INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Plataforma de Logística aplicada à Prestação de Serviços

EMANUEL FERNANDO PAIVA DA SILVA MARQUES
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Logistics Platform on Service Provision

Emanuel Fernando Paiva da Silva Marques

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialization Area of Software Engineering

Supervisor: Prof. António Rocha

Evaluation Committee:
President:

Members:

Porto, October 11, 2019

Dedicatory

This work is dedicated to my parents, for all the work they had with me and everything they taught me. They made me who I am today and I will be forever grateful for the excellent education they gave me.

My greatest sincere thank you!

Dedicatória

Este trabalho é dedicado aos meus pais, por todo o trabalho e dedicação que tiveram comigo e tudo o que me ensinaram. É graças a eles que hoje sou quem sou, e estarei para sempre grato pela excelente educação que eles me deram.

O meu mais sincero obrigado!

Abstract

Society is increasingly tied to technology. If you look around you, you see countless pieces of engineering to which people are becoming more and more accustomed and outstanding. This is due to the efficiency with which technological solutions respond to the needs of their users. Nowadays it is possible to do almost everything online: buying clothes, traveling, ordering food and being delivered at home, are examples of things that were unthinkable 30 years ago.

The e-commerce brought a revolution that has shaken many industries. From music to retail, through to services, there was none to which the appearance of this new kind of business was indifferent. The possibility of selling online to any part of the world made it possible for even small businesses to export and compete with large companies.

However, the application of this type of trade to services is not as strong as when compared to the sale of products. There are many more platforms for selling products via the Internet than platforms for providing services.

With this, some companies started to provide this service, but without using technology. All the management of orders and receipts is done by telephone, and the records made in paper or on an Excel sheet, which causes employees to spend time on tasks that are not their area of expertise.

This thesis aims at creating a platform to meet these needs by providing a range of tools to facilitate the work of service providers. To achieve this, a platform capable of supporting multiple service types and multiple marketplace service providers will be designed and developed.

This platform will be composed of several applications, from the platform's main site and back-office application, to mobile applications for couriers. These are aimed at a segmented response to the needs of the three existing types of customer: service providers, remittance agents and, of course, the final customer.

In this context, a pilot laundry will be used as the pilot, which will lead to the evaluation and testing phase.

Keywords: Logistics, Services, Micro-Services, Architecture, E-Commerce

Resumo

A sociedade está cada vez mais ligada à tecnologia. Se olharmos à nossa volta, vemos inúmeras peças de engenharia, às quais as pessoas estão cada vez mais habituadas e dependentes. Tal acontece devido à eficiência com que as soluções tecnológicas respondem às necessidades dos seus utilizadores. Nos dias de hoje é possivel fazer praticamente tudo via internet: comprar roupa, viagens, encomendar comida e esta ser entregue em casa, são exemplos de coisas que há 30 anos era impensável que se conseguisse.

O e-commerce trouxe uma revolução que abanou muitas indústrias. Da música ao retalho, passando pelos serviços, não houve nenhuma ao qual o aparecimento deste novo tipo de comércio fosse indiferente. A possibilidade de vender online para qualquer parte do mundo possibilitou que mesmo pequenos negócios pudessem exportar e competir com as grandes empresas.

No entanto, a aplicação deste tipo de comércio a serviços, ainda não é tão forte como quando comparado à venda de produtos. Existem muito mais plataformas de venda de produtos via internet, do que plataformas de prestação de serviços.

Com isto, algumas empresas começaram a prestar esse serviço, mas sem recurso à tecnologia. Toda a gestão de pedidos e estafetas é feita via telefone, e os registos feitos em papel ou numa folha de Excel, o que faz com que os funcionários gastem tempo em tarefas que não são a sua área de especialização.

Esta tese visa a criação de uma plataforma que dê resposta a estas necessidades, providenciando uma série de ferramentas que facilite o trabalho dos prestadores de serviços. Para tal, será idealizada e desenvolvida uma plataforma capaz de suportar múltiplos tipos de serviços e múltiplos prestadores de serviços, ao estilo marketplace.

Esta plataforma será composta por várias aplicações, desde o site principal da plataforma e aplicação back-office, até aplicações móveis para os estafetas. Estas têm como objetivo dar uma resposta segmentada às necessidades dos três tipos de cliente existentes: os prestadores de serviços, os estafetas e, obviamente, o cliente final.

Neste contexto, será tido como cliente piloto uma lavandaria, que conduzirá a fase de testes e avaliação.

