D5.4-Documentation (auto parts demonstrator)

Project acronym:	**ReCiPSS**
Project full title:	**Resource-efficient Circular Product-Service Systems — ReCiPSS**
Grant agreement no.:	**776577-2**
Author/s:	**Dominik Kuntz, Catalin Petrea, Adrian Dragomir, Cristina Prunaru**
Reviewed:	**Perttu Korpela, Ruud de Bruijckere, Niloufar Salehi, Alena Klapalova**
Approved:	**<list>**
Document Reference:	**D5.4**
Dissemination Level:	**PU**
Version:	**2.1 DRAFT**
Date:	**<29.05.2020>**

This is a draft document and subject to approval for final version. Therefore the information contained herein may change.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

History of Changes

Version	*Date*	*Modification reason*	*Modified by*
0.1	20.02.2020	Initial Draft	Dana Oniga
0.2	21.04.2020	Technical chapters	Cristina Prunaru, Adrian Dragomir
0.3	08.05.2020	Completion	Dominik Kuntz
0.4	08.05.2020	Completion in Chapter 6	Dana Oniga, Catalin Petrea
0.5	09.05.2020	Peer review	Perttu Korpela
0.6	11.05.2020	Peer review	Ruud de Bruijckere
0.7	18.05.2020	Peer review	Niloufar Salehi
0.8	22.05.2020	Peer review	Alena Klapalova
0.9	29.05.2020	Final draft review	SIMAVI
0.10	dd.mm.yyyy	Quality check	Author Name
1.0	dd.mm.yyyy	Final reviewed deliverable	Author Name

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

List of figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

List of abbreviations

Abbreviation	*Explanation*
API	Application Programming Interface
CE	Circular Economy
CPU	Central Processing Unit
D5.2	Deliverable 5.2
D5.4	Deliverable 5.4
EC	European Commission
ERP	Enterprise Resource Planning
ICT	Information and Communications Technology
IoT	Internet of Thing
IT	Information Technology
JWT	JSON web token
M7,18,24	Month 7, 18 or 24
MS Azure	Microsoft Azure
MS6	Milestone 6
OE	Original Equipment
OS	Operating System
PbD	Privacy by Design
TL	Trade Level
T5.1	Task 5.1
T5.2	Task 5.2
T5.3	Task 5.3
T5.4	Task 5.4
UI	User Interface
WP	Work package

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

1. Executive Summary

One of the most important work packages in ReCiPSS project is WP5- ICT platforms for Multiple Lifecycles, which creates an IoT platform for white goods (T5.1) and a part data management platform (T5.2). Both platforms will serve as a foundation for WP6 – White Goods Demonstrator and WP7-Automotive Parts Demonstrator.

This report is a result of task T5.4 -Integrating, testing and documenting the part data management platform for the automotive parts demonstrator. In the same time, this document is closely linked with D5.2 - Automotive part data exchange platform and with Task 5.2: Implementing a part data management platform including standard data exchange protocols ensuring transparency and efficiency of trade networks and reverse logistics, documenting all software development processes that were necessary to create, integrate and test the automotive part data management platform since the starting of WP5 in December 2018 –M7 of the project.

While D5.2 was focusing more on functionalities developed by M18- November 2019, this deliverable will complete the whole picture of the automotive part data management platform.

The part data management platform was build upon the existing system by C-ECO's service brand CoremanNet, which includes IT-components supporting the physical-processes or customer-interaction (e.g. core-evaluation-system or customer-data and reports, material-master-data).

The newly developed ReCiPSS-platform focuses solely on all functions needed to incorporate and manage return-options. Main functionalities are detailed in Chapter 6, alongside with technologies used during the development process:

- Building development infrastructure in Microsoft Azure Cloud,
- Creating front- and backend of ICT-platform for return-options,
- Defining interfaces and connect to existing systems environment, company management features and reporting features
- Create a concept for “Underlying-Logic” as an integral feature needed for defining an option/warrant in a generic way.

Business Requirements in the context of applying the Circular Economy Concept that were the foundation of the development process are explained in Chapter 3, completed with Key Features and Overall Architecture of the Platform. A brief description of concepts used within the platform was considered necessary to better understand the way the new ReCiPSS platform works.

Chapter 4 presents three main Use Cases illustrating how the platform answers business requirements and clients' needs. Chapter 5 describes the new platform IT architecture, while the main functionalities are detailed in Chapter 6.

Chapter 7 is dedicated to the testing and validation of the actual version of the Automotive Part Data Management platform.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

2. Introduction

Deliverable D5.4 – Documentation (auto parts demonstrator) is the result of task T5.4 - Integrating, testing and documenting the part data management platform for the automotive parts demonstrator, within WP5 - ICT Platforms for Multiple Life cycles.

