TQS: Product specification report

Gonçalo Silva [103668], João Matos [103182], Pedro Rasinhas [103541]
v2022-05-19

1.1 Overview of the project....1
1.2 Limitations....1
2.1 Vision statement....2
2.2 Personas and scenarios....4
2.3 Project epics and priorities....8
4.1 Key requirements and constraints....9
4.2 Architectural view....9
4.3 Deployment architecture....10

1 Introduction

1.1 Overview of the project

This project assignment, which is part of the TQS course, attempts to create a workable software product while using a Software Quality Assurance (SQA) approach. Additionally, we are asked to distinguish between the specialized region of the store (that uses the pickup points platform) and the general elements of many businesses (pickup points and logistical services).

Our software, "LoadConnect," is a software solution establishing and managing a network of pickup points as a service. It aims to streamline and optimize the process of order collection and package pickup for businesses and their customers.

1.2 Limitations

We found some limitations in backend module:

LoadConnect-Core

- Query by nearby locations is implemented in the backend (endpoint) but not implemented in frontend
- Lack of locker implementation.
- Notifications endpoints created

2 Product concept

2.1 Vision statement

LoadConnect will provide a core API that allows companies to integrate and implement a pickup points mechanism for their goods. The system enables companies to provide their consumers the flexibility of choosing a pickup place for their orders, such as a partner store or a parcel locker. By implementing *LoadConnect*, companies can streamline their delivery operations, enhance customer satisfaction, and improve overall logistics efficiency.

2.1.1 Use Cases

LoadConnect (Partner Store)

#	Use Cases	Description
1.1	Partner Store Registration	Partner Stores should be able to register on our platform;
1.2	Partner Store Login	Partner Store should be able to login on our platform, to be able to access all the functionalities and features provided;
1.3	Pickuppoint registration	A Partner Store should be able to register a new pickup point;
1.4	Check Orders	Pickup Points should be able to check their current orders;
1.5	Change order status	When the order reaches the pickup point and is ready to be picked up the Pickup Point should be able to change the status of the order.
1.6	Check Notifications	Pickup points should be able to check their notifications such as “New order incoming!”;

LoadConnect(admin)

#	Use Cases	Description
1.1	Admin Login	Admin should be able to login to access their available resources
1.2	Check all current pickup points	Admin should be able to check a list with all the current pickup points registered in the platform
1.3	Check pickup points details	Admin should be able to know all the current pick up details, such as location and their orders history.
1.4	Check partner stores	Admin should be able the partner stores;
1.5	Check orders;	Admin should be able to check all the orders;
1.6	Manage pickup points;	Admin should be able to accept/decline pickup points.
1.7	Check notifications	Admin should be able to check their notifications, such as, for example “New pickupoint registered”;

eStore

#	Use Cases	Description
1.1	Client Registration	Clients should be able to register an account
1.2	Client Login	Clients should be able to log in into their accounts to access a client's related features.
1.3	Customize Profile	Clients should be able to customize/change their profile information
1.4	Add Products to Cart	Clients should be able to add products to their cart
1.5	Filter Products	Clients should be able to filter products based on the price and name;
1.6	Check Cart Status	Clients should be able to check the contents of their cart at any time
1.7	Order Products	Clients should be able to place an order for the products that are already placed in the cart
1.8	Select pickup point	On checkout, the client should be able to select the service
1.9	Check Order Status	Clients should be able to check any order status
1.10	Check Order History	Clients should be able to view their order history
1.11	Check Notifications	Clients should be access their notifications;

2.1.2 Requirements gathering

The concept and requirements of *LoadConnect* were developed by gathering insights from similar services, such as *Pickupp* and *Upela*, as well as conducting online research to identify the most beneficial features for pickup point services.

Requirements:

PPs Platform:

- Accept a package;
- Manage (check in/out) packages;
- Manage partner stores / lockers;
- Manage partner stores rating and reviews;

Store (client):

- Package shopping;
- Provide details on the pickup point;
- Search filters (name, location, …)
- Profile customization and Registration;
- Review & rate the pickup point;

2.2 Personas and scenarios

Amy Adams

Job Title

ByteWave manager

Age

25

Highest Level of Education

12th grade

Goals or Objectives

Amy is not being able to get clients to her store so she is trying to bring the store to them

Stories

- Register her store in the platform
- Check Current Orders

Jonathan Majors

Job Title

Programmer

Age

33

Highest Level of Education

Masters Degree

Goals or Objectives

Jonathan is an introvert programmer and wants to buy a new monitor interacting as little as possible with other people

Stories

- Register in ByteWave application
- Set personal data
- Order Monitor from the store
- Track Order

Dave Smith

Job Title

CEO of DxD

Age

57

Highest Level of Education

Masters Degree

Goals of Objective

To get an extra income, Dave decided to allow his store, in Aveiro, to work as a Pickup Point.

