TQS: Product specification report

Alexandre Gazur (102751)

Daniel Ferreira (102885)

Emanuel Marques (102565)

Ricardo Pinto (103078)

v09-05-2023

[**1**](#_gjdgxs) **Introduction 1**

[1.1 Overview of the project 1](#_30j0zll)

[1.2 Limitations 1](#_1fob9te)

[**2**](#_3znysh7) **Product concept 2**

[2.1 Vision statement 2](#_2et92p0)

[2.2 Personas](#_tyjcwt), scenarios, user stories 2

[2.3 Project epics and priorities 2](#_3dy6vkm)

[**3**](#_1t3h5sf) **Domain model 2**

[**4**](#_4d34og8) **Architecture notebook 3**

[4.1 Key requirements and constraints 3](#_2s8eyo1)

[4.2 Arc](#_17dp8vu)hitecture 4

[**5**](#_26in1rg) **API for developers 4**

[**6**](#_lnxbz9) **References and resources 4**

# Introduction

## Overview of the project

The objective of this project is to design and build a system with thorough and extensive tests. We will apply the software testing standards, tools and methodologies learned in classes.

PickUs is a solution for managing deliveries of products and managing delivery points.

Separately, we also have a web store where end users can buy products and have them delivered to an ACP (Associated Collection Point).

## Limitations

We don’t plan on implementing courier services nor lockers system for deliveries, but that may change in the future.

Initially, we won’t have notifications about order state changes (e.g. on the way -> arrived) in the web store, but we plan on implementing them once the core of our system is finished.

# Product concept

## Vision statement

Managing deliveries of products can be quite complicated. Our solution makes managing product deliveries and managing delivery points a much easier task. Any external store can use our solution to manage their product delivery.

The main part of our solution are the admins and ACPs (associated collection point).

In the ACP dashboard, ACPs can manage orders associated with that ACP (orders whose delivery point is that ACP). The ACP may refuse and accept orders, check state of current associated orders and change their state (e.g. on the way, delivered, delayed).

In the admin dashboard, admins can accept or refuse ACPs, check all orders (current and past) and visualize analytics.

The second part of our solution is an example web store.

End users can buy products on our web store and, with PickUs, have them delivered to an ACP, and collect it there. They can also see the state of current orders.

## Personas, scenarios, user stories

User stories are in our Jira for now: <https://grupo-ies.atlassian.net/jira/software/projects/TQSPROJ/boards/4>

## Project epics and priorities

Initiative: eStore

Epic: Frontend

Task: Home page

Task: Product listings

Task: Product details page

Task: User profile

Task: User orders dashboard

Task: Order creation

Task: API calls to PickUs API (to get user orders)

Epic: Backend

Task: Logic and API to get products

Task: Logic and API for authentication

Epic: SQL database

Task: Store products

Task: Store user accounts

Initiative: PickUs management website

Epic: Admin dashboard UI

Task: Analytics UI

Task: Dashboard with current orders (parcels)

Task: List of pending ACP requests

Task: Accept/refuse

Epic: ACP dashboard UI

Task: Dashboard with associated orders

Task: Change state of an order

Epic: connection to PickUs API (API calls), for the tasks in this initiative

Initiative: PickUs backend

Epic: REST API endpoints

Task: Get all orders and their details

Task: Get details of a specific order

Task: Get all orders of a user

Task: Get all orders associated with a specific ACP

Task: Create (POST) a new order

Task: Update an order (e.g. change its state)

Task: Analytics (for admins)

Epic: Business logic (services) for the 7 tasks above

Epic: Containerization

Task: Containerize database

Task: Containerize backend

Epic: Deployment

Task: Deploy PickUs to be accessed remotely, maybe in a UA server

Initiative: QA and tests

Epic: Static code analysis

Task: Static code analysis on every pull request

Epic: Backend tests

Task: Unit tests

Task: ACP API tests

Task: ACP business logic tests

Task: Store API tests

Task: Store service business logic tests

Task: Admin management API tests

Task: Admin management business logic tests (e.g. analytics)

