TQS: Product specification report

Andreia Portela [97953]

Ricardo Ferreira [98411]

Tiago Matos [98134]

Vitor Dias [98396]

v2022-06-17

Index

1 - Introduction

1.1 - Overview of the project

The main objective for building this project in the scope of the TQS course is to create two full-stack apps whilst testing every aspect of them.

We will have 2 separate systems, a delivery system and an online clothing store called *aDress*, the latter will be used to make sure everything is functional, as it will be made with the purpose of using the delivery system as any other online store would.

The delivery system will work with other businesses, in a B2B relationship, by receiving requests to deliver packages that have been purchased by a customer using the other businesses' website/app. The delivery system then chooses couriers based on their location and ongoing deliveries.

Our system is useful for new businesses and for existing ones to expand because it will allow them to keep selling while not having to worry about the delivery side of it, by letting us take over.

1.2 - Limitations

The main limitation of this project is the time and stipulated deadlines. Since this project was done in parallel to the Projeto em Engenharia Informática, we had little time available to implement all the intended functionalities. Some of these include the continuous delivery pipeline that isn't fully operational and the cityDelivery mobile app.

2 - Product concept

2.1 - Vision statement

The system will be used to allow businesses that do not have a way to deliver orders but want to get in on the market to access it in an easy and fast way.

The main idea for this project is to create a "last-minute" delivery service, which will be called CityDelivery. This service can be used by companies or users to send their products to a nearby destination (e.g.: a client's house). When a delivery is requested, the system will assign an available carrier to pick up the products and deliver them to their destination. Besides the CityDelivery system, we will also develop an online clothing shop called aDress. This application will rely on the CityDelivery services to send their products to their clients and will work as a proof of concept.

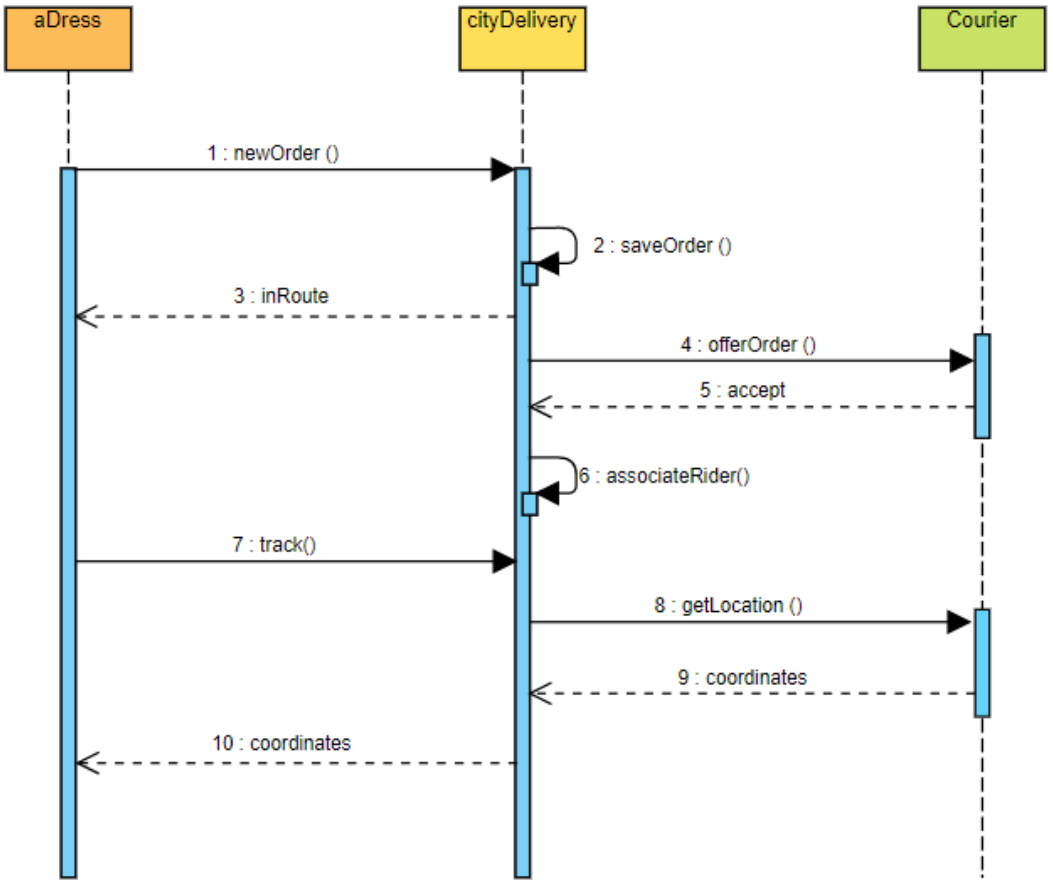

Diagram 1: Interactions between modules

2.2 - Personas

Júlia is a 50 years old manager at cityDelivery. She wants to be able to check where their delivery guys are at all times and also how many deliveries are planned.

