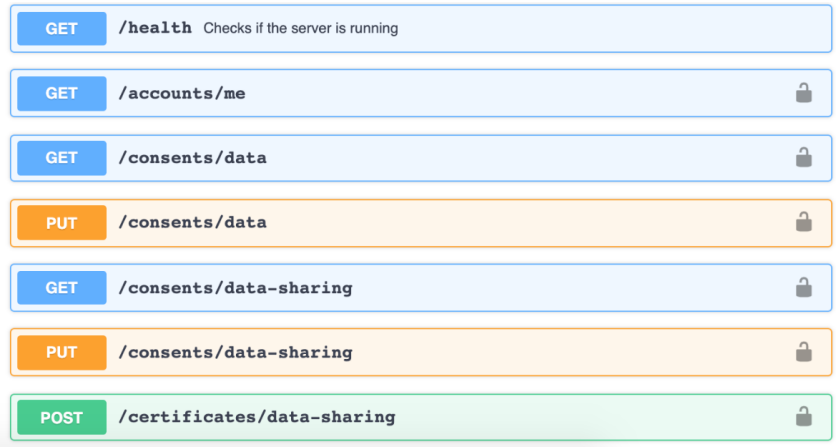



*Figure 7 - Open APIs for Personal Consent Manager*

### 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[21]. The Open API implementation of Personal Data Safe component is available at official PIMCity's Gitlab code repository, under WP5 project[22].

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.

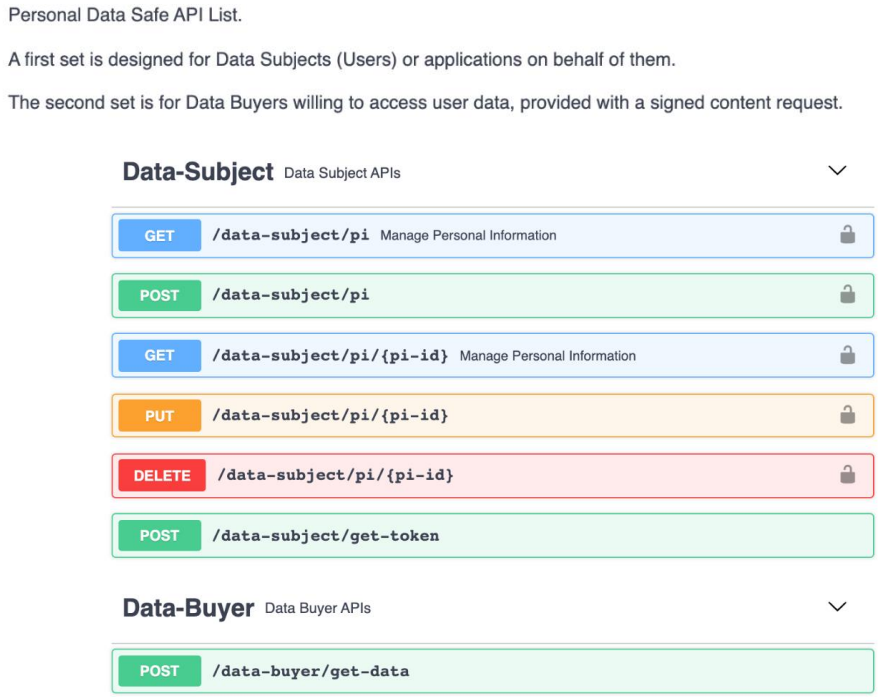



*Figure 8 - Open APIs for Personal Data Safe*

[21] https://gitlab.com/pimcity/wp2/personal-data-safe

[22] https://gitlab.com/pimcity/wp5/open-api/-/blob/master/WP2/personal-data-safe.yml

#### 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[23]. The Open API implementation of Personal-Privacy Preserving Analytics component is available at official PIMCity's Gitlab code repository, under WP5 project[24].

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.

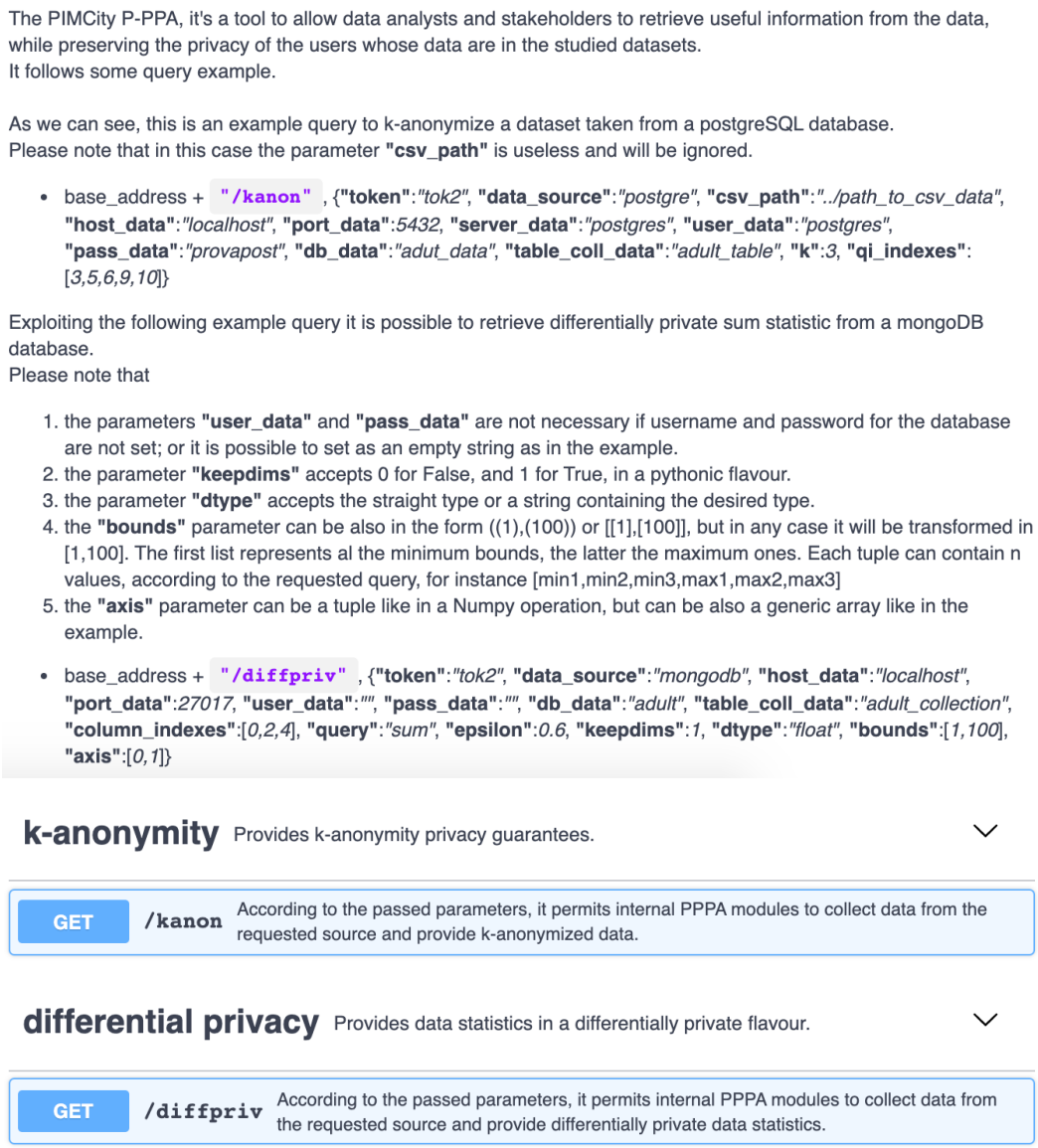



*Figure 9 - Open APIs for Personal Privacy Preserving Analytics*

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

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

#### 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[25].

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.

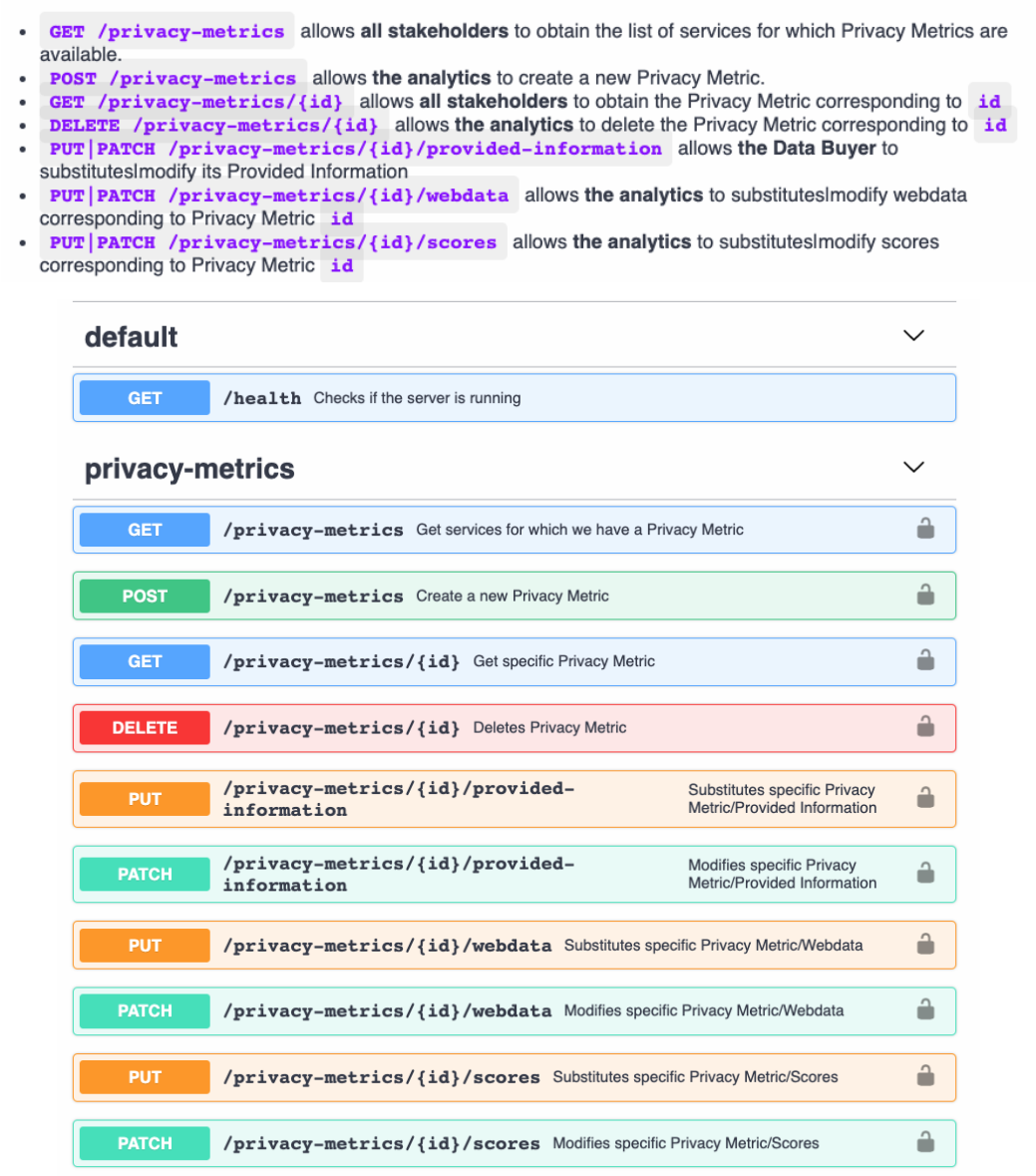



*Figure 10 - Open APIs for Privacy Metrics*

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

### 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 complete code of the module is available in the project Gitlab[26]. 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[27].

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

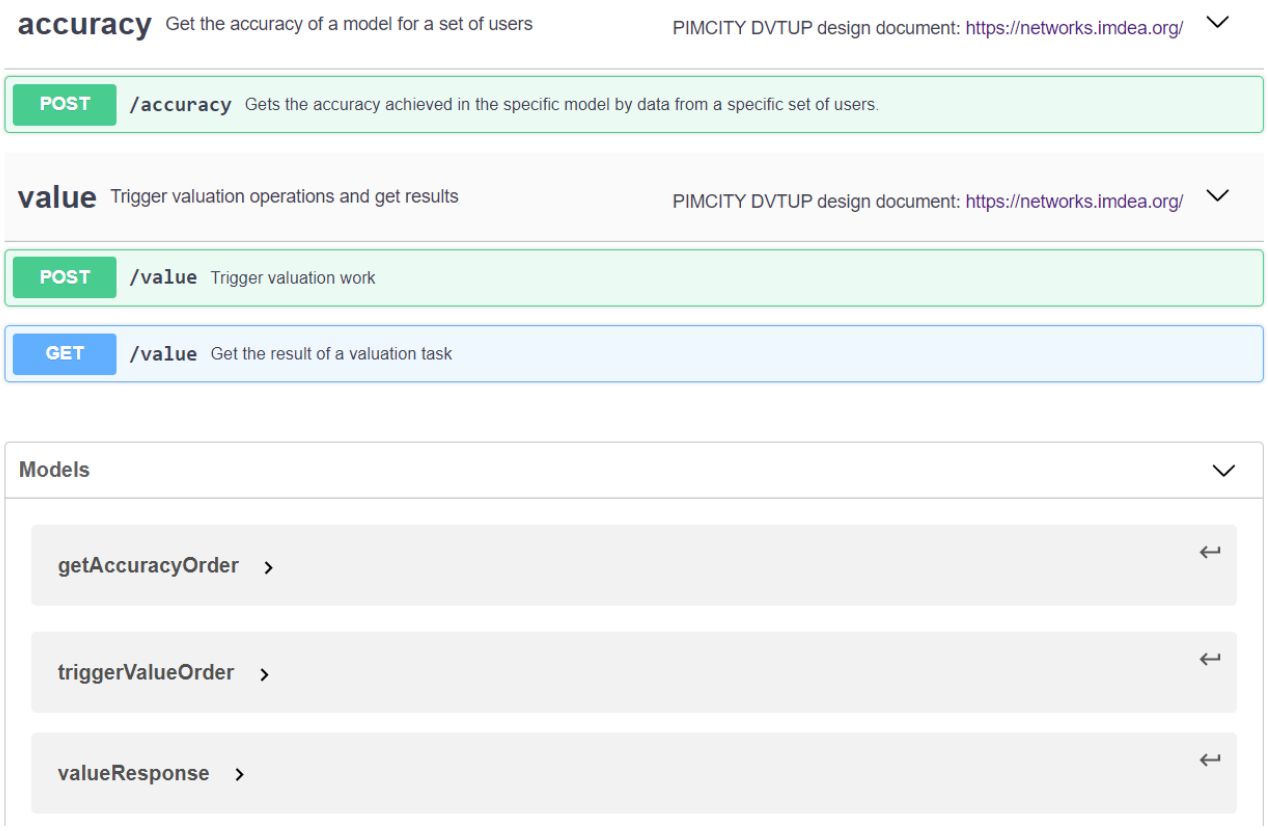



*Figure 11 - Design Open APIs for Data valuation tool user perspective*

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

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

#### 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[28]. 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[29]. Below you can find a report from the Module API draft documented by Swagger UI. As it 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.