This document is closely linked with the report D5.2 - Automotive part data exchange platform, that presented the results obtained in T5.2 - Implementing a part data management platform including standard data exchange protocols ensuring transparency and efficiency of trade networks and reverse logistics by M18- November 2019. D5.4 is a continuation of D5.2 and shows all stages of the development process of the Automotive Part Data platform between M7- M24, when the platform is ready to be deployed.

The part data management platform for the automotive parts developed in WP5 will be deployed in a production environment accessible and will be the foundation for WP7 - Automotive Parts Demonstrator.

2.1. Document Scope

The scope of this document is the description and testing of the Part Data Management platform for the Automotive Demonstrator as at M24 of the ReCiPSS project, in order to meet Milestone MS6 - ICT platforms ready, M24 - May 2020.

2.2. Document Structure

The document is structured in eight chapters, starting with the Executive Summary, which briefly presents the whole document. Introduction explains the Scope and the Structure of the deliverable. Chapter 3 presents the business requirements to which the new functionalities developed in ReCiPSS project answer, continued by Chapter 4, which illustrates the end users' perspective in three Use Cases.

Chapter 5 presents the interaction at high level between the new created components of ReCiPSS platform. Chapter 6 describes in detail the development process for each new component. Chapter 7 describes the testing activities done so far.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

3. Business requirements and use cases

The business background and the concept of introducing the options were already described in the deliverable D5.2. The platforms will, therefore, enable the options creation and options transfer in a way that those business processes are not attached to physical artefacts. The clearinghouse will manage those digital instances to incorporate the options concept. By the combination of that, circular economy can go from specific bilateral agreements to an open standard for core-return options.

The platform will be open for all stakeholders involved to enable or intensify business between actors that could not unfold to its full extent, partly due to missing artefacts that are now being originated by these options.

3.1. Use case 1 - Wholesaler challenge - Intransparency & Complexity

The wholesaler finds himself in the middle of the trade channels in between the remanufacturers and his clients. This gives him the largest complexity as he needs to trace financial flow, remanufactured parts and core flow in both directions. But the complexity does not only arise from multitude of business relationships; the fact that the different reman suppliers all use different procedures and conditions makes it even more complicated.

The differences are:

- diverse logistic rules as minimum core return quantity or weight,
- deviating technical criteria when evaluating a core if it is still acceptable or not
- a large variety in feedback after a remanufacturer has received cores and has to pay out the surcharge.

Especially the last-mentioned point results in intransparency because without the proper feedback of the supplier, the wholesaler is not able to synchronize his purchases with his rights to return the corresponding used parts. If there is no alignment with the supplier about what has been accepted, what has been rejected, why something has been rejected, or what I have been refunded for, then the whole business of remanufacturing is at stake, at least for the wholesaler. It is not only about feeling this complexity and intransparency, in the end, it is about losing real money.

Those issues were made clearly visible in one of the co-creation sessions with a wholesaler. The core outflow was compared to a black hole, because the wholesaler in this case has no information system for reverse logistics. Even though this might not be the case for some remanufacturers that do a good job with transparent feedback, the situation for the wholesaler needs to be resolved in a holistic approach.

On the ReCiPSS platform, the wholesaler can see his full portfolio of options. Across all suppliers, he can see the details of each option consist of underlying, option value, expiration date, and

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

option writer. This gives the wholesaler the full visibility to know how many options he has available at which supplier. In this setting, the wholesaler will be independent from his suppliers reporting, especially in combination with the possibility also to process the exercise of the options via the clearing house.

3.2. Use case 2 - Wholesaler challenge - Avoid losing options value

The newly gained transparency does not end in itself. It is the basis to take action, to avoid losing options value or in the best case to even make more money out of the core business. As every option will expire if it is not exercised in time, the size of that risk gets tangible with the help of the ReCiPSS platform. As said before, the fact to be aware of that now is not only for the sake of transparency, but it offers a tool to the wholesaler to work with those options.

If the cores are not available in time, the full options-value is lost. The wholesaler can try to realize at least a part of that value by offering to sell such options if it is probable to not have the cores available until the expiry of the options. Someone else could have the corresponding used parts available and could make use of those options and exercise the full option value at the option writer. It is up to the two parties of selling and buying the option to determine the market value of the option with which the option can then change hands from the initial holder to the new holder of the option. Not only the core-brokers seem to be a potential stakeholder for that business case but also other wholesalers having more cores available than options. The reason for this could be expired options, cross-channel purchases where the cores don´t go back the same way, or simply the fact that also used parts outside of the reman program of the remanufactures could correspond to an underlying of an option deriving from the reman program.

Consequently, for a wholesaler, this can go into two directions: either having cores options that cannot be exercised in time or having cores without the corresponding options. Optimizations in both ways can certainly help to close the gap of lost money – as described in the previous paragraph.

