# D3.1 ICT Platform and Dissemination Dashboard

## Document Identification

<table>
<thead>
<tr>
<th>Status</th>
<th>Final</th>
<th>Due Date</th>
<th>30/04/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.0</td>
<td>Submission Date</td>
<td>06/08/2018</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Related WP</th>
<th>WP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Deliverable(s)</td>
<td>D9.3, D1.2, D9.3, D1.3, D2.1</td>
</tr>
<tr>
<td>Document Reference</td>
<td>D3.1</td>
</tr>
<tr>
<td>Dissemination Level (*)</td>
<td>PU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lead Participant</th>
<th>ICCS</th>
</tr>
</thead>
</table>
| Lead Author      | Richardos Drakoulis (ICCS)  
|                  | Fay Misichroni (ICCS)  
|                  | Maria Krommyda (ICCS)  
|                  | Anna Antonakopoulou (ICCS) |
| Contributors     | ICCS, AKKA, TBridge, AAU |
| Reviewers        | Antonella Berardi, Alessandro Rinaldi (ICOOR POLIBA)  
|                  | Juan Sancho, Gabriel Martinez (ATOS) |
# Document Information

## List of Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evangelia Portouli</td>
<td>ICCS</td>
</tr>
<tr>
<td>Marco Perotti</td>
<td>AKKA</td>
</tr>
<tr>
<td>Djibrilla Amadou-Kountche</td>
<td>AKKA</td>
</tr>
<tr>
<td>Benoit Baurens</td>
<td>AKKA</td>
</tr>
<tr>
<td>Simone Porru</td>
<td>TBridge</td>
</tr>
<tr>
<td>Kristian Torp</td>
<td>AAU</td>
</tr>
</tbody>
</table>

## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change editors</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>01/01/2018</td>
<td>Kristian Torp (AAU)</td>
<td>Started draft document</td>
</tr>
<tr>
<td>0.2</td>
<td>16/02/2018</td>
<td>Evangelia Portouli (ICCS)</td>
<td>Structure, architecture, open points comments</td>
</tr>
<tr>
<td>0.3</td>
<td>16/04/2018</td>
<td>Evangelia Portouli (ICCS)</td>
<td>ToC Restructure and initial content</td>
</tr>
<tr>
<td>0.4</td>
<td>06/06/2018</td>
<td>Evangelia Portouli (ICCS)</td>
<td>Comments and proposals from TBridge</td>
</tr>
<tr>
<td>0.5</td>
<td>07/06/2018</td>
<td>Evangelia Portouli (ICCS)</td>
<td>General Architectural schema defined</td>
</tr>
<tr>
<td>0.6</td>
<td>18/06/2018</td>
<td>Richardos Drakoulis (ICCS)</td>
<td>Semi-final version of all the document (AKKA and ICCS inputs)</td>
</tr>
<tr>
<td>0.7</td>
<td>02/07/2018</td>
<td>Richardos Drakoulis (ICCS)</td>
<td>Format and grammar check</td>
</tr>
<tr>
<td>0.8</td>
<td>17/07/2018</td>
<td>Anna Antonakopoulou (ICCS)</td>
<td>Introduction, Conclusions and References added, send for internal review</td>
</tr>
<tr>
<td>0.9</td>
<td>26/07/2018</td>
<td>Fay Misichroni (ICCS), Richardos Drakoulis (ICCS)</td>
<td>Internal Review Comments addressed</td>
</tr>
<tr>
<td>1.0</td>
<td>27/07/2018</td>
<td>Anna Antonakopoulou (ICCS)</td>
<td>Final version sent to QM</td>
</tr>
</tbody>
</table>
### Quality Control

<table>
<thead>
<tr>
<th>Role</th>
<th>Who (Partner short name)</th>
<th>Approval Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable leader</td>
<td>Richardsos Drakoulis (ICCS), Fay Misichroni (ICCS), Maria Krommyda (ICCS), Anna Antonakopoulou (ICCS)</td>
<td>27/07/2018</td>
</tr>
<tr>
<td>Technical manager</td>
<td>Evangelia Portouli (ICCS)</td>
<td>27/07/2018</td>
</tr>
<tr>
<td>Quality manager</td>
<td>Ivan Zaldivar (ATOS)</td>
<td>31/07/2018</td>
</tr>
<tr>
<td>Project Coordinator</td>
<td>Angelos Amditis (ICCS)</td>
<td>06/08/2018</td>
</tr>
</tbody>
</table>
# Table of Contents

Document Information ............................................................................................................. 2  
Table of Contents ..................................................................................................................... 4  
List of Figures .......................................................................................................................... 5  
List of Tables ........................................................................................................................... 6  
List of Acronyms ...................................................................................................................... 6  
Executive Summary ................................................................................................................ 8  
1. Introduction ......................................................................................................................... 9  
   1.1 Purpose of the document ................................................................................................. 9  
   1.2 Relation to other project works ..................................................................................... 9  
   1.3 Structure of the document ............................................................................................ 9  
2. ICT Platform Architecture ..................................................................................................... 11  
   2.1 Introduction .................................................................................................................... 11  
   2.2 General Architecture of ELVITEN ............................................................................... 12  
   2.3 Functional Requirements of ELVITEN ICT Platform .................................................... 13  
   2.4 The users of ELVITEN .................................................................................................. 14  
   2.5 ELVITEN ICT Platform Components ........................................................................... 18  
       2.5.1 ELVITEN Middleware ............................................................................................ 18  
       2.5.2 ELVITEN Identity Manager .................................................................................... 18  
       2.5.3 ELVITEN Data Staging .......................................................................................... 18  
       2.5.4 ELVITEN Data Warehouse .................................................................................... 19  
       2.5.5 ELVITEN Dashboard ............................................................................................. 20  
3. Configuration Management ................................................................................................... 22  
4. Data Quality Assurance ....................................................................................................... 23  
   4.1 Validation Rules .............................................................................................................. 23  
   4.2 Metadata ....................................................................................................................... 23  
5. Data privacy ........................................................................................................................... 27  
   5.1 ELVITEN Identity Manager ........................................................................................... 27  
   5.2 ELVITEN Middleware ................................................................................................... 28  
   5.3 ELVITEN Data Warehouse ............................................................................................ 28
List of Figures

Figure 1: ELVITEN ecosystem .................................................................................. 11
Figure 2: ELVITEN ICT Platform Architecture .................................................. 12
Figure 3: High-level use cases of the ELVITEN EL-V user ................................ 14
Figure 4: High-level use cases of the Regional Support Group members .......... 15
Figure 5: High-level use cases of the general public ........................................... 16
Figure 6: High-level use cases of the Scientist .................................................... 17
Figure 7: Data model of the ELVITEN Data Warehouse ..................................... 20
Figure 8: The General architecture of RabbitMQ adapted to ELVITEN .......... 29
Figure 9: Communication use case between the ELVITEN Middleware and the Incentives System 30
Figure 10: OpenID Connect components .............................................................. 31
Figure 11: OpenID Connect workflow ................................................................. 33
Figure 12: Model-View-Controller architecture .................................................. 35
Figure 13 Illustration of OAuth 2.0 in ELVITEN project .................................. 43
Figure 14: Keycloak’s authentication page ............................................................ 45
Figure 15: Keycloak - Main page .......................................................................... 45
Figure 16: Keycloak – Add role ............................................................................. 45
Figure 17: Keycloak – Add user ............................................................................ 46
Figure 18: Keycloak – User list ............................................................................. 46
Figure 19 Keycloak – ELVITEN User info ............................................................... 46
Figure 20: Keycloak – Assign user to role ............................................................. 47
List of Tables

Table 1: ICT platform functional requirements ........................................................................... 13
Table 2: ICT component-requirement relations ........................................................................ 13
Table 3: Example of metadata for time history Field Operational Test data ................................. 23
Table 4: Example of DDI compliant metadata attributes ............................................................ 24
Table 5: Privacy concerns and countermeasures ........................................................................ 27
Table 6: ID Token features .......................................................................................................... 42

