deti universidade de aveiro departamento de eletrónica, telecomunicações e informática

TQS: Product specification report

Alexandre Gazur (102751)
Daniel Ferreira (102885)
Emanuel Marques (102565)
Ricardo Pinto (103078)
v16-05-2023

1 Introduction

1.1 Overview of the project

The goal of this project is to create a robust system for efficient product deliveries and management of delivery points. To ensure the quality and reliability of our system, we conduct thorough and extensive testing. We apply industry-standard testing practices, use appropriate testing tools, and follow established methodologies to validate the functionality and performance of our solution.

PickUs is a solution designed to handle the logistics of product deliveries and streamline the process of managing delivery points. Additionally, In addition, we

have a web store where customers can purchase products and choose an Associated Collection Point (ACP) for delivery.

1.2 Limitations

There are some limitations to our current system implementation. These include:

- No Courier Services and Locker Systems: Our system does not integrate with courier services or lockers for product deliveries. Currently, our system primarily serves as a simulation or product concept.

- Absence of Order State Notifications: We do not have notifications in the web store that inform customers about changes in their order status. For example, customers do not receive updates when their order is en route or has arrived at the designated location. However, they can still obtain feedback on the progress of their orders by refreshing the web page.

- No Code System for User Validation: At the moment, we do not have a code system in place that allows users to validate their identity when picking up products. This would be important for ensuring secure and efficient product retrieval.

2 Product concept

2.1 Vision statement

Managing product deliveries can be complex, but our solution simplifies the process. It offers an easy way to manage both product deliveries and delivery points for any external store.

The core components of our solution are the admins and ACPs (associated collection points).
In the ACP dashboard, ACPs can efficiently handle orders associated with their specific ACP. They can view the status of current orders and update their state (e.g., on the way, delivered, delayed).
In the admin dashboard, admins have the ability to accept or decline ACPs, review all orders (past and present), and access visual analytics.

The second part of our solution is an example web store.
End users can conveniently purchase products from our web store and use PickUs to have them delivered to a designated ACP for collection. They can also track the status of their current orders.

2.2 Personas, scenarios, user stories

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

Personas:

André is a 21 year old college student who is working hard to get his degree. He enjoys going out drinking with his friends and watching football games. He lives alone in an apartment in the outskirts of the city because rent prices were too high. He finds it difficult to buy a lot of the products he wishes to buy because he lacks a car and has to walk everywhere, normally between 3-5 kilometers.

Motivation: André would like to have his purchased goods be delivered closer to him.

Tiago is a hardworking 41-year-old store clerk with over 10 years of experience in the retail industry. He has witnessed the evolving landscape of consumer behavior with the rise of online shopping and e-commerce. Tiago understands the challenges faced by traditional brick-and-mortar stores in keeping up with changing customer expectations and demands. He takes pride in providing excellent customer service and believes in the value of personal interactions and the role that physical stores play in creating meaningful connections with customers. However, he also recognizes the need for stores to adapt and embrace new technologies to stay relevant in a rapidly changing market.

Motivation: Tiago would like to modernize his store, increase his clientele and generate more revenue.

Scenarios:

Tiago registers his store as an ACP—Tiago goes to the ACP website and creates an ACP account, registering his store. After completing the ACP application online he awaits a decision from program administrators.

Tiago delivers a product and marks the order as completed—Tiago, an ACP owner, receives a notification that a customer has a product ready for pickup at his store. The customer arrives and provides the necessary details. Tiago hands it over to the customer. Using the ACP website, Tiago updates the order status as “completed”.

André purchases and picks up a product—André places an order for a desired product from an online store. During checkout, he selects a nearby ACP point as the pickup location. He is able to view the details and an estimated pickup time. After being notified of the delivery, André walks a short distance to the designated ACP point, where he presents his order confirmation. The ACP staff verifies the details and retrieves the package.

2.3 Project epics and priorities

- Initiative: eStore
  - Epic: Frontend
    - Task: Account creation (register page)
    - Task: Login page
    - 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: Dashboard with current orders (parcels)
    - Task: List of pending ACP requests
    - Task: Accept/refuse ACP requests
  - 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