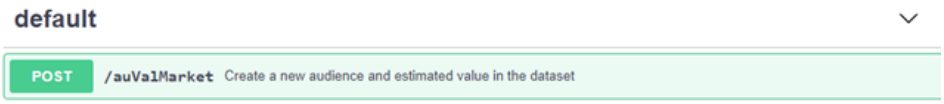



*Figure 12 - Design of Open APIs for Data valuation tool Market perspective*

#### 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.
- Making the component efficient, fast, and scalable to accommodate a huge number of transactions for a very large set of users and Data Buyers.

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

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

The complete code of the module is available in the project Gitlab[30]. The Open API documentation is available in the WP5 code repository of the project[31]. Below we report the module API draft documented by Swagger UI.

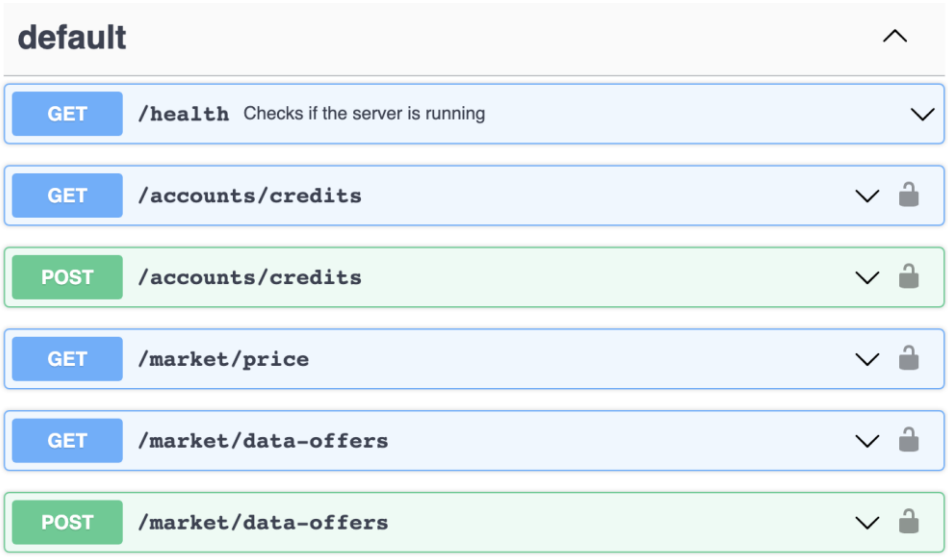



*Figure 13 - Design of Open APIs for Trading Engine*

### 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[32].

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

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

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

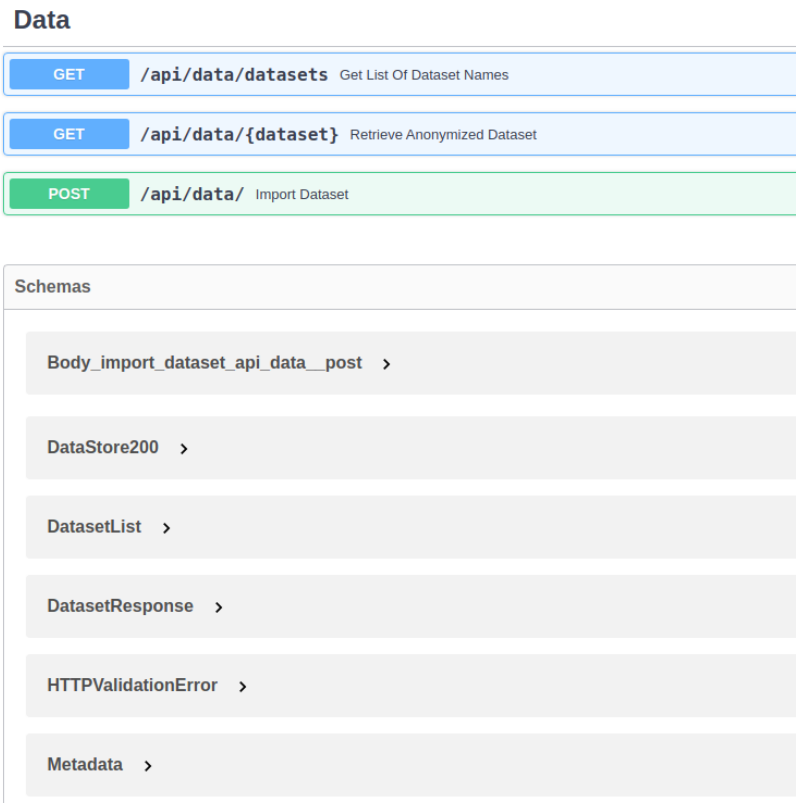



*Figure 14 - Design Open APIs for Data Aggregation tool*

#### 6.3.3.2 Data Portability and 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., user unavailability due to event attendance) 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[33].

In the following we report the design draft 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 D4.2 deliverable [10].

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

## default

| | | |
|---|---|---|
| GET | /health | Checks if the server is running |

## data-sources

| | | |
|---|---|---|
| GET | /datasources | Get a list of all available Data Sources. |
| POST | /datasources | Create a new Data Source |
| GET | /datasources/{id} | Get a specific Data Source |
| PUT | /datasources/{id} | Modifies an existing Data Source |
| PATCH | /datasources/{id} | Modifies a specific property of an existing Data Source |
| DELETE | /datasources/{id} | Deletes an existing Data Source |

## tranformations

| | | |
|---|---|---|
| GET | /transformations | Get a list of all available Data Transformations. |
| POST | /transformations | Create a new Data Transformation |
| GET | /transformations/{id} | Get a specific Data Transformation |
| PUT | /transformations/{id} | Modifies an existing Data Transformation |
| PATCH | /transformations/{id} | Modifies a specific property of an existing Data Transformation |
| DELETE | /transformations/{id} | Deletes an existing Data Transformation |

## exports

| | | |
|---|---|---|
| GET | /exports | Get a list of all available Data Exports. |
| POST | /exports | Create a new Data Export |
| GET | /exports/{id} | Get a specific Data Export |
| PUT | /exports/{id} | Modifies an existing Data Export |
| PATCH | /exports/{id} | Modifies a specific property of an existing Data Export |
| DELETE | /exports/{id} | Deletes an existing Data Export |

*Figure 15 - 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 X, out of which the most relevant for the Data Trading Engine (which interacts with the Data Provenance here) are the two we list below.

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

*Table 1- API endpoints for getting back Watermarked datasets*

In principle, these are the main endpoints for POST and GET as requests are coming from the DTE in WP5. 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 Privacy Metrics component is available at official PIMCity's Gitlab code repository under WP5[34].

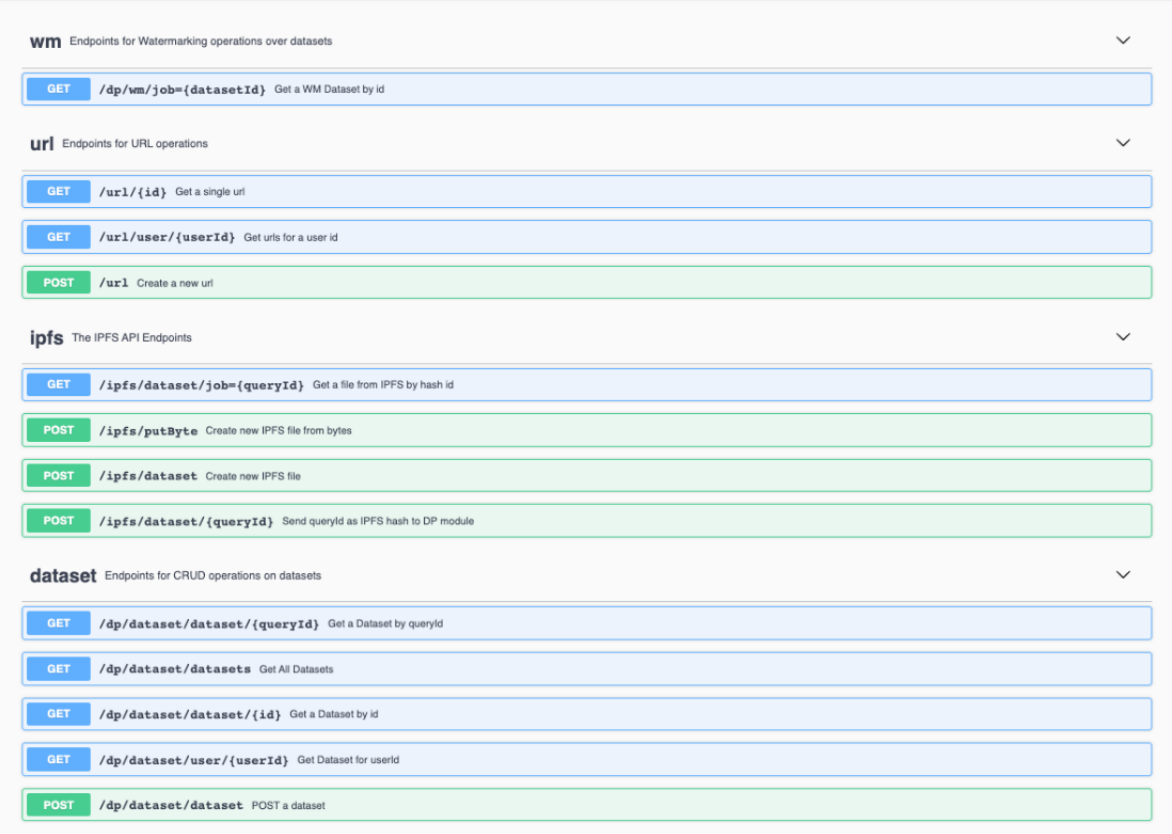



*Figure 16 – Design of the Open APIs for Data Provenance tool*

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

### 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).

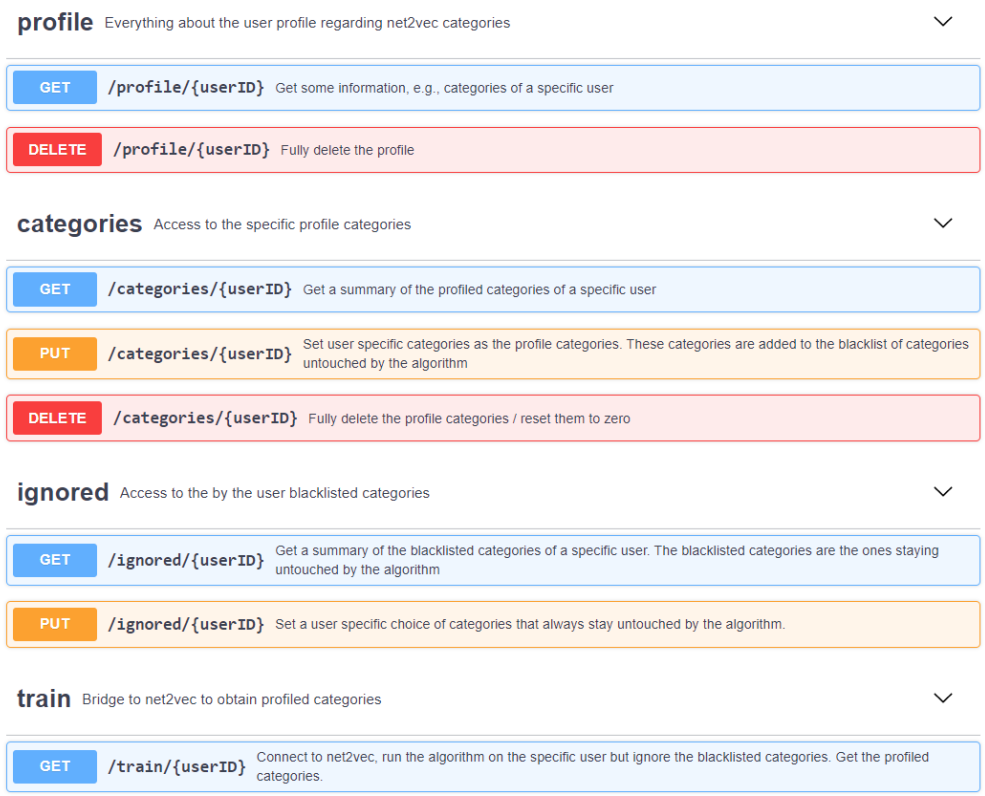



*Figure 17 - Design of Open APIs for User Profiling component*

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[35]. 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[36].

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

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

# 7 Design of the Personal Data Avatar (PDA)

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.

## 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).

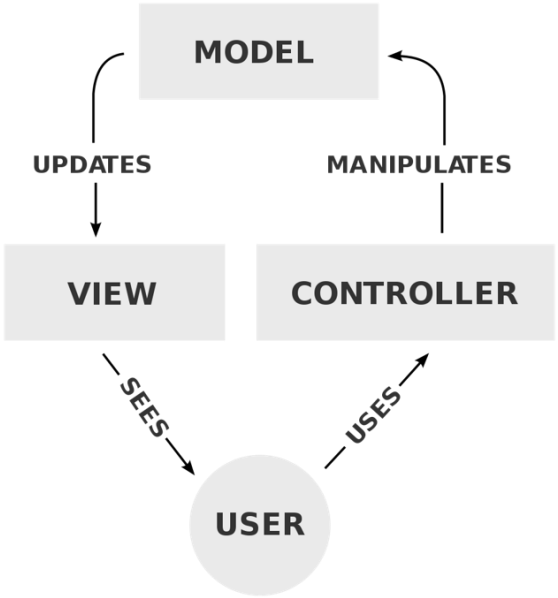



*Figure 18 - MVC design of PDA*

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:

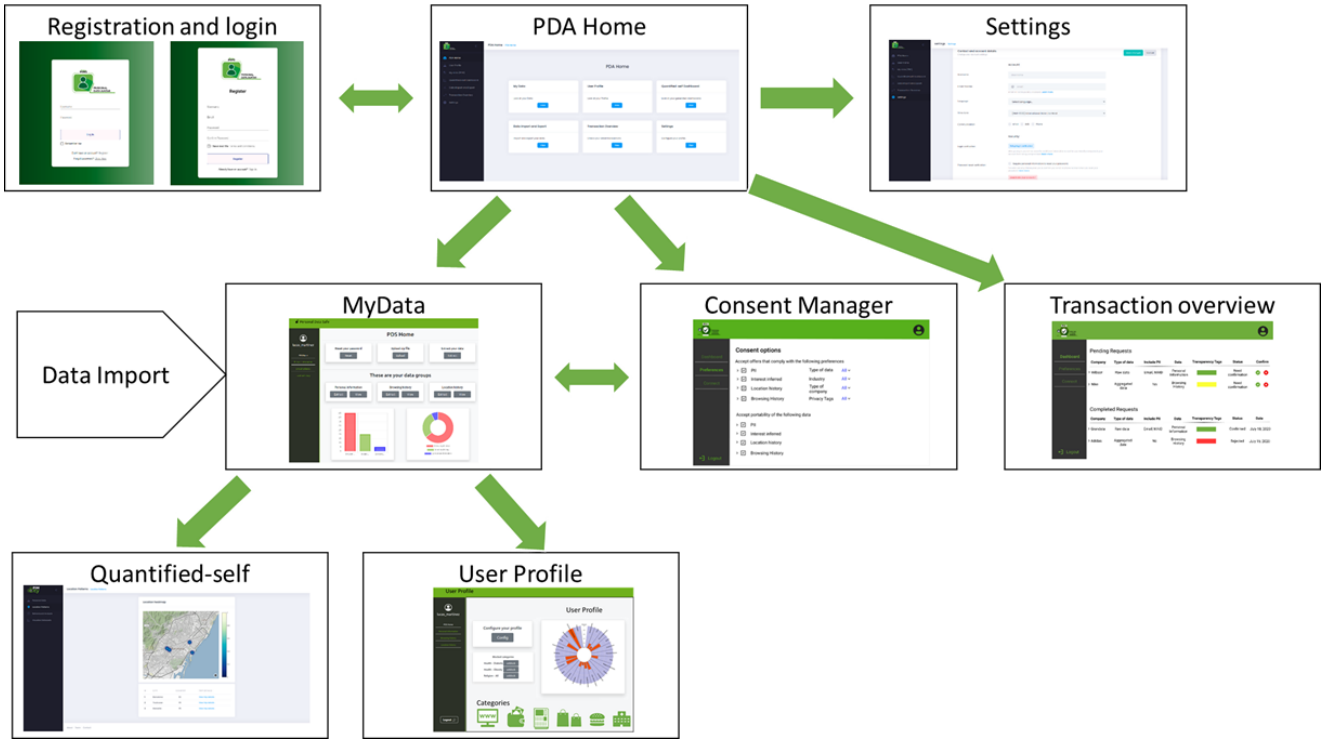



*Figure 19 - Webmap with navigation options*

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 Transcation 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.

