universidade de aveiro
deti departamento de eletrónica, telecomunicações e informática

TQS: Product specification report

Daniel Ferreira [102442], Guilherme Antunes [103600], Mariana Andrade [103823], Vicente Barros [97787]

v2022-06-06

1 Introduction

1.1 Overview of the project

The project in focus, RoadRunner, is a comprehensive pickup management system that operates in the B2B space, partnering with various retail stores and delivery companies. RoadRunner provides a valuable service by offering customers of these stores the option to have their orders delivered to select pickup points. The system is composed of two types of users: admins and pickup personnel. Admins have broad access to all order statistics, they manage new partner store requests, search for pickups by location, view order statuses by pickup point, and accept or deny delivery requests. Pickup personnel are responsible for checking in and out orders at their location, viewing the status of all orders linked to their pickup (incoming, stored, or forgotten), and processing returns. RoadRunner uses a full-stack approach with a SpringBoot backend with MySQL persistence layer, and a React-based presentation layer. This system is designed to streamline the package pickup process and improve the overall customer experience in online retail.

1.2 Limitations

Though RoadRunner is a comprehensive solution for managing pickups, there are a few known limitations and unimplemented features:

- The system currently does not support real-time tracking of packages. This means that users cannot monitor the movement of their package from the store to the pickup point, on a second-by-second basis, with the status of the order only being updated periodically.
- The system does not currently support multi-language functionality. The interface is available in English only, potentially limiting its accessibility for non-English speaking users.
- While RoadRunner supports returns processing, the feature for handling exchanges is currently not implemented. Therefore, customers who wish to exchange a product must return the original item and place a new order.
- The system does not provide a mobile application. As such, it relies on the web platform for all interactions, which might not offer the same convenience and accessibility as a mobile application. RoadRunner currently does not have an integration with all major e-commerce platforms. Therefore, some stores may need to do some custom integration work to be able to use the system.

These are areas for future development that are planned to enhance the functionality and user experience of the RoadRunner system.

2 Product concept

2.1 Vision statement

The RoadRunner system is envisioned to revolutionize the B2B and B2C retail landscape by streamlining and simplifying the pickup process. The system aims to enable retailers to expand their customer service offering and improve order efficiency through an easily manageable pickup solution. Furthermore, it is designed to offer customers greater flexibility in receiving their orders, contributing to improved satisfaction and convenience. The high-level problem being solved by the system is the inconvenience and inefficiency associated with traditional delivery methods. By allowing customers to pick up their purchases from designated locations at their convenience, RoadRunner addresses a significant pain point in the e-commerce customer journey. Initially, the system was planned to be a simple solution for pickup management, but as the concept developed, it became apparent that integrating features like returns processing and interfacing with multiple retail stores would provide more value. RoadRunner shares similarities with other pickup services but stands out in its detailed B2B focus. The system is not just about the pickup process for customers but also about enabling businesses to better manage and monitor their pickup services.

2.2 Personas and scenarios

Personas:

Admin - Alex is a 35-year-old operations manager working at RoadRunner. He oversees the partner store relations and delivery processes of the system. He's tech-savvy and prefers tools that provide detailed reports and allow for seamless operation management. He values efficiency and accuracy.

Pickup Personnel - Sarah, aged 28, is a pickup point employee. She handles multiple orders daily, checking them in and out of the pickup location. She's comfortable with technology but values simplicity and intuitiveness in the tools she uses.

Store Owner - John, 45, is a small business owner who has recently started offering online ordering. He's not overly tech-savvy but understands the value of a reliable pickup management system. He appreciates easy-to-use, hassle-free solutions.

Customer - Emily, a 26-year-old working professional, often shops online due to her busy schedule. She appreciates the flexibility of picking up her packages at a convenient location and time. She values clear communication about her orders' status.

Scenarios:

Admin - Alex starts his day by logging into the RoadRunner system. He reviews the stats for all orders, checks for new partner store requests, and monitors the status of each order by pickup point. He notices a delivery request from a new partner store and approves it.

Pickup Personnel - Sarah arrives at the pickup point and logs into the RoadRunner system. She sees a list of incoming orders and prepares the storage space. Throughout the day, she checks in and out orders, updates their status in the system, and processes a couple of returns.

Store Owner - John logs into the RoadRunner system to check the status of his customers' orders. He uses the system to notify customers when their orders are ready for pickup. Later in the day, he receives a notification of a returned item and updates his inventory accordingly.

Customer - Emily receives a notification that her order is ready for pickup at her chosen location. After work, she visits the pickup point, collects her package, and the status of her order is updated as "collected" in the RoadRunner system.

2.3 Project epics and priorities

Based on the project backlog, the implementation of the RoadRunner project will be realized through the following epics:

Epic 1 - Eshop Implementation: This epic involves the creation of a user-friendly, customer-centric online shopping platform, acting as a virtual marketplace. Here, customers can explore the product catalog, add items to their shopping cart, finalize their purchases, and monitor their order's status. The objective is to provide a seamless shopping experience from browsing to checkout, with additional functionalities such as secure login and real-time order tracking.