Acknowledgement

Since this is very likely to be the end of my academic education, I feel that I should take this opportunity to thank all the people who have come across me over the last eighteen years of study and have, somehow, contributed to my academic, professional and personal education.

I begin by thanking all the elementary and secondary school teachers who taught me the various disciplines of knowledge. They showed me how the world has come to this day and taught me how to speak a language without which I couldn't communicate with most people on this planet. To do head counts in order to gain logical reasoning and proved me that every action has a reaction. Essentially they taught me to think.

The various professors I came across during this journey at Instituto Superior de Engenharia do Porto, who not only taught me how to be a software engineer, taught me how to be a good engineer, showing me the best practices of the industry and taught me to have a critical attitude when faced with new problems.

To all my colleagues with whom I took this walk side by side, especially the colleagues I came across in designing and developing the Núcleo de Estudantes de Informática do ISEP. They allowed the development of a range of skills that are not possible to be learned in the classroom.

To all my family, who has always been by my side, to celebrate the good times and to hug me in the least good ones. The fact is, I'm lucky to have such people around me.

To my thesis advisor, Prof. António Rocha for, besides having taught me a lot in the classes in which he was my teacher, he accepted to guide me through this challenge.

To the fantastic people I came across at Farfetch for helping me grow so much as a professional over so little time. A very special thank you to Fernando Costa for being my adopted mentor and for continuing to be a great colleague and friend who readily offered to support me in this project.

To everyone who somehow contributed to my professional success in one way or another without you was not the person I am today.

Thank you very much.

Emanuel Marques

Agradecimentos

Sendo este o muito provável terminar da minha formação académica, sinto que devo aproveitar esta oportunidade para agradecer a todas as pessoas que se foram cruzando comigo ao longo desta caminhada de mais de dezoito anos de estudo e de alguma forma contribuíram para a minha formação académica, profissional e pessoal.

Começo por agradecer a todos os professores do ensino básico e secundário, que me ensinaram as várias disciplinas do conhecimento. Que me mostraram como o mundo chegou aos dias de hoje e me ensinaram a falar uma língua sem a qual não conseguia comunicar com a maioria das pessoas deste planeta. Que me ensinaram a fazer contas de cabeça de forma a conseguir obter um raciocínio lógico e me provaram que cada ação tem uma reação. Em suma, ensinaram-me a pensar.

Aos vários professores com os quais me fui cruzando ao longo desta jornada pelo Instituto Superior de Engenharia do Porto, que não só me ensinaram a ser engenheiro de software, ensinaram-me a ser um bom engenheiro, mostrando-me as melhores práticas da indústria e a ter uma atitude crítica quando confrontado com novos problemas.

A todos os meus colegas com os quais dei esta caminhada lado a lado, em especial aos colegas com que me cruzei na conceção e crescimento do Núcleo de Estudantes de Informática do ISEP. Estes permitiram o desenvolvimento de uma série de competências que não se aprende em sala de aula.

A toda a minha família, que sempre esteve ao meu lado, para celebrar os bons momentos e para me abraçar nos menos bons. Facto é que sou um sortudo por ter tais pessoas perto de mim.

Ao meu orientador de tese, Prof. António Rocha por, para além de muito me ter ensinado nas aulas que foi meu professor, ter aceite orientar este desafio.

Às fantásticas pessoas com que me cruzei na Farfetch por terem-me ajudado a crescer tanto enquanto profissional num espaço de tempo. Um obrigado muito especial ao Fernando Costa, por ter sido o meu mentor adotado e por continuar a ser um grande colega e amigo que prontamente se disponibilizou para me apoiar neste projeto.

A todos os que de alguma forma contribuíram para o meu sucesso profissional, de uma forma ou de outra, sem vocês não era a pessoa que sou hoje.

Muito obrigado.

Emanuel Marques

Contents

List of Figures **xix**

List of Tables **xxi**

List of Source Code **xxiii**

List of Acronyms **xxv**

1 Introduction **1**
1.1 Context . . . 1
1.2 Problem Statement . . . 2
1.3 Objectives of the Project . . . 2
1.4 Document Structure . . . 2

2 Context **5**
2.1 E-Commerce Industry . . . 5
2.2 E-Commerce applied to services . . . 6

3 State of the Art **7**
3.1 Existing Solutions . . . 7
3.1.1 UberEats . . . 7
3.1.2 Glovo . . . 8
3.2 Technology . . . 9
3.2.1 Architecture . . . 9
Service Oriented Architecture . . . 9
Micro-Services Architecture . . . 10
Command Query Responsibility Segregation . . . 11
Event Sourcing . . . 13
3.2.2 Software Development Frameworks . . . 13
.NET Core . . . 14
Java Spring . . . 15
Node.js . . . 16
3.2.3 Swagger . . . 18