### 7.2.1 Registration and Login

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

#### Functionalities

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

### Mockup

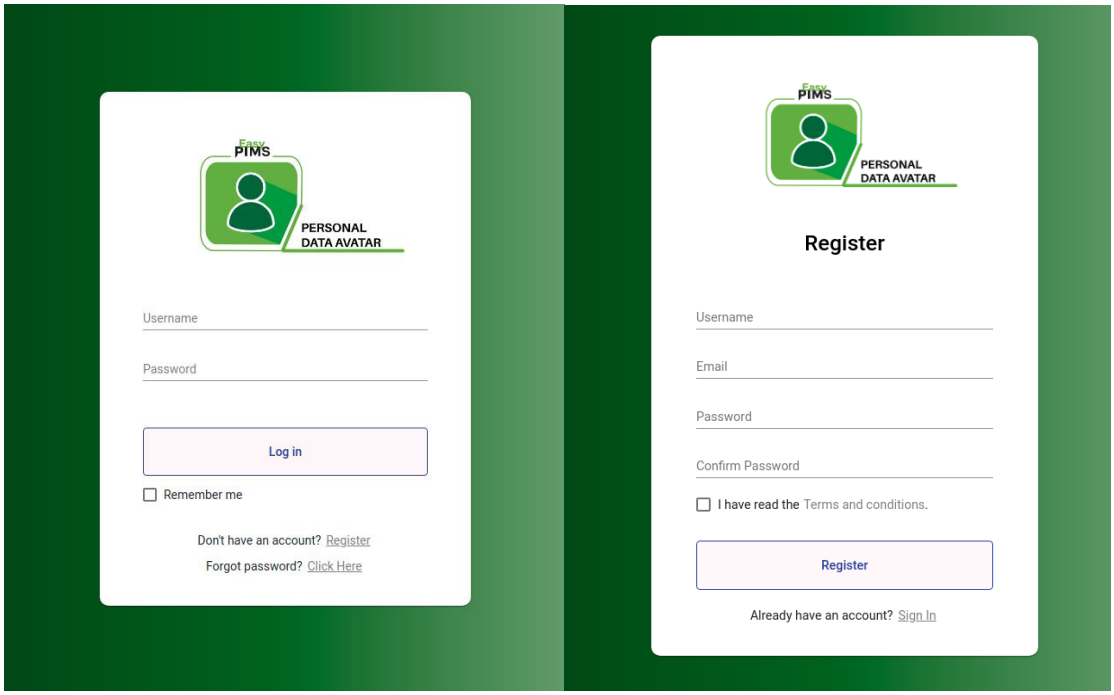



*Figure 20 - PDA's Login and Registration screen mockups*

### 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).

## 7.2.2 Main Screen

In the main screen, after user logs in, there will be a list of the options the user has. Selecting one of these the user will be directed to the specific screen of each tool. These options are:

- Transaction overview
- myData (PDS)
- User Profile
- Quantified-self dashboard
- Data import and export
- Settings

The details of each one of the tools are described in the following sections.

### Mockup

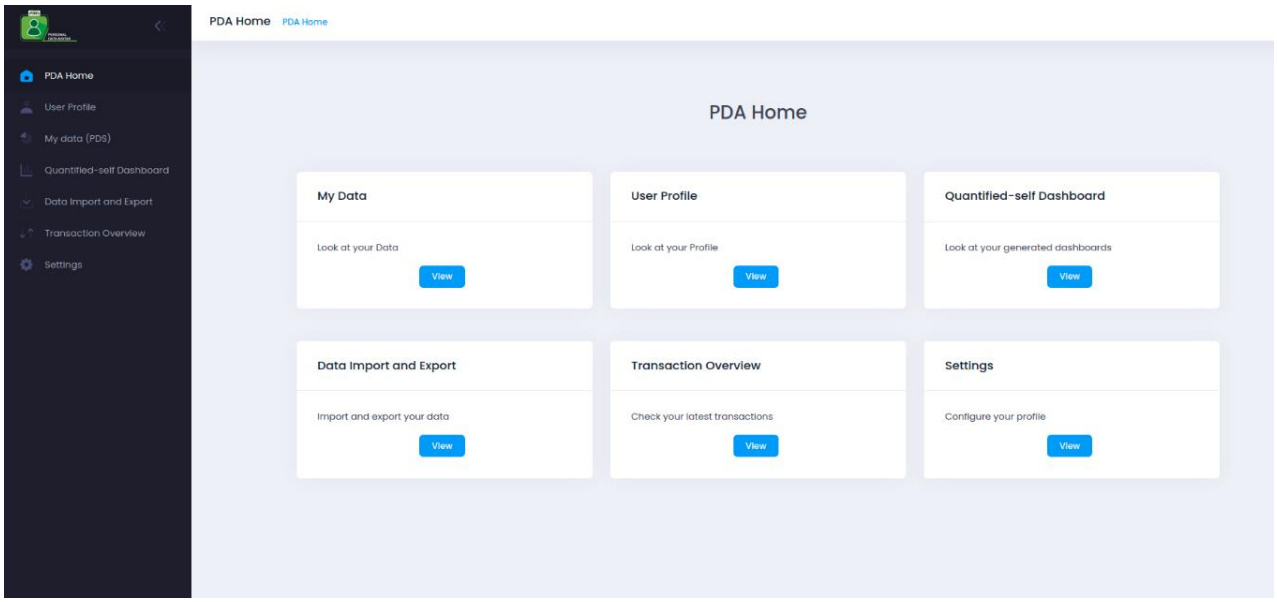



*Figure 21 - PDA's Main screen mockup*

### Interactions

The main screen is used as an entry point/ menu of the PDA, so, the interactions of this screen are those to the items of this main menu.

## 7.2.3 Transaction overview

The Transaction Overview presents a list of all past Data Transactions done by the user or in which he or she participated. It works as a log of all data shared and all the companies' information that had access to that data and in which form. In the same list, the user can see also the credits earned for the transactions done.

### Functionalities

This module lists all Data Transactions performed by or in which the user participated. It also shows all the details of all Data Transactions listed: data shared, reward earned, information about the company or companies that had access to the data shared, dates, and so on.

With these functionalities, the user can have and read the log of all the transactions involving his or her personal data.

### Mockup

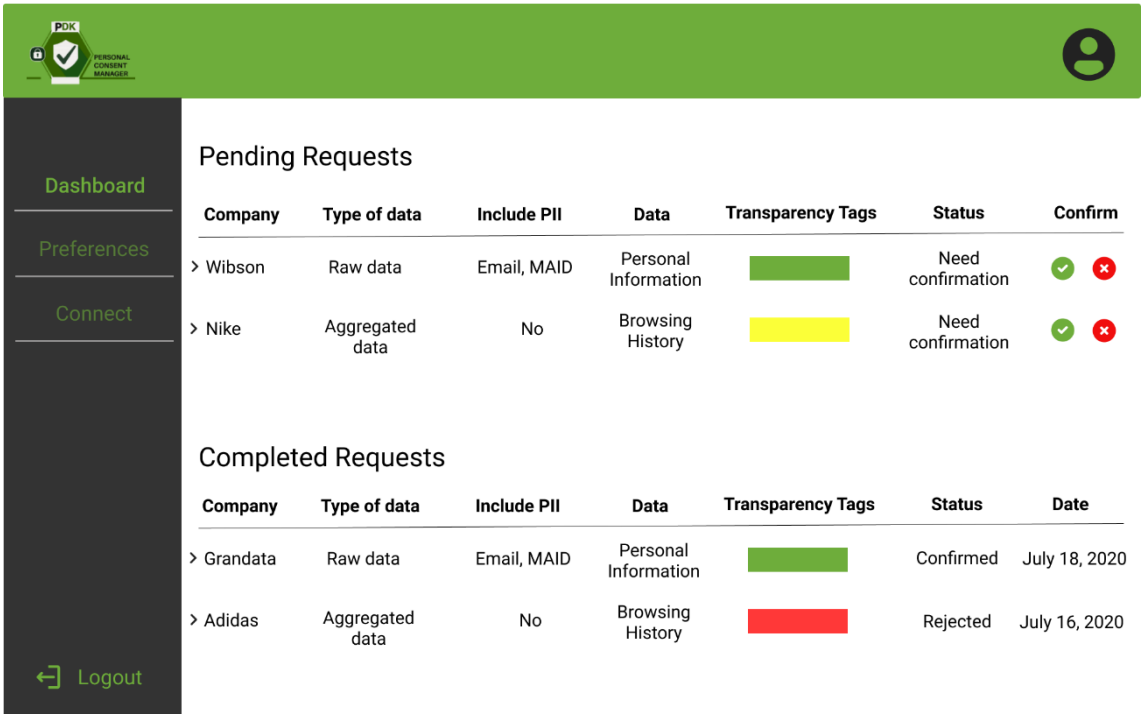



*Figure 22 - PDA's Transactions Overview mockup*

### 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 user logged in is used.

## 7.2.4 myData

The myData dashboard is a simple web interface for presenting the user's data stored in the PIMCity Personal-Data Safe. It presents a summary of the Personal Data Safe information in a simple and user-friendly fashion. Its goal is to provide the user a simple and intuitive way to visualize and control her personal information.

### Functionalities

On the home page, the user visualizes a summary of her personal information stored on the system and has the possibility to browse to the myData subpages to access specific content. Each group of personal information is presented on a dedicated subpage. Accessing the per-group subpage, the user can visualize and manipulate her personal information. Different types of information are presented in a different way. For example, geographical data are presented on a map, while the browsing history is arranged on a paginated table.

### Mockup

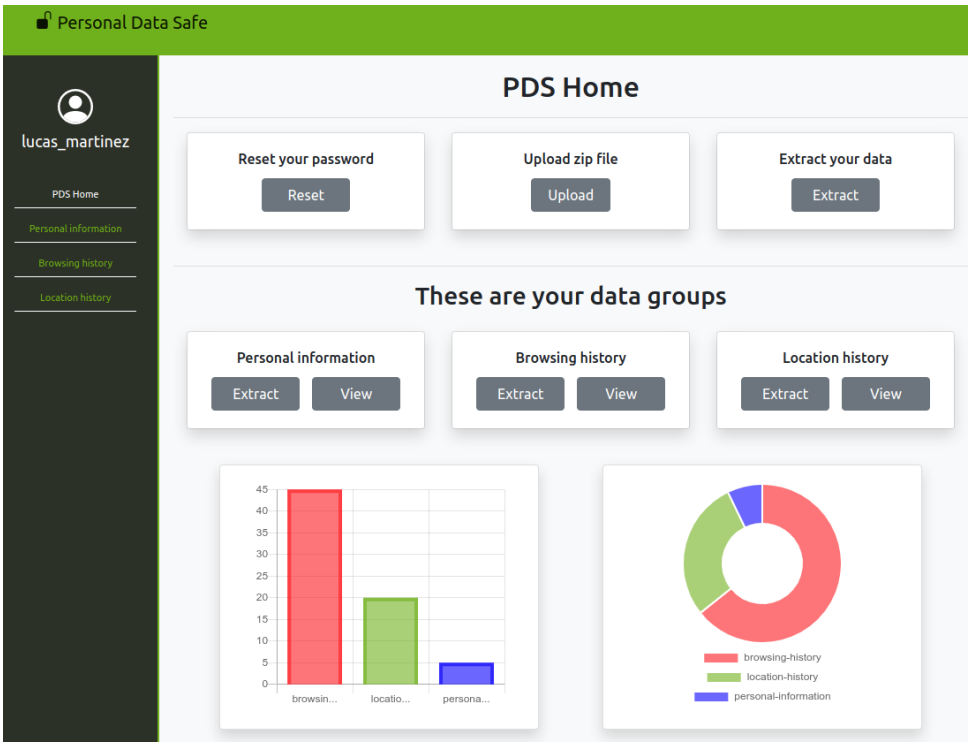



*Figure 23 - PDA's myData mockup*

### Interactions

The myData interface acts as a visual front end of the PIMCity Personal-Data Safe (P-DS). It communicates with the P-DS using JSON web APIs using a frontend-backend logic. It is also integrated with the PIMCity cloud controller offering authentication and authorization services.

## 7.2.5 User Profile

The User Profile section shows the user the profile automatically created by the system for the user. In order to obtain that profile, the EasyPIMS platform should have access to the user browsing history. The profile created is a representation of the interest of the user in the different categories defined in the IAB taxonomy[28].

### Functionalities

This screen allows to visualize the profile automatically assigned by the system to the user based on her browsing history. It also allows to modify the profile to include those interest the user considers to be important. Finally, it let the user choose the profile parts for sharing. This will allow the user to share only the parts of her profile she wants to be shared (i.e., she can hide his “health related” interests).