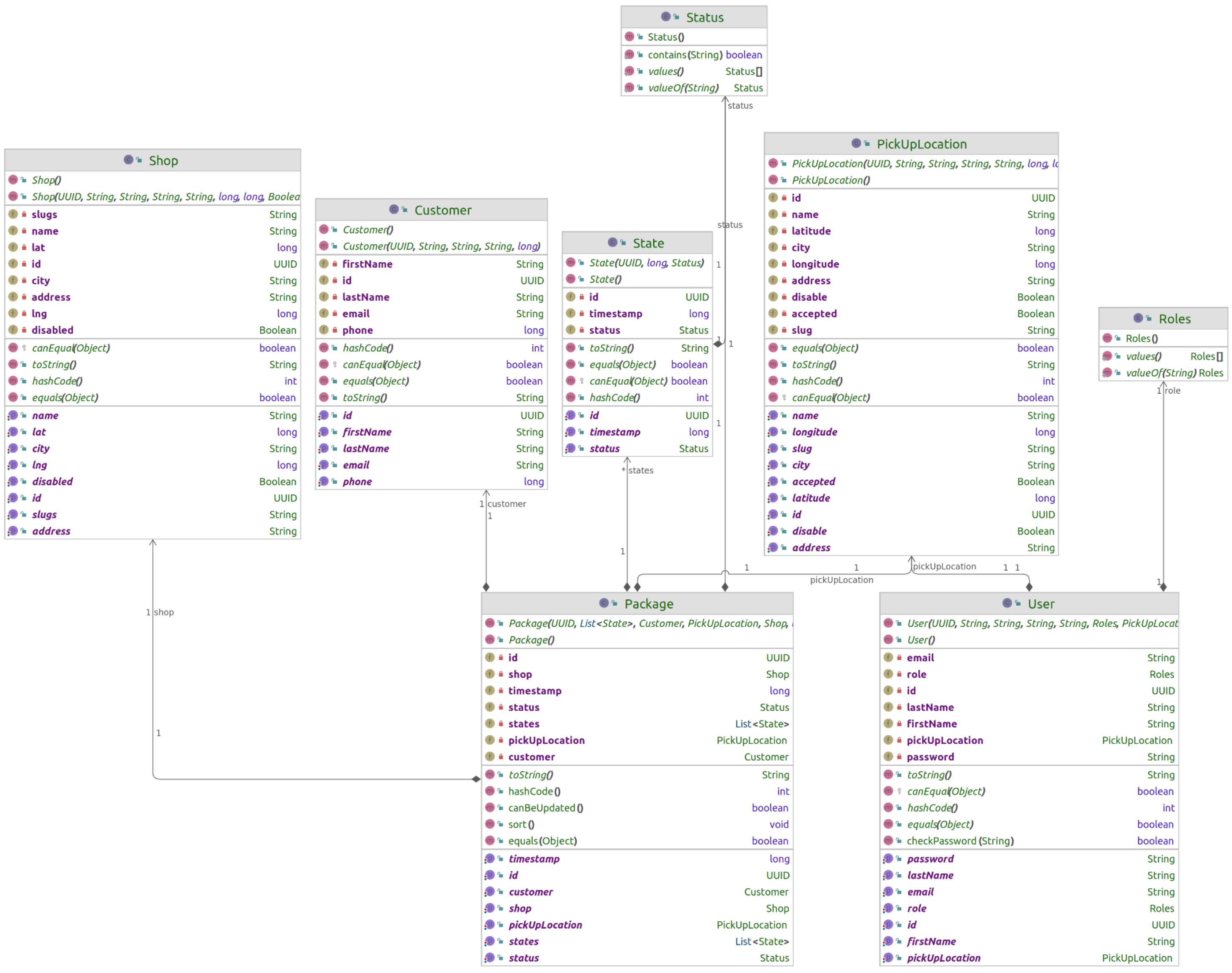
Epic 2 - RoadRunner Platform Implementation for Pickup Point Personnel: This epic focuses on the development of the RoadRunner platform from the perspective of Pickup Point Personnel. These users, associated with specific pickup locations, are granted access to manage incoming and outgoing packages at their respective locations. This functionality ensures accurate tracking and management of the package flow, thereby enhancing efficiency in operations and customer service.

Epic 3 - RoadRunner Platform Implementation for Admin: This epic caters to the Admin users of the RoadRunner platform. Admin users, unlike Pickup Point Personnel, have comprehensive access to the platform. Their responsibilities include viewing and managing statistics for all orders, overseeing all pickup points, and managing the partner stores associated with the platform. This level of access ensures thorough oversight of logistics and operations. Each epic provides incremental value, forming an integrated part of the overall project scope. The priorities among these epics will be decided based on the stakeholder feedback and strategic project objectives. Each epic is also subject to iterative development and improvement as the project advances.

