TQS: Product Specification Report

***Rui Oliveira, Filipe Antão, Pedro Matos, Simão Antunes***

[1 Introduction 1](#_Toc39943375)

[1.1 Overview of the project 1](#_Toc39943376)

[1.2 Limitations 1](#_Toc39943377)

[2 Product concept 2](#_Toc39943378)

[2.1 Vision statement 2](#_Toc39943379)

[2.2 Personas 2](#_Toc39943380)

[2.3 Main scenarios 2](#_Toc39943381)

[2.4 Project epics and priorities 2](#_Toc39943382)

[3 Domain model 2](#_Toc39943383)

[4 Architecture notebook 3](#_Toc39943384)

[4.1 Key requirements and constrains 3](#_Toc39943385)

[4.2 Architetural view 3](#_Toc39943386)

[4.3 Deployment architeture 4](#_Toc39943387)

[5 API for developers 4](#_Toc39943388)

[6 References and resources 4](#_Toc39943389)

# Introduction

## Overview of the project

The following report provides an overview of the project that focuses on the development of a viable Minimum Viable Product (MVP) using software enterprise architecture patterns. The project aims to deliver a comprehensive solution by incorporating software quality assurance (SQA) practices throughout the software engineering process.

The project was carried out by a team of students who assumed various roles to ensure effective collaboration and successful completion of the project objectives. The roles assigned to the team members include Team Coordinator, Product Owner, QA Engineer, DevOps Master, and Developer. Each team member contributes as a developer while also taking on additional responsibilities related to their respective roles.

The scope of the project revolves around the concept of a network of pickup points as a service and the scenario centers around an e-commerce platform called BioStore, which offers the opportunity to buy biologically raised fruits and vegetables.

The system includes multiple components such as the eStore web application, the platform backend, web user interfaces for the admins and for the pickup points and a locker kiosk module. All the web applications and interfaces were developed using React and for the backend we used SpringBoot. The database is a MySQL database.

Throughout the project, several best practices are adopted to ensure efficient development and high-quality outcomes. These practices include agile backlog management using user stories, feature-branching workflow with code reviews, comprehensive software testing, establishment of a CI pipeline, and deployment using containers.

In addition to the project repository containing the source code, the project documentation includes the Product Specification report, the QA Manual, API documentation, and system observability features. These artifacts provide detailed information about the project's specifications, quality assurance practices, API functionalities, and the monitoring and analysis of the deployed system.

## Limitations

There are features on the ACP frontend such as the verification of the delivery that are not working correctly. We didn’t test so we can’t assure its functionality.

We deployed the backend and the eStore page, but we couldn’t the admin page and ACP page. The reason why we couldn’t do so is because our Azure subscription didn’t support more than two public IP addresses.

# Product concept

## Vision statement

Our system aims to provide users with a network of pickup points, offering a seamless and accessible solution for order retrieval. By leveraging the power of Associated Collection Points (ACPs) and Lockers, PickUpLine eliminates the traditional constraints of home delivery. Users can conveniently collect their orders from designated pickup locations, such as local businesses or strategically placed lockers in public spaces. This approach solves the challenge of missed deliveries and the need for customers to be present at home during delivery hours.

## Personas and scenarios

The design and development of PickUpLine was supported on three Personas deemed relevant for a proper characterization of the context.

**John** is a **client** who wants to buy organic lettuce. He accesses BioStore and buys the lettuce. Then he follows the progress of his order.

**Maria** is the **administrator** of our service. A new ACP wants to join the service. Maria wants to accept. She accesses her admin dashboard and accepts the ACP into the service.

**Antonio** is an **operator of an ACP**. He joined our service and wants to know what orders have been placed and their status. He accesses his dashboard and checks the list of orders.

## Project epics and priorities

Initially, we defined three initiatives: developing the front, the backend and connecting both ensuring everything would work as expected.

We broke down those initiatives into epics:

* The frontend development was divided into the eStore, the admin dashboard and the ACP interface development.
* The backend was divided into the database and the API development and then we connected the front and backend on each interface separately and deployed the backend on Azure.

# Domain model

<which information concepts will be managed in this domain? How are they related?>

<use a logical model (UML classes) to explain the concepts of the domain and their attributes>

# Architecture notebook

## Key requirements and constrains

<**Identify issues that will drive the choices for the architecture** such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?

Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions? Are there integrations with external systems? Is the system to be offered in different user-interfacing platforms (web, mobile devices, big screens,…)?

E.g.: (the references cited in [XX ] would be hypothetical links to previous specification documents/deliverables )

We decided to use Kubernetes because it makes the system easily scalable.

## Architectural view

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

## Deployment architecture

KUBERNETS

# API for developers

The documentation of the API is available on the following link: <http://20.93.198.246/swagger-ui/#/>

# References and resources

<document the key components (e.g.: libraries, web services) or key references (e.g.: blog post) used that were really helpful and certainly would help other students pursuing a similar work>