TQS: Product specification report

Alexandre Rodrigues [92951], Diogo Bento [93391], Pedro Laranjinha [93179], Pedro Silva [93011]

v2021-06-22

1 Introduction

1.1 Overview of the project

LogisticsMarshall is a management service that allows entities to benefit from their delivery network for last-mile deliveries.
It provides an API to allow other services to automate service usage.
We will also be developing a food delivery service called **MarchingFood** to demonstrate usage of our API.
All components are to be tested according to standards set by TQS classes, and external services like SonarQube and Docker will be used to ease testing, quality control, and deployment.

1.2 Limitations

Our logistics system does not log payments from our clients on a per-request basis, rather than payment being deducted immediately, a periodic payment is expected based on an analysis of requests made over a time interval, and said payment is not part of the web service.

2 Product concept

2.1 Vision statement

Our project consists of two sub-projects that interact with each other, the first one is ***LogisticsMarshall,*** a management app that allows a company to register with us and be able to communicate with a network of drivers employed to us, this app provides an API that enables a client to post a request for delivery, to know the state of any delivery posted, to rate any of our drivers, and check our drivers' statistics.

The second one is ***MarchingFood,*** a food delivery service (restaurant) that enables customers to order food, which talks with our logistics service for the delivery part, but handles communication with the client and payments on its own.
Furthermore, ***LogisticsMarshall*** also possess an API to be used for our drivers, which enables them to accept any requests posted, update any delivery they have accepted, see what delivery requests are available and see his own profile including ratings and comments and his delivery history.

2.2 Personas

Personas:

James is a 20 year old student at a local university who doesnt have time to cook so he needs a service that allows him to order food and have it delivered to some place he desires. He would prefer a simple system that alows him to check on the state of a delivery so he is able to be where the delivery is planned to arrive.

Peter is 35 year old business man that is interested in taking his restaurant to the next level by offering a food delivery service to his costumers, in order to do this he decides to register with LogisticsMarshall as he is looking for a clear system that allows him to be able to provide a service without having to invest capital in contracting drivers and managing them.

Alejandro is a 25 year old unemployed man that needs money in order to pay rent and decides to enlist in our service.



Fig 1,2,3: Project Personas

2.3 Main scenarios

James orders a “DonDinis” Burger menu – James starts by opening the application for MarchingFood in his web browser and is greeted by a menu with the latest offers put up by restaurants. He searches for “DonDinis Burger” in the search bar at the top right corner of the screen and is greeted with the various menus available, after selecting the Burger menu he clicks the checkout icon also at the top right corner of the screen and confirms the order. He is redirected to a window showing the status of his order and after some time and he receives a notification in said page to get his meal.

Peter decides to register his app “ColchonEnterprise” with LogisiticsMarshall - Peter starts by registering at LogisticsMarshall, he goes to the register section of the website and filles his business details, the admin team confirms this request by using the management side of the website. After being informed his credentials are in by email, Peter immediately starts integrating his app with LogisticsMarshall by using its API to post delivery requests, checking their status and checking the state of our drivers.

Alejandro decides to register himself as a driver in LogisticsMarshall - Alejandro registers at the LogisticsMarshall website by going to the driver register section and fillling out his personal details, the admin team after that confirms this request by using the management side of the website. After being informed his credentials are in by email, Alejandro can now check for new delivery requests attributed to him in his profile page.

Alejandro looks for tasks to perform – Alejandro logs into his account, looks at the tab for available deliveries for a list of nearby tasks suited to him, chooses one and tries to assign it to himself.
He is successful and must now complete the task.

2.4 Project epics and priorities

Full delivery worker usability – LogisticsMarshall
Drivers can register, accept and perform tasks for pay, and view their own performance stats.
Full customer-side usability – LogisticsMarshall
Customers can sign up, edit their profile, set up delivery requests and see information about their assigned delivery workers.
Admin features – LogisticsMarshall
Admin can view stats for all entities, ban or remove customers and workers based on transgressions.
Admin features – MarchingFood
Admin can register new products on the website.
Customer features - MarchingFood
Customer can sign in, make orders, monitor order delivery status and rate delivery worker.