List of Acronyms

<table>
<thead>
<tr>
<th>Abbreviation / acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMQP</td>
<td>Advanced Message Queuing Protocol</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CESSDA</td>
<td>Council of European Social Science Data Archives</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>DDI</td>
<td>Data Document Initiative (DDI)</td>
</tr>
<tr>
<td>DSA</td>
<td>Data Staging Area</td>
</tr>
<tr>
<td>DW</td>
<td>Data Warehouse</td>
</tr>
<tr>
<td>EL-V</td>
<td>Electrified Light Vehicle</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract, Transform, Load</td>
</tr>
<tr>
<td>FAIR</td>
<td>Findable, Accessible, Interoperable and Re-usable</td>
</tr>
<tr>
<td>GDPR</td>
<td>General Data Protection Regulation</td>
</tr>
<tr>
<td>GIN</td>
<td>Generalised Inverted Index</td>
</tr>
<tr>
<td>GiST</td>
<td>Generalised Search Tree</td>
</tr>
<tr>
<td>Abbreviation / acronym</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technologies</td>
</tr>
<tr>
<td>JEE</td>
<td>Java Platform, Enterprise Edition</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>JWE</td>
<td>JSON Web Encryption</td>
</tr>
<tr>
<td>JWS</td>
<td>JSON Web Signature</td>
</tr>
<tr>
<td>JWT</td>
<td>JSON Web Token</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>KVM</td>
<td>Kernel-based Virtual Machine</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>LXC</td>
<td>Linux Containers</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-View-Controller</td>
</tr>
<tr>
<td>OP</td>
<td>OpenID Provider</td>
</tr>
<tr>
<td>RAM</td>
<td>Random-access Memory</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RFC</td>
<td>Requests for Comment</td>
</tr>
<tr>
<td>RP</td>
<td>Relying Party</td>
</tr>
<tr>
<td>RSG</td>
<td>Regional support group</td>
</tr>
<tr>
<td>SCM or S/W CM</td>
<td>Software Configuration Management</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSO</td>
<td>Single sign-on</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>VCS</td>
<td>Version Control System</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifiers</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>
Executive Summary

ELVITEN project organises demonstrations of electrified light vehicles (EL-Vs) in six European cities and will collect trip data and users’ perceptions and experiences in order to boost the market uptake of such vehicles. For this, it will deploy the “ELVITEN ICT Platform” in order to handle the data flow from the various ICT tools operating in each City and to allow data extraction for analysis and data visualisation.

This deliverable presents the reference architecture and the design of the functional building blocks of the platform, which are the Identity Manager, the Middleware, the Data Warehouse, the Data Staging and the Dashboard, their relation and interaction. The reference architecture of the platform aims to fulfil the functional requirements needed for the project objectives and to cater for security, safety and privacy issues. The deliverable presents the software configuration management process, the data quality assurance and preparation procedures and the data protection mechanisms. Data privacy assurance requirements are fulfilled through the evaluation and subsequent adoption of suitable security and privacy enhancing techniques (e.g. pseudonymisation). Quality assurance requirements are fulfilled by implementing a data management layer for validating and storing the data collected from the various sources and Cities. The deliverable also presents the implementation and deployment strategy to be followed for each of the Platform components.

The ELVITEN ICT Platform will allow partners, stakeholders and the general public to get access to the data generated during the ELVITEN demonstrations in an efficient and comprehensive way, thus enabling the calculation and analysis of indicators for the studies and analyses in the framework of the project and after its completion.
1. Introduction

1.1 Purpose of the document

ELVITEN project organises demonstrations of electrified light vehicles (EL-Vs) in six European cities and will collect trip data and users’ perceptions and experiences in order to boost the market uptake of such vehicles. For this, it will deploy an “ICT Platform and Dissemination Dashboard” in order to handle the data flow from the various ICT tools operating in each City and to allow data extraction for analysis and data visualisation. This deliverable, entitled “D3.1 ICT Platform and Dissemination Dashboard”, presents the reference architecture and the design of the functional building blocks of the platform (in short, the “ELVITEN ICT Platform”), their relation and interaction. The reference architecture of the platform aims to fulfil all functional requirements, including security, safety and privacy issues. This deliverable is the result of the work performed in two tasks of the project:

1. Task 3.2 is related to the design and implementation of the ICT platform. This task defined how to link the data flows generated by the services and tools in the Demonstration Cities, to the central ELVITEN ICT platform. This task listed the sub-systems and organised them, by describing the main relationships among them.

2. Task 3.1 described the basic system functionalities to be provided by the ELVITEN ICT Platform as a whole. The work was based on the service-orientation paradigm which is appropriate for large-scale distributed systems consisting of loosely-coupled components, and on Cloud Platforms, allowing the fast and cost-competitive creation of innovative ICT services for electromobility.

1.2 Relation to other project works

This deliverable is the output of task T3.2 and its purpose is to describe the ELVITEN ICT platform and its functionalities, and to provide an overview of its implementation. This deliverable is also based on task “T3.1 ICT requirements analysis and assets inventory”.

1.3 Structure of the document

The document is structured as follows:

- Section 2 presents the ELVITEN ICT platform architecture and analyses the functionality of its main components.
- Section 3 outlines the software configuration management process.
- Section 4 describes how the data received from the cities will be quality assured and cleansed.
- Section 5 explains how data-related privacy issues are handled from the ELVITEN ICT platform.
- Section 6 illustrates the implementation strategy towards the final and fully functional versions of the software components.
- Section 7 analyses the deployment strategy applied on the various software components.
- Section 8 reports the main conclusions of the deliverable.
2. ICT Platform Architecture

2.1 Introduction

The ELVITEN ICT ecosystem consists of two major software blocks:

1. The ICT tools operational in the Demonstration Cities, which will be presented in detail in deliverable D3.2.
2. The ELVITEN ICT platform, which is the subject of this deliverable.

These blocks are illustrated in Figure 1.

![Figure 1: ELVITEN ecosystem](image)

The ELVITEN ICT platform includes the components that are responsible for the data flow from the ELVITEN ICT Tools to the ELVITEN Data Warehouse, as well as the software for data extraction and visualisation. This section supplies an overview of the ICT platform architecture followed by a detailed description of each software component.
2.2 General Architecture of ELVITEN

The general architecture is depicted in Figure 2.

Figure 2: ELVITEN ICT Platform Architecture
2.3 Functional Requirements of ELVITEN ICT Platform

The first step of the design process was the definition of the platform’s functional requirements. The ICT requirements defined in deliverable D1.2 [1] and task T3.1 have been taken into consideration, although they are not directly applicable to the ICT Platform, since they are focusing on the ICT Tools side. The main functional requirements for the ICT platform are reported in Table 1.

Table 1: ICT platform functional requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICTP01</td>
<td>The ICT platform must support multiple and heterogeneous data sources.</td>
</tr>
<tr>
<td>ICTP02</td>
<td>The ICT platform should enable communication among the different ICT Tools.</td>
</tr>
<tr>
<td>ICTP03</td>
<td>Data from all data sources should be stored in a central repository.</td>
</tr>
<tr>
<td>ICTP04</td>
<td>The data repository must scale to very large data sizes without affecting the performance of computations depending on the stored data.</td>
</tr>
<tr>
<td>ICTP05</td>
<td>Data privacy and user pseudonymization must be respected.</td>
</tr>
<tr>
<td>ICTP06</td>
<td>Each user should be authenticated/authorized across the whole system with a single authentication call.</td>
</tr>
<tr>
<td>ICTP07</td>
<td>Stored data must be available to scientists for analysis.</td>
</tr>
<tr>
<td>ICTP08</td>
<td>Calculated KPIs (from the stored data) should be available to the general public.</td>
</tr>
</tbody>
</table>

Based on the platform functional requirements, the main architectural components have been derived. Table 2 presents the list of ICT platform’s components along with the requirements relevant for each of them. The functionality of each component is described in detail in its dedicated subsection.

