

TQS: Product specification report

**Malwina Schonhofer [116718], Juliusz Szymajda [116726], Piotr Czapla [116634],
Maciej Adamczewski [116748]**

V2023-06-26

1.1	Overview of the project.....	1
1.2	Limitations.....	2
2.1	Vision statement.....	2
2.2	Personas and scenarios.....	2
	Personas.....	2
4.1	Key requirements and constrains.....	4
4.2	Architeturual view	5
4.3	Deployment architecture	5

1 Introduction

1.1 Overview of the project

The project's objective was to create a multi-layered web application with the use of a *Software Quality Assurance (SQA)* strategy in order to develop a viable software product.

The "InPostOr" project aims to provide delivery services for individuals and businesses alike. It caters to the needs of online shoppers seeking a convenient and secure method to receive their packages, as well as business owners requiring reliable logistics solutions. With an expansive network of automated lockers and associated collection points (ACPs), InPostOr ensures that customers can conveniently collect their parcels at their preferred time and location.

Our project was divided into two parts:

- 1) **E-shop** - a basic online store, where the user can purchase products and order them directly through our delivery platform.
- 2) **InPostOr** – the delivery platform responsible for the distribution and delivery of orders by the method selected by the customer.

1.2 Limitations

Little experience in the development of Spring Boot applications, first exposure to the Quality Assurance methods used.

2 Product concept

2.1 Vision statement

The InPostOr application is a comprehensive delivery management system designed to address the high-level business problem of efficient and convenient parcel delivery for both individuals and businesses. The system provides a user-friendly interface and a range of features to streamline the delivery process and enhance customer satisfaction.

2.2 Personas and scenarios

Z komentarzem [IO1]: or, in alternative, actors and use cases

Personas

- *Greg* - is an owner of a shop for brewers in his 30s. He has just opened a eShop with ingredients for beer brewers and is looking for a way to deliver his products to his customers. He is an IT geek and knows how to use external APIs and is looking for one that could manage the deliveries for him.
- *Felix* - is a brewer in his 40s. He is looking for a way to effortlessly buy beer ingredients in the internet and would like to have them delivered to a delivery point instead of his home since he is a busy man and is very rarely at his home.
- *Isabel* - is a grocery store owner in her 40s. Although she makes a living by selling groceries, she wants to bring more customers to her shop and possibly make a delivery pickup point in her store. In day to day life, she uses the internet only to check the news and weather, she is not very experienced.

Scenarios

Greg adds InPostOr delivery service to his eShop

Actor	Greg
Motivator	Opens an eShop.
Intention	Needs someone to deliver his products to his customers.
Action	Greg browses the documentation site to find which API endpoints he can use to order deliveries through the InPostOr system.
Resolution	He finds the endpoints he wants to use and creates a delivery page on his site that interacts with the API.

Felix buys ingredients in an eShop

Actor	Felix
Motivator	Is going to brew a new batch of beer.
Intention	Needs to buy ingredients to be delivered to a pickup point.
Action	Felix opens Greg's eShop and selects all products he wants to buy. In the delivery page he selects to use the InPostOr service and puts all relevant information about the delivery into the form.
Resolution	InPostOr service and Greg receive a new delivery order that has a delivery code assigned. Later the InPostOr administration will assign a delivery service provider to complete the delivery.

Isabel creates an ACP

Actor	Isabel
Motivator	Wants to have more customers.
Intention	Needs to create an ACP at her grocery store.
Action	Isabel opens the InPostOr website, registers her business and fills in the form with all of the relevant information (address, opening hours, store capacity, etc.) and clicks the register button.
Resolution	Her grocery store gets registered in the InPostOr service as an ACP and it will be possible to retrieve the information about it through the InPostOr API to be used by eShops and delivery services.