Luís is a 23 year old man who has been unemployed for more than a year. He spends most of his day roaming around Aveiro driving his bike. Luís wants to do some odd jobs in order to earn some quick money. Since he is already driving his bike around most of the day, he thinks it is a good idea to make deliveries. He hears about cityDelivery and downloads the application. He signs up to be a courier and gets approved.

Samuel is a 31 years old who works 12 hours shifts and has no time to go shopping in physical stores. He wants to be able to buy clothes and have them delivered to his work or his house.

2.3 - Main scenarios

2.3.1 - Júlia

1. Júlia needs to do the monthly business report. For this, she checks how many deliveries they had done that month.
2. Júlia wants to know how many couriers are doing deliveries at that moment. She checks the map to see their location.
3. Júlia received complaints about one of their couriers. She needs to ban him from cityDelivery and block his access to the platform.
4. Júlia needs to accept new delivery men. She needs to verify his identity and accept or deny their request to join the platform.

2.3.2 - Luís

1. Luís receives a notification of a new delivery near him. He opens the application and starts driving to the pick-up business location. When arriving, an employee hands him the client's order. Luís takes it to the client's delivery address and hands it off to them and marks the delivery as done.
2. Luís receives a notification of a new delivery near him. He is not able to perform the delivery. He opens the application and declines the delivery.

2.3.3 - Samuel

1. Samuel would like to order from aDress. He creates an account and fills in his information. His information is saved in his profile.
2. Samuel needs to buy a new pair of pants. Since he has no time to go to a physical store, he goes to aDress and picks up the pants he wants. After paying, he chooses to have it delivered and fills the form with his information.
3. Samuel bought a new item from aDress and chose to have it delivered. He decides to check the app to see where his package is at the moment.

2.4 - Project epics and priorities

The project is divided into 4 Epics: 2 for cityDelivery and another 2 for aDress. The following table will describe the different epics and their goals

Side	Epic	Goal
CityDelivery	Deliver Product	CityDelivery accepts orders, sends them to the couriers who can accept and deliver them, and lastly notify the client about its order location.
	Management System	Have statistics about deliveries and their whereabouts.
ADress	Store Navigation	Have a functional static frontend where users can add things to the cart, alter their information etc (no business logic yet).
	Make Purchase	Have an interactive application where the frontend is functional and works with the backend. The user can create their profile, add things to the cart, make purchases, see their order and track them.

Table 1: Description of the project Epics

The first priority was to create the static frontend to see what needed to be implemented in the backend (ex.: there is a profile page in the aDress frontend therefore the backend needs to have an endpoint to get the client information).

Then, it was important to work on the creation and reception of orders on both sides. Lastly, we handled the tracking.