4 Value Analysis **21**
4.1 New Concept Development . . . 21
4.1.1 Opportunity Identification and Analysis . . . 23
4.1.2 Idea Genesis and Selection . . . 23
Idea Selection using the Analytic Hierarchy Process . . . 23
4.1.3 Concept & Technology Development . . . 25
4.2 Value Proposition . . . 25
4.2.1 Value Proposition for Final Customers . . . 26

4.2.2 Value Proposition for Service Providers . . . 27
4.2.3 Value Proposition for Couriers . . . 27
4.3 Value for the Customer . . . 27

5 Requirements Engineering **31**
5.1 User Groups . . . 31
5.2 Functional Requirements . . . 32
5.2.1 FF01 - Register as Customer . . . 32
Requirement Description . . . 32
5.2.2 FF02 - Apply as a Service Provider . . . 33
Requirement Description . . . 33
5.2.3 FF03 - Apply as a Courier . . . 33
Requirement Description . . . 33
5.2.4 FF04 - Log In . . . 34
Requirement Description . . . 34
5.2.5 FF05 - Recover Password . . . 34
Requirement Description . . . 34
5.2.6 FF06 - Place an order for a service . . . 34
Requirement Description . . . 34
5.2.7 FF07 - Check all available service providers . . . 35
Requirement Description . . . 35
5.2.8 FF08 - Check order history . . . 35
Requirement Description . . . 35
5.2.9 FF09 - Evaluate Experience . . . 35
Requirement Description . . . 35
5.2.10 FF10 - Manage available services . . . 36
Requirement Description . . . 36
5.2.11 FF11 - Manage orders . . . 36
Requirement Description . . . 36
5.2.12 FF12 - Change Area of Actuation . . . 37
Requirement Description . . . 37
5.2.13 FF13 - Change service profile . . . 37
Requirement Description . . . 37
5.2.14 FF14 - Assign orders for pickup . . . 37
Requirement Description . . . 37
5.2.15 FF15 - Manage Service Providers . . . 38
Requirement Description . . . 38
5.2.16 FF16 - Manage Couriers . . . 38
Requirement Description . . . 38
5.3 Non-Functional Requirements . . . 38
5.3.1 Usability . . . 38
5.3.2 Security . . . 38
5.3.3 Availability . . . 39
5.3.4 Scalability . . . 39
5.3.5 Resilience . . . 39
5.3.6 Performance . . . 39
5.3.7 Internationalization and Localization . . . 39
5.3.8 Maintainability . . . 40
5.4 Process View . . . 40

5.5 Order Process States 42

6 Analysis of the Existent Solutions **45**
6.1 Architectural Approaches 45
6.1.1 Micro-Services 45
6.1.2 Event Sourcing 46
6.1.3 CQRS 46
6.1.4 Summary 47
6.2 Software Development Frameworks 47
6.2.1 .NET Core 48
6.2.2 Java Spring 48
6.2.3 Node.js 48
6.2.4 Summary 49

7 Design and Architecture **51**
7.1 Architectural Study 51
7.2 Domain Model 54
7.3 Internal Project Structure 56
7.3.1 Presentation 57
7.3.2 Application 57
7.3.3 Domain 57
7.3.4 Data 57
7.3.5 Infrastructure 58

8 Implementation **59**
8.1 Code Structure 59
8.2 Price Calculation 59
8.3 Code Quality 62
8.4 User Interfaces 63
8.4.1 SnapTasks Portal 63
Login Page 64
Service Provider Listing Page (SPLP) 64
Service Listing Page (SLP) 65
Service Display Page (SDP) 65
Checkout Page 66
Order Details Page 67
Order History Page 68
8.4.2 SnapTasks Back-Office Management 69
Service Provider Management Pages 69
Courier Management Pages 70
Order Management Pages 70

9 Evaluation of the Solution **73**
9.1 Evaluation Methodology 73
9.2 Evaluation Results 74
9.3 Usability Surveys 75
9.3.1 Customer Perspective Survey 75
How do you rate the ease of registration and log in? 75
How do you rate the presentation of service providers? 75
How do you rate a service provider's presentation of services? . . . 75