3.3. Use case 3 - Remanufacturer challenge → Get missing cores

In case the remanufacturer is the writer of an option and he initiates the process of selling a remanufactured part, thus, makes a core return option available. On the other hand, that gives him the obligation to pay out the surcharge value as soon as such an option is exercised. As not all cores will be returned – may it be due to the fact that some of the reman parts find their way to markets without an established reverse logistics, or may it be because some cores do not meet the technical criteria when being taken out of a vehicle – there will always be missing cores

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

that are needed for the production of remanufactured parts. In this case, either the remanufacturer (if he is the OEM) is able to produce new as an alternative – but this could be a lot more expensive than remanufacturing – or he is just not able to fulfill the customer demand. Both variants are not very satisfying for the remanufacturer showing the importance of a reliable core return. Already today, the remanufacturer is actively buying cores from core brokers. However, this process is a manually driven process with only little system support. For example, excel sheets are used to be distributed to the potential core brokers. The feedback is not standardized, as the answers could be within the excel sheets but potentially in deviant format, or the answer could come in a separate e-mail or even by phone.

The ReCiPSS platform could offer a systematic solution for this process. The remanufacturer could create options for the cores he is looking for. Those options can be offered to a core broker, and the options can then be transferred to the one that has given the best price. As the nominal option value needs to be set when the option is created, the price to sell the option to the core broker should then cover the difference between nominal and market value.

In addition to this operational benefit for the remanufacturer, the idea behind ReCiPSS is to strengthen remanufacturing as a whole. As soon as the wholesaler benefits, he will definitely be willing to buy remanufactured over new products. And as a consequence, the remanufacturer will cash in on that as well.

3.4. Use case 4 – Core-Broker challenge – reduce risk of professional gambling

Similar to the co-creation workshop that was done with the wholesaler, another session took place with core-brokers, including some of their counterparts, the core purchasers of the remanufacturers. It provided a lot of useful information for the ReCiPSS project.

As already described in the use case of the remanufacturer and wholesaler, the core-broker will be a strong business partner for the other stakeholders. The use cases will strengthen the position of the core-broker as an intermediate between wholesaler and remanufacturer and as a direct business partner. The ReCiPSS platform can help minimize risk and reduce the uncertainty of future demand by making expiring options visible even if the physical artefact of the core it is not in hands of the option holder.

For the core-broker, it could also be seen as a possibility to get out of the niche of being a professional gambler by turning the core know-how into more cash by entering the market of trading options. Overall, with strengthening the idea of circular economy, all of those stakeholders will be winners.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

4. ReCiPSS IT platform architecture

A modular service-oriented architecture (Figure 1), commonly referred to as microservices architecture, was considered the best fit in order to achieve the ambitions of the new ReCiPSS Automotive Platform. Technologically this is supported by Linux containers running on a managed Kubernetes cluster in Microsoft Azure.