Table 2: ICT component-requirement relations

<table>
<thead>
<tr>
<th>Component name</th>
<th>Related requirement ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELVITEN Middleware</td>
<td>ICTP01, ICTP02</td>
</tr>
<tr>
<td>Identity Manager</td>
<td>ICTP05, ICTP06</td>
</tr>
<tr>
<td>Data Staging</td>
<td>ICTP01, ICTP04</td>
</tr>
<tr>
<td>Data Warehouse</td>
<td>ICTP03, ICTP04, ICTP05</td>
</tr>
<tr>
<td>ELVITEN Dashboard</td>
<td>ICTP07, ICTP08</td>
</tr>
</tbody>
</table>
2.4 The users of ELVITEN

The specification of the ELVITEN architecture addresses the following users.

**EL-V users**

Electrified Light Vehicle (EL-V) users, as described in deliverable D1.2 [1], will use the ELVITEN ICT Tools depicted in Figure 2 and they will be the ones that will generate the data to be stored in the Data Warehouse. After their registration, using the ELVITEN Identity Manager, the EL-V users will be able to access the ELVITEN ICT Tools using proper authentication and authorization mechanisms.

Figure 3 illustrates the high-level use cases of this type of users.
Members of the Regional Support Groups

These users will be responsible for verifying that key objectives of the project are assured, such as the usage level of the EL-Vs, of the parking places and of the charging points of their cities. Therefore, they will need information about the Key Performance Indicators and targets set. Upon authentication and authorisation, these users will access the aggregated data of their city related to the key objectives of the project. Their use cases are depicted in Figure 3: High-level use cases of the ELVITEN EL-V user.
Figure 4: High-level use cases of the Regional Support Group members
The General Public

These users will not require any authentication and authorisation. They will have access to aggregated data related to the benefits of using EL-Vs.

Figure 5 describes the use cases of the general public users.
Scientists

Scientists will access the data harvested from the EL-V users. They will need to be authenticated and authorized in order to access EL-Vs related data. In addition, they will not be able to re-identify the users or correlate their activities. Their use cases are shown in Figure 5.

Figure 5: High-level use cases of the general public
Figure 6: High-level use cases of the Scientist
The ELVITEN Project Partners

The ELVITEN partners will have access to the data after authentication and authorisation. Special care is taken in relation to the personal data of the ELVITEN EL-Vs users, following the principles described in the deliverable D9.3 [2]. Particular caution is taken by the administrators of the ELVITEN Identity Manager, the ELVITEN Middleware and the ELVITEN Data Warehouse on this issue.
2.5  ELVITEN ICT Platform Components

2.5.1  ELVITEN Middleware

In order to allow the decoupling of the different ELVITEN ICT Tools and to provide a secure and centralized messaging system, a Middleware has been introduced in the general ELVITEN architecture. It enables communication among the ELVITEN components by routing messages sent from the ICT Tools to the Data Warehouse and by handling communication between the ICT Tools themselves. Generally, the following main patterns of communication are considered:

1. Publisher / Subscriber
2. Request / Response

However, several more complex communication patterns are supported as well through the Orchestrator sub-module. The Middleware’s separate sub-modules, namely the Message Bus, the Caching, the Orchestrator and the Security, and the technologies adopted for their realisation are presented in Section 6.

2.5.2  ELVITEN Identity Manager

Authentication is the process of identifying an individual, entity or website based, for example, on a username and password, a client ID and client secret or biometric data (fingerprint, iris scan, face recognition, etc.). On the other hand, authorisation is the process of granting or denying access to a resource. In plain words, authentication ensures that the individual is who he or she claims and authorization ensures that he or she has the proper access rights (read/write) on a specific resource.

Within ELVITEN, the system that is responsible for authentication/authorisation is called Identity Manager. The ELVITEN Identity Manager handles both processes by providing a Single sign-on (SSO) access control. SSO is an authentication process in a client/server relationship where the user, or client, can authenticate once (with only one user/password pair for example) and have access to more than one applications or access to a number of resources within an enterprise or the whole web. SSO adoption in ELVITEN enables the users to switch among multiple ICT Tools without the need to be authenticated multiple times. Consequently, all ELVITEN ICT Tools must be integrated with the ELVITEN Identity Manager.

Upon registration, each ELVITEN user is assigned a unique ID. This ID is generated by the Identity Manager and is used horizontally in every ELVITEN ICT Tool.

2.5.3  ELVITEN Data Staging

The ELVITEN Data Warehouse stores data in a predefined way and with a predefined structure. Due to the heterogeneity of the data sources, i.e. from the ELVITEN ICT Tools, the received data have to be converted in order to conform to the requirements of the Data Warehouse design.
The whole process, from the extraction of the data to their storage, is called extract, transform, load (ETL) and includes: (i) the data extraction from homogeneous or heterogeneous data sources, (ii) the data transformation to a proper format or structure and (iii) the data loading into the final target Data Warehouse. Considering the volume of data to be extracted from the sources, the ETL process is likely to cause performance problems. A Data Staging Area (DSA) is included in the architectural design in order to solve this kind of issues. A Data Staging Area is a temporary storage area between the data sources and the data targets. It is mainly used to increase efficiency of ETL processes, ensure data integrity and support data quality operations.

In the proposed architectural design, the role of the Data Staging is to receive from the ELVITEN Middleware the correlated data from all the ICT Tools and transform them to the appropriate formats to be stored in the Data Warehouse. During data staging, the format of the data is also validated, allowing the identification of those that are not compliant with the data model.

2.5.4 ELVITEN Data Warehouse

A Data Warehouse (DW) is a system used for reporting and data analysis and it is considered a core component of business intelligence. Data Warehouses are central repositories of integrated data from multiple sources. They store current and historical data and are used for creating analytical reports and computing Key Performance Indicators (KPIs). The Data Warehouse creates a layer optimized for and dedicated to analytics to ensure easier reporting and analysis, it is structured to speed up analytics (queries) and make them easy, and it is optimized for efficiently reading/retrieving large data sets and for aggregating data.

The ELVITEN Data Warehouse contains a number of fact tables for each of the information types that must be stored in the ELVITEN ICT platform. These information types include the following:

- the trip data (GPS and CAN bus data) from the EL-Vs;
- the data from the various ELVITEN ICT Tools;
- data from users’ questionnaires during the demonstrations.

As presented in the beginning of this section, the main requirements of the ELVITEN Data Warehouse, concerning the data storage, are:

1. the ability to store all relevant data in a way that allows the efficient and consistent computation of every defined KPI as described in D1.3 [4] and D2.1 [3];
2. the ability to scale to very large data sizes.

The design of the ELVITEN Data Warehouse is based on the top-down model, which means that the data are stored based on a normalized enterprise data model. The data are stored in the Data Warehouse at the greatest level of detail, but the repetition of the information is eliminated. The normalized structure divides data into entities creating several tables in a relational database.

This architecture approach has been chosen for the Data Warehouse due to a number of advantages. To begin with, this approach is straightforward as far as the addition of information into the database is concerned; facilitating the role of the Data Staging and its ability to scale to very large data sizes without lag in performance. In addition, it can easily scale horizontally to include any additional services that may be of interest to the platform.
The main disadvantage of this approach is that, because of the number of tables involved, it can be difficult for users to join data from different sources into meaningful information and to access the information without a precise understanding of the sources of data and of the data structure of the Data Warehouse. This disadvantage is not materialized here given that the tables are not heavily interconnected and have been carefully designed to facilitate the computation of the defined KPIs in an efficient and meaningful way.