Regarding the detail given on the page of a service, was the information sufficient? . . . 75
Was the price of a given service clear to you? . . . 76
How do you rate the ease of placing an order? . . . 76
Was the data presented about the order placed (after purchase) clear? 76
How do you rate the information given in the order history? . . . 76
Summary . . . 76
9.3.2 Service Provider Perspective Survey . . . 77
How do you rate the ease of log in? . . . 77
How do you rate the presentation of service providers? . . . 77
How do you rate a service provider's presentation of services? . . . 78
Regarding the detail given on the page of a service provider, was the information sufficient? . . . 78
How do you rate the ease of updating the price of a service? . . . 78
Was the presentation of the service data clear? . . . 78
Was the price presentation clear? . . . 78
How do you rate the information given in the order history? . . . 79
How do you rate the information given in the details of an order (click "Details")? . . . 79
How do you rate the ease of changing an order status? . . . 79
Summary . . . 79
9.3.3 Results . . . 80

10 Conclusion **81**
10.1 Achievements and Results . . . 81
10.2 Future Work . . . 82

Bibliography **83**

A Thesis Formalization **87**

B QEF Solution Evaluation **91**

C Business Model Canvas **97**

D Customer Perspective Inquiry **99**

E Service Provider Perspective Inquiry **103**

F Customer Perspective Inquiry - Answers **107**

G Service Provider Perspective Inquiry - Answers **113**

List of Figures

2.1 Worldwide e-commerce sales growth from 2014 to 2021 6

3.1 Mobile share of total e-commerce 2016 to 2021 8
3.2 Service Oriented Architecture Example 10
3.3 Microservices Architecture Example 11
3.4 CQRS Typical Implementation 12
3.5 Event Driven Architecture using Kafka 13
3.6 Top 5 most used IDEs by Web developers 15
3.7 Node.js Thread management 17
3.8 Back-Office API Endpoints on Swagger page 19
3.9 Open endpoint on Swagger 19

4.1 NCD model 22
4.2 Hierarchy Decision Tree 24
4.3 Longitudinal Perspective on Value for the Customer (VC) 28

5.1 Use Case Diagram 31
5.2 Process diagram 41
5.3 Order State diagram 43

7.1 Possible architectural solution (Monolith) 52
7.2 Separation of FO and BO APIs 52
7.3 Proposal of a Micro-Services Architecture 53
7.4 SnapTasks Domain Model 55
7.5 Generic representation of the internal project structure 56

8.1 Code structure on Visual Studio 60
8.2 Sequence diagram for price update 61
8.3 Codacy grade to SnapTasks projects 63
8.4 SnapTasks login page 64
8.5 SnapTasks service provider listing page 64
8.6 Service Listing Page of a Service Provider 65
8.7 SnapTasks service display page 65
8.8 SnapTasks checkout page 66
8.9 SnapTasks order details page 67
8.10 SnapTasks order history page 68
8.11 List of service providers page 69
8.12 Service Provider details page 70
8.13 Courier list page 70
8.14 Order Processing Page 71
8.15 Order details page 71

9.1 Summary of results of the Customer Perspective Survey 77
9.2 Summary of results of the Customer Perspective Survey 80

List of Tables

4.1 AHP Evaluation Table 24
4.2 Weights of each criteria 25
4.3 Contingent VC Temporal Stages Values mapping 29

6.1 Summary of Pros and Cons of each evaluated architecture 47
6.2 Summary of Pros and Cons of each evaluated software development framework 49

List of Source Code

3.1 Callback Hell example[1] 18

List of Acronyms

AHP	Analytic Hierarchy Process.
AoA	Area of Actuation.
AOP	Aspect Oriented Programming.
ASP	Active Server Pages.
BO	Back-Office.
CBAM	Cost-Benefit Analysis Method.
CLR	Common Language Runtime.
CPU	Central Processing Unit.
CQRS	Command Query Responsibility Segregation.
CRUD	Create, Read, Update and Delete.
CVP	Customer Value Proposition.
DBO	Database Object.
DDD	Domain Driven Design.
DLL	Dynamic Link Library.
DTO	Data Transfer Object.
ESB	Enterprise Service Bus.
FCL	Framework Class Library.
FEI	Front End of Innovation.
FFE	Fuzzy Front End.
FO	Front-Office.
GPS	Global Positioning System.
GUI	Graphic User Interface.
HTTP	Hypertext Transfer Protocol.
I/O	Input/Output.
IDE	Integrated Development Environment.
IoC	Inversion of Control.
JDK	Java Development Kit.
JPA	Java Persistence API.
JS	JavaScript.
JVM	Java Virtual Machine.
KLOC	Thousand Lines Of Code.