Stories

- Register his store as a new pickup point on LoadConnect.
- Change orders status;

Joe Spielberg

Job Title

Employee at DxD

Age

28

Highest Level of Education

12th Grade

Goals or Objectives

To be able to keep his job and his income, Joe has to adapt to the new functionality of DxD, acting as a pickup point.

Stories

- Check in orders;
- Check out orders;

Martin Scorsese

Job Title

LoadConnect Admin

Age

54

Highest Level of Education

Bachelor's Degree

Goals or Objectives

Martin needs to make an overview of the LoadConnect app to show to the CEO of the company

Stories

- Check current statistics

Jonathan wants to place an order

In the last few weeks, Jonathan has been thinking of upgrading his monitor so checks ByteWave's website to check their current offers.

Martin wants to make an report/overview of the LoadConnect app

After making login as an Admin he can check any stats that he wants, since the number of partners to profit made.

Dave wants to make an extra income

Dave decides to register his store on the LoadConnect service, to act as a pickup point.

Joe wants to keep his job

Joe wants to keep his job so he has to adapt to the new functionality of DxD, acting as a pickup point.

Amy, ByteWave's manager uses LoadConnect's services

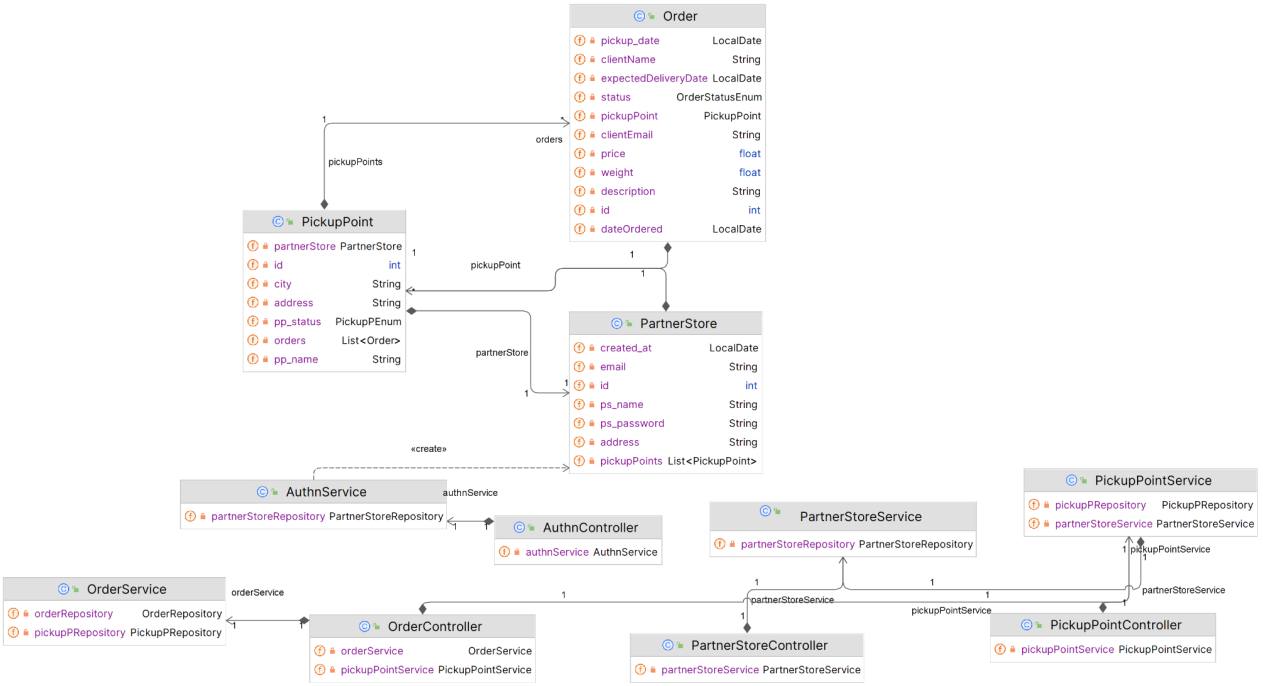
As the owner of the store, she is trying to reach more clients to her tech store so she decides to use the LoadConnect's services.