### Mockup

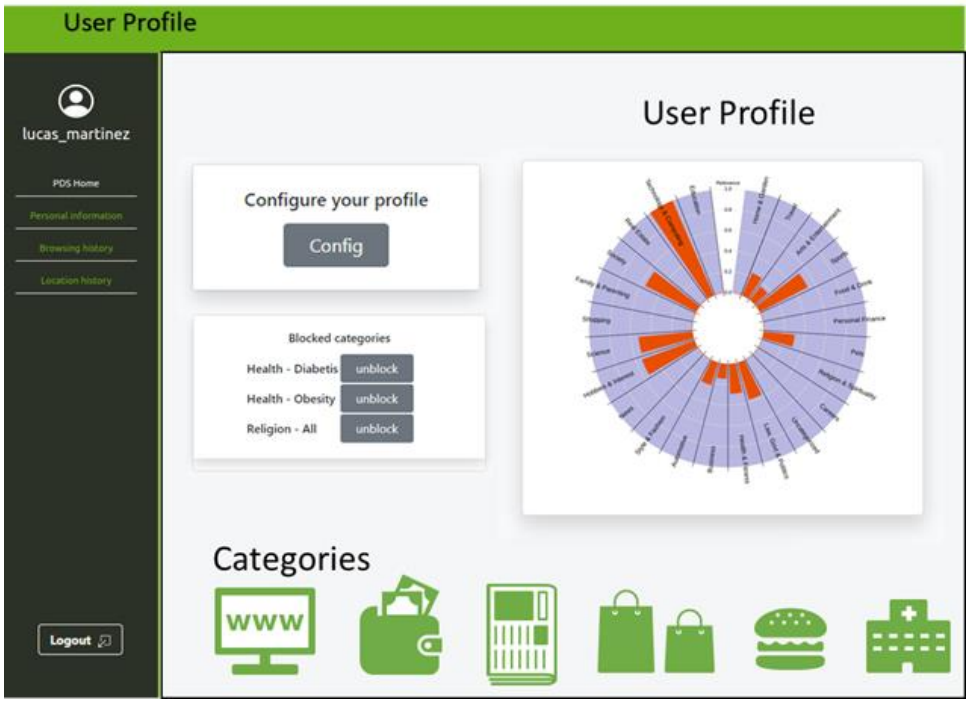



*Figure 24 - PDA's User Profile mockup*

### Interactions

The User Profile web view connects directly with the user profiling module of the PDK developed in WP4. Moreover, it connects with the Data Trading engine developed in WP3 to store the “sharing decisions” of the users.

## 7.2.6 Quantified-self dashboard

The purpose of the Quantified-self dashboard is to present to the user an analysis of their behavior based on the data available, especially related their location. Through comparing their data with other users’ aggregated data, the user will be able to better understand not only their behavior in relation to other similar users, but also to see the benefits of sharing data in the EasyPIMS setting and the value of her data.

### Functionalities

The Quantified-Self is designed into several screens that showcase its functionality in PIMCity. Depending on the available dataset the user can have various insights of the user data. We aim to use user mobility data from telco provider in order to exploit and present mainly location patterns and behavioral patterns of the user, also in comparison with other users with similar profile (based on age, gender, location, etc.). Following are the main operations that we envisage for this module, along with the related initial mock-up screens.

The user, after logging-in will be able to access 3 main operations that are listed on the left of the screen. The Location patterns, the behavioral patterns and the dataset or taxonomy patterns.

## Mockup

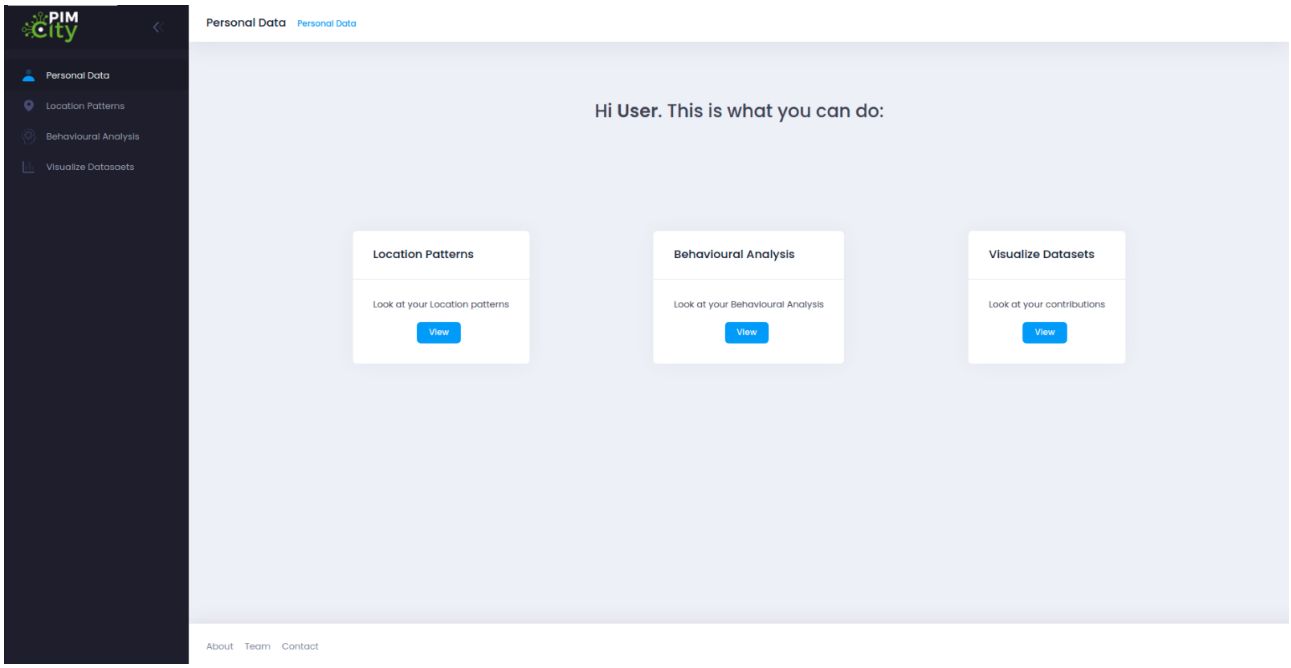



*Figure 25 - Mockup of PDA's Quantifield-self Dashboard main screen*

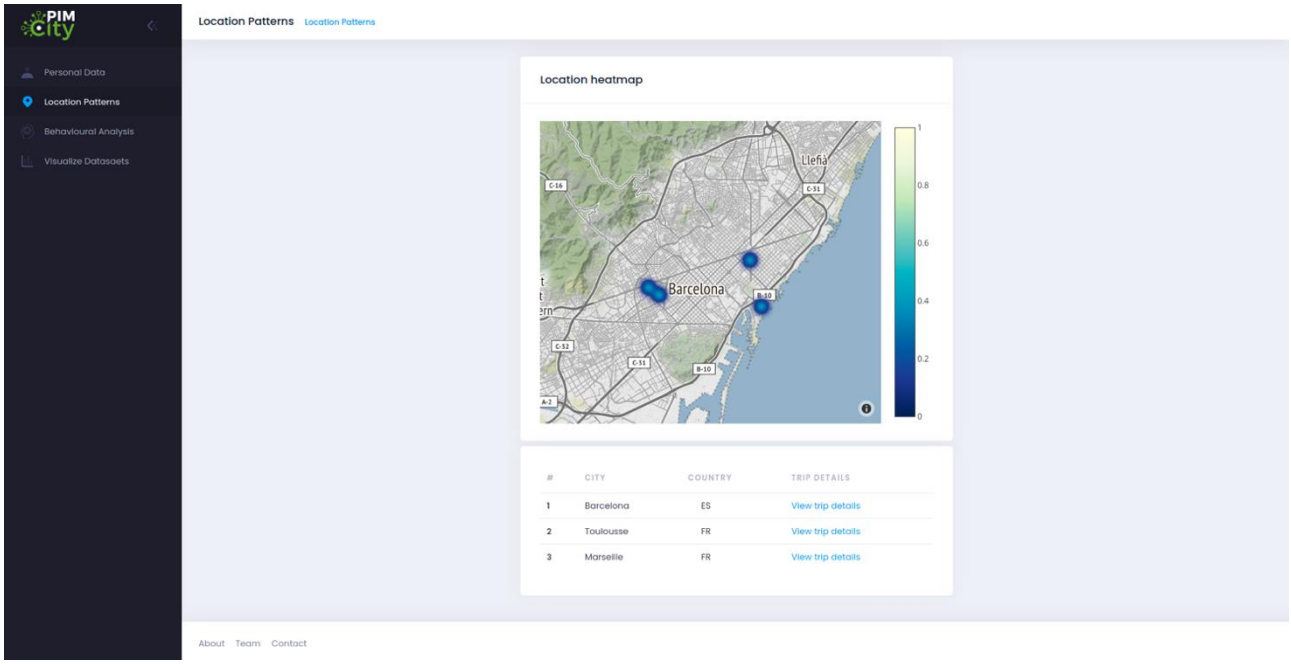



*Figure 26 - Mockup of PDA's Quantifield-self Dashboard locations analysis*

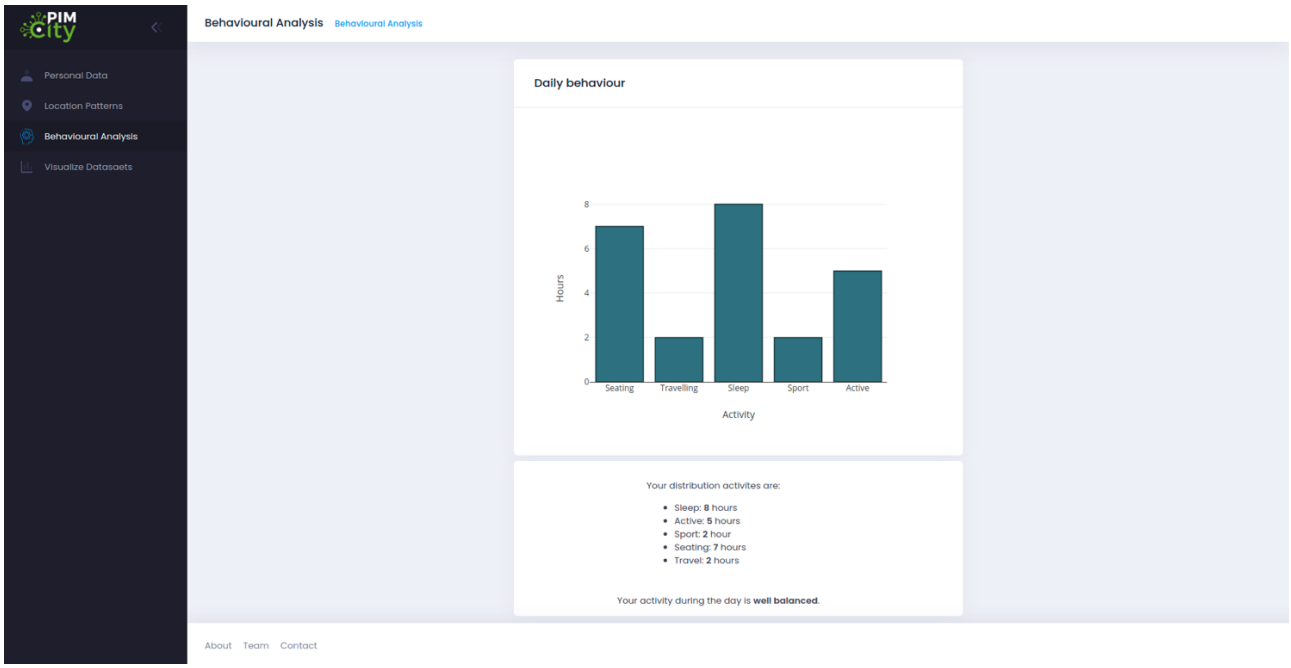



*Figure 27 - Mockup of PDA's Quantifield-self Dashboard behavioral analysis*

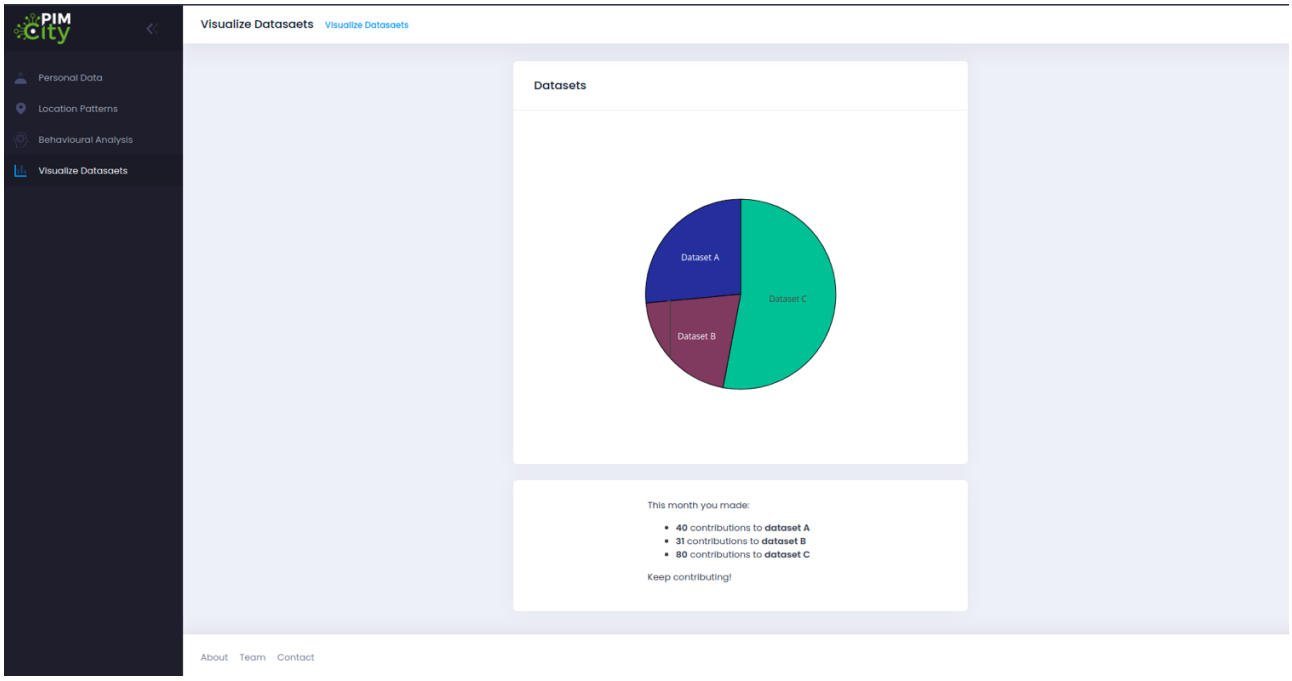



*Figure 28 - Mockup of PDA's Quantifield-self Dashboard screen showing available datasets*

## Interactions

This component is getting input from the P-DS, the Data Aggregation component, the User profiling and the Data Knowledge Extraction module through the respective APIs.

### 7.2.7 Data import and export

The right to data portability is a novelty of the EU's General Data Protection Regulation (GDPR), that allows individuals to obtain and reuse personal data from one environment to another. The purpose of this tool is to support such novelty by allowing individual users to migrate their data to new platforms, in a privacy-preserving fashion.