MVP	Minimum Viable Product.
NCD	New Concept Development.
NPD	New Product Development.
NPPD	New Product and Process Development.
ORM	Object-Relational Mapper.
OS	Operating System.
PoC	Proof of Concept.
PoP	Point of Parity.
QEF	Quality Evaluation Framework.
RAM	Random Access Memory.
ROI	Return on Investment.
SDP	Service Display Page.
SEI	Software Engineering Institute.
SLP	Service Listing Page.
SOA	Service Oriented Architecture.
SP	Service Provider.
SPLP	Service Provider Listing Page.
SRP	Single Responsibility Principle.
UI	User Interface.
UX	User Experience.
VAT	Value Added Tax.
VC	Value for the Customer.
VCS	Version Control System.
VIP	Very Important Person.
XML	Extinsible Markup Language.

Chapter 1

Introduction

The first chapter has as its purpose the presentation and contextualization of the subject of this dissertation. Here will be presented the key concepts, indispensable for the understanding of the project. This chapter will start by an overview of the context, followed by a description of the problem and what are the main objectives of this project. Lastly, a structure of the whole document will also be presented.

1.1 Context

Consumers are looking forward to use products and services through digital platforms. As we look around ourselves, we see people more connected to technology than at any other time in history. If we scale down to just e-commerce, the behavior is very similar, and even more if we check the buying of services. This can be proved by the growth of platforms like Uber and Lyft. In just three years, the number of Uber's daily trips has grown from about 70 thousand daily trips to about 500 thousand, just in New York City. This statistical behavior is very similar to other competitors. This kind of platforms are gaining market share to more traditional business models, such as taxis (Schneider 2018). As millennials are becoming adults and gaining purchasing power, the openness of society to these platforms is estimated to grow, due to the tech savviness of the millennial generation and since they are one of the core spenders of these platforms (Lansat 2018).

With this market growth, in 2014, Uber started a new service, also in the transportation business. The objective was to create a food delivery service that would solve the problems that other food delivery platforms already on the market hadn't solved yet. UberEats was released to connect several restaurants that didn't do home delivery and sell them that service on the Uber platform. Unlike their competitors, UberEats would only deliver to a limited area, near the restaurants in the platform (Carson 2016).

This opened the door to the creation of other companies dedicated to the delivery of alternative products, like Glovo. Glovo started in 2015 in Barcelona and now is present in 22 countries and provides delivery services of several products like food, gifts, and pharmacy. This is a perfect example of a "Uber like" company that provides different services but with a similar business model.

The growth and success of this kind of platforms reveals a great acceptance by the consumers, and the lack of platforms applying this concept to other business areas brings an opportunity for those who are willing to take it.

1.2 Problem Statement

Despite the existence of several companies/platforms that provide logistic services for products using Uber's business model, none is able to make the transportation of goods that will be target of a service. An example of this are laundries that provide home pickup and delivery services.

Unlike the case of a simple transportation of goods, there is no platform on the market that enables the customer to make a purchase of a service (a laundry service, for example), enables the Service Provider (SP) to manage their orders, and provides a network of couriers to make the transportation. A platform like this could be used for different kinds of services: laundry, equipment repair, document delivery/signing, etc.

Due to the nonexistence of a platform with these features, some laundries have already started to provide a service of pickup and delivery at the customer's address. However, since this is being done mostly by the laundry calling the courier and the customer directly, and managing the orders on paper or an excel file, it entails a great amount of time and work for the laundry. Furthermore, this creates a lot of problems for the laundry. There must be at least one person assigning work to the couriers, the couriers are not informed directly of the pickup address and the routes taken by them, might also be inefficient.

1.3 Objectives of the Project

The purpose of this project is to analyze, design, propose, implement and evaluate a software solution that suppresses the needs presented on section 1.2. It should provide tools to carry out several essential business operations, like for a customer to place and pay for an order, for an SP to manage his/her orders, and for a courier to check currently unassigned orders, assign himself to it and provide information of the pickup and delivery locations.

It is expected for this solution to be prepared to support different types of services (laundry, mechanics, electronic repairs, etc.). There can also be examples of services with more than a two-leg journey (for example, repairing services which require different specialists that are located in different locations).

In this thesis, the focus will be on two leg services (where the goods are sent from the customer, to the SP and then sent back to the customer), and having the laundry use case as a Proof of Concept (PoC). This happens due to having a pilot client already on-board. However, as it was already mentioned, the final solution must support multiple SPs, with different types of services.