Based on these and considering the data description for all data types found in the ELVITEN Data Management Plan as described in D9.3 [2], the following data model has been developed:

![Data model of the ELVITEN Data Warehouse](image)

The relational data model presented in Figure 7 consists of 14 tables to store the ICT Tool’s generated data. Each questionnaire is stored in a dedicated table allowing the efficient aggregation of the information per question as required by specific KPIs. The model is also enriched with multiple indices that support all the needed queries for the calculations of the KPIs as presented in D2.1 [3].

### 2.5.5 ELVITEN Dashboard

The ELVITEN ICT platform must make sure that all data stored in the Data Warehouse are accessible by scientists for analysis purposes. Additionally, it has to offer a visualisation of the calculated KPIs to the general public. Towards this direction, a dashboard application, namely the ELVITEN Dashboard, will be developed. In order to be easily accessible from everywhere and facilitate its integration with the Data Warehouse, the Dashboard will be available as a web application.
The Dashboard will consist of single or multiple screens with charts and tables. It will provide common data visualizations, such as bar charts, maps, line charts, scatterplots, pie charts, gauges, tables, chosen according to the specific KPIs to be presented. For simple KPIs, it will be possible to directly retrieve them from the Data Warehouse in table format or as text files.

Apart from the visualisation of KPIs, the Dashboard will incorporate tools for data extraction, which will be available only to authorised users. Using own credentials, the Scientists will be able to login to the platform in order to access dedicated tools for data analysis. In particular, they will have read access to the Data Warehouse content and they will be able to extract any set of data via the user interface. The extracted data will be formatted such as they can be easily manipulated also by external tools.
3. Configuration Management

To ensure keeping track of changes, integrate ICT Tools, and roll back to safe previous configurations of the overall ELVITEN platform, it is necessary to properly manage software configuration. Software configuration management can be divided into two equally important activities:

- **Server configuration management**, which aims at tracking all the server configuration changes. This is done essentially for two reasons: provide a base upon which to build new servers (by taking the last configuration and applying it) or provide a safe rollover point in case of server fault (the last modification/s can be rolled back to a previous safe configuration).

- **Software configuration management (SCM or S/W CM)**, that consists in tracking and controlling changes in the software. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can mark it (TAG) to ensure that a valid rollback point is defined.

Among other goals, SCM also aims at identifying configurations and their items (configuration identification), managing the process and tools used for builds (build management), and making sure every defect has traceability back to the source (defect tracking).

It is necessary to ensure that due attention is given to configuration activities, especially considering that the different ICT Platform components is realized by different partners, and that final goal is to integrate heterogeneous applications. To ensure that basic configuration activities are put in place, a version control system (VCS) has been deployed on ICCS servers.

Version control systems track changes in computer files, maintaining the complete history and full version tracking, and help to coordinate work among multiple developers. Those can be used to track changes in the source code as well as in the server configuration files. A single version can be tagged with a name for marking important milestone or release and can be branched (so that modifications can happen in parallel along both branches). The branches can be later merged to integrate changes back onto the parent branch. Other more complex tools can be added to a configuration management system to provide additional functionalities, such as continuous integration and deployment.

The version control system that is used in ELVITEN is based on Git [5]. For managing the Git repository, the Gitlab [6] software is used. In GitLab, the code is organised in projects and groups. Only authenticated users have access to the resources stored in the Git repository, by applying appropriate access rights to each resource. ELVITEN Git repository contains all the code, documentation, and test cases. GitLab advanced features can be used for continuous integration and deployment. It must be noted that ELVITEN Git repository and GitLab is not part of the ELVITEN ICT Platform. Nevertheless, this software provides the baseline for the Integration Tests preliminary to the actual start of usage of the ELVITEN ICT Platform.
4. Data Quality Assurance

This section describes the quality assurance mechanism applied on the data received from the cities. Its main purpose is to ensure that the data from the demonstration cities can be integrated and compared as required.

4.1 Validation Rules

The validation rules ensure the validity of data with regards to some constraints. These constraints correspond to conditions that data must fulfil before being stored in the Data Warehouse. Examples of constraints are:

(i) non-empty columns;
(ii) values falling within defined ranges;
(iii) right data formats;
(iv) compliance with other rules depending on the originator of the data.

Tools like JSON validators or software dedicated to data clean-up (e.g. OpenRefine [7]) are proposed to be implemented for the validation rules. These tools perform automated validation tasks including identification of rule violation and reformatting when possible. It must be noted that these rules apply not only on the data but also on the metadata.

4.2 Metadata

Metadata will be used as a reference to ensure comparability between data originated from different cities. In other terms, if data received from multiple sources follow the same metadata schemas, high level comparability is ensured. In the Data Management Plan (D9.3 [2]), a general metadata schema is described. The schema is based on the DataCite metadata schema v3.1 [8] and it is common to all collected data. Some attributes are mandatory to describe the datasets that are going to be publicly available at the end of the project.

Specialized metadata for trip data and questionnaire data are proposed. These metadata follow existing standards that are relevant to the type of collected data. For the trip data, the framework developed in the FOT-Net Data project [9] is considered to design the metadata schema. This project dealt with different types of data generated from real usage of vehicles. On Table 3, an example of metadata attributes for time history data, as extracted from the FOT-Net Data framework, is shown. It is encouraged to reuse this framework to manage data resulting from any vehicle usage experiments.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Instruction/example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>What is the accuracy of the measure?</td>
</tr>
<tr>
<td>Attribute</td>
<td>Instruction/example</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Unit</td>
<td>What is the unit of the measure (e.g., m/s, RPM or if an enumeration)?</td>
</tr>
<tr>
<td>Sample rate</td>
<td>What is the current frequency of the measure (e.g., speed resampled at 10 Hz or 1 Hz)?</td>
</tr>
<tr>
<td>Filter</td>
<td>Which filters are applied (e.g., low-pass, interpolation or outlier filters)?</td>
</tr>
<tr>
<td>Origin</td>
<td>How is the measurement generated and from what data source? For instance, it is important to know if the speed measures are originated from CAN at 20 Hz or GPS at 1 Hz. This could also refer to another described measure.</td>
</tr>
<tr>
<td>Type</td>
<td>Is the measure an integer, float, string or picture file?</td>
</tr>
<tr>
<td>Range</td>
<td>What is the expected range (minimum and maximum values) of the measure?</td>
</tr>
<tr>
<td>Error codes</td>
<td>Which values trigger error codes? What is a null value? It is also important to describe how the errors are managed.</td>
</tr>
<tr>
<td>Quality</td>
<td>Are there any quality measures related to this measure and how are they defined? The quality could be set on a per-trip, per-measure or even per-sample level (e.g., for GNSS data: HDOP, number of satellites).</td>
</tr>
<tr>
<td>Enumeration specification</td>
<td>Can enumerations be translated into readable values (e.g., 1 means left and 2 means right for the turn indicator)?</td>
</tr>
<tr>
<td>Availability</td>
<td>Can the measure be shared? What are the conditions to access it?</td>
</tr>
</tbody>
</table>

For the questionnaires data, the Data Document Initiative (DDI) international standard [10] is proposed. It is advised to select a subset of this metadata schema including the most relevant attributes for the type of surveys that will be conducted in ELVITEN.