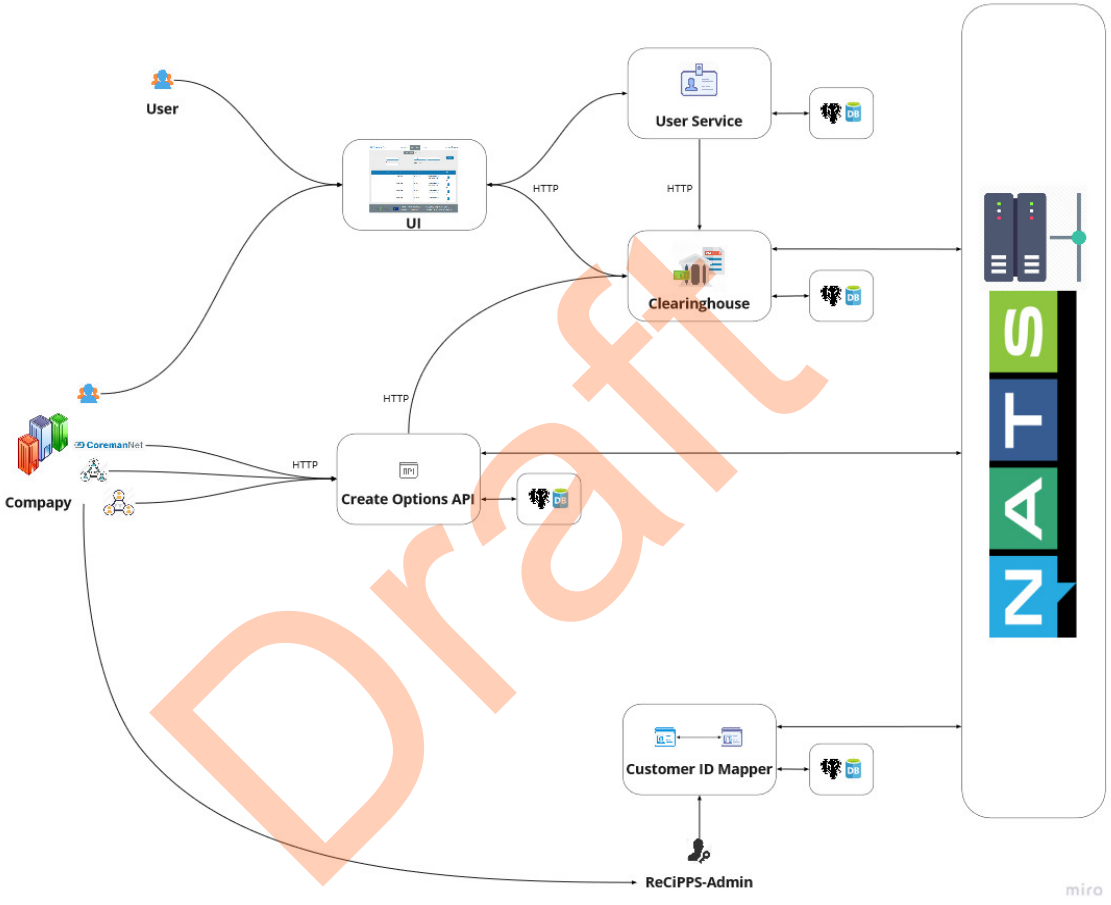

Figure 1: ReCiPSS IT platform architecture

Figure 1 depicts the service architecture implemented for the platform. It comprises five microservices, each responsible for a specific set of closely related tasks. The “UI” and the “Create Options API” form the interface to the public to facilitate the interaction of users and company ERP systems with the platform; both are accessible on the internet. The “User Service” manages the identity and roles of the ReCiPSS users providing the login and authorization functionality. The “Clearinghouse” is the heart of the platform. It manages the creation, the transfer and the execution of options. Finally, the “Customer ID Mapper” creates the connection of customer IDs, maintained by each company, to the ReCiPSS user IDs.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

As the platform is accessible online, the communication of the public interfaces runs on the HTTP protocol whereas the internal communication between the microservices uses NATS as a message queuing and streaming system, NATS also enables us to use an event-sourced architecture for each microservice. Each microservice stores all processed information in dedicated databases.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

5. Automotive part data exchange platform development

5.1. Development methodologies

As development methodology, the team has chosen from the start of WP5 the Agile Methodology, which is one of the adaptive software development methods [1].

This method was preferred because it allows a more efficient communication when member of teams are in different countries. The delivery of code is easier, constant feedback is given, testing is done for each task, and finally, a better quality of software is obtained. The response to changes and to product manager is faster and will lead to greater customer satisfaction.

5.2. Main Technologies used

Standard web technologies like HTML5 and CSS3, as well as more specific languages & frameworks like TypeScript, Angular 8.2.1. and NodeJS were used for the web-based user interfaces of the platform.

The backend system is developed using Java 11 with the Spark web framework, combined with NodeJS/ExpressJS-based microservices. Inter service communications are encoded as JSON and send via NATS Streaming and HTTP. For the persistence of aggregated data, Postgres and Redis were used.

Platforms used were Azure, Docker, Kubernetes, and Bamboo.

5.3. Core development tasks

Currently, the platform is capable of providing the core features for the management of return-options. Main functionalities to be mentioned here are:

- Build development infrastructure in Microsoft Azure Cloud,
- Create front- and backend of ICT-platform for return-options,
- Define interfaces and connect to existing systems environment
    - Company management features
    - Reporting features
- Create concept for “Underlying-Logic” as an integral feature needed for defining an option/warrant in generic way
- Privacy by design features

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

5.3.1. Build development infrastructure in MS Azure Cloud

Azure offers a lot of services for deploying applications. Azure takes care of the infrastructure to run and scale the software products, so in this way, the developers are able to focus on the business logic part and avoid all types of distributed system problems: scalability, reliability, latency, management.

The ReCiPSS platform infrastructure is based on Azure services: Container Registry, App Services, Storage Accounts, Application Gateway and Virtual Machines.

Every part of the platform is encapsulated in a Docker container. Using Docker containers, the application can run virtually anywhere and it offers instant portability.

Azure Container Registry is the place where the ReCiPSS platform parts could be safely stored and manage in Docker containers. Besides storing and managing containers, Azure Container Registry is integrated with a lot of Azure Services. So, from this registry, the docker images are pulled to the Azure App services, which are created for every backend or REST API from the ReCiPSS platform.

Azure App Service is a fully managed Platform as a Service (PaaS) that takes care of auto-scaling and traffic management capabilities, offering zero data loss, minimal downtime and high-performance. It is a good solution for ReCiPSS backend and REST APIs handling successfully many requests in no time.

The platform backend and REST APIs are accessible from anywhere in the world over HTTPS running Dockers containers which are pushed to Azure Container Registries and pulled by Azure App Services.

The frontends builds contains data objects (scripts, images) which are pushed to different Azure Storage Account. This service can store data objects in a secure, durable, highly available and massively scalable way. Azure Storage Account provides a secure unique namespace that is accessible from anywhere in the worlds over HTTPS. Every request to a storage account, from security reasons, must be authorized, including an Authorization header.