1.4 Document Structure

In this section, it will be presented the structure used in this thesis. Providing an overview of the whole document, it will be possible to have an idea of all the aspects that will be approached in this thesis.

This dissertation is divided into **ten** chapters.

The first chapter, *Introduction*, the reader is firstly provided by a brief contextualization, followed by a description of the problem and the objectives to be achieved along the development of the present thesis.

The next chapter, *Context*, provides a more detailed explanation of the context of the project. In this case, it provides information regarding the state of e-commerce around the world and, specifically, the application of e-commerce to services.

In the third chapter, *State of the Art*, analyzes, as its own name suggests, the literature regarding the state of the art in both existing business solutions that solve similar problems and in technology, presenting several architectures and frameworks that help to design and implement a platform of this degree of complexity. The main purpose of this chapter is to gain inspiration to solve the problem proposed by this dissertation in the best way possible.

The *Value Analysis* provides a theoretical approach to present how an opportunity is identified, followed by the appearance of an idea. Here it also analyzes the value of this project for the final customers, the service providers, and the couriers of the platform.

The fifth chapter starts the more technical half of this thesis. The chapter *Requirements Engineering* firstly presents the several user groups in the platform and the functional requirements of the project. Lastly, it presents a process view of the buy flow.

In the next chapter, *Analysis of the Existent Solutions*, a deeper analysis of the architectural approaches and frameworks is made. Here is also presented a comparison between each of them. As an outcome, there is the selection of both architectural style and development framework to be used in this project.

The *Design and Architecture* presents the study of possible architectures that respond to the platform needs. In the end of this chapter, there is also a proposal of the architecture to be implemented.

The eighth chapter presents details of the *Implementation* of the proposed solution. It starts by presenting and explaining the domain model. This enables the reader to understand the most common concepts of the domain of the platform. Next, it overviews the project structure used in the many components that compose the solution. Afterwards, there is an overview about code quality and the resultant analysis of the code of the solution regarding this matter and, lastly, the user interfaces of the platform are also presented.

In the *Evaluation of the Solution* chapter, it is presented the evaluation methodology and its results. Here some metrics are also presented regarding the accomplishment of the proposed objectives.

The last chapter, *Conclusion*, presents the achieved results and objectives. Future work, enhancements and improvements are also presented.

Chapter 2

Context

The concepts that were described in section 1.1 are essential for the correct understanding and analysis of the problem. For that reason, this chapter will present a deeper investigation on the concepts that are inherent to the business domain.

Despite this project not being about retail of goods, it is still, in fact, a project about e-commerce, since it involves a commercial trade. The start of the chapter will start by introducing what is the E-Commerce Industry and explaining how e-commerce can be applied to services.

2.1 E-Commerce Industry

By definition, "e-commerce refers to the use of electronic means and technologies to conduct commerce (sale, purchase, transfer or exchange of products, services and/or information" (Manzoor 2010). This concept is as considered as old as internet itself. However, since internet was only for military use until 1991 (Leiner et al. 1997), the commercial activity wasn't very present until that time.

Amazon was one of the first online stores. Launched in 1995, this website was initially focused just on book selling. However, Jeff Bezos (Amazon's CEO) knew from the beginning that Amazon had to be "an everything store" (Hartmans 2018). This vision propagated to other entrepreneurs, and after Amazon, many other competitors, like eBay (1995) and Alibaba (1999) appeared on the market.

The number of worldwide e-commerce sales has been increasing consistently and it is expected to continue this trend for the next years. The Figure 2.1 shows how this trend has been for the last years and how it is predicted to be in the next few years.[1]

This behavior is result of two facts. Firstly, society is more trustful on technology and the fear of fraud is more reduced now due to the current legislation, certifications given to trusted online stores and because new web-oriented payment methods having risen. The second reason is the high market competition that exists nowadays. Unlike the 90s and the early years of the 21st century, there are a lot of online stores, selling almost everything that a customer might need.

Despite the existence of ever more online stores and e-commerce platforms, the great majority of these platforms are engaged in the trade of physical products.