Iteration 1
Develop the architecure for both MarchingFood and LogisticsMarshall
Plan functional and non functional requirements
Design main use cases for both apps
Design APIs for LogisticsMarshal
[FInish or remove]

Iteration 2
Start unit tests for LogisticMarshall
Start integration tests for LogisticMarshall
Start development of backend component of LogisticsMarshall
Start development of frontend components of LogisticsMarshall
Start unit tests for MarchingFood
Start integration tests for MarchingFood
Start development of backend component of MarchingFood
Start development of frontend components of MarchingFood

Iteration 3
Continuation of LogisticMarshall and MarchingFood testing
Finish development for LogisticMarshall
Finish development for MarchingFood

3 Domain model

On **LogisticsMarshall** we associate users with an instance of Driver or Company.
Both are associated to deliveries, either as the worker or as the request.
Payment has been envisioned as a monthly event based on deliveries performed and is thus not represented on this database schema.
Every delivery instance may have an associated Driver review.

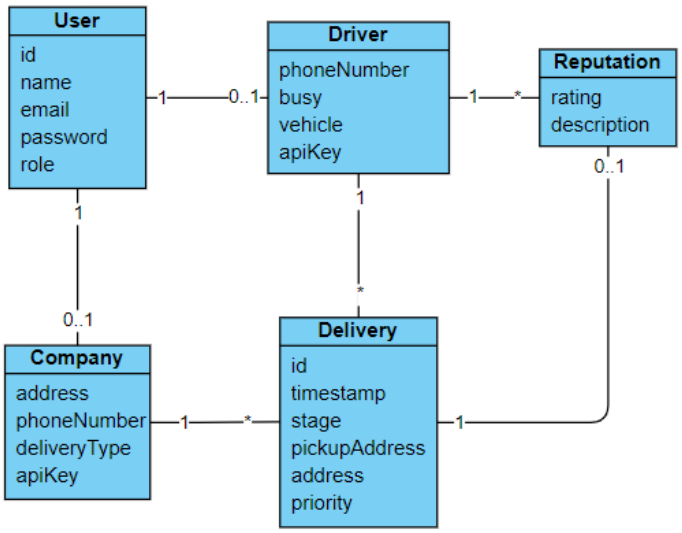

Fig 4: Logistics Component Data Schema

MarchingFood has its own database, focused around storing available menus, Users which can only be Customers or Admins, and managing business transactions on the customer side.

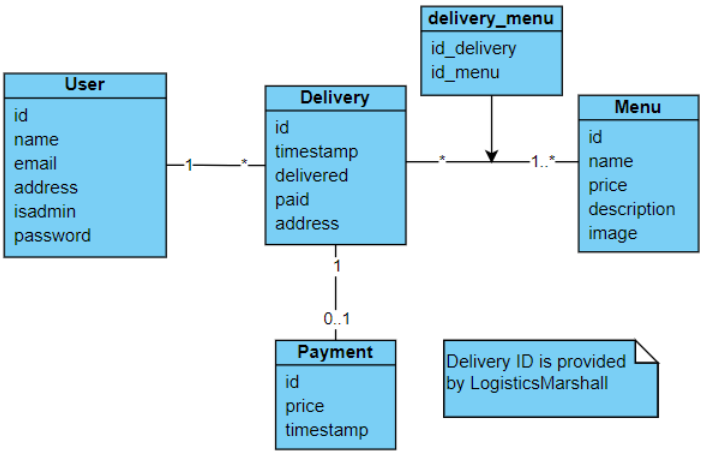

Fig 5: Client Component Data Schema

4 Architecture notebook

4.1 Key requirements and constrains

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- The delivery worker part of the web portal must be easy to use on mobile
- The system must be stable
- The system should be easy to use
- The system must not allow 2 workers to be assigned to the same task
- The system must not allow a task to be unperformed indefinitely
- The system must log transactions accurately
- The reputation system must notify administration of underperforming staff
- The separate servers should be deployable in different devices

4.2 Architectural view

We created 2 simple spring boot applications with added spring security configurations, based on PostgreSQL databases.
Each component has a series of controllers that relies on services to perform tasks.
The client app queries the Logistics provider for certain tasks, such as posting a new delivery via RestTemplate.
We provide an API for both riders and companies, where authentication is based on API access keys.