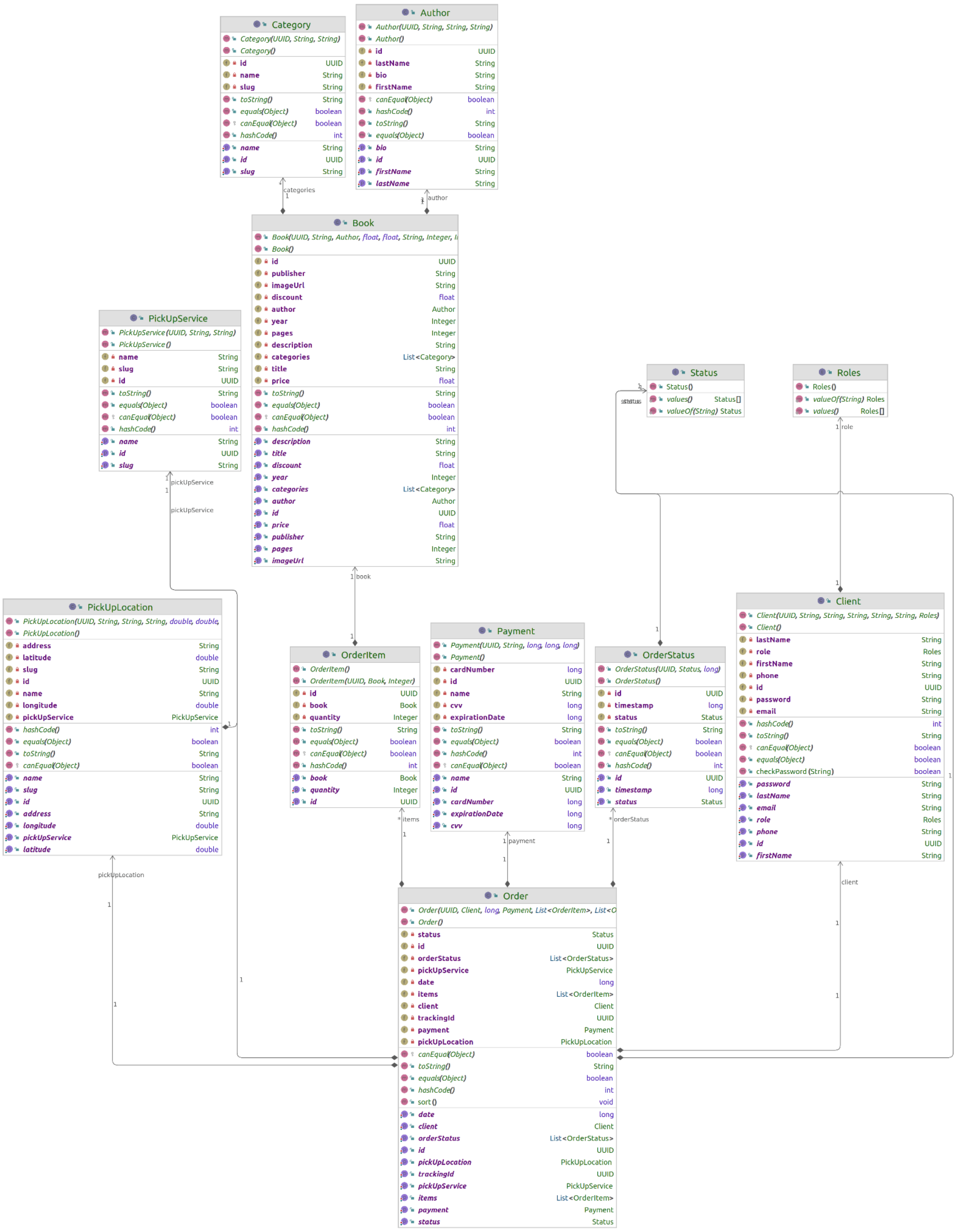
3 Domain model

The specific service database, roadrunner, and the general service, Eshop, are both developed in MySQL.