[1] data retrieved from https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

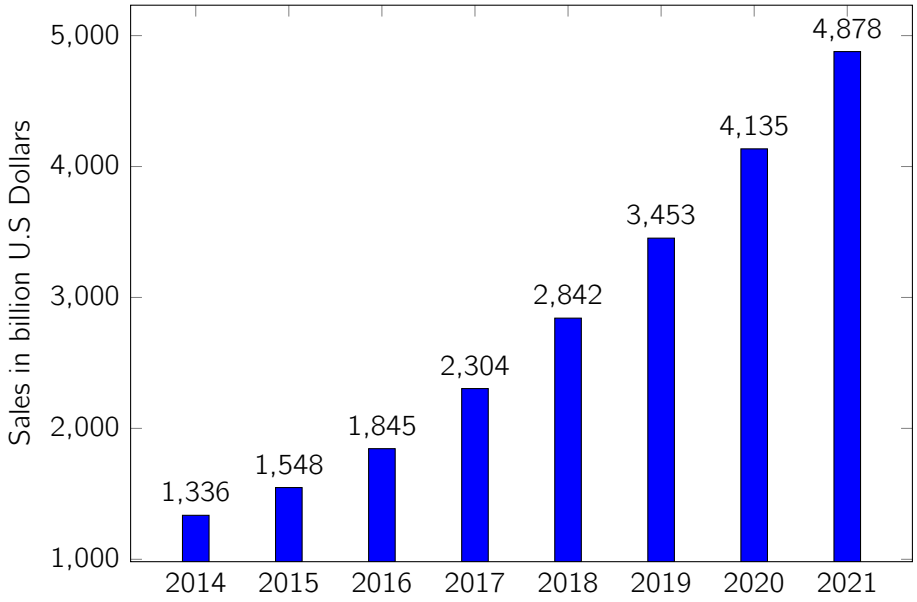


Figure 2.1: Worldwide e-commerce sales growth from 2014 to 2021

2.2 E-Commerce applied to services

As was explained on section 2.1, most of e-commerce is done for buying physical products. This is done in the so called "online stores". However, the sale of services is a more contained, niche, market within this industry. Most of the services that are sold online are typically also virtual. Examples of those services are web hosting, cloud services and email accounts. It is, nevertheless, also possible to order some services online, like ordering an insurance contract but the range of options that exist is much more restricted.

Chapter 3

State of the Art

Developing a platform that supports multiple types of services and with the ability to grow in terms of both features and traffic, requires it to be built with the best quality standards of software engineering. In this chapter, we will firstly analyze approaches of other platforms already on the market, and present the state of the art on software architecture, by researching on several architectural designs that might be applicable to this project.

3.1 Existing Solutions

In this section will be presented some existing solutions that have similar requirements to this project. The analysis of these platforms will help to understand what problems may arise, and how these companies tackled them.

3.1.1 UberEats

With the success that Uber was having in the passenger transportation sector, in 2014, the company started on the food delivery business. Initially, this was a side project that was integrated Uber's main app. The service was only available for lunch, and number of options were scarce. Note that this occurred in a time where there were already other competitors offering food delivery services, without these restrictions (Carson 2016).

UberEats is a food delivery service offered by Uber, that connects several restaurants to the customers in the restaurant's surrounding area. Contrary to what the competition was doing, UberEats limited the access to place orders to customers in zones where there were restaurants in their network (Carson 2016). This decision allowed three things: first, to increase the quality. It is impossible to keep the quality of a product after it had to go through several kilometers in order to be delivered to the customer. Second, it diminished the delivery time. People want to be able to order food minutes before lunch/dinner time. By limiting the areas where it was available, it was also possible to increase speed time. Last, but not the least, it enabled to keep a low, flat rate for all delivery services.

Uber, and by consequence UberEats, is a company that was, since the beginning, very mobile oriented. Mobile e-commerce has grown a lot in the last ten years. In fact, since 2017, it counts for over half of the whole web traffic (statista.com 2018). The rise of the smartphones and with telecommunication companies offering better and cheaper services was a key fact that enabled this rise in the mobile share. This behavior is even more noticeable when the data relative to e-commerce is analyzed. The graph in figure 3.1 refers to the

growth of the market share for mobile e-commerce[1]. In 2018, the market share for mobile e-commerce was 63.5%, and it is expected to grow as high as 72.9% until 2021.