3 - Domain model

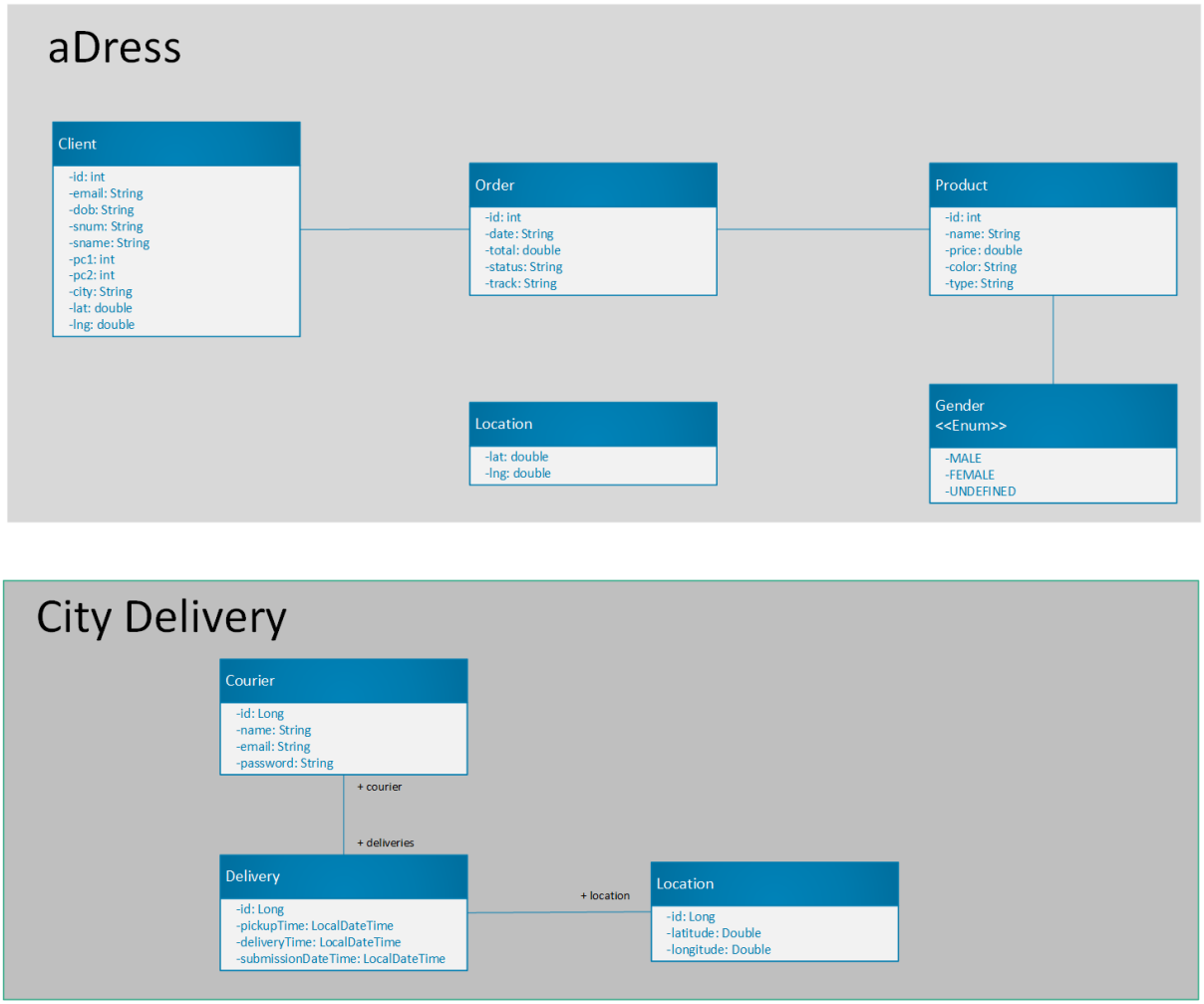

Figure 1: Domain Model of the system

4 - Architecture notebook

4.1 - Key requirements and constraints

- Both applications must have a register and log-in page.
- Couriers need to have a mobile app to accept orders and share their location.
- Couriers should be able to get a notification when a new delivery request is available.
- CityDelivery needs to be able to see statistics about orders and their whereabouts.
- aDress needs to allow new orders to be placed and tracked.
- aDress clients should have their order status updated automatically.

4.2 - Architectural view

For the architecture of this project, there will be two back-end infrastructures implemented with Spring Boot. One of them will compose the CityDelivery system, which will not only provide the web application but also provide an API to be used by other systems. This API will then be called by the second infrastructure that composes the aDress system and its web application. Both mentioned systems will contain a layer of persistence implemented with MySQL and a front-end layer that will be implemented with HTML, CSS and JS.