2.3 Project epics and priorities

Sprint	Epics	Tasks
Sprint 1 (4th to 11th May)	Documentation	Define system architecture. Define system requirements. Create Main Scenarios. Create Personas. Create Domain Model.
Sprint 2 (11th to 18th May)	Documentation; Core Frontend; Service Frontend; Core Backend; Service Backend;	Configure CI for the different projects. Draft the QA manual. Implement the entire core frontend; Implement the entire service frontend; Implement 'Pickup Point Registration'; Implement 'Pickup Point Login'; Implement 'Admin Login'; Implement 'Client Registration'; Implement 'Client Login'; Implement 'Check all current pickup points'; Implement 'Check pickup points details'; Implement 'Check Revenue';
Sprint 3 (18th to 25th May)	Documentation; Core Frontend; Service Frontend; Core Backend; Service Backend;	Configure the CD pipeline; Finish the QA manual; Implement 'Add Products to Cart'; Implement 'Filter products'; Implement 'Check Cart status'; Implement 'Order products'; Implement 'Select pickup point'; Implement 'Check Order Status';
Sprint 4 (25th May to 1st June)	Core Frontend; Service Frontend; Core Backend; Service Backend;	Implement 'Change Order status'; Implement 'Check Orders History'; Implement 'Customize Profile';
Sprint 5 (1st to 6th June)	Documentation; Core Frontend; Service Frontend; Core Backend; Service Backend;	Finish Product Specification report (1.2); Implement 'Check Notifications', for all entities;

3 Domain model

- LoadConnect Core - Final



4 Architecture notebook

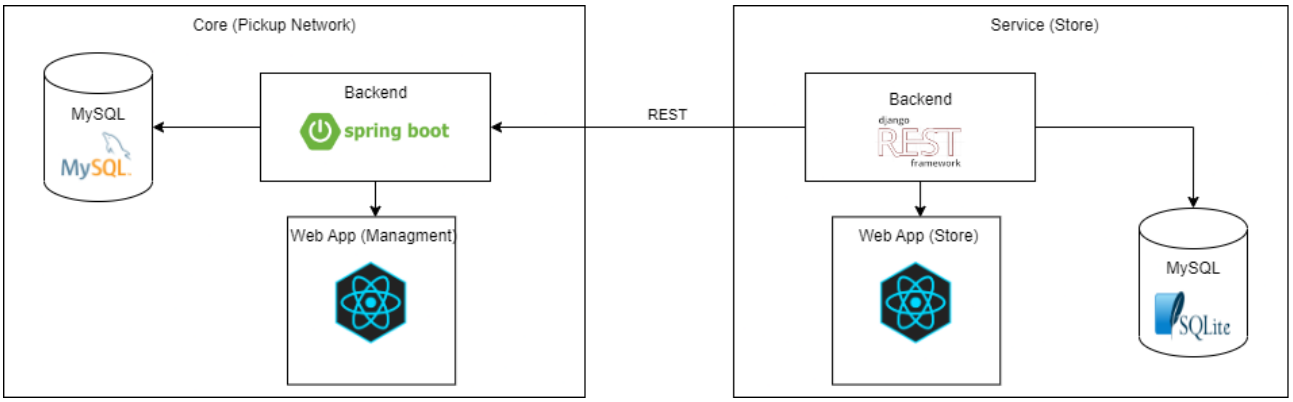
4.1 Key requirements and constraints

One of the main requirements for our system is the ability to provide a general API that can be used by several small or medium businesses to provide pickup points to their clients. This means our API needs to be scalable and be able to support the load generated by all the partner businesses.

This being said, there are some key requirements and system constraints that have a direct impact on the architecture. These are:

- There are different users with different authorizations;
- The system should be able to persist the data;
- The business should be able to communicate with the partner stores (B2B), using a REST API.

4.2 Architectural view



The architecture of our system is fairly simple. It's separated into two main modules, the core module, that is the one that will implement the Pickup Points Platform, i.e., the central API that provides the services to other businesses. The other module, called Service, is the representation of a business (an e-store) that has its own architecture but wants to use our Core API to provide pickup points to their customers.

Each of these modules have their own submodules.

The core module is divided into:

- Backend: Using spring boot;

The backend submodule is responsible to expose the REST API that will be then consumed by the Frontend submodule;