#### Functionalities

The user can:

- Extract data from one PIMS (e.g., Facebook, bank, mobile phone)
- Process it by filtering out sensitive information such as platform-inferred data (e.g., user unavailability due to event attendance) or user-inputted data (e.g., remove login credentials or debit card numbers)
- Outport it into the PDK module, a new PIMS (e.g., EasyPIMS) or an exported file in a common data interchange format (e.g., JSON).

#### Mockup

This tool only provides an API that is consumed by the User Dashboard. In next figure we provide an example UI that will be available in the User Dashboard that lists all available Data Sources, Data Transformations and Data Exports, together with relevant actions to edit, deactivate and delete them.

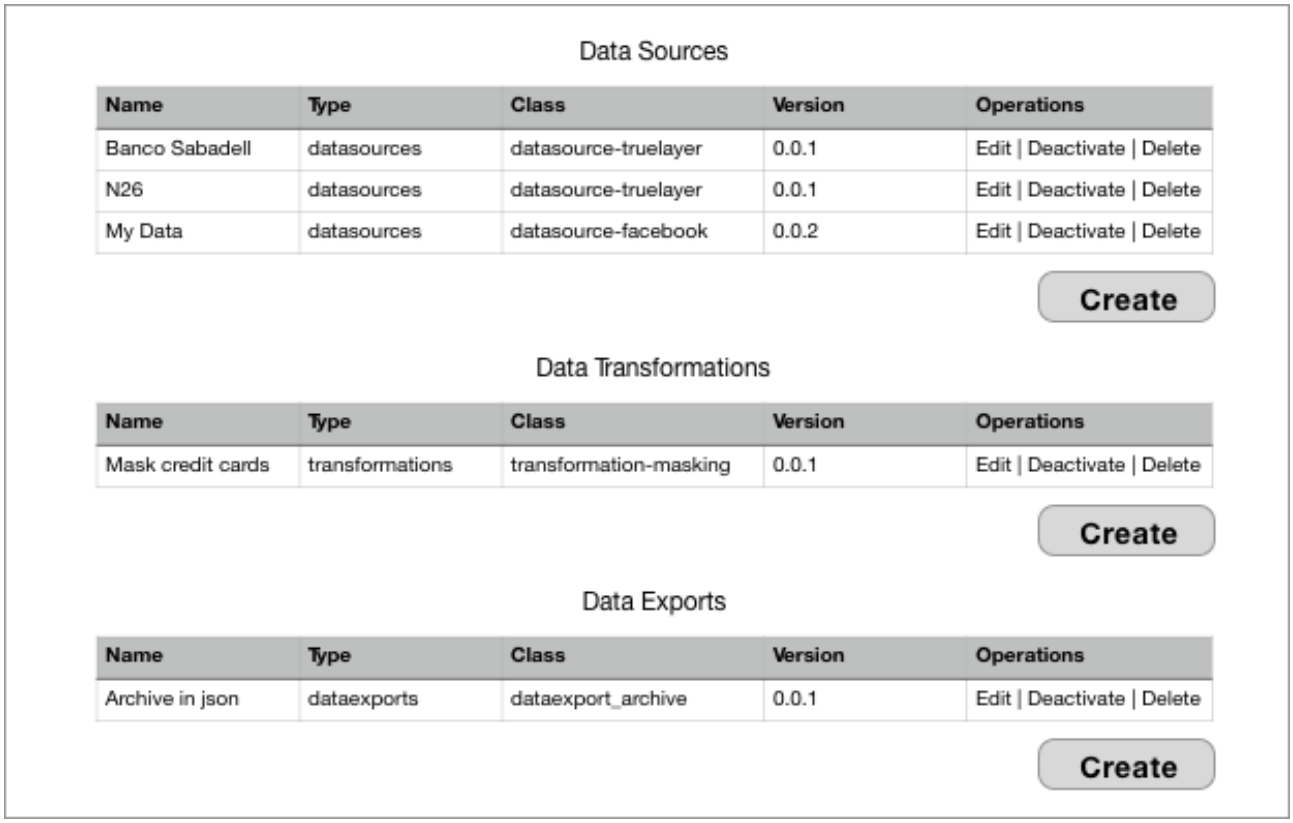



*Figure 29 - Mockup of presentation of data sources*

#### Interactions

This tool interacts directly with the User Dashboard to perform all the supported functionalities. As the PDK service requires authentication, the JWT obtained from the Cloud Controller when the user logged in is used.

### 7.2.8 Consent Manager

Its primary objective is to give the users transparency and control over their data in a GDPR compliant way. That is, give them the possibility to decide which data can be uploaded and stored in the platform, as well as which (raw, extracted or aggregated) data can be shared with Data Buyers in exchange for value when the opportunity arises.

Consent Management over the internet is still brand new as regulations define new rights and responsibilities. Therefore, there is no standard way of gathering and providing consents to store and use data. With PIMCity's P-CM, we aim at:

- Defining a standard interface to store, update, delete, audit and activate consents received for multiple purposes, such as storing data and sharing it with third parties.
- Providing a web application as an implementation of such standard interface.
- Implementing state-of-the-art techniques, using standard community-approved software libraries.
- Putting in place an integration flow that eases the deploy and allows for quick upgrades to the tool in the production environment.
- Making the component efficient, fast and scalable to accommodate the consents of a very large set of users.

#### Functionalities

The user can:

- Read the current consent settings. This shows the user which data is being shared and under which conditions.
- Modify the consent settings, allowing users to change which data is being shared, with which companies and under which conditions.

#### Mockup

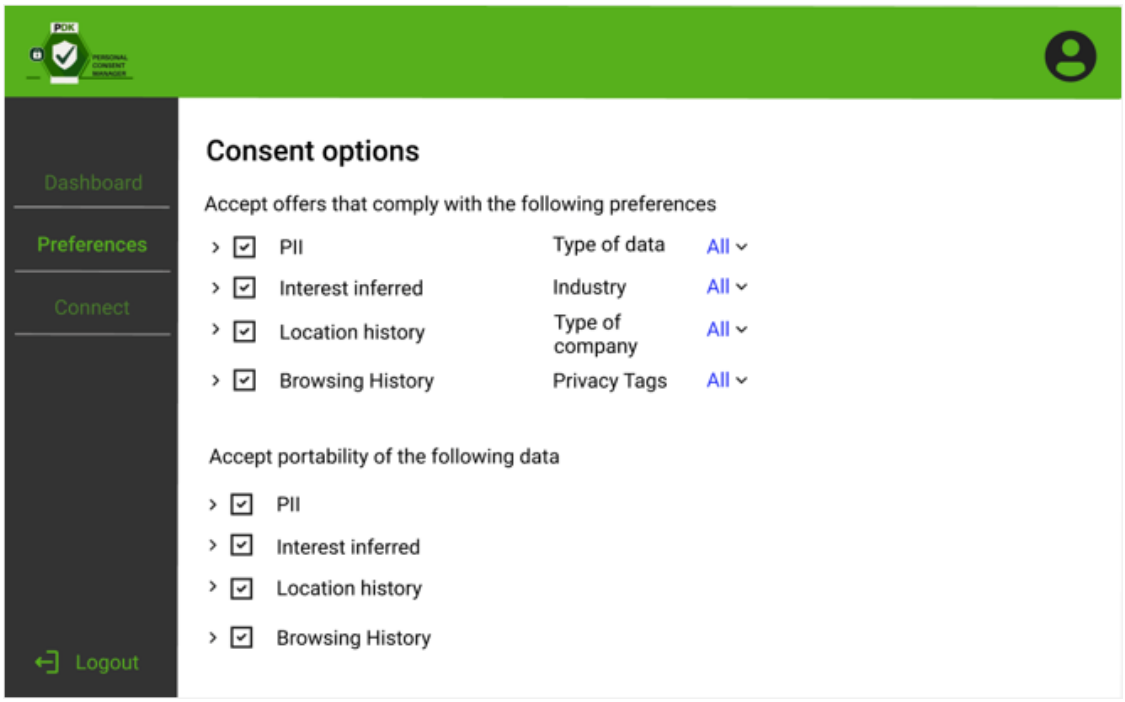



*Figure 30 - Mockup of PDA's Consent Manager screen*

#### Interactions

This screen interacts directly with the Personal Consent-Manager to perform all the supported functionalities. As the PDK service requires authentication, the JWT obtained from the Cloud Controller when the user logged in is used.

### 7.2.9 Settings

This part of the PDA will allow the user to view and change their settings related to the general profile and the parameters related to the account, like the security options, the contact details and possibly settings related to the general appearance and the UI of the PDA.

#### Functionalities

The user should be able to view and edit their personal details, like the username, email, password and possibly other appearance options.

Settings related to the individual components of the PDA will be available through their own interface.

#### Mockup

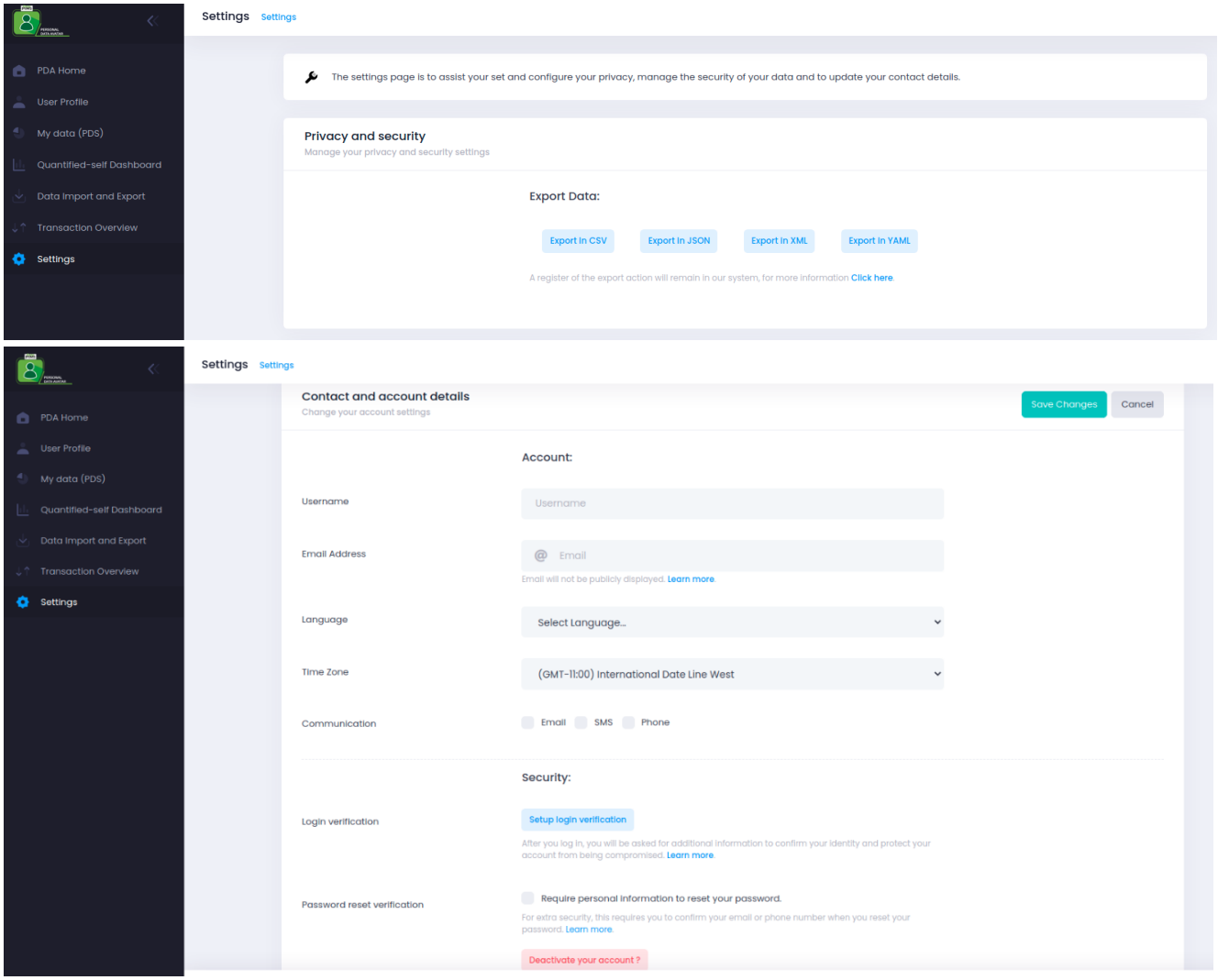



*Figure 31 - Mockups of PDA's settings screen*

#### Interactions

The settings component will not interact with other PDK components but is part of the PDA.

# 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[37] 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 is 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.

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

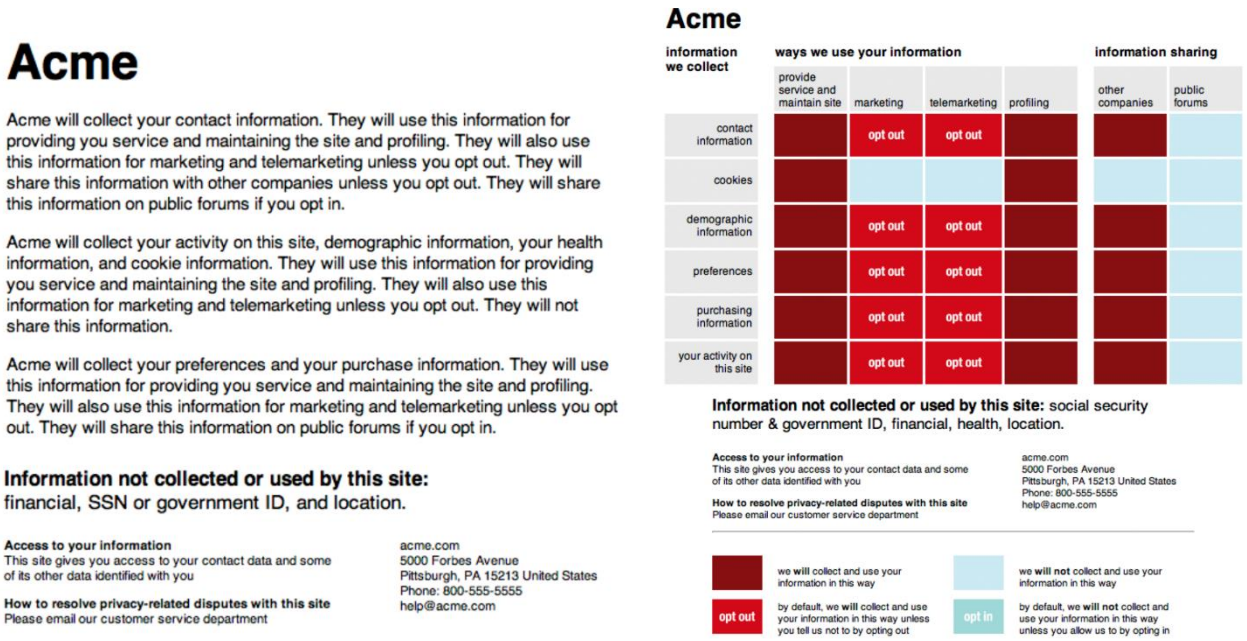



*Figure 32 - Examples of formats for Privacy Nutrition Labels developed by CUPS researchers. Standardized short-text (left) and standardized table (right)*

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[38]. 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.