The ReCiPSS platform contains many parts, distinct services, interfaces, frontends and backends. The global routing to these differences is made with the Azure Application Gateway. This service is a web traffic load balancer that takes care of traffic to ReCiPSS platform. The Azure Application Gateway does URL-based routing (application layer – OSI layer 7). Based on the incoming URL (i.e. overview), the traffic is routed to a specific Azure Storage account or app service.

The Application Gateway is integrated with many Azure services like Azure App Services and Storage accounts. It knows all small parts of the ReCiPSS platform, which are deployed as Azure App Services or Azure Storage Accounts and based on URL; it knows every request where to be routed. It provides protection for the platform against SQL injection, cross-site scripting, many web vulnerabilities and it offers strong encryption to platform data from frontend to backend. This service scales new instances based on platform traffic load.

Azure Virtual Machine is a scalable computing resource from Azure. This type of Azure resource was chosen for installing and storing the Postgres database, Redis in-memory data structure and

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

NATS messaging system. The data is encrypted and the instances are scaled by traffic. All platform Azure Services could communicate and transfer data safely with this virtual machine.

5.3.2. Create frontend and backend of ICT-platform for return-options

The main user interface of the platform is made of two parts:

- The login page where the user can log on the platform

- The UI (user interface) for the products management

The login page is made of multiple functionalities for users: login to the platform, change/retrieve the password for the associated user account and create a new user account.

The main frontend of the platform is written in Typescript using Angular 8. The frontend contains data objects link scripts and images.

The main pages of the products management platform are :

1. Surcharge options (Analyse and select options):

This is the view where the user can see all the options that he holds. As a standard case, the platform created those options for the user in the moment he was purchasing a reman product (remanufacturable product) from his supplier (writer of the option).

2. Transfer wallet (Transfer selected options):

When the user decides to trade options or to at least offer options to be traded, the selected options will be moved to the transfer wallet for further processing. In the wallet, the user can transfer options to another user or create a container of multiple options for further processing.

3. Containers (View and trade bundled options):

Containers themselves can also be transferred. Whilst trading options from the wallet means that every single option is transferred with an own value to another user, the transfer of a container only requires a single value for the whole container containing multiple options.

Containers can also be shared with other users, which gives them the possibility to view the details of the contained options upfront of trading for them.

4. Payment obligations (Track options you created):

The view is similar to the view within the “surcharge options”, but the options those payment obligations correspond to the options that the user has written or created. In a standard case, the platform created those options for the user in the moment he was selling a reman product to his customer (holder of the option).

5. Create option (Create new options):

Options could either be created automatically by transmitting sales/purchases via a defined interface, or they could be created manually within the platform.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

All of these views can be accessed from the Welcome page , where each view has an additional description and a link to a specific page.

The first step for a new company user is to add options, to do so it has to go on “Create options” view and complete a form.

The form should be filled in with the following information:

- Underlying / Part number: is a valid product number of the parts that will be become new options

- Option value: is the price of each option newly created. This price is the amount of money that the writer is willing to give for every core that they will receive if it fulfills the criteria (underline).

- Expiration date (option to auto select 12 months or 18 months or select a date from the calendar): is the period while the option writer waits to receive cores. After this expiration date, the option is not valid anymore (and if someone returns a core after this date, the core will not be paid back.)

To set the expiration date, you can use the option to auto select 12 months or 18 months from the current day, or you can use calendar picker.

- Quantity: is the amount of options that the writer wants to issue.

After you press on the “create button, a new option will be created and will appear in “Surcharge options” view table, on the first line, and include some information about the options: status, part number (the identifying code of the option), quantity, value, currency (euro), expiration date , writer name and the actions column. Here the user has some additional filtering to apply for the table of options and sort by each column.

Holding an option means to have the right to return a core with the corresponding option for a certain value. This right can be executed towards the option writer.

By default 10 records will be listed with the option to change to 5 or 20 records per page. In the top of the list, the user has the possibility to filter data by value, expiry date or writer name.

The following analysis can be done based on that view:

- sort and filter options by option writer to see how much surcharge value is open (

- sort and filter options by expiry date to see when the next options will expiry (and consequently focus on the ones that will expire soon)

- sort and filter options by value to see the most valuable options (and consequently focus on the most valuable ones)

In the menu “containers,” the user will not only find the containers that the user created by himself, but in “my shared containers” will also find the containers that were created by other users and that were shared with the user.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

In the overview, the name of the container, the number of options and the sum of the nominal value is displayed. For shared containers also the holder of the options with that shared container is shown.

Both for own containers and for shared containers, the user has the possibility to view the details of the container.

The reporting feature is used by holders and writers to visualize data display by charts. The default view of this page is the pie chart which presents supplier(companies) status according to value (euro) and the other data visualization is another pie chart that displays supplier (companies) and the number of pieces that were sold in last month.

The last view is “Payment obligation” view and it is used by the writer of options , here is a table with some columns about options (status of an option , part number, quantity, value, currency, expiration date, and holder name. The table is almost the same with the table of options depot, it also has the additional filtering and sorting possibility.