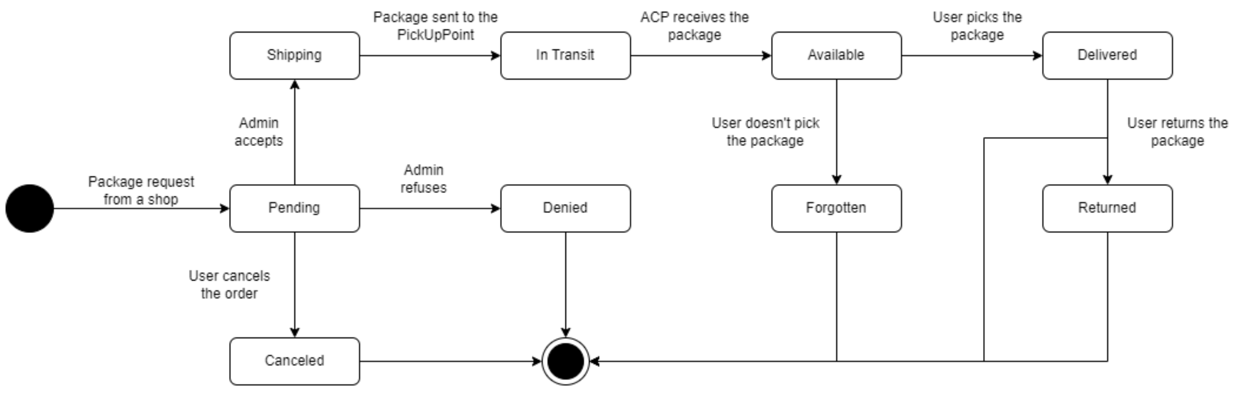
3.1 RoadRunner



3.2 SHOP


Category
Category(UUID, String, String)
Category()
id UUID
name String
slug String
toString() String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
name String
id UUID
slug String
Author
Author(UUID, String, String, String)
Author()
id UUID
lastName String
bio String
firstName String
canEqual(Object) boolean
hashCode() int
toString() String
equals(Object) boolean
bio String
id UUID
firstName String
lastName String
categories
author
Book
Book(UUID, String, Author, float, float, String, Integer, I
Book()
id UUID
publisher String
imageUrl String
discount float
author Author
year Integer
pages Integer
description String
categories List<Category>
title String
price float
toString() String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
description String
title String
discount float
year Integer
categories List<Category>
author Author
id UUID
price float
publisher String
pages Integer
imageUrl String
PickUpService
PickUpService(UUID, String, String)
PickUpService()
name String
slug String
id UUID
toString() String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
name String
id UUID
slug String
pickUpService
Status
Status()
values() Status[]
valueOf(String) Status
status
Roles
Roles()
valueOf(String) Roles
values() Roles[]
role
book
PickUpLocation
PickUpLocation(UUID, String, String, String, double, double,
PickUpLocation()
address String
latitude double
slug String
id UUID
name String
longitude double
pickUpService PickUpService
hashCode() int
equals(Object) boolean
toString() String
canEqual(Object) boolean
name String
slug String
id UUID
address String
longitude double
pickUpService PickUpService
latitude double
OrderItem
OrderItem()
OrderItem(UUID, Book, Integer)
id UUID
book Book
quantity Integer
toString() String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
book Book
quantity Integer
id UUID
Payment
Payment(UUID, String, long, long, long)
Payment()
cardNumber long
id UUID
name String
cvv long
expirationDate long
toString() String
equals(Object) boolean
hashCode() int
canEqual(Object) boolean
name String
id UUID
cardNumber long
expirationDate long
cvv long
OrderStatus
OrderStatus(UUID, Status, long)
OrderStatus()
id UUID
timestamp long
status Status
toString() String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
id UUID
timestamp long
status Status
Client
Client(UUID, String, String, String, String, String, Roles)
Client()
lastName String
role Roles
firstName String
phone String
id UUID
password String
email String
hashCode() int
toString() String
canEqual(Object) boolean
equals(Object) boolean
checkPassword (String) boolean
password String
lastName String
email String
role Roles
phone String
id UUID
firstName String
items
payment
orderStatus
pickUpLocation
client
Order
Order(UUID, Client, long, Payment, List<OrderItem>, List<O
Order()
status Status
id UUID
orderStatus List<OrderStatus>
pickUpService PickUpService
date long
items List<OrderItem>
client Client
trackingId UUID
payment Payment
pickUpLocation PickUpLocation
canEqual(Object) boolean
toString() String
equals(Object) boolean
hashCode() int
sort() void
date long
client Client
orderStatus List<OrderStatus>
id UUID
pickUpLocation PickUpLocation
trackingId UUID
pickUpService PickUpService
items List<OrderItem>
payment Payment
status Status