Table 4 shows an example of selected attributes with the instructions provided within the Nesstar Publisher [11] tool, which helps in designing DDI compliant metadata schemas.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Instruction/example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Title</td>
</tr>
<tr>
<td></td>
<td>The title is the official name of the survey as it is stated on the questionnaire.</td>
</tr>
<tr>
<td>Version</td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td>- v0.1: Basic raw data, obtained from data entry (before editing).</td>
</tr>
<tr>
<td>Attribute</td>
<td>Instruction/example</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Attribute</td>
<td>Instruction/example</td>
</tr>
<tr>
<td>Overview</td>
<td>• v1.2: Edited data, second version, for internal use only.</td>
</tr>
<tr>
<td></td>
<td>• v2.1: Edited, anonymous dataset for public distribution.</td>
</tr>
<tr>
<td>Abstract</td>
<td>The abstract should provide a clear summary of the purposes, objectives and content of the survey.</td>
</tr>
<tr>
<td>Unit of analysis</td>
<td>Individuals, families/households, groups, facilities, institutions/organizations, administrative units, physical locations, etc.</td>
</tr>
<tr>
<td>Description of scope</td>
<td>The scope is a description of the themes covered by the survey.</td>
</tr>
<tr>
<td>Topic classification</td>
<td>Topics should be selected from a standard thesaurus, preferably an international, multilingual thesaurus.</td>
</tr>
<tr>
<td></td>
<td>Example: Council of European Social Science Data Archives (CESSDA)</td>
</tr>
<tr>
<td>Keywords</td>
<td>Keywords summarize the content or subject matter of the survey.</td>
</tr>
<tr>
<td>Country</td>
<td>-</td>
</tr>
<tr>
<td>Geographic coverage</td>
<td>Example: city of Genoa</td>
</tr>
<tr>
<td>Primary investigator</td>
<td>The primary investigator will in most cases be an institution but could also be an individual in the case of small-scale academic surveys.</td>
</tr>
<tr>
<td>Funding</td>
<td></td>
</tr>
<tr>
<td>Dates of collection</td>
<td>Enter the dates (at least month and year) of the start and end of the data collection.</td>
</tr>
<tr>
<td>Mode of data collection</td>
<td>Examples: face-to-face, computer assisted interview ...</td>
</tr>
<tr>
<td>Notes on data collection</td>
<td>This element is provided in order to document any specific observations, occurrences or events during data collection.</td>
</tr>
<tr>
<td>Questionnaire</td>
<td>This element is provided to describe the questionnaire(s) used for the data collection.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Instruction/example</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Data processing</strong></td>
<td></td>
</tr>
<tr>
<td>Data editing</td>
<td>The data editing should contain information on how the data was treated or controlled for in terms of consistency and coherence.</td>
</tr>
<tr>
<td>Other processing</td>
<td>Use this field to provide as much information as possible on the data entry design.</td>
</tr>
<tr>
<td><strong>Data appraisal</strong></td>
<td></td>
</tr>
<tr>
<td>Other forms of data appraisal</td>
<td>This section can be used to report any other action taken to assess the reliability of the data, or any observations regarding data quality.</td>
</tr>
<tr>
<td><strong>Data access</strong></td>
<td></td>
</tr>
<tr>
<td>Confidentiality</td>
<td>-</td>
</tr>
<tr>
<td><strong>Contacts</strong></td>
<td></td>
</tr>
<tr>
<td>Contact persons</td>
<td>-</td>
</tr>
</tbody>
</table>

One of the tasks of checking the quality of the collected data is ensuring that attributes of the different metadata schemas are correctly filled out. The previously cited validation tools ensure the correctness of the metadata and point out any invalid entry. Data with incomplete metadata should not be inserted into the Data Warehouse. In fact, lack of metadata is particularly disadvantageous for data analysis and can even make data useless. For instance, statistical analyses are particularly enhanced with good metadata documentation [12]. The data quality assurance process takes place within the Data Staging module.
5. Data privacy

5.1 ELVITEN Identity Manager

The ELVITEN Identity Management platform only stores the unique user ID, as described in the deliverable D9.3 [2]. In addition, no identity data is accessible from the ELVITEN modules thus the risk on privacy at the Identity Manager can be considered as low. Nonetheless, Table 5 summarizes privacy concerns and countermeasures usually implemented when using OpenID connect (i.e. the technology the Identity Manager is based on, which is presented in detail in section 6).

<table>
<thead>
<tr>
<th>Privacy concerns</th>
<th>Possible Countermeasures</th>
<th>Applicable to ELVITEN</th>
</tr>
</thead>
</table>
| **Personal Identifiable Information contained in UserInfo** | - Obtain End-User consent about its information release  
- Register the purpose of the usage of the information  
- Client stores only the necessary UserInfo  
- Client keeps associations between the user data and the purpose of usage | - Yes  
- Yes  
- Yes  
- Yes |
| **Data access Monitoring** | - The End-Users have access to the logs of the Resource server and can monitor the applications that accessed their data | - Yes |
| **Correlation** | - The usage of Pairwise Pseudonymous Identifier (PPID) as the sub (subject) is considered to prevent the Clients from correlating the End-Users | - No |
| **Offline Access** | - No offline or Explicit consent for offline mode  
- The server shall Detect the issuing point of the Access Token (User Agent or Token End-Point) and deny access | - No  
- No |

Confidentiality and integrity are the two security principles that are enforced at the identity provider servers in order to protect the user identity from malicious actions in case of data breach. In addition, high availability of the identity provider is assured as it is a central component in the ELVITEN global architecture. Finally, the encrypted user database will be regularly backed up. The implementation section of this deliverable gives more details about the mechanisms for achieving these principles.
5.2 ELVITEN Middleware

The ELVITEN Middleware will not apply any processing on the data originating from the ICT Tools. As its sole purpose is to route the messages, it assumes that appropriate privacy preservation mechanisms have been implemented before the messages are sent. In addition, as in the case of the identity provider server, security mechanisms shall be enforced at the Middleware to guarantee the confidentiality and the integrity of the message passing. The adopted security techniques are presented in section 6.

The ELVITEN ICT Tools produce data before sending them through the Middleware to the Data Warehouse. These data contain the ELVITEN User ID and other data that are necessary to the process of the tool. Therefore, at the level of the ICT Tool, the privacy preservation mechanism implemented is a pseudonymisation by the usage of the ELVITEN User ID.

5.3 ELVITEN Data Warehouse

All the data that will be stored in the Data Warehouse will be pseudonymised. The User ID will be used in the calculation of the KPIs related to the user behaviour with regards to the EL-Vs but will not be shown in the Dashboard, thus protecting the privacy of the user. The data will be received through the abovementioned secure communication methods adopted by the ELVITEN Middleware.
6. Implementation

The data flow within the ICT platform determines the dependencies between the separate components. The development procedure, however, can be performed in parallel for each one of them as soon as the interfaces between them are defined.

6.1 ELVITEN Middleware

The implementation of the ELVITEN Middleware is based on RabbitMQ [13], an open source message broker, which is lightweight and can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements.

**Message Bus**

The Message Bus has the objective of sending the data from the ICT Tools to the Data Warehouse. Since the architecture is distributed, the need of a message bus that acts as a broker for delivering the messages arises.

![Figure 8: The General architecture of RabbitMQ adapted to ELVITEN](image)

The general architecture of RabbitMQ, regarding the communications with the ICT Tools and the Data Warehouse, is the following: The Publishers send messages to an Exchange, which addresses them to the right Queue, from which the Consumers extract the messages. Currently there is only one Consumer, which takes all the messages, thus only one queue is needed. Nevertheless, as RabbitMQ supports more than one Queue, there can be more than one Consumer, ensuring in this way the extendibility of the ELVITEN ICT Platform.

Figure 8 shows the communication architecture. The Consumer part of the broker is located in the same place as the Data Warehouse and it is implemented in the Data Staging part. The Publisher part of the
broker is distributed to the ICT Tools’ providers as a software library, in order to allow them to send messages in a JSON format to the Exchange.

The Middleware supports many more communication types. An example of communication with the Incentive System is showed in Figure 9.

![Figure 9: Communication use case between the ELVITEN Middleware and the Incentives System](image)

When the Incentive System needs data from one ICT asset, the request is sent to an Exchange, which routes it to the correct receiver. Every ICT asset has its own message queue, where the request messages are stored. When the ICT Asset consumes the message from its queue, the response is sent back and stored into the Incentive System’s own queue, waiting to be consumed.