Besides that menu of pages, on the navigation bar will appear on the right side the “Logout” button, the first two letters of the logged user and the language of the platform(En or DE), and on the left side is the logo of the platform and a link by welcome page.

5.3.3. Clearinghouse Backend

The clearinghouse microservice is the main part of the ReCiPSS platform. All the other microservices communicate and share data with the clearinghouse.

This microservice is a REST API that handles a lot of requests related to platform options and containers. It has been written in Spark Java using Java 11.

The functionalities that clearinghouse offers are:

- Creating new options. Each option requires a part number, a price, an expiration date and a writer.
- Getting, filtering and sorting the options already created. The options could be filtered by value, date or writer name. The sorting is available by value, currency, expiration date, writer name or status.
- Transfer options to wallet state or even to other companies
- Creating containers of options
- Viewing, sharing or transferring containers of options
- Deleting containers of options

The communication between the clearinghouse and the other microservices is via HTTPS or using NATS messaging system. In both ways, the common format for sending and receiving data is JSON (JavaScript Object Notation). This format means that data has the form of property – value ({ “property”: “value” }).

All user input data that comes in clearinghouse is strong checked for preventing SQL injections and the other attacks that threaten clearinghouse data integrity.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

All data created and managed by the clearinghouse is stored in a Postgres database using an Event Sourcing and CQRS pattern. Event Sourcing means that all changes in these microservices (Example: creating or transferring options) are stored as a sequence of events. Some of these events could be useful to other microservices and besides storing them, the clearinghouse sends them further. The options created and transferred events are useful to the reporting microservice too. The clearinghouse will send these events to reporting via NATS.

There is a configured Flyway as a database migration tool that allows moving data from one database to another one or delivering database changes from one environment (local) to the other (i.e. testing environment) without data loss or corruption, in a secure way. Flyway supports the undo command; therefore, the latest production database changes could be reverted safely.

The communication between the clearinghouse and all the other services is very secure because of JSON Web Token . Before anything else, the first step of handling each request is decoding the token and checking if the user is authorized for that request. The authorization process means not only the user to exist and to be valid, but also to have the proper permissions and role for the action that he started.

5.3.4. Define interfaces and connect to existing system environment

The platform contains distinct services, interfaces that have backend and frontend or only backend. Each of these interfaces: clearinghouse, customer-id-mapper, reporting and company management has backend and frontend parts, and the fifth microservice , create options API has only a backend part. Each microservice stores all processed information in dedicated databases.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

The Auth0 manages the identity and roles of the ReCiPSS users providing the login and authorization functionality.

The “UI” and the “Create Options API” form the interface to the public to facilitate the interaction of users and company ERP systems with the platform; both are accessible on the internet.

The global routing of these different interfaces is made using Azure Application Gateway, how is an URL based routing.

The system environment contains these separate microservices :

- Clearinghouse
- Create-options API
- Customer id mapper
- Company management features
- Reporting features

Clearinghouse is the main microservice for the platform, contains the backend of the platform that manage options and the frontend for the users to use this microservice

The clearinghouse backend is written in Spark Java 11 and the frontend is written in Angular 8 with Typescript .

The clearinghouse API is accessed via HTTP protocol and uses token based authentication. All information exchange with the API is based on the JSON format.

Requests to the API should use the "Authorization" header with a specific value where the token is an access token obtained through the Auth0 workflow.

When the client makes a request to the authentication server where is sending the “client Id” and the “client secret” along with the audience, then the authentication server validates the request and if successful sends a response with an access token, after that the client can use the access token to interact with the API.

The communication between the main backend, clearinghouse and the other microservices is via HTTPS or using NATS messaging system. This communication is very secure because it contains JSON Web Token . The authorization process means not only the user to exist and to be valid, but also to have the proper permission and role for any actions that he needs.

Create-options-API provide public API to create options, this microservices has only a backend how is wrote in Java 11.

The endpoint “/options/create” is used to create options and requires a JSON string describing the options. Each option contains a list with some additional information: transferred (unique UUID) and a list that contains (writer name, holder name, expiryDate ,underlying (with partNumber), number of options, surcharge, order ID, and delivery ID.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

The **customer ID mapper** lets an admin create connections between ERP users on the customer side and our own internal user or company accounts. That means this microservice creates the connection of customer IDs, maintained by each company, to the ReCiPSS user IDs.

Company-management features

Company management is the microservice that creates and manages company accounts. It has a separate backend and frontend, these are written in Typescript. The fronted used preact-hyperscript and parcel for the configuration web application bundler.

This interface is different for ReCiPSS admin users and company admin users.

The UI for ReCiPSS admin contains a default page, where it is a form for creating company admins. The form contains some specific details and from the technical point of view, each field has specific validations, the format of the email must be correct; the VAT-ID must also be valid. Email, company name and VAT-ID cannot be duplicated. When the company is created, the new company admin will receive an email and follow the instructions of this email. Besides that, these types of users can see a technical documentation accessed from the top "Documentation" link.