3.3 Diagram the state do package



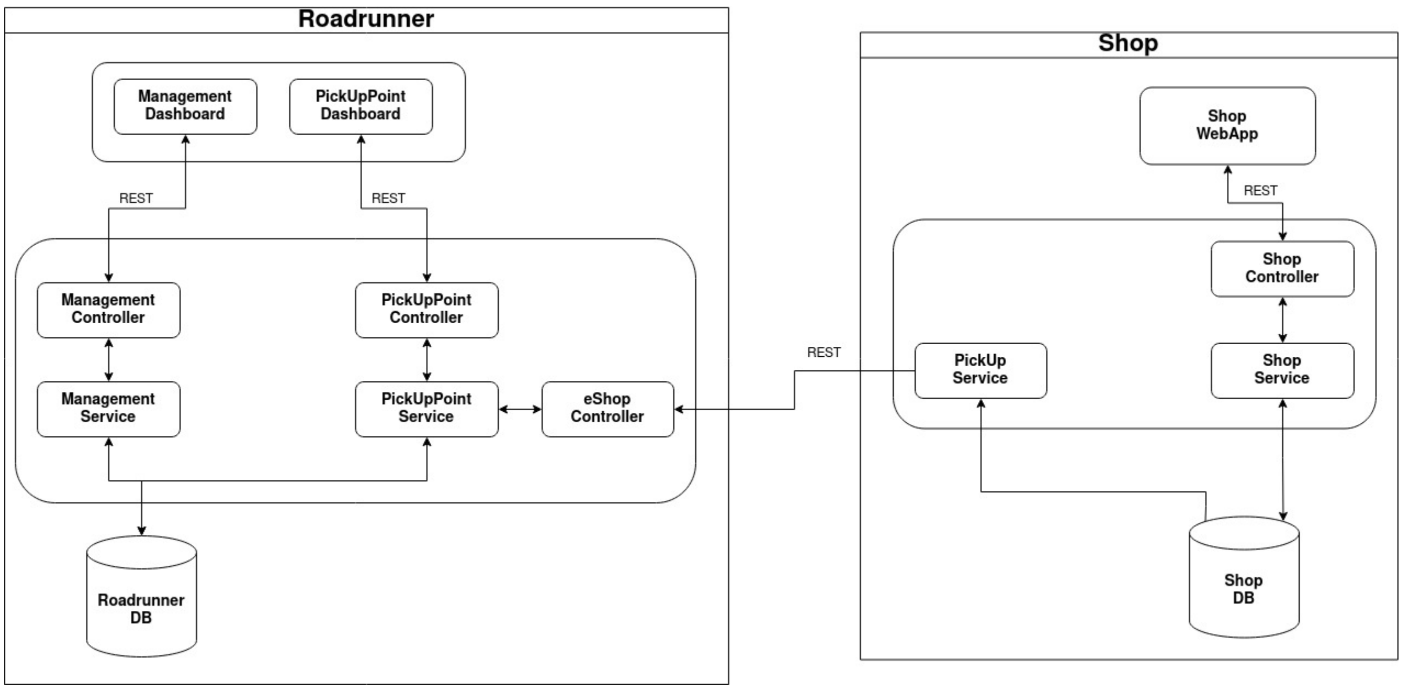
4 Architecture notebook

4.1 Key requirements and constraints

The choice of architecture was made based on the following requirements:

- The system should have a secure authentication system prepared for roles.
- It should allow the user to track the order status in real-time.
- It should allow stores to associate PickUpPoints for collection.
- The RoadRunner system should allow applications for new ACPs
- The RoadRunner system should have different functionalities for different roles.

4.2 Architectural view



The architecture implemented in our system has been divided into two main modules. One of these modules, RoadRunner (Core), is responsible for implementing the pickup platform. The second module, Shop (Service), represents the architecture of the store, which utilizes our CORE API to provide pickup services to its customers.

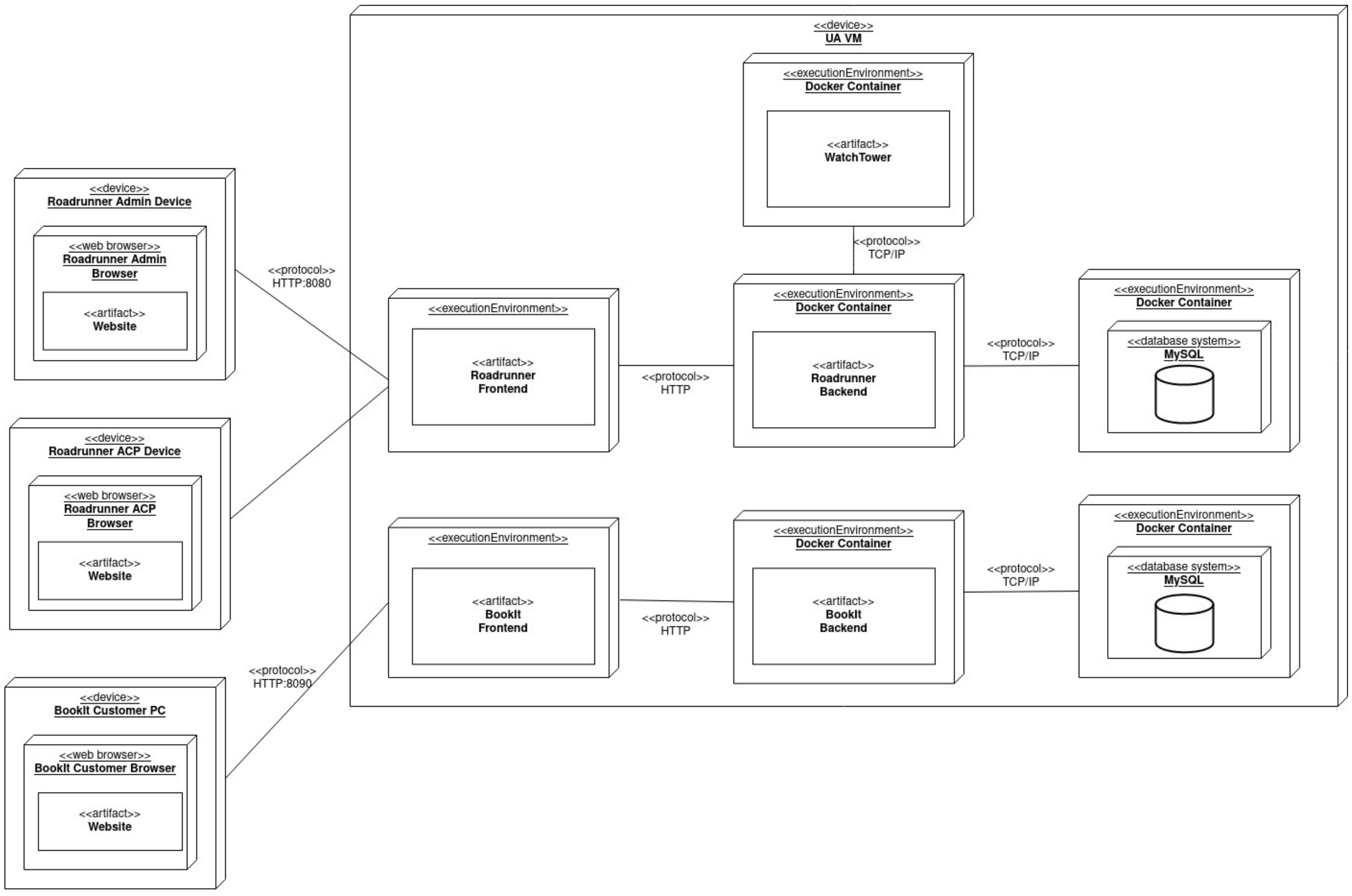
Each of these modules is further subdivided. In the RoadRunner module, we have the main backend and the main frontend. The backend leverages a REST API that handles all the business logic for the pickup points platform, developed in SpringBoot. The frontend module consists of two web applications developed in React. These integrations will be used by stores to register a new package and track its status. These connections are established by connecting to the backend technology and obtaining the necessary information using the exposed REST API.