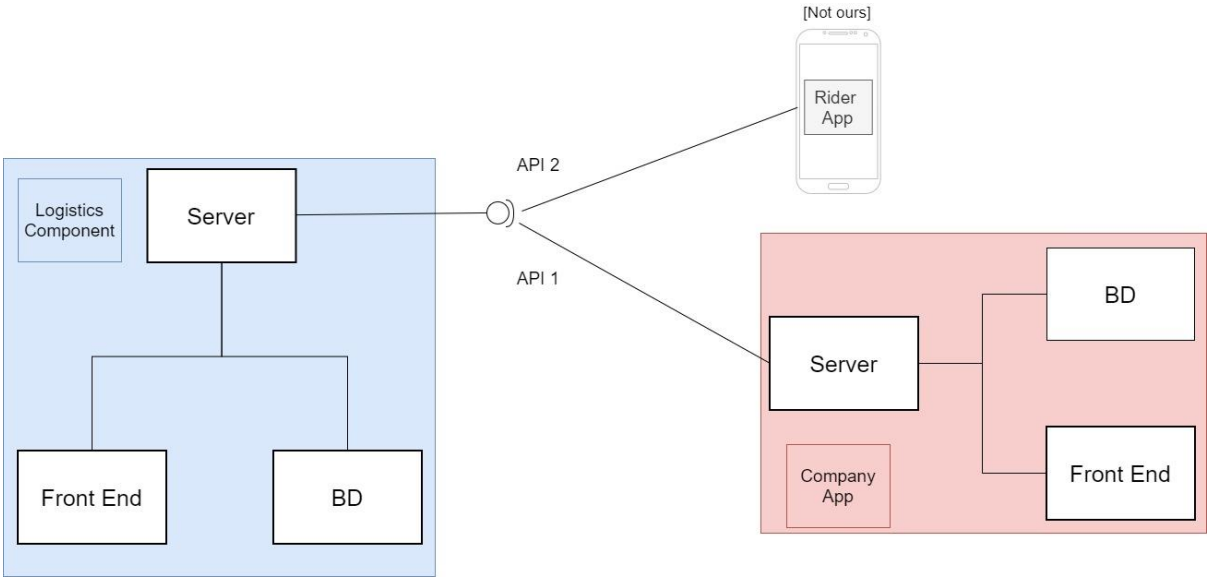

Fig 6: Architectural diagram

4.3 Deployment architecture

Our solution will deploy the logistics provider as 2 Docker containers via docker-compose , one with the actual backend server and one with the database that will provide persistence for server data.

The service depending on our logistics provider can be deployed in the same manner, either on the same host or on a different one.
The dependant service will be able to communicate with the logistics provider via a REST API.