3 Domain model

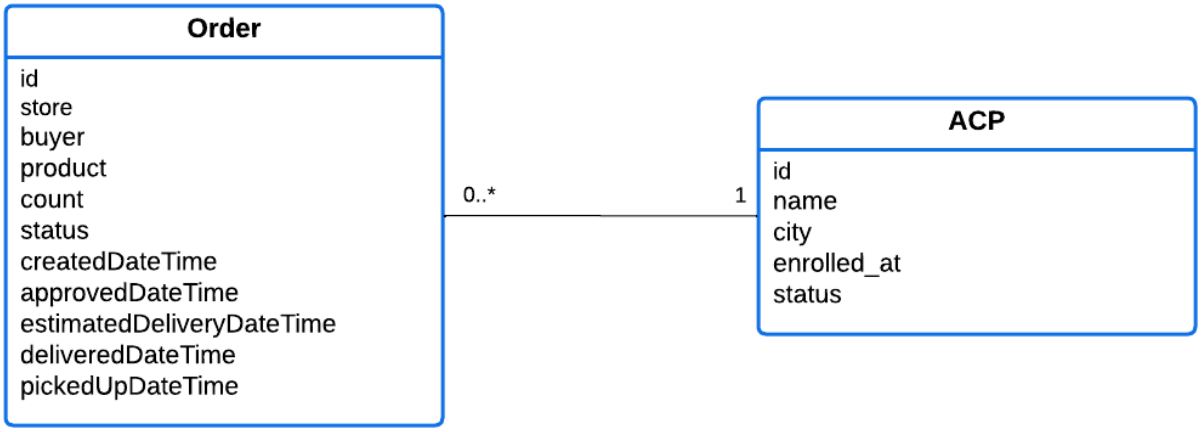

Fig. 1 - PickUs System domain model

In the provided domain model, two information concepts are managed:

1. Order:
   - Attributes: id, store, buyer, product, count, status, createdDateTime, approvedDateTime, estimatedDeliveryDateTime, deliveredDateTime, pickedUpDateTime
   - Description: Represents an individual order placed by a buyer in a store.
2. ACP:
   - Attributes: id, name, city, enrolled_at, status
   - Description: Represents an Association of Certified Providers entity.

Relationships between these concepts:

1. An ACP can have no Orders or at least one.
2. An Order is associated with only one ACP.

This is an example of a Many-to-one relationship - Multiple instances of an Order can be related to a single instance of ACP.

The user accounts, along with their related functionality such as authentication and authorization, are managed separately by the respective web applications. Therefore, are not included in this module.

4 Architecture notebook

4.1 Key requirements and constraints

- It is crucial to thoroughly test PickUs, particularly its backend components such as the API, business logic, and database, to ensure high quality and reliability.

- PickUs is designed to be used by any type of store. Any store interested in using PickUs can refer to our API documentation and easily adapt their system to integrate with it.