The SHOP module represents a store that should have associated pickup points where its customers can collect their orders. It has a similar architecture to the RoadRunner module. The Backend sub-module, implemented in Spring Boot, implements a REST API that is accessed by the Frontend sub-module.

The Frontend module contains a web application developed in React, which is used by store customers to place orders for products from that store. Both modules have a MySQL database.

4.3 Deployment architecture

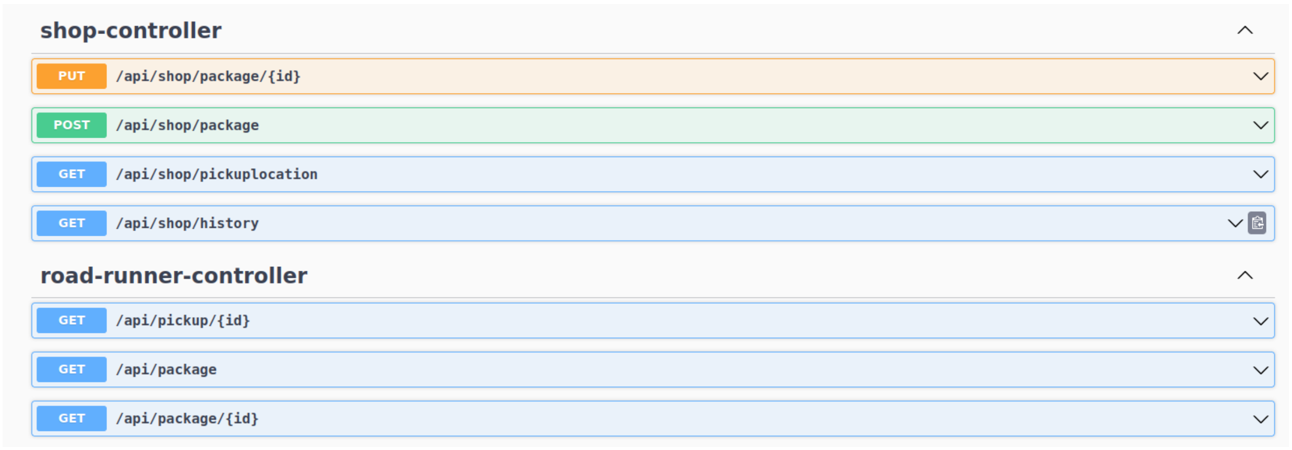
As you can see from the diagram, the deployment was done on a Virtual Machine provided by the university specifically for our group. The backends were containerized using SpringBoot, the databases were set up with MySQL, and WatchTower was utilized for CD/CI purposes.



5 API for developers

Documentation of the developed controllers. Note that the shop-controller corresponds to the Shop-side API, and the remaining controllers belong to the RoadRunner API. The auth-controller is common to both APIs.