If a user is logged with a company admin, by default he can see the company and client details on "Company overview" and on the other page view, he can create users accounts for his company. To create an user account it is necessary to add user email address, after that the new user has to get an email and set a password from his account and will have access to login on the main platform and will be able to work.

Reporting features

The reporting microservice used for data visualization of the options, this options comes from Create-options-API. This feature is created using "react-chartjs" module. It has a separate backend and frontend, both are written in Typescript .

The charts are using only the open options. The data for the charts come from the backend of this microservice and are created in the JSON format and brought to the frontend as lists; these data contain writer name, total amounts of options and total pieces. Then each type of chart (open option by value and open options by pieces) uses information about writer name and total amounts of options or writer name and total of pieces.

The reporting feature is an interest to both the option writer and the holder because, in this view, they can see the data held by their company according to value (euro) or the number of pieces that was sold in the last month.

5.3.5. Create concept for "Underlying Logic" as an integral feature needed for defining an option/warrant in generic way

An option is defining the right to buy or sell a defined good at a later time for a defined price. This means an option (or warrant) is a standardized contract that can be traded and transferred as a product as such.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

Option is granting the owner the right to sell (“put-option”) or the right to buy (“call-option”) something. Depending on the time-frame for execution related to the option. Execution means the owner of the option can either buy (call) or sell (put) the so called underlying for a predefined price, which is also part of the options-contract. The predefined price is referred to as the nominal value of the option. That is not to be mistaken with the market price of the option in case it is traded. The nominal value is the price that has to be applied to the underlying in the event of the execution of the option, whereas the option-price or market-value is what stakeholders are willing to pay to acquire the right defined by the option (transfer of option).

A very important point to mention is that the “underlying” is the artefact that option refers to. The underlying in a precision that a stakeholder acquiring the option can be sure of what it is referring to, but at the same time allowing quite flexible. The underlying might not only define the “thing” or artefact as such, but also its state, conditions or location.

Two major roles associated with an option will be :

The “option-writer” is the entity who is offering the contract represented by the option. The option-writer is responsible for the fulfillment of the contract and the obligations resulting from that.

The“option-holder” is the current owner of the option. This user is the one who could either sell and transfer an option to a new holder or execute the option towards the options-writer. Option-holder can change with the transfer of the option, whereas the option-writer always stays the same for a specific option as he/she is part of the definition of an option.

Changing of any of the characteristics of an option includes the options-writer is not possible, as this would mean to do unilateral adaption to contract-conditions, which would influence the value of the option.

An option can only be terminated by the following events:

1. It is executed by the holder towards the writer. This means that the contract defined in the option is fulfilled.

2. It is not executed and time has passed over the due-date. In this case, the option cannot be executed anymore, is, therefore, useless and loses all its value.

3. Option-writer is buying back the option from free market and terminates it.

The platform provides a public API, which can be used by customers to transmit data in an automated way. The clearinghouse API is accessed via HTTP protocol and uses token based authentication. All information exchange with the API is based on the JSON format.

Requests to the API should use the "Authorization" header with a specific value where the token is an access token obtained through the Auth0 workflow.

When the client makes a request to the authentication server where is sending the “client ID” and the “client secret” along with the audience, then the authentication server validates the request and if successful sends a response with an access token, after that the client can use the access token to interact with the API.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776577-2

This endpoint "/options/create" is used to create options and requires a JSON string describing the options. Each option contains a list with some additional information: transferred (unique UUID) and a list that contains :

- writer name (is the seller of an option and is obligated to accept a returned core)
- holder name (how owns the option and has the right, but not the obligation to return a core)
- expiryDate (the date when an option expires)
- underlying (is a construct defining the conditions to return a core; this is defined by partNumber, which is the reman part number)
- number of options (corresponds to the number of purchased reman products)
- surcharge (additional fee paid for the reman product, to be returned by the option writer)
- orderID (allows to connect the option to an order of a reman product)
- deliveryID (allows to connect the option to delivery of a reman product)

5.3.6. Privacy by design

The ReCiPSS Platform is created, taking into account all principles of Privacy by Design (PbD) approach. The privacy implications were considered before writing the first line of code being a core function of it, not an add-on.

The ReCiPSS platform does the best to protect the users from attacks on their privacy, their dignity, and even their safety.

The user passwords are not stored in the database in cleartext. If an attacker was to break into the database and steal the passwords table, the attacker could then access each user account. Taking that into account, ReCiPSS never store passwords in cleartext. Passwords are always hashed and salted using bcrypt. Additionally, data encryption is offered at rest and in transit by using TLS with at least 128-bit AES encryption.

The platform requires minimum personal data for creating a new company: an email address, a company name, address and VAT ID. All this data is required for security reasons: logging into account and checking and authorizing the company. All this data is stored safely in the database. The email address and profile data of the logged in user is encoded in a JSON Web Token with a private key.