**Caching**

One of the advantages RabbitMQ is the Lazy Queues feature. Lazy Queues are queues that move their contents to disk as early as practically possible, and only load them in RAM when requested by consumers, therefore the lazy denomination. Such implementation can be useful when consumers are offline or slower than normal.

**Orchestrator**

A most common solution for communication of several applications is represented by the utilization of an Orchestrator. In ELVITEN, the Orchestrator will be a central business process system that will handle the communications among the different ICT Tools in a centralized way.
RabbitMQ does not execute the single operations between the applications, but it provides several different configurations of message exchange, such as multi-publisher, multi-consumer schemas, enabling a central communication system.

**Security**

RabbitMQ manages Access Control through its exchanges, queues and channels by the usage of virtual hosts: certain user can access certain virtual hosts only with beforehand permissions. Every different operation (configures, write, read) needs permission as well. RabbitMQ also supports TLS, in order to establish an encrypted communication channel.

### 6.2 ELVITEN Identity Manager

The authentication/authorisation system selected for the ELVITEN user’s authentication is based on OpenID Connect. OpenID Connect 1.0 [14] is an identity layer on top of the OAuth 2.0 protocol [15]. It enables clients (each ELVITEN ICT Tool acts as a client) to verify the identity of the end-user based on the authentication performed by an Authorisation Server, as well as to obtain basic profile information about the end-user in an interoperable and REST-like manner. OAuth 2.0 Authentication Servers implementing OpenID Connect are also referred to as OpenID Providers (OPs) and OAuth 2.0 Clients using OpenID Connect are referred to as Relying Parties (RPs). Information about the authentication performed is returned in a JSON Web Token (JWT), called ID Token. The ID Token should not be confused with the ELVITEN ID.

The OpenID Connect framework includes the following steps:

1. An ELVITEN User starts using an ICT Tool referred as “Application” and asks for a protected resource (step 1).
2. The ELVITEN Application, or Relying Party (RP) on OpenID’s terms, prepares an authorisation request and sends it to the authorisation endpoint in Open ID Provider Server (step 2).
3. The authorisation endpoints examine the authorisation request and authenticate the end user if he is not already authenticated. The Authorisation Server sends an authentication page to the Application that is displayed to the ELVITEN User afterwards in order to enter his credentials (steps 3, 4, 5).
4. If the credentials provided by the user are correct, the authorisation endpoint sends an authorisation code to the client (step 6) that will be sent afterwards to the Token Endpoint for verification (step 7) and granting an ID Token and an Access Token to the Application (step 8).

5. Once these tokens are given, the access for the protected resources is granted to the ELVITEN User.

6. If after authentication and authorisation, the Application asks for certain resources from the UserInfo Endpoint, the Application will present the ID Token to the UserInfo endpoint asking for the resource (step A).

7. The UserInfo endpoint asks the Token Endpoint token verification (step B). After ensuring the validity (step C), the resources will be given to the client (step D).

The above steps are illustrated in Figure 11. More details regarding OAuth2.0, OpenID connect and the ID Token are presented in Annex 1.
Figure 11: OpenID Connect workflow
Different implementations of OpenID Connect are tested. The local testing environment at AKKA includes PHPOIDC [29], PingIdentity [30], Keycloak [16], and Gluu Server [31].

Keycloak [16] is an open source Identity and Access Management solution aimed at modern applications and services, providing SSO technology and enabling login with social networks. It has built-in support to connect to existing LDAP, Kerberos or Active Directory servers and provides Client Adapters that makes it easy to secure applications and services. Adapters are available for a number of platforms and programming languages. Keycloak has been tested using a JEE Sample application that expose three services to the users “Public, Secured and Admin”. In order to have access, the user needs to authenticate himself by clicking on the “Logging” Button that transfers him to a Keycloak interface asking for username and password. After providing correct credentials, the user will be able to access the services depending on his role.

Further details about Keycloak and clients configurations are presented in Annex 2.

6.3 ELVITEN Data Staging, Data Warehouse and Dashboard

From the implementation point of view, both the Data Staging and the Dashboard need to be tightly coupled with the Data Warehouse in order to achieve the desired performance of their individual functionalities. More specifically, they should be implemented in a way that enables them to have direct access to the Data Warehouse, ideally without any intermediate interface.

However, all three components have to offer interfaces for communicating with their dependent components. To avoid this, the design pattern of Model-View-Controller (MVC) has been adopted to link the Data Staging, the Data Warehouse and the Dashboard together and provide interfaces for communication with all other components.

In the MVC pattern, an application is divided into the following three interconnected components:

- the model, which represents the knowledge. A model could be everything from a single object to a large structure of objects.
- the view, which is a visual representation of the model.
- the controller, which controls the flow of information between the model and the view.

In our case, the model represents the structure and is responsible for managing the data in the Data Warehouse. The view is the user interface of the Dashboard, which is responsible to visualise the model in a particular format. Finally, the controller handles all communications and interactions among the Data Staging, the Data Warehouse and the Dashboard. It additionally offers the possibility to expose a Web API for communication with external components. The proposed architecture is showed in Figure 12. It is clear that each one of the three ELVITEN component interacts with only one of the MVC components.
The data model is derived directly from what has been already presented in Section 2.5.4. It was implemented in the Data Warehouse in a relational database that allows the definition of clear roles with appropriate rights, allowing the Data Staging to store information and the Dashboard to query the data. Given the purpose and role of the Data Warehouse, and the importance of the historical data to the computation of the KPIs no deletion of data is allowed.

The Data Staging (using the controller) has two main responsibilities:

1. To link the ELVITEN Middleware with the Data Warehouse.
2. To validate and convert the received data to conform to the requirements of the Data Warehouse.

In order to receive messages from the Middleware, it implements a subscriber to act as a message consumer. When the consumer receives the message, it validates it, by checking for violations on used data types and ranges, data structure or duplicate values etc. If an error occurs, it is corrected and converted accordingly and then it is forwarded to the Data Warehouse for storage purposes.

Finally, the communication between the Dashboard and the Data Warehouse is handled by the selected MVC framework. The Dashboard is accessed through a web browser and forwards all user actions to the controller, which communicates with the Data Warehouse and updates the view accordingly.

**Data Staging Technologies**

Django [19] has been selected as the underlying framework to enable the functionalities and communications presented above. Therefore, the Data Staging is being implemented in Python programming language. The data extraction part, which is the consumer part of the RabbitMQ architecture, uses the Pika library [20], a pure-Python implementation of the AMQP 0-9-1 protocol, which is the recommended solution by the RabbitMQ team [21]. To support the data validation part, the Django framework has been enhanced with the Django REST framework [22], which offers the ability to define custom data serializers. Data validation is based on JSON schema and on serializers’ validation...
functionality. Upon validation, data are deserialized and stored directly in the Data Warehouse using the Django mechanisms.

Data Warehouse Technologies

The Data Warehouse is deployed as a PostgreSQL[23] database. The structured, clearly defined and unmodifiable data model led to the decision of using a relational database for the data storage.

PostgreSQL offers full support for all the functionalities of the Structured Query Language (SQL), geometric data types including a point of floating numbers necessary for the storage of the GPS coordinates as well as a wide range of indices that can be used to support the computation of the KPIs.

PostgreSQL supports GiST indexes which are not a single kind of index, but rather an infrastructure where many different indexing strategies can be implemented. Accordingly, the operators with which a GiST index can be used vary depending on the indexing strategy (the operator class). This is very important as it allows spatial queries over several two-dimensional geometric data types. In addition, it offers GIN indices which are appropriate for data values that contain multiple component values, such as arrays. This allows the flexibility for multiple choice answers. Multicolumn indices and indices on expressions will also be used to support the computation of the KPIs when needed.