RoadRunner API documentation - http://192.168.160.227:8080/swagger-ui/index.html
Shop API documentation - http://192.168.160.227:8090/swagger-ui/index.html

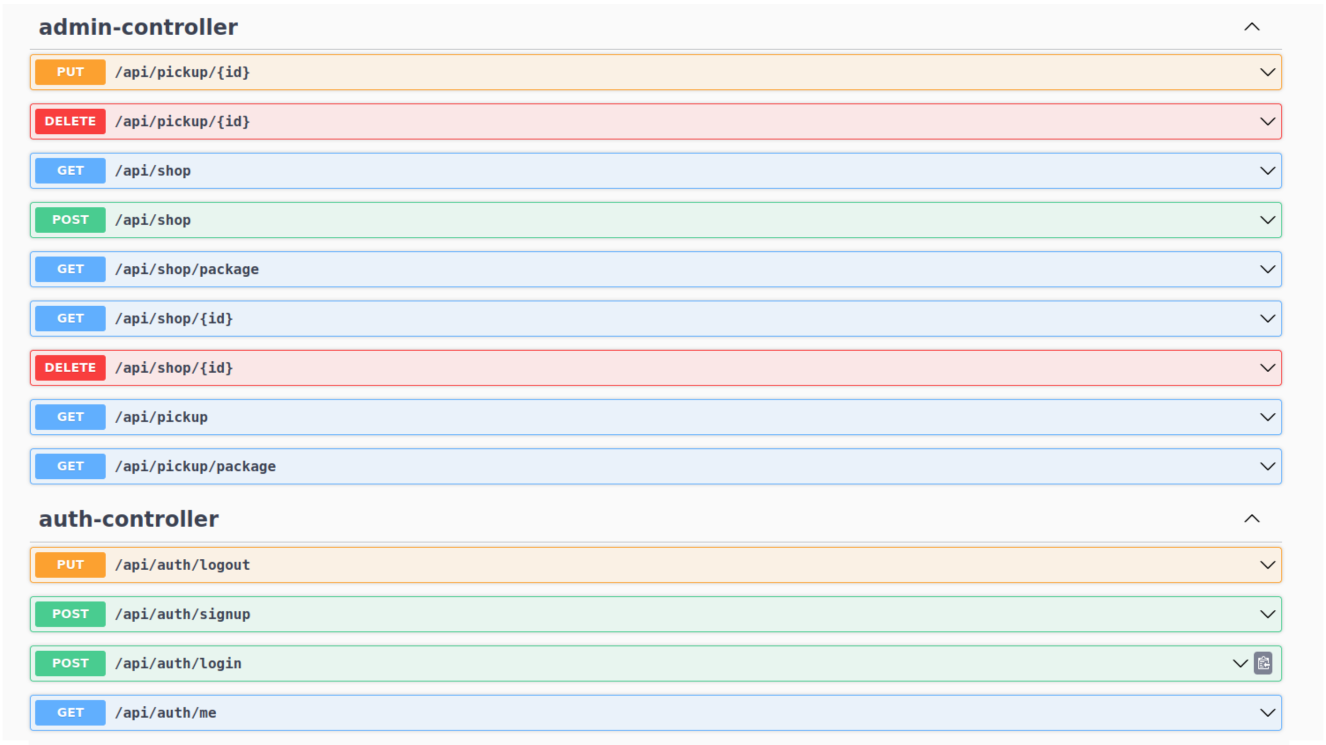


shop-controller

```
PUT     /api/shop/package/{id}
POST    /api/shop/package
GET     /api/shop/pickuplocation
GET     /api/shop/history
```

road-runner-controller

```
GET     /api/pickup/{id}
GET     /api/package
GET     /api/package/{id}
```

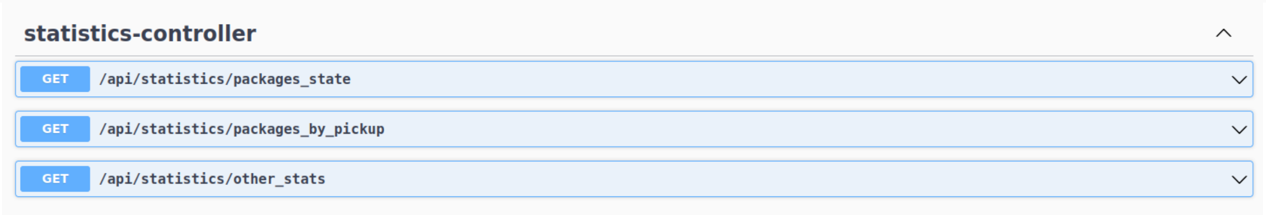


admin-controller

```
PUT     /api/pickup/{id}
DELETE  /api/pickup/{id}
GET     /api/shop
POST    /api/shop
GET     /api/shop/package
GET     /api/shop/{id}
DELETE  /api/shop/{id}
GET     /api/pickup
GET     /api/pickup/package
```