It doesn't require unnecessary app permissions (contacts, microphone) or personal data (birthday, gender) just email address and the profile data (avatar and name) when the user is logged in the first time.

The platform doesn't share any data with third parties and doesn't require social media registration. It's removed the data of users whose accounts have been closed or deleted.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

Each data created on the platform (company users, options, containers, reports) are stored safely and uniquely identified.

There's end-to-end lifecycle protection using Azure services like: Storage Accounts, App Services. The platform is divided into small parts that are deployed on Azure in Storage Accounts and App Services. Azure Storage encrypts user's data automatically when it is persisted to the cloud. Data is encrypted and decrypted transparently using 256-bit AES encryption.

Azure provides tools for web vulnerability scanning too. In this way can be prevented all kind of vulnerabilities as SQL injections, cross site scripting and more.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

6. Integration, testing and validation

6.1. Testing

During the Agile development process each task was tested until being declared solved inside each Sprint.

For being declared solved, each task has to meet some technical and product requirements.

- Technical requirements:
  - Unit tests passed (passing tests for a new feature and existing tests still passed)
  - Functional tests passed
  - Code reviewed by someone not involved in the implementation
  - Technical documentation (README for each repository is up-to-date, Architecture decision are recorded as ADR in each repository)
  - Feature is deployed to the development system
- Product requirements
  - Acceptance criteria met according to user story
  - Non-functional requirements are met
  - User documentation (Demo credentials are sent to testers as needed)
  - Product owner accepts the user story.

6.2. Integration

Integration tests were created for checking if the different parts of the platform are working fine combined together. After the functionalities were tested first through unit tests, then the integration tests check if the functionalities combined give the desired output.

The integrations test checks the platform dependencies (Postgres Databases, Redis, Azure Virtual Machine) at both each microservice level and platform level.

At the microservice level, most integrations tests check the links between the routing part with the database connection and HTTP response sent to the interface. At the platform level, the integration tests check the communication between microservices made via NATS Streaming channels.

After integration of different components, the plan of testing the platform before deployment contains:

1. Test preparation
   - preparation and verification of test specifications (creation of test cases for functional testing and integration testing),
   - checking the coverage of the requirements with test cases,
   - prioritization of test execution,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

- establishing the specifications and testing criteria for acceptance

2. Test configuration
   - preparation of the test environment and preparation/verification of the test data sets
3. Test execution
   - running the tests (making the test reports and detailing the non-conformities identified when running)
4. Test evaluation
   - structuring and analyzing the test results (synthesis of the results from the test reports and the list of identified non-conformities).

6.3. Validation

The software evaluation process (testing validation) will be performed continuously both during the development process and at the end of the development process (until the deployment to the production environment) to determine if the software solution meets the requirements.

In each validation stage (unit testing, functional testing, integration testing (at a singular functionality level and at a level of components), system testing, acceptance testing) the results will be analyzed and corrective actions will be planned (iterative, fixing nonconformities (coding), validation (fixing testing with ensuring regression)) until the completeness of the successful execution of the tests, according to the established acceptance criteria.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

7. Conclusions

This deliverable documented all the activities performed within tasks T5.4- Integrating, testing and documenting the part data management platform for the automotive parts demonstrator and T5.2 - Implementing a part data management platform including standard data exchange protocols ensuring transparency and efficiency of trade networks and reverse logistics, since the beginning of WP 5 – ICT platforms for Multiple Life Cycles.

The part data management platform will be deployed in WP7- Automotive Parts Demonstrator.

The report highlighted the business requirements and uses cases that translated the end user's point of view. The Key Features and overall platform architecture were presented to make the understanding of the existing platform and functionalities added through the ReCiPSS project easier. New concepts introduced by the project were described to facilitate the explanations on the platform's functionalities.

Descriptions of IT Platform Architecture, development methodology and main technologies used in the development process added important details to the whole picture of the Automotive Part Data Management platform.

Core functionalities were described in fine details, emphasizing the use of Privacy by Design concept.

The ReCiPSS Automotive part data platform is functional and core features are implemented: Cloud Infrastructure, Clearinghouse, Interfaces and connections to existing systems environment, Company management features, Reporting features and Underlying Logic concept.

During the deployment process, more customization and testing will be performed, as necessary.

The new developed Part Data management platform will streamline the reverse logistics flow, supporting the identification of cores and evaluation only once, bypassing trade levels and shipping the cores directly to the final destination.

The ReCiPSS part data management platform will support the increase of parts lifetime and will rise the degree of recyclability of materials.

Sustainable Development Goals (SDGs), in particular SDG 12 "Ensure sustainable consumption and production patterns", will be supported by the results of using the Automotive Part Data Management platform.

Other positive impacts include reducing waste generation through remanufacturing, recycling, and reuse of white goods and automotive parts, encouraging manufacturing companies to shift to circular product-service systems and integrating sustainability information into their reporting cycle.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2

8. References

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

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776577-2