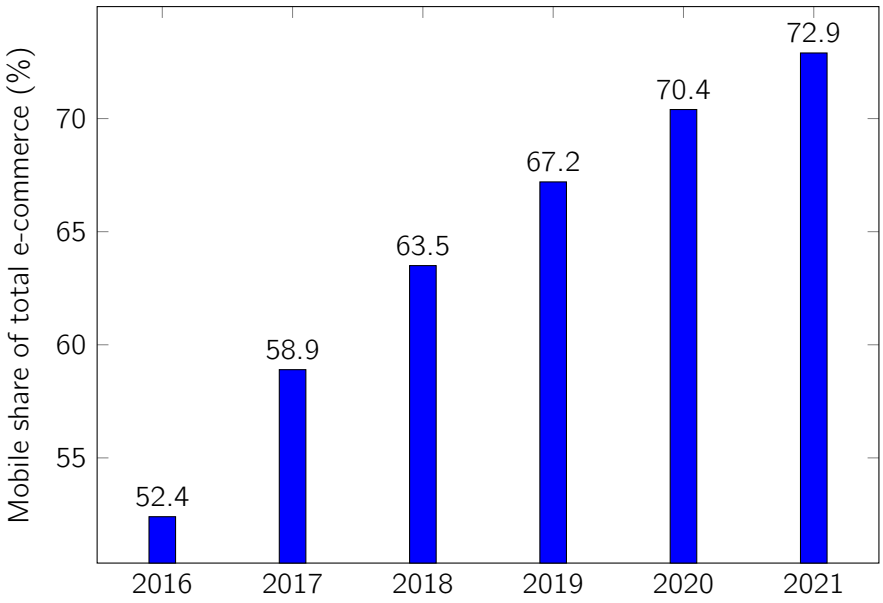


Figure 3.1: Mobile share of total e-commerce 2016 to 2021

This means that Uber was right on the start of the mobile trend, when it started, which was a very positive factor when evaluating the growth of their business.

3.1.2 Glovo

Glovo started in 2015 with a business model similar to UberEats. However, the two companies differed on what products they delivered. While UberEats is focused on food delivery, Glovo has a broader range of products that they are able to deliver. According to Glovo's website, they are able to deliver "food, gifts, markets, pharmacy, snacks & juices, anything". As the company advertises, in addition to the services that they provide (ordering food, medicine...), the platform is also able to process and deliver miscellaneous items, defined by the customer.

Despite the possibility for the customer to order almost anything on Glovo, the most used service is, actually the food delivery. This makes Glovo a competitor to Uber Eats which customers choose, typically, based on price and delivery time. Glovo also offers certain options that are not available in UberEats. An example of that is the possibility to add instructions of how well done the meat is wanted. On the other hand, UberEats provides an estimated delivery time, while Glovo only offers abstract information about the status (for example, "your Glovo is being collected") (Costa 2018).

Similarly, the approach was unsurprisingly very mobile driven, since when the company started, the mobile phenomenon was already very prominent. Nevertheless, taking into account the figure 3.1, it was a good decision, since the mobile share on total e-commerce keeps growing.

[1]data retrieved from https://hostingfacts.com/internet-facts-stats/

3.2 Technology

In this section the current state of the art in terms of technology will be analyzed. Architecture and developing frameworks are the topics that will be discussed in the next pages. The research done for these topics is essential to fully understand the pros and cons of each approach, and how they fit (or not) this project.

3.2.1 Architecture

As it is in the software development life cycle, the first topic that will be tackled is the architecture. This research will help the understanding of the best practices in terms of software design, and which approaches are a better fit for this project.

Architecture can be defined as the structure of the various components within a system, how the communicate with each other, the constraints, principles and guidelines. It must be taken into account that these last three points are dynamic and often change over the time (Zhang et al. 2014). In the next few sections, the architectural styles Service Oriented Architecture (SOA), Micro-Services, and Command Query Responsibility Segregation (CQRS) will be presented.

Service Oriented Architecture

SOA is an architectural style that has as its main purposes, the focus on the business, modularity and reuse. It consists in splitting the system's functions into small applications, each one representing a real business activity. That business activity is what is called a service. The services that exist in a system must provide an interface in order to communicate with other services.

The systems that follow this architecture may be called service-oriented applications. These applications typically follow an architecture similar to the one presented in figure 3.2.

This architectural style is one of the most common on companies. In fact, "it has become the excellent methodology of system construction" (Zhu et al. 2013).

The architecture is composed of three main blocks. The topmost level is where are the responsible for the integration of the application with external services/applications. This is often accomplished with an Enterprise Service Bus (ESB). An ESB is communication system that abstracts the conversion of different protocols into a single component. This layer communicates with the business services. The business services are located in the second layer of the application and processes low-level business operations. The business logic inherent to the system, would be implemented in this layer. The integration layer will call several business services in order to process a business operation. Taking the e-commerce example, when processing an order, the business services could be fetching the information of the product, calculate order values and process invoice. Lastly, the third level of the architecture consists of data-access services, responsible to retrieve and persist information in the database. The business services use this layer in order to have access to the information and then return it, when processed, to the integration layer. It may also

[2] retrieved from `https://www.ibm.com/support/knowledgecenter/en/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html`