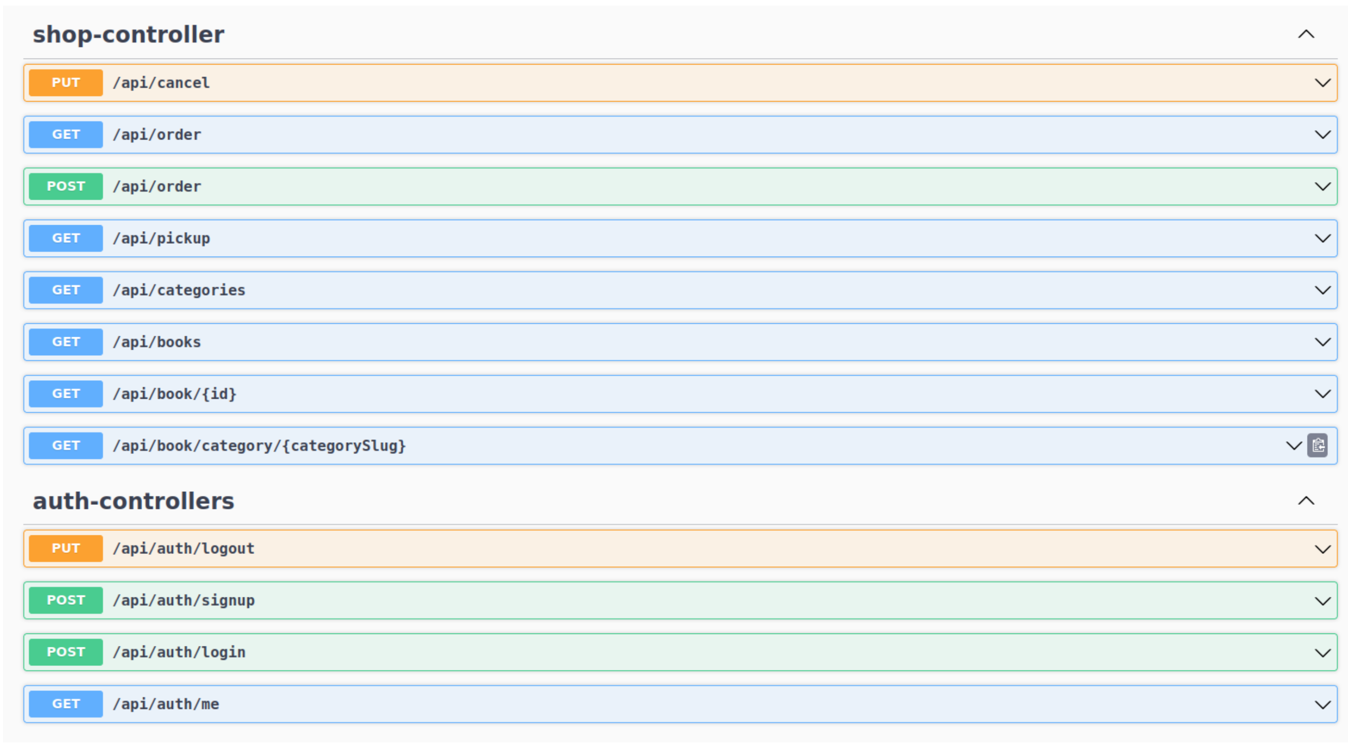
auth-controller

```
PUT     /api/auth/logout
POST    /api/auth/signup
POST    /api/auth/login
GET     /api/auth/me
```



statistics-controller

```
GET     /api/statistics/packages_state
GET     /api/statistics/packages_by_pickup
GET     /api/statistics/other_stats
```



shop-controller

PUT /api/cancel

GET /api/order

POST /api/order

GET /api/pickup

GET /api/categories

GET /api/books

GET /api/book/{id}

GET /api/book/category/{categorySlug}

auth-controllers

PUT /api/auth/logout

POST /api/auth/signup

POST /api/auth/login

GET /api/auth/me

6 References and resources

Spring, "Spring Framework Reference Documentation," Spring Framework, 2021. OnlineOnline. Available: https://docs.spring.io/spring-framework/reference/. Accessed: May 30, 2023.

Docker, "Docker Documentation," Docker, 2023. OnlineOnline. Available: https://docs.docker.com. Accessed: May 31, 2023.

Watchtower, "Watchtower Documentation," Containrrr, 2023. OnlineOnline. Available: https://containrrr.dev/watchtower/. Accessed: May 31, 2023.

"GitHub Actions Documentation," GitHub, 2023. OnlineOnline. Available: https://docs.github.com/en/actions. Accessed: June 1, 2023.

"SonarCloud Documentation," SonarCloud, 2023. OnlineOnline. Available: https://docs.sonarcloud.io/. Accessed: June 2, 2023.

"Apache JMeter User Manual," Apache JMeter, 2023. OnlineOnline. Available: https://jmeter.apache.org/usermanual/index.html. Accessed: June 2, 2023.

"JUnit 5 Documentation," JUnit, 2023. OnlineOnline. Available: https://junit.org/junit5/. Accessed: June 3, 2023.

Baeldung, "JUnit 5 Tutorial," Baeldung, 2023. OnlineOnline. Available: https://www.baeldung.com/junit-5. Accessed: June 4, 2023.

Baeldung, "Spring Boot Testing Tutorial," Baeldung, 2023. OnlineOnline. Available: https://www.baeldung.com/spring-boot-testing. Accessed: June 5, 2023.

"Mockito Documentation," Mockito, 2023. OnlineOnline. Available: https://site.mockito.org/. Accessed: June 5, 2023.