Task: Database tests

Task: Integration tests

# Domain model

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

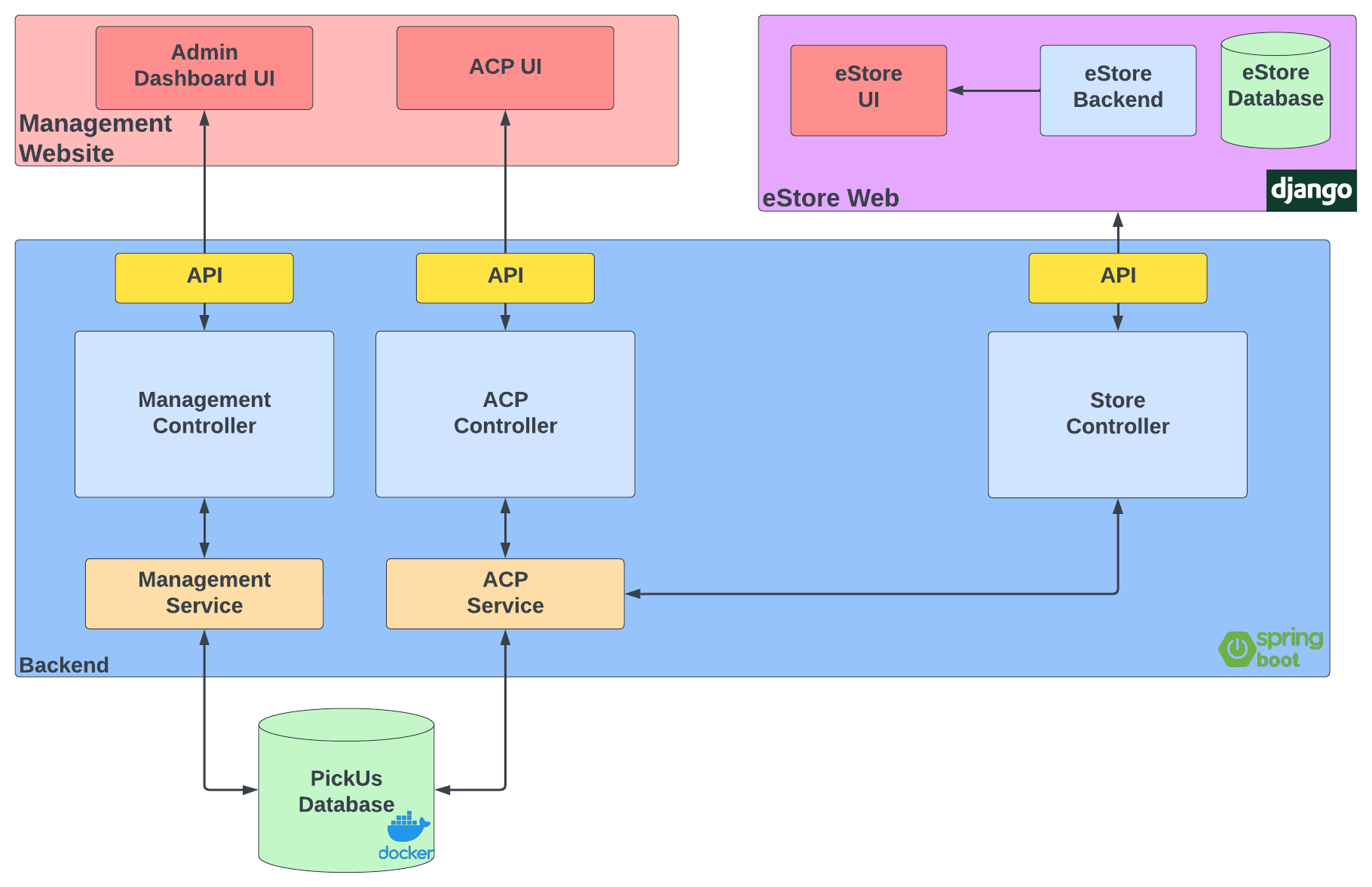
<use a logical model (UML classes) to explain the concepts of the domain and their attributes>

# Architecture notebook

## Key requirements and constrains

* PickUs must be extensively tested, specially the backend (API, business logic, database, etc) in order to assure quality.
* PickUs should be usable by any type of store. Any store that wants to use PickUs can read our API documentation and adapt their system to it.

## Architecture



The eStore can get the orders of the logged in user through our API, which fetches them from the PickUs database, processes them, and responds to the API GET request.

The eStore can also create a new order by making a POST request to our API. This order is then saved in the PickUs database.

The management website gets data from the other section of our API. The management website services fetch the requested data from the database, processes them, and responds to the request.

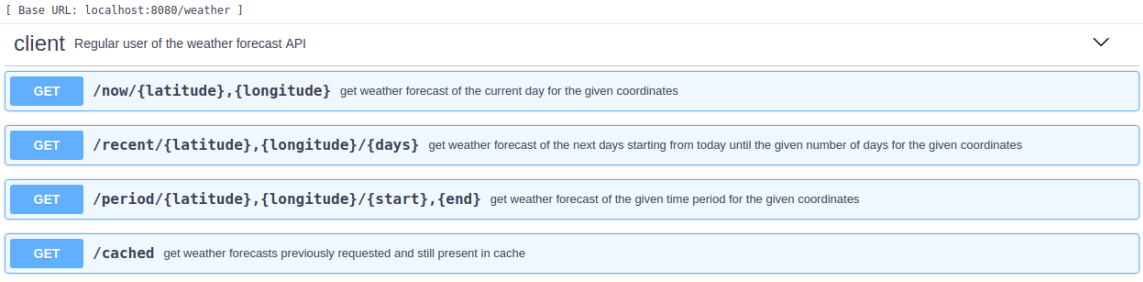
To accept or refuse ACPs, the admin website make a POST to the API. To change an order’s state, the ACP website makes a POSt to the API. The respective services then process the POST request and update the PickUs database.

# API for developers

[Explicar a organização da API. Os detalhes detalhes/documentação dos métodos devem ficar numa solução *hosted* de documentação de APIs, como o [Swagger](https://swagger.io), Postman documentation, ou incluída no próprio desenvolvimento (e.g.: maven site)

<what services/resources can a developer obtain from your REST-API?>

<document the support endpoints>



# References and resources

Github: <https://github.com/orgs/TQSProject>

Jira (backlog): <https://grupo-ies.atlassian.net/jira/software/projects/TQSPROJ/boards/4>