The data stored in the Data Warehouse have to be accessible not only from the Dashboard but also from external parties. Since the ELVITEN data should be Findable, Accessible, Interoperable and Re-usable (FAIR) a suitable API should be offered. For this purpose, a RESTful API has been selected and implemented for its simplicity, performance, scalability, and modifiability. REST stands for Representational State Transfer. It relies on a stateless, client-server, cacheable communications protocol and, in virtually all cases, HTTP (Hypertext Transfer Protocol) is used. In a REST API, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs). Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. Any data marked as cacheable may be reused as the response to the same subsequent request.

Since the ELVITEN data should not be modified or deleted, only GET operation is supported by the Data Warehouse API. JSON has been the selected format for the returned data. A preliminary version of the exposed API has been created using the Swagger [24] tool and is presented in Annex 3.

Dashboard Technologies

The ELVITEN Dashboard will be a web application that accesses the data using the abovementioned RESTful API and it will be using common web technologies, i.e. HTML, CSS and JavaScript. For the KPI visualisation purposes, JavaScript libraries such as Chart.js [25] and Plotly [26] will be used.
7. Deployment

The ELVITEN ICT platform is a distributed system, consequently, its individual components are deployed on separate machines which communicate to each other using the network.

On the one hand, the data adapters will be distributed and reside to the corresponding ELVITEN ICT Tool, in order to enable messages transmission to the ELVITEN Middleware. The various copies of the message publishers will be delivered to all interested partners, along with the appropriate credentials to access the exchange. In order to activate the API and send the JSON messages, the Data Adaptor part needs to be created by the partner that is responsible for the relevant ELVITEN ICT tool.

On the other hand, the components of the ELVITEN ICT Platform, namely the ELVITEN Middleware, the Data Staging, the Data Warehouse and the ELVITEN Dashboard, will be deployed on ICCS’ private cloud infrastructure. The Identity Manager will be deployed on a different hardware than the one mentioned above, having a different administrator in order to be compliant with privacy guidelines imposed by GDPR.

Some of the deployment techniques that are considered are presented below:

- In order to assure the high availability of the provided services, a cluster of servers is considered for each ELVITEN ICT Platform component.
- Usage of load balancers is being investigated in order to reduce latency and to ensure a fault-tolerant configuration. For example, NGINX web server can be used as a load balancer and reverse proxy. It must be noted that the load on the majority of the ELVITEN ICT Platform components is expected to be small.
- Multiple instances of components that are expected to have workload can be deployed. The instances can perform their work in parallel and independently from each other. For instance, this mechanism can be applied in Data Staging in order for the data manipulation (i.e. consuming, processing and validating data coming from the Middleware) and the storage to be performed faster and the fault tolerance of the system to be increased.
- Floating IPs is also being considered for fault-tolerance. In this case, the users access the services using a specific IP, pointing by default to the primary server. If a fault is detected (by a heartbeat service) and the primary server becomes unavailable, the secondary server automatically runs a script to reassign the Floating IP to it. Thus, subsequent network traffic to the Floating IP is directed to the secondary server, which acts as the active server until the primary server becomes available again. When the primary server becomes available again, the primary server reassigns the Floating IP to itself, and the secondary server reassigns the alternative IP to itself.
- In components that incorporate a database (such as the Data Warehouse or the Identity Manager), it must be ensured that the stored data is always available and cannot become unusable, corrupted or lost. For this purpose, the underlying databases must be replicated on other machines. To deal with data synchronisation, a synchronous solution is being examined: a data-modifying transaction is not considered committed until all servers have committed the transaction. This guarantees consistency after a failure.
• Applying quality of services, such as durability, is also considered. For example, in order to avoid losing messages in the broker we need to cope with broker restarts, broker hardware failure and in extremis even broker crashes. To ensure that messages and broker definitions survive restarts, we need to ensure that they are on disk. The AMQP standard has a concept of durability for exchanges, queues and of persistent messages, requiring that a durable object or persistent message survives a restart.

• Integrity of data is ensured by using TLS in all relaying parties and resources servers. Additionally, integrity of data is ensured by encryption technics (e.g. hashed passwords in Identity Manager) and regular backups.

• Installation of firewalls in every server of ELVITEN is mandatory, as the majority of this servers are exposed to the internet. Firewalls enables the blocking of unwanted access to your server without disrupting its normal operation. MAC and IP filtering is being considered for machine to machine communication. As most of the servers are based on Linux operating systems either IP table firewalls or advanced policy firewall are considered.

• To ensure that the broker survives hardware failure, RabbitMQ's clustering is used. In a RabbitMQ cluster, all definitions (of exchanges, bindings, users, etc) are mirrored across the entire cluster. Queues behave differently, by default residing only on a single node, but optionally being mirrored across several or all nodes. Queues remain visible and reachable from all nodes regardless of where they are located. Mirrored queues replicate their contents across all configured cluster nodes, tolerating node failures seamlessly and without message loss.

• For the purposes of ELVITEN, ICCS provides resources from two of its servers. The characteristics of these servers are: a) 40 Cores, 512 GB RAM, 10 TB Storage, 2x1GB Ethernet and b) 32 Cores, 512 GB RAM, 10 TB Storage, 2x1GB Ethernet. The servers are part of ICCS’ private cloud infrastructure.

• The first of the aforementioned servers will be managed by Proxmox VE [32]. Proxmox VE is an open-source server virtualization platform to manage two virtualization technologies a) KVM [33] (Kernel-based Virtual Machine) for virtual machines and b) LXC for containers [34], with a single web-based interface. It also integrates out-of-the-box tools for configuring high availability between servers, software-defined storage, networking, and disaster recovery.

• Micro-service architecture is realized by using Dockers or Linux containers (LXC).

The aim of ELVITEN is to collect information from at least 84,000 trips and 26,000 questionnaires. In the above, one should add data for the booking, the charging points, the incentive platform and the gaming ICT Tools, resulting in a rough estimation of one million data objects to be stored. Given the average size of the corresponding JSON objects the total volume of data, along with the indices needed, is estimated to be a few tens of GBs.
8. Conclusions

The present deliverable presents the functional architecture of the ELVITEN ICT Platform. The functional elements are derived from a) the usage schemes from D1.2 “ELVITEN usage schemes and functional requirements” and b) the system requirements in terms of functionality derived from T3.1 “ICT requirements analysis and assets inventory”. The analysis of the information flow among the main sub-systems has facilitated the extraction of the functional requirements.

Server and software configuration management techniques are going to be followed in ELVITEN, in order to systematically handle changes to the system in a way integrity over time is maintained. Specific quality assurance mechanisms, such as validation rules and use of metadata, are going to be applied on the data derived from the cities. Special attention is paid on the privacy of user data, by following the privacy guidelines imposed by GDPR. In this context, the Identity Manager, is deployed on a separate hardware from the other sub systems, having a different administrator.

The ELVITEN technical partners that are involved in the development of the ICT platform will follow a concrete implementation strategy. The identification of the inter-dependencies between the different components has led to the definition of the corresponding interfaces. Such approach has enabled the development of each functional block in parallel.

ELVITEN is based on a distributed deployment model where components are geographically dispersed with functionalities being exposed through proper web service interfaces. Furthermore, the threats that a system such as ELVITEN is facing do not differ greatly from other ITS in general. Toward this direction, several deployment strategies will be applied during the pilot demonstrations phase, in order to ensure high availability and secure communications. Sensitive data like EV location will be stored, handled and transmitted securely throughout the system. Additionally, system availability and data reliability will be ensured by avoiding attacks, aimed at the infrastructure, which can compromise the security.