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

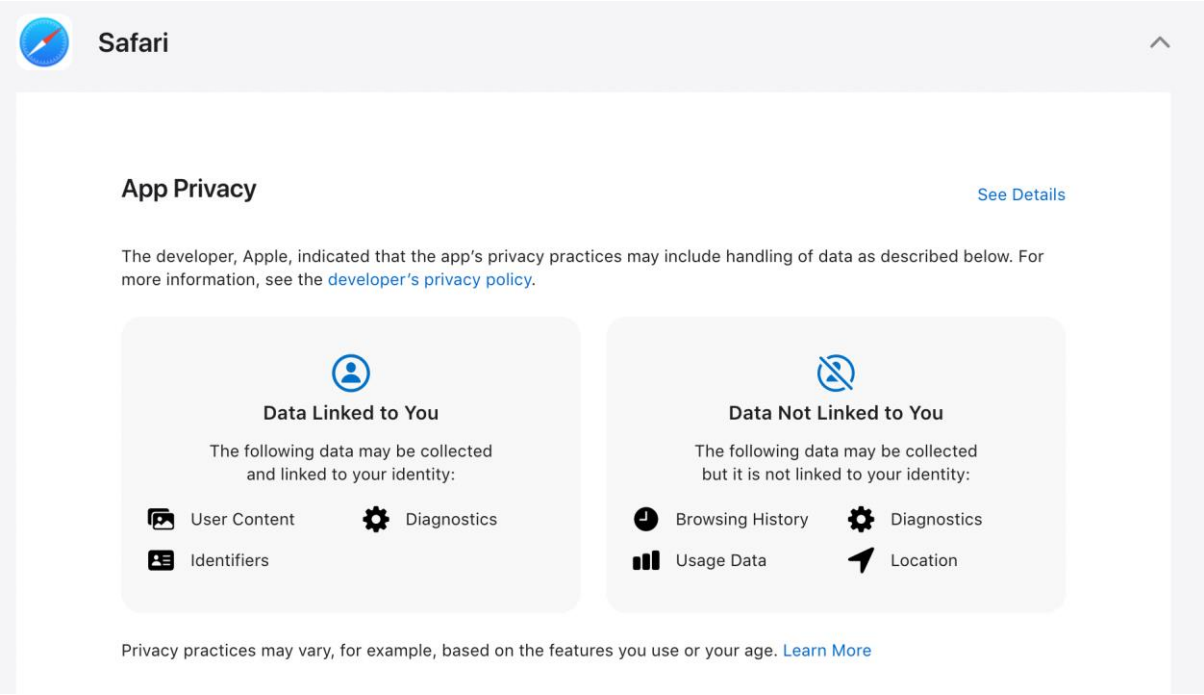



*Figure 33 - Example of Apple's Privacy Label for Safari browser.*

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[39] (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[40] (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 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.

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

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

For all reasons above, TT's UI is quite specific from the perspective of design requirements, and since user interaction might be needed, we have to 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 have to 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[41].

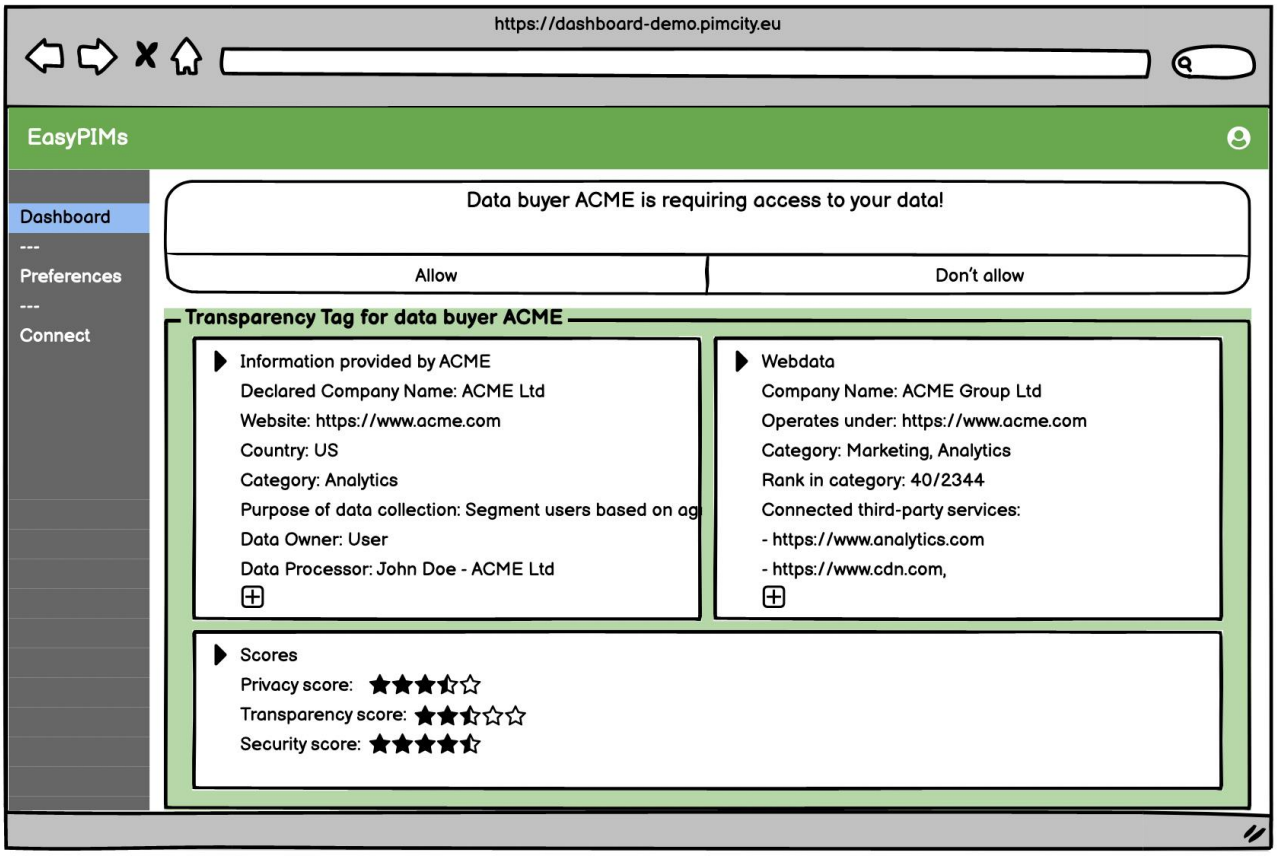
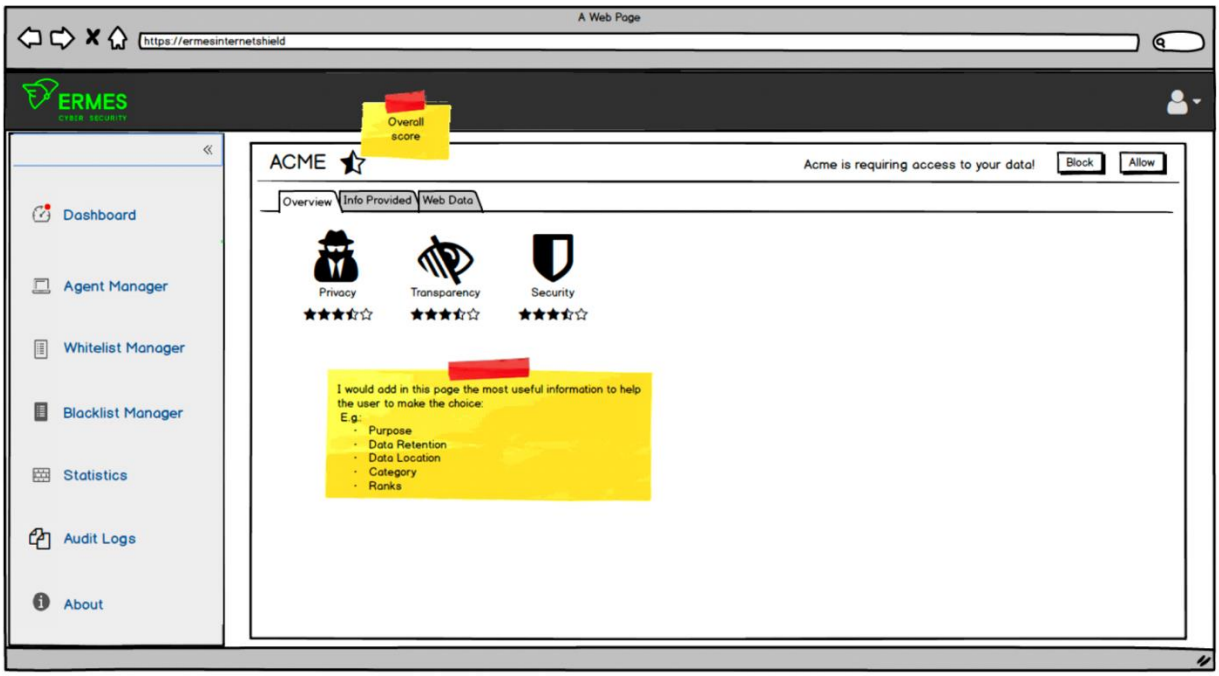
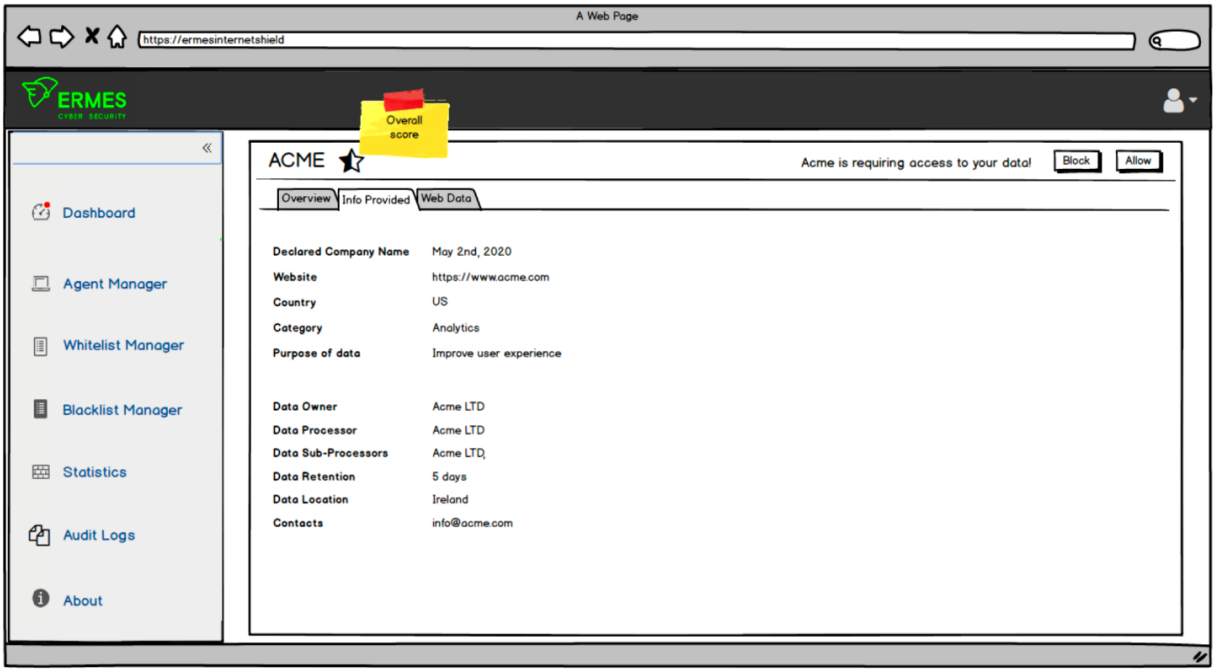



*Figure 34 - Single-page TT mockup for desktop*

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.