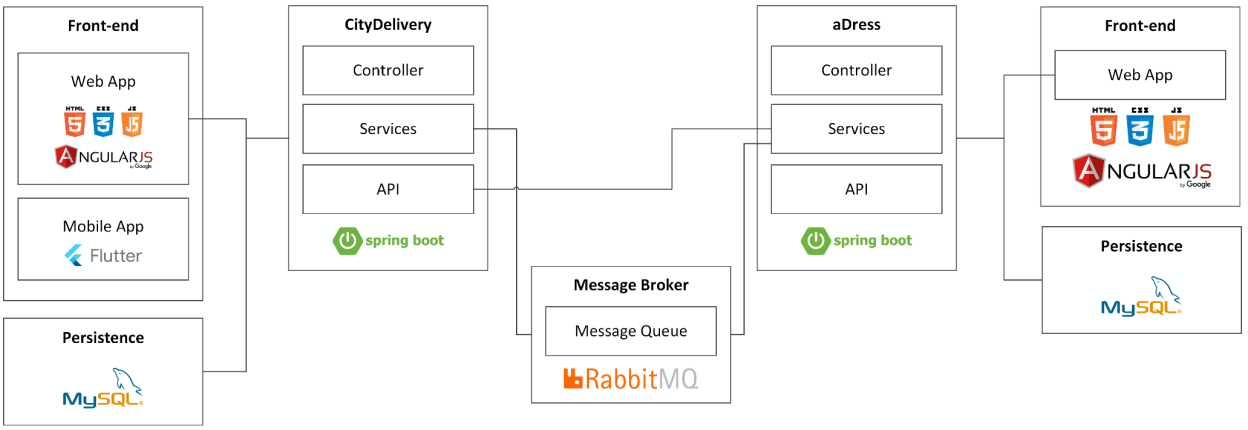

Diagram 2: Architecture diagram

4.3 - Deployment architecture

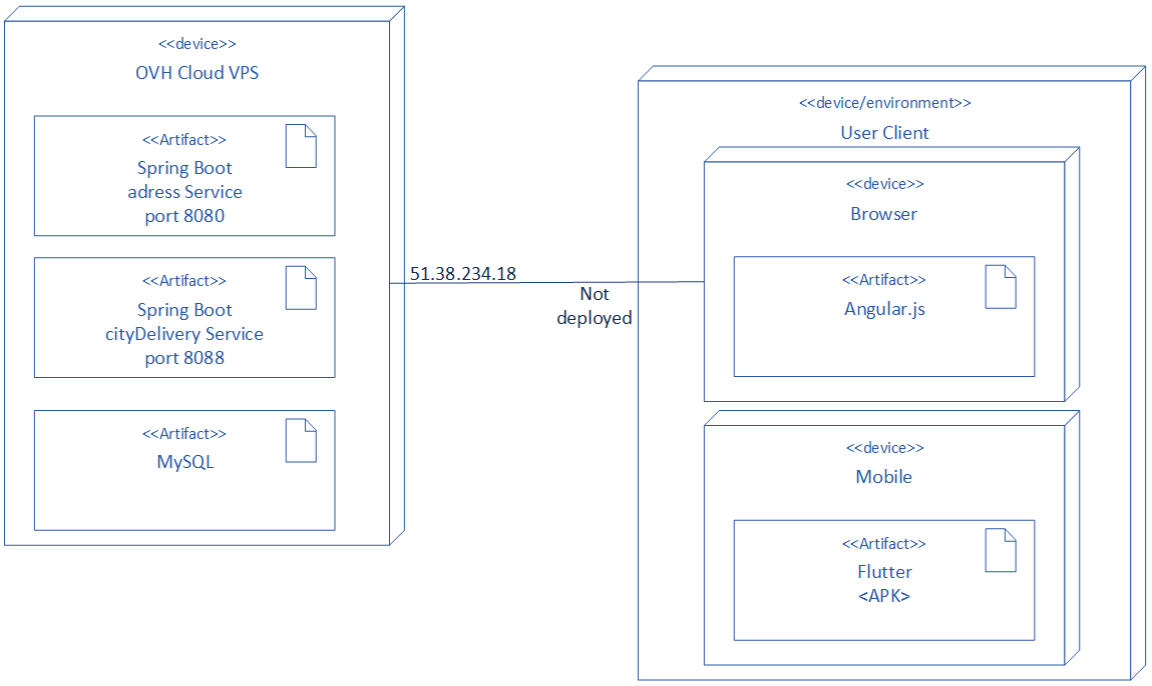

Diagram 3: Deployment diagram

5 - API for developers

The cityDelivery API allows services to have an easy delivery system. All the clients have to do is call the API with the delivery information and the system will assign a courier to deliver it. If the client needs to track its order, they can call the API to obtain this information.

The API documentation for the CityDelivery is available in https://app.swaggerhub.com/apis-docs/tiagomrm/city-delivery/1.0.0-oas3

We also have an internal API on the aDress side which has the following documentation: https://app.swaggerhub.com/apis/vfrd00/aDress-API/1.0.0

6 - References and resources

https://angular.io/docs

https://www.baeldung.com/unirest

https://swagger.io/

https://docs.flutter.dev/