4.2 Architecture

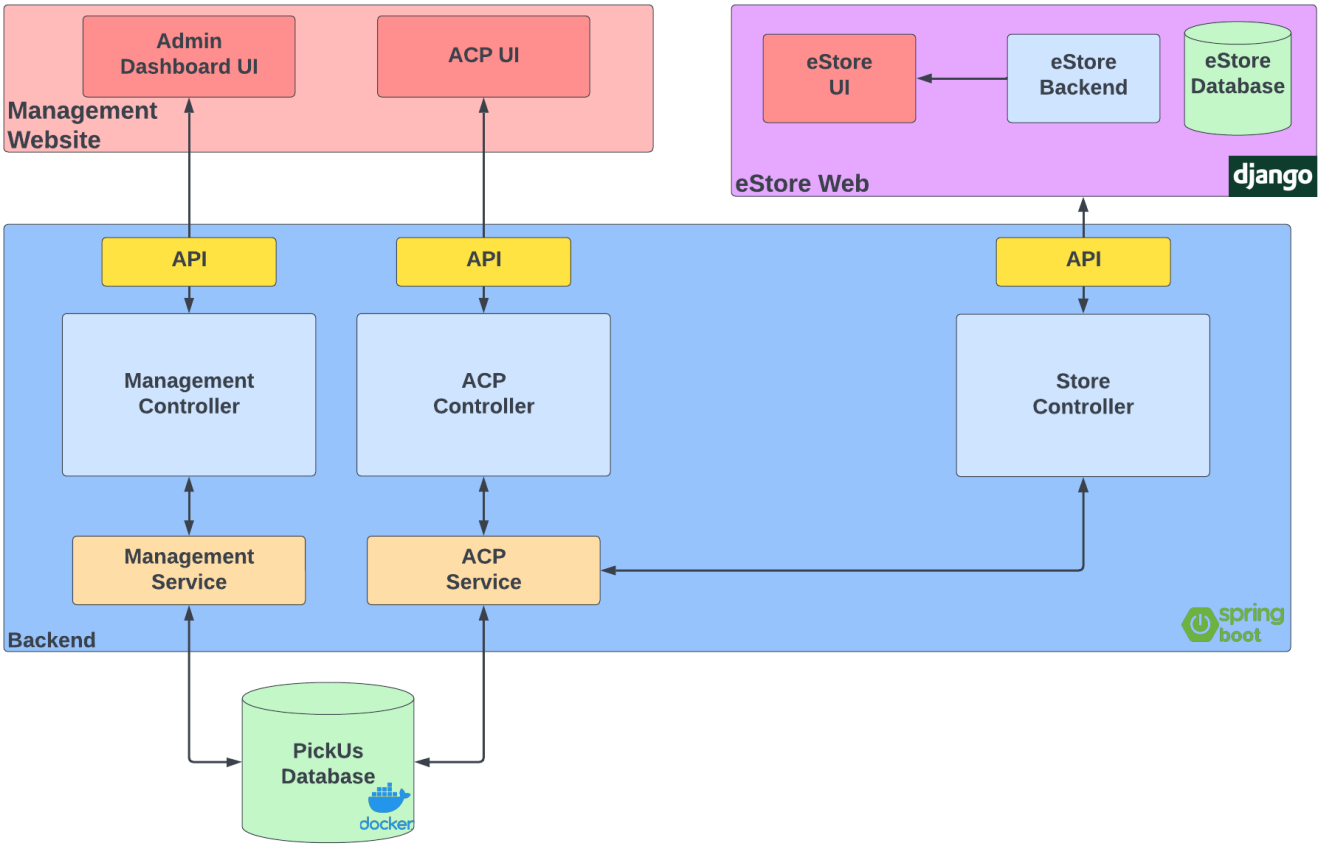

Fig. 2 - PickUs System Architecture

The eStore uses our API to retrieve orders for logged-in users from the PickUs database. It processes the orders and responds to the GET request.

Additionally, the eStore can create new orders by sending a POST request to our API, which then saves the order in the PickUs database.

The management website obtains data from another section of our API. The website's services retrieve the requested data from the database, process it, and respond accordingly.

For accepting or refusing ACPs, the admin website sends a POST to the API. To change an order's status, the ACP website also sends a POST to the API. The corresponding services then process the request and update the PickUs database.

5 API for developers

Developers can:

- Create an ACP.
- Update an ACP's status.
- Obtain an ACP by its ID.
- Obtain all ACPs or filter them by name, city or status.
- Obtain all the orders assigned to an ACP.
- Create an order.
- Update an order's status.
- Obtain an order by its ID.
- Obtain all orders and filter them by store, buyer, ACP, product or status.

The API Documentation can be found here.

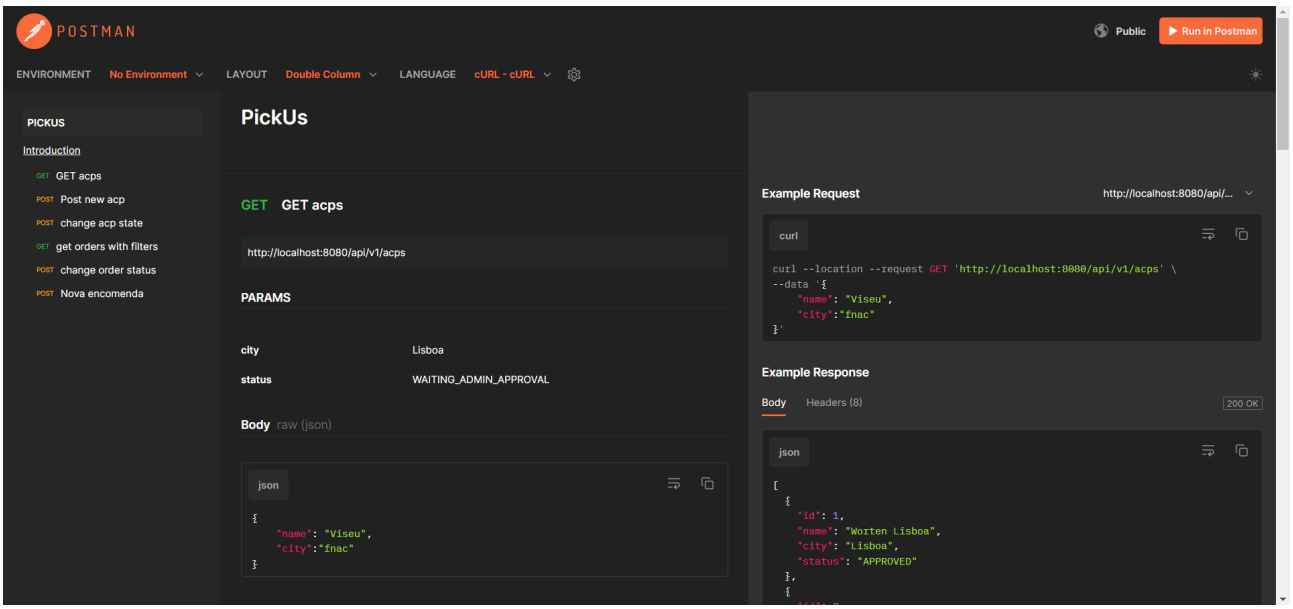

Fig. 3 - Preview of API documentation using Postman

6 References and resources

Github: https://github.com/orgs/TQSProject
Jira (backlog): https://grupo-ies.atlassian.net/jira/software/projects/TQSPROJ/boards/4
Postman (API doc.): https://documenter.getpostman.com/view/13973483/2s93m7X25U