[41] 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

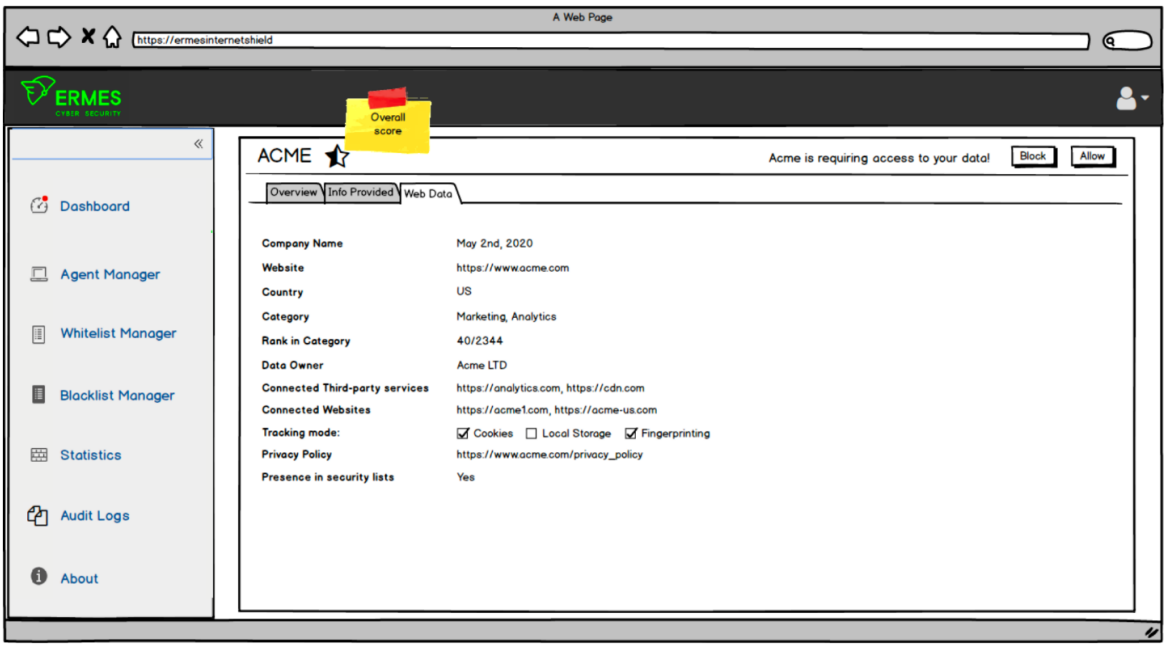



*Figure 35 - Multi-tab TT mockup for desktop*

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

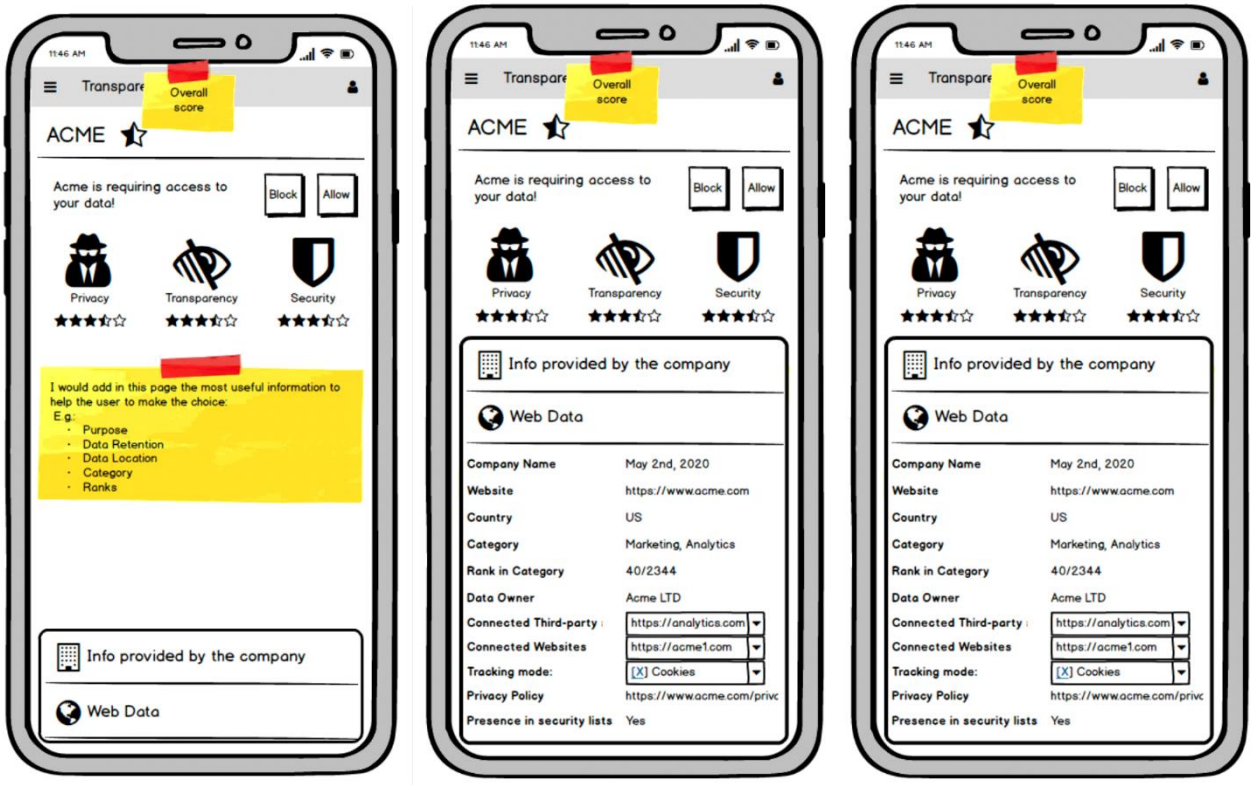



*Figure 36 - Multi-tab TT mockup for mobile*

## 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 is organizing multiple focus groups at the moment of this writing, and results will be considered and presented in development of this task.

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.

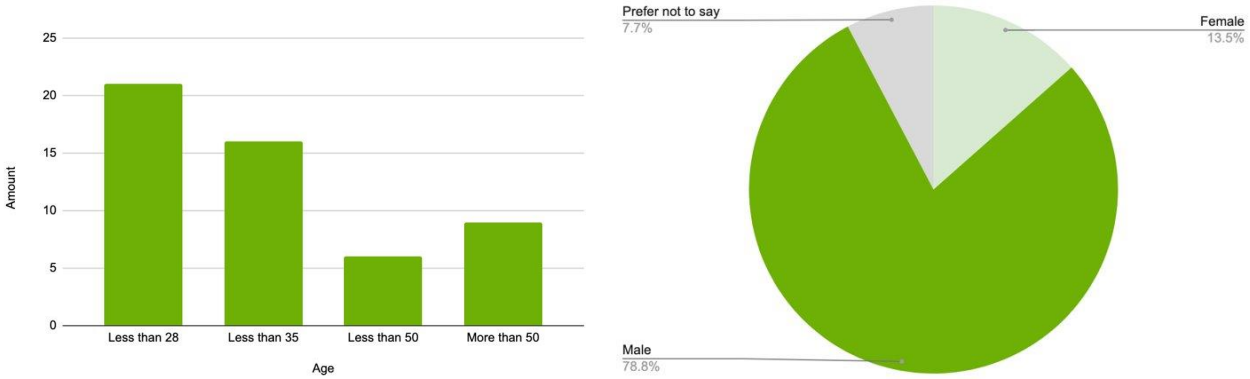
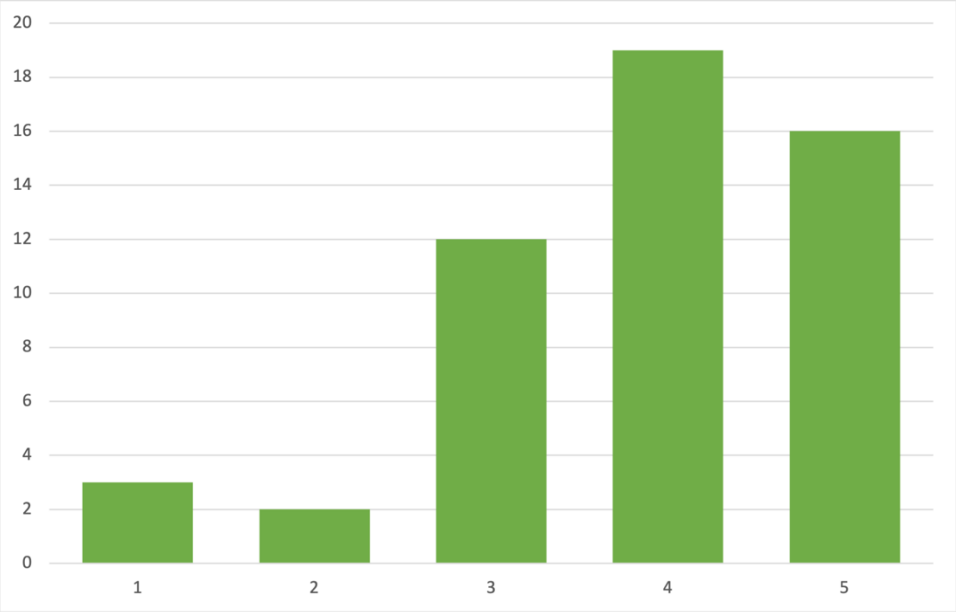


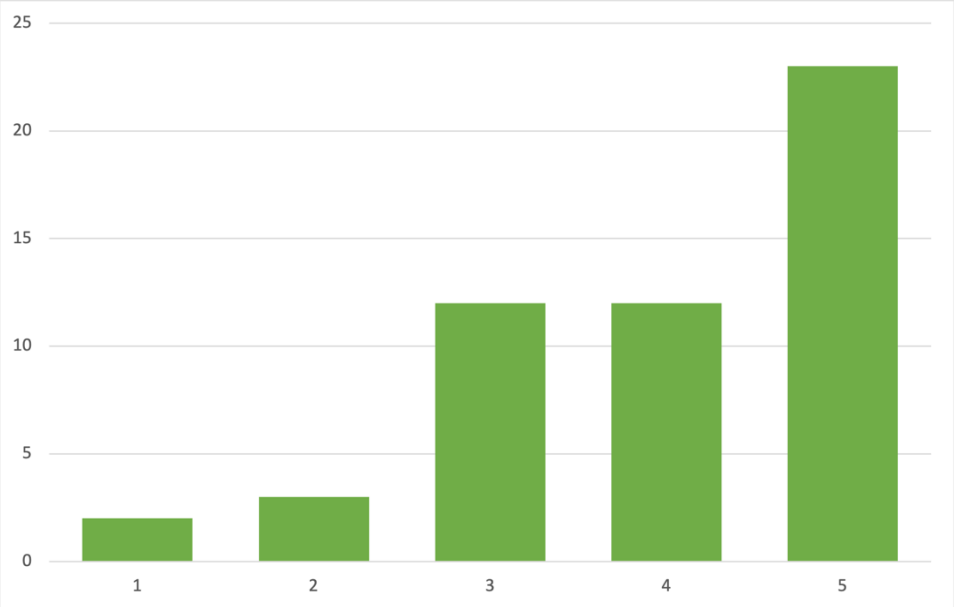

*Figure 37 - Demographics of end users who participated the online survey*

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.

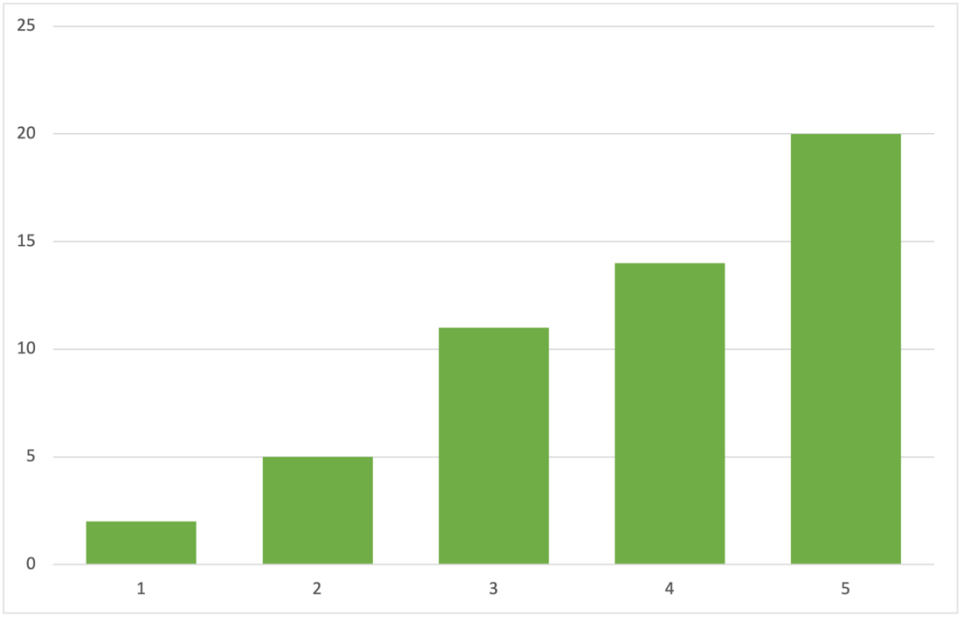
**How important is to know the category of the company buying your data?**
As shown below, users greatly appreciate to know the category of the company asking to collect and process their personal data. 47 (90%) users rate this information greater or equal than 3.




**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.

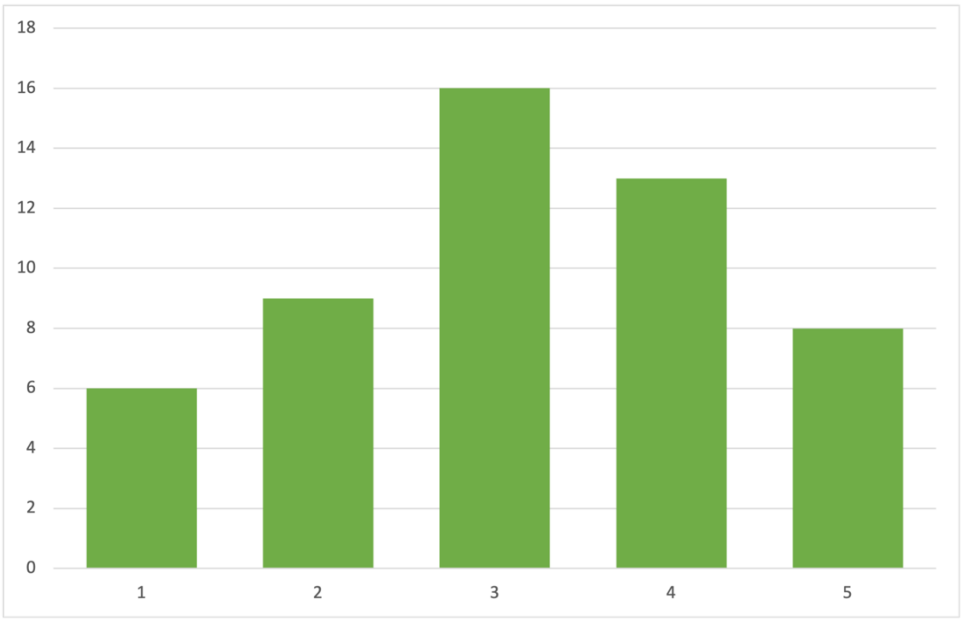



**How important is to know the security and privacy rating of the company buying your data?**
Yet users want to get information about the reliability of company asking to collect and process their personal data. Also in this case 47 (90%) users rate this information greater or equal than 3.



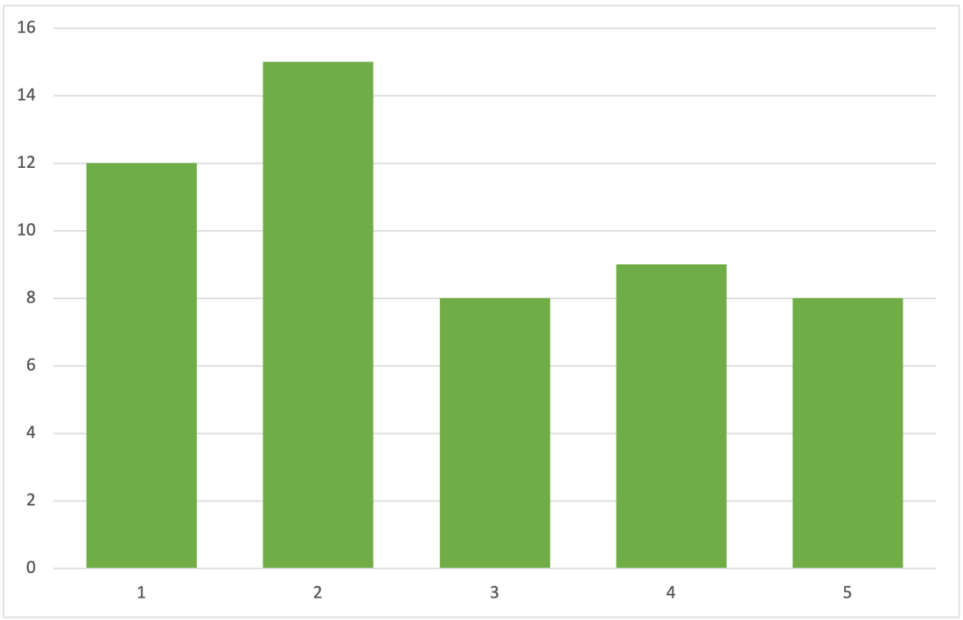

**How important is to have a description of the company asking to buy your data?**

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.