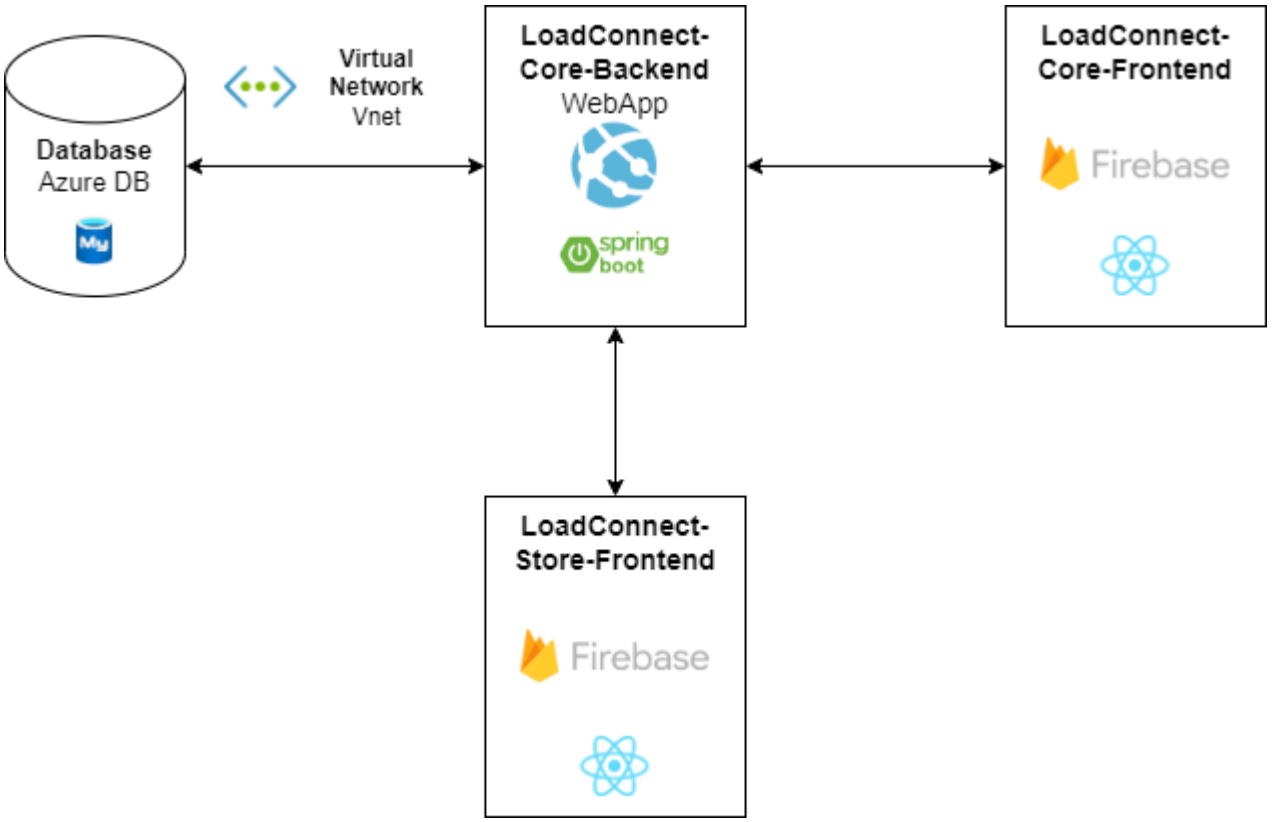
- Database: Using MySQL;

This layer is responsible for the data persistence.

- Frontend: Using ReactJS;

This architecture was our first draft and was not fully realized in its entirety.
The core module was fully implemented, as described above, but the service module was not.
The service module was not fully implemented as described on the architecture diagram because the team member responsible for that part failed to fulfill his responsibilities and neglected to complete his assigned tasks.
So what we did was create a small react page that allowed us to post orders and check the orders based on a user's email. This was not the expected outcome but it was somehow enough to show the minimum flow of an interaction with an e-store.

4.3 Deployment architecture



Our project is divided into 3 different modules: Core-Backend, Core-Frontend, Store-Frontend. Both frontends were deployed with firebase cloud service and that deployment is made automatically when we push a commit to the main branch. All that workflow is executed through GithubActions.

The Core-Backend we opted to use the Microsoft Azure service and created a WebApp in the service. Also in Azure, we create a database (AzureDb) which is connected to the Core-backend through a Virtual Network (Vnet). Even though the backend is deployed through github actions when there is a push or merge to main, the database is persistent.

5 API for developers

Documentation of all the controllers.
Note that the link is only for the core API, as the store was just not fully implemented, as discussed above.

Core API: https://loadconnect.azurewebsites.net/swagger-ui/index.html

6 References and resources

Documenting API :
https://swagger.io/resources/articles/documenting-apis-with-swagger/

What's a deployment diagram? :
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/

How to use Java DTOs :
https://snyk.io/blog/how-to-use-java-dtos/

Hosting with Firebase :
https://firebase.google.com/docs/hosting/quickstart?hl=pt-br

Configure CI/CD with Github Actions- Azure App service :
https://learn.microsoft.com/en-us/azure/app-service/deploy-github-actions?tabs=applevel