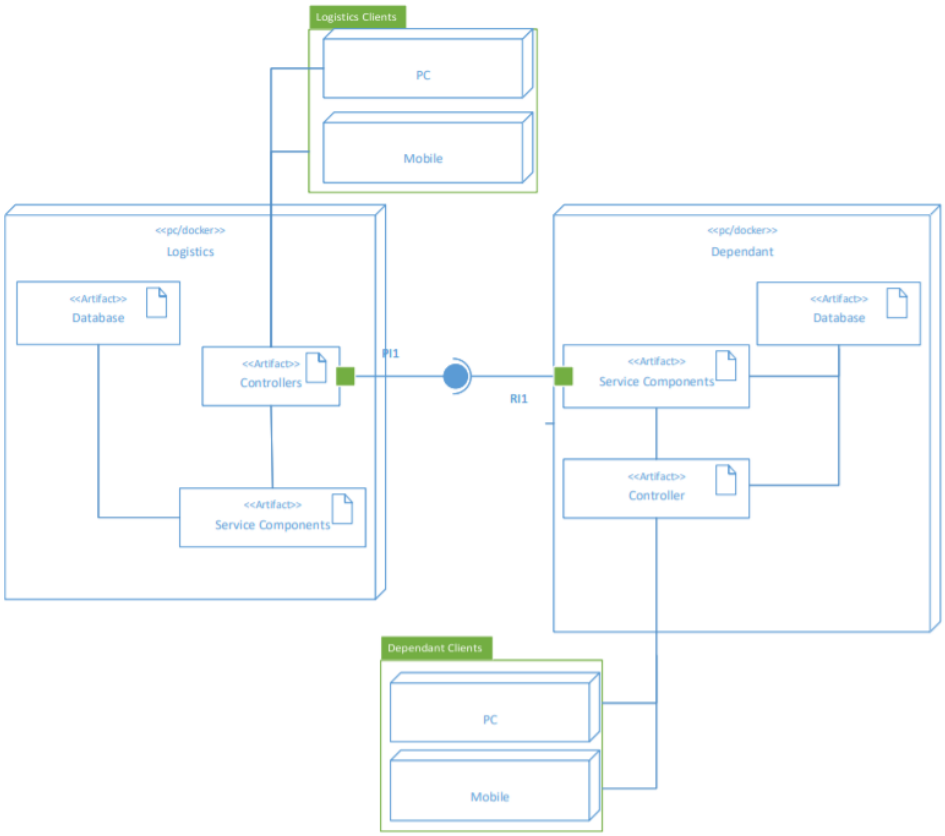

Fig 7: Deployment diagram

5 API for developers

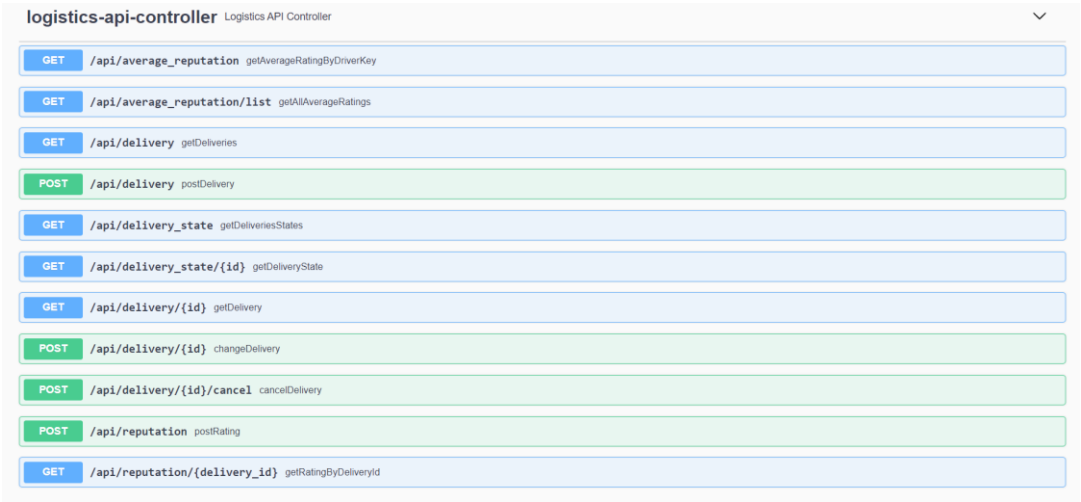

Fig 8: LogisticsMarshall API endpoints

6 References and resources

- RestTemplate is useful for querying APIs
- Thymeleaf is a satisfactory template engine for designing web pages
- Spring Security greatly eases the creation of an authentication system
- MockMvc is very useful for testing Spring controllers
- Mockito
- Watchtower helps continuously deploy your system, assuming Docker is being used
- SonarCloud greatly helps with monitoring quality, especially when aided by Git Actions
- PivotalTracker is a good task management dashboard