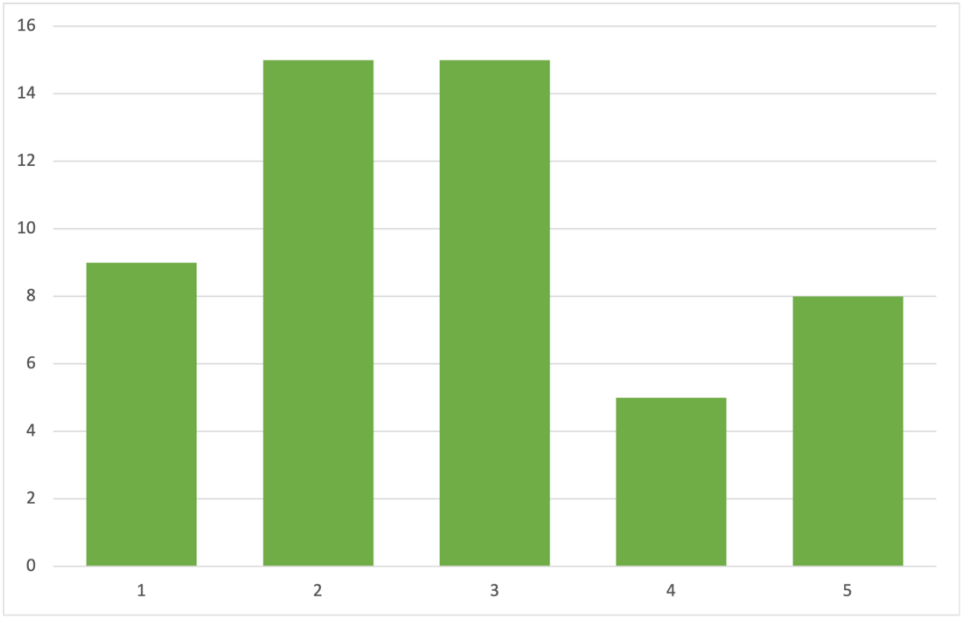

**How important is to get the link to the company's privacy policy?**

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 emerges 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

All in all, the feedback we received so far about the current design of Transparency Tags is quite positive. Users we contacted welcome the information and the presentation contained in Transparency Tags, but they also suggest modifications and improvements that we will evaluate in the development of the project and in the next design cycles. In general, we register 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 so far seems to validate the design choices taken so far. We will take care of iterating over such choices as new feedack will be acquired.

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

# 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 for 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.

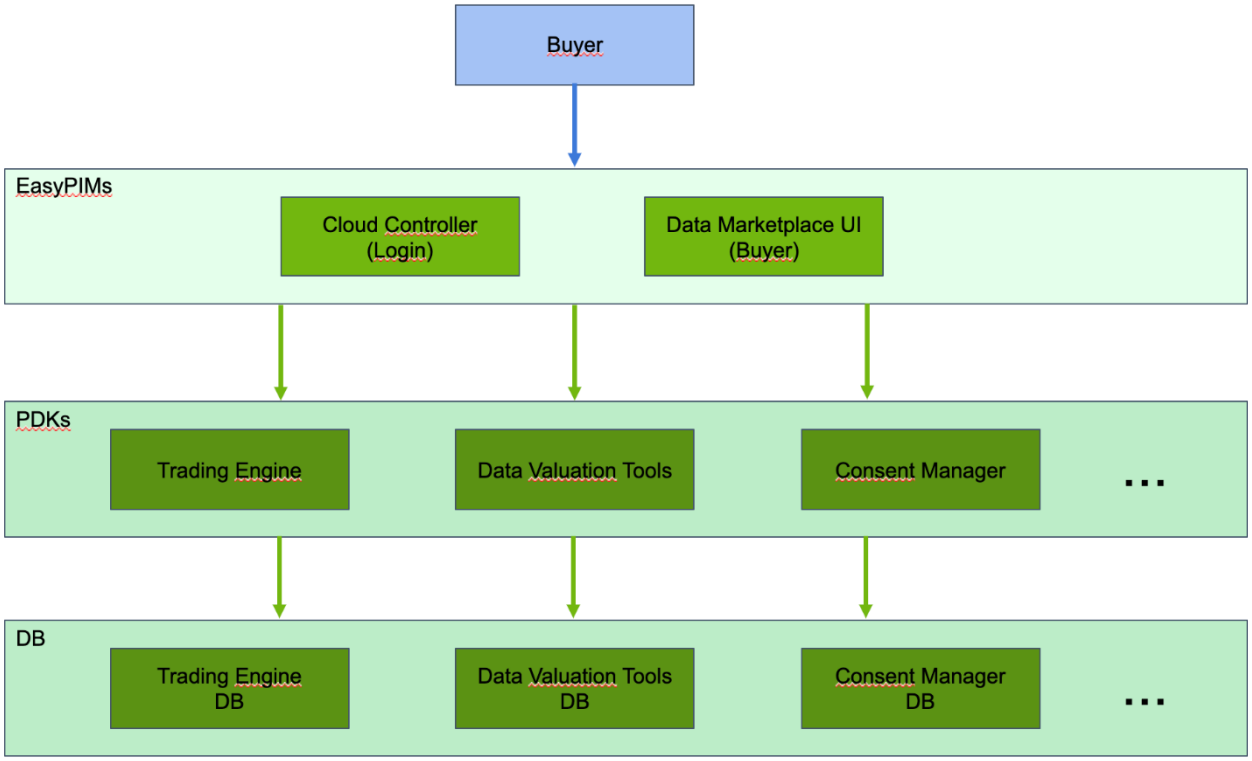



*Figure 38 - Data Marketplace webmap*

### 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 its account, with its 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 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
- Job Title

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

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

Mockups presented in the section below have been used to run surveys and collect feedback from different communities and stakeholders.

**Mockup**


*Figure 39 – Registration screen mockup for Data Marketplace*


*Figure 40 - Data Marketplace's login screen mockup*

### 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
- Feed of last/most popular five data transactions recently closed in the market (by other buyers)

### Mockup

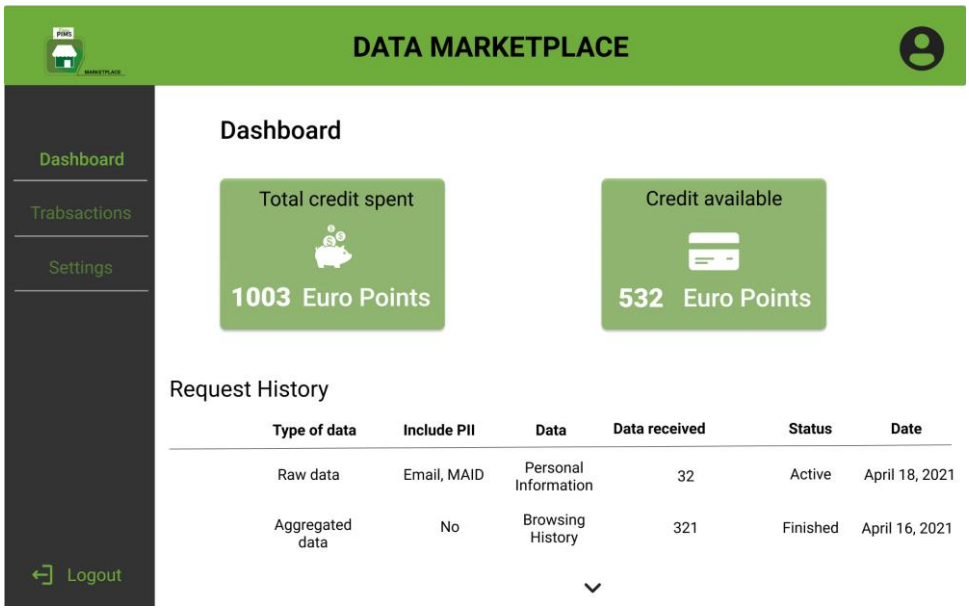



*Figure 41 - Mockup of Data Marketplace's main screen*

### 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.

### Mockup

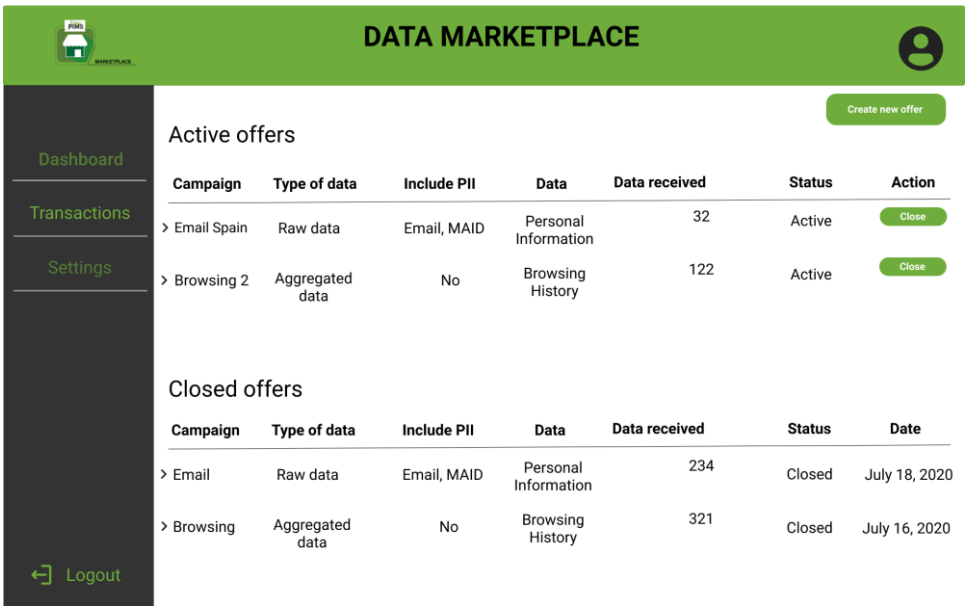



*Figure 42 - Mockup of Data Marketplace's transactions overview screen*

### Interactions

This screen interacts directly with the Trading Engine to extract the list of the 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.

### Mockup

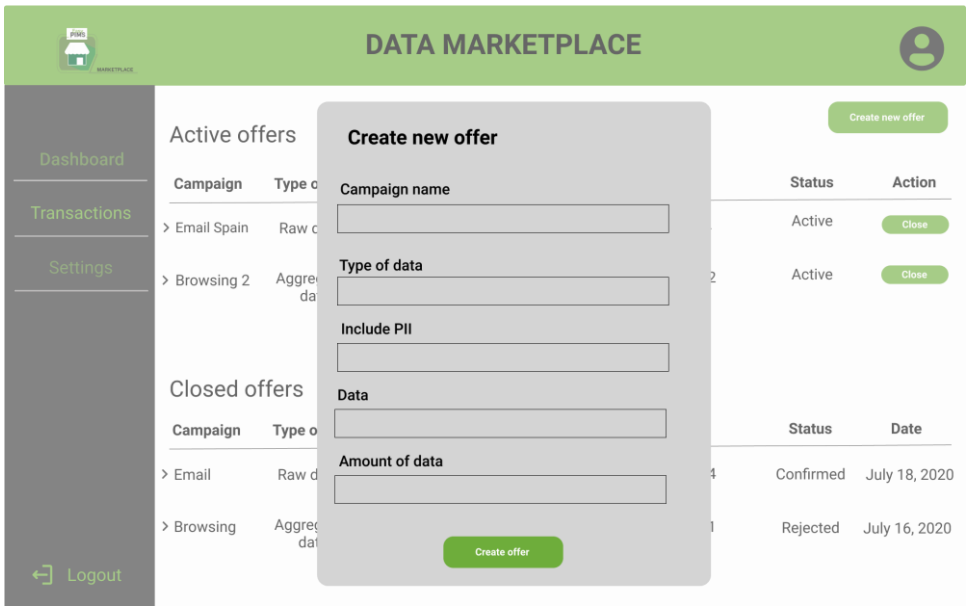



*Figure 43 - Mockup of Data Marketplace's screen to create new data offer*

### 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 Account Profile

This screen allows the buyer to read all the account information and company details, as well as to edit the settings of the account. Also, the buyer can read the current balance in credits and notify a new deposit done to update the amount in the platform.

Last but not least, the buyer can edit the Transparency Tags, change the password or delete the account.

### Functionalities

This screen allows the buyer to perform the following actions:

- Read all the company details submitted during the registration phase
- Edit privacy policy link for future Data Transactions.
- Points management:
  - Read the current balance
  - Notify a new deposit
- Other settings:
  - Password change
  - Delete account
  - Edit Transparency tags

### Mockup

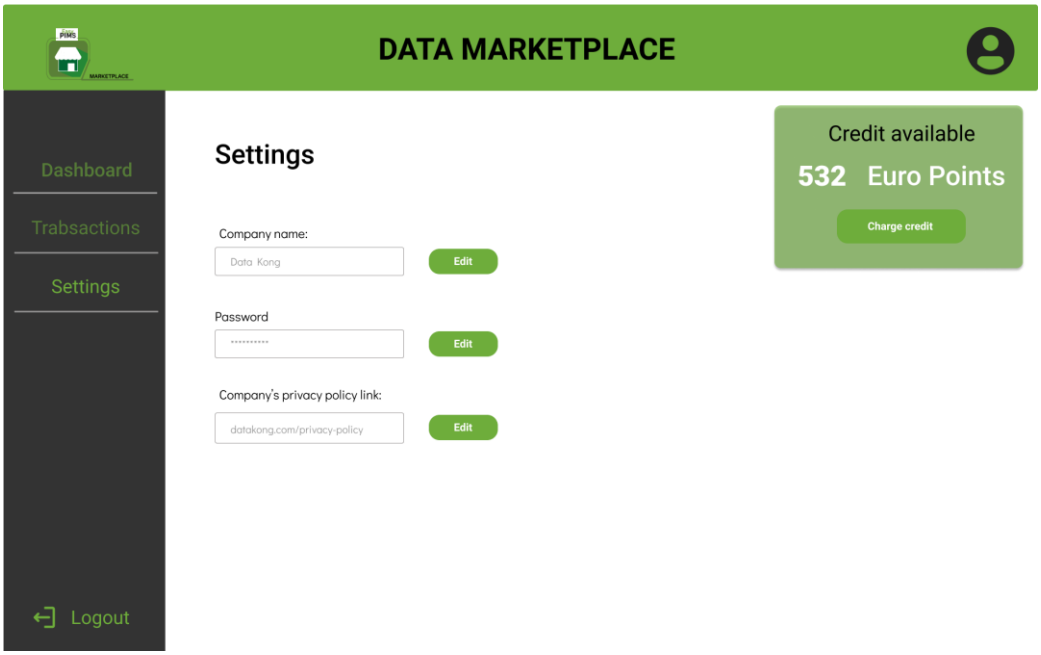



*Figure 44 - Mockup of Data Marketplace's settings screen*

### Interactions

This screen interacts directly with the Trading Engine to extract all the information about the company, except for the Transparency Tags that is connected with the P-PM PDK. As both TE and 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.

# 10 Conclusions

This deliverable presented the first design of EasyPIMS' fundamental components. We described the design for the Cloud Controller, the Open APIs, the Personal Data Avatar, the Transparency Tags and the Data Marketplace. For each component we described the technical requirements, the design choices, and when possible, the preliminary feedback we collected to drive the design.

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

# 11 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.

# 12 - Appendix

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 |