By exploiting already available knowledge and ICT development, adapting and extending it where necessary ELVITEN aims to have contributed to the definition of a standardized ICT architecture for electromobility.
References

[1] ELVITEN Deliverable 1.2, ELVITEN Usage Schemes and Functional Requirements, June 2018
[3] ELVITEN Deliverable 2.1, ELVITEN Demonstration methodology and baseline scenarios, July 2018
[4] ELVITEN Deliverable 1.3, ELVITEN Study Questions and KPIs, April 2018
[17] https://searchsoftwarequality.techtarget.com/definition/authorization
[29] https://bitbucket.org/PEOFIAMP/phpoidc
[31] https://www.gluu.org/
[32] https://www.proxmox.com/
[33] https://www.linux-kvm.org/
[34] https://linuxcontainers.org/
Annex 1: OAuth 2.0 and OpenID Connect

OAuth 2.0

In this section, definitions taken from the RFC6749 (OAUTH 2.0) [27] are given:

- **Authorization** is the process of granting or denying access to a resource after authenticating the End User asking for the resource, which ensures that a user is who he or she claims to be.
- **OAuth 2.0** is an authorization framework, which allows a third-party application to access resources on an HTTP service by obtaining the approval of the owner of these resources or on its own behalf. Therefore, the third-party application does not need to store any credentials (username and password).
- **The resource owner** is an entity capable of granting access to a protected resource.
- **The resource server** is the server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.
- **The Client** is an application making protected resource requests on behalf of the resource owner and with its authorization.
- **The authorization server** is the server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

OpenID Connect

In this section definitions taken from OpenID Connect Specification are provided:

- **Identity Provisioning** is the process of creating, updating, retrieving and deleting user’s digital identities, which carry information such as username, password, email, address, phone number, roles, groups, etc. about registered user.
- **Authentication** is the process of identifying an individual, entity or website based, for example, on a username and password, a client id and client secret or biometric data (fingerprint, iris scan, face recognition, etc.). Authentication ensures that the individual is who he or she claims to be but says nothing about the access rights of the individual.
- **Single Sign On (SSO)** is an authentication process in a client/server relationship where the user, or client, can authenticate once (with only one user/password pair for example) and have access to more than one application or access to a number of resources within an enterprise or the whole web. Single Sign On takes away the need for the user to enter further authentications when switching from one application to another.
- **A Claim** is a piece of information asserted about an Entity. It is return by the Authentication/Authorization server. The Claim can have the form of the ID Token.

The ID Token

The primary extension that OpenID Connect makes to OAuth 2.0 to enable end-users to be Authenticated is the ID Token data structure. The ID Token is a security token that contains claims
about the authentication of an end-user by an Authorisation Server when using a client, and potentially other requested claims. The features of the ID Token are presented in Table 6.

**Table 6: ID Token features**

<table>
<thead>
<tr>
<th>Sub</th>
<th>REQUIRED. Subject Identifier. A locally unique and never reassigned identifier within the Issuer for the End-User, which is intended to be consumed by the Client. It must not exceed 255 ASCII characters in length. The sub value is a case sensitive string.</th>
<th>In ELVITEN, the Sub will be automatically generated by the Identity Manager deployed on ICCS server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iss</td>
<td>REQUIRED. Issuer Identifier for the Issuer of the response. The Iss value is a case sensitive URL using the HTTPS scheme that contains scheme, host, and optionally, port number and path components and no query or fragment components.</td>
<td>In ELVITEN, the Issuer is the URL of the Identity Manager.</td>
</tr>
<tr>
<td>Aud</td>
<td>REQUIRED. Audience(s) that this ID Token is intended for. It must contain the OAuth 2.0 client_id of the Relying Party as an audience value. It may also contain identifiers for other audiences. In the general case, the Aud value is an array of case sensitive strings. In the common special case when there is one audience, the Aud value may be a single case sensitive string.</td>
<td>In ELVITEN, the audiences are the ICT Tools.</td>
</tr>
<tr>
<td>Nonce</td>
<td>String value used to associate a Client session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authentication Request to the ID Token.</td>
<td>In ELVITEN, this value will be generated automatically by the Identity Manager.</td>
</tr>
<tr>
<td>Iat</td>
<td>REQUIRED. Time at which the JWT was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.</td>
<td>In ELVITEN, this value will be generated automatically by the Identity Manager.</td>
</tr>
<tr>
<td>Exp</td>
<td>REQUIRED.Expiration time on or after which the ID Token must not be accepted for processing.</td>
<td>In ELVITEN, this value will be generated automatically by the Identity Manager.</td>
</tr>
</tbody>
</table>

The ID Token may contain other claims and it can both be encrypted using JSON Web Encryption (JWE) and digitally signed using JSON Web Signature (JWS), so it can be verified by the intended recipients and will be encrypted for confidentiality.
Global view of OAuth 2.0

Figure 13 illustrates the underlying steps of the OAuth 2.0 framework.

![Diagram of OAuth 2.0](image)

Figure 13 Illustration of OAuth 2.0 in ELVITEN project

First step

The ELVITEN user launches the ELVITEN application for booking an EL-V for example.

Authorization request

The ELVITEN application (the client) requests authorization from the ELVITEN user (the resource owner). The client has two ways to ask the ELVITEN user this authorization. In ELVITEN, the client uses the authorization server as an intermediary to request the authorization.

Authorization grant

The client receives an authorization grant, which is a credential representing the resource owner’s authorization expressed as one of the four types:

- authorization code
- implicit
- resource owner password credentials,
- client credentials
- as well as an extensibility mechanism for defining additional types

More details about these authorization grants can be found in the RFC6749. In the ELVITEN project, the authorization code is the type of authorization grant that is used.
The authorization code is obtained by using an authorization server as an intermediary between the client and the resource owner. The client directs the resource owner to an authorization server (via its user-agent as defined in RFC7235 [28]. The authorization server will:

- Authenticates the ELVITEN user using the ELVITEN ID and obtains authorization
- Then directs the resource owner back to the client with the authorization code.

**Access Token Request**

The client requests an access token by authenticating with the authorization server and presenting the authorization grant. An Access Token is a string generated by the authorization server to denote an authorization granted to a client. It contains credentials sent to the resource server to determine the authorization. The Access Token is validated by the resource server and is opaque to the client.

The security requirements of the resource server conditioned the format, the structure and the methods of utilization of the Access Token. For the ELVITEN project, each resource server’s security requirements are taken into consideration and the appropriate Access Token are derived.

**Access Token Response**

The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token.

**Restricted resource Request**

The client requests the protected resource from the resource server and authenticates by presenting the access token.

**Restricted resource Response**

The resource server validates the access token, and if valid, serves the request.

The previous steps of OAuth 2.0 have been refined by OpenID connect. These refinements are adopted by the ELVITEN Identity Manager and are presented in section 6.
Annex 2: Keycloak and clients’ configurations

After navigating to the OpenID Connect provider’s URL and choosing to be linked to the console admin, an authentication page is shown.

![Keycloak’s authentication page](image)

Figure 14: Keycloak’s authentication page

After entering correct credentials, Keycloak’s main page is accessed, where new users and roles can be added by the administrator.

![Keycloak - Main page](image)

Figure 15: Keycloak - Main page

![Keycloak – Add role](image)

Figure 16: Keycloak – Add role
Figure 17: Keycloak – Add user

Figure 18: Keycloak – User list

Figure 19: Keycloak – ELVITEN User info
Users can then be assigned to roles, as shown in the figure below.

![Role Assignment](image)

**Figure 20: Keycloak – Assign user to role**

After deploying a sample application, the Keycloak’s administrator register the application as a client in the OP.

![Client Registration](image)

**Figure 21: Keycloak – Client’s registration**
After registering the client, a JSON file will be downloaded and put in the application /config directory.

The client is registered now and configured in order to use Keycloak as an OP. After opening the application, nothing is prompted before performing a successful authentication request. To do so the user needs to push the log button and enter his credentials in Keycloak’s interface. After that the user is able to call the resource assigned to his role.
Annex 3: Data Warehouse API

The preliminary version of the Data Warehouse API is presented in Figure 24.

![Data Warehouse API Diagram]

**Figure 24: Data Warehouse API**
As an example, a detailed view of the trip interface is illustrated in Figure 25.

Figure 25: Example request for trips