3 Domain model



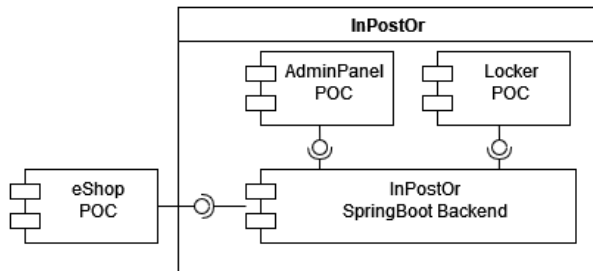
- 1) **Controller Layer:** The topmost layer of the application responsible for handling REST controllers. It consists of four controllers: ShopController, ACPCController, DeliveryServiceController, and AdminController. These controllers handle incoming requests from different stakeholders, such as eShops, ACPs (Automated Collection Points), delivery services, and the administration team. They communicate with the service layer to perform CRUD operations and provide the necessary data to the requesting clients.
- 2) **Service Layer:** The middle layer of the application that manages the data flow between the controllers and repositories. It contains the business logic of the application and ensures the implementation of CRUD functionalities. The service layer receives requests from the controllers, performs necessary operations, and communicates with the repository layer to access and manipulate the data. It encapsulates the core logic of the application, making it reusable and testable.
- 3) **Repository Layer:** The lowest layer of the application that deals with data access. It contains the data access objects and repositories responsible for interacting with the underlying database or data storage.

4 Architecture notebook

4.1 Key requirements and constraints

- The system must be available 24/7 to allow customers to access their packages at any time and to facilitate deliveries to ACPs.
- The system must have fast response times, even during periods of high usage, to ensure a smooth customer experience.
- The system must be user-friendly and easy to navigate for both customers and administrators, with clear instructions and intuitive interfaces.
- The system must be scalable to accommodate increasing numbers of customers and package deliveries as the service expands.
- The system must provide sending and delivery notification to customers via email.

4.2 Architectural view



- The system architecture consists of four modules, three of which are the InPostOr web application.
- The most important module is the InPostOr backend is written in Springboot and extends REST Api's to other modules to provide CRUD functionalities for potential API users.
- The rest of the modules were created as a proof of concept (POC) of InPostOr functionalities and all could serve as separate web applications, since they all hook up to the REST API's extended from the backend. For the time being the POC modules are hosted by the same application as the backend, but as it is stated before, could be easily separated.
- All POC modules use a combination of Thymeleaf and Bulma technologies.


4.3 Deployment architecture

For deployment of your product we used Github Actions connected to Google Cloud Platform. Initially the tests are run in order to ensure that everything works as intended, the .jar file is built and dockerized. Then the credentials for Google Cloud Platform are checked and the containers gets pushed to the cloud. Next is the Google Cloud SDK which is set up, configured and authenticated. Once all the actions are done the deploy is initialized on to the closes european server and who whole app is activated.



5 API for developers

To build API documentation we included a Spring Boot Swagger 3.0 dependency that automatically creates documentation based on annotations held in the methods of the controllers. By defining the API specification in Swagger, both the client and server teams can have a clear understanding of the API's structure, endpoints, and data models.

locker-controller		^
PUT	/locker/update/{id}	▼
POST	/locker/create	▼
GET	/locker	▼
GET	/locker/{id}	▼
GET	/locker/address/{address}	▼
DELETE	/locker/delete/{id}	▼
admin-controller		^
PUT	/admin/update/{id}	▼
POST	/admin/login	▼
POST	/admin/create	▼
POST	/admin/acp	▼
GET	/admin/{id}	▼
GET	/admin/deliveries	▼
GET	/admin/acp/{id}	▼
DELETE	/admin/delete/{id}	▼
acp-controller		⌵
PUT	/acp/update/{id}	▼
POST	/acp/create	▼
GET	/acp	▼
GET	/acp/{id}	▼
GET	/acp/get/all	▼
GET	/acp/address/{address}	▼
DELETE	/acp/delete/{id}	▼ 
order-controller		^
POST	/order/update	▼
POST	/order/delete/{id}	▼
POST	/order/create	▼
GET	/order/getByShopName/{name}	▼
GET	/order/getByOwner/{owner}	▼
GET	/order/getByLockerAddress/{address}	▼
GET	/order/getByDeliverer/{deliverer}	▼
GET	/order/getByACPAAddress/{address}	▼
GET	/order/get/{id}	▼

6. References and resources

https://sonarcloud.io/summary/overall?id=specklew_InPostOrTQS

45426 Teste e Qualidade de Software



<https://seszuave.atlassian.net/jira/software/projects/GT/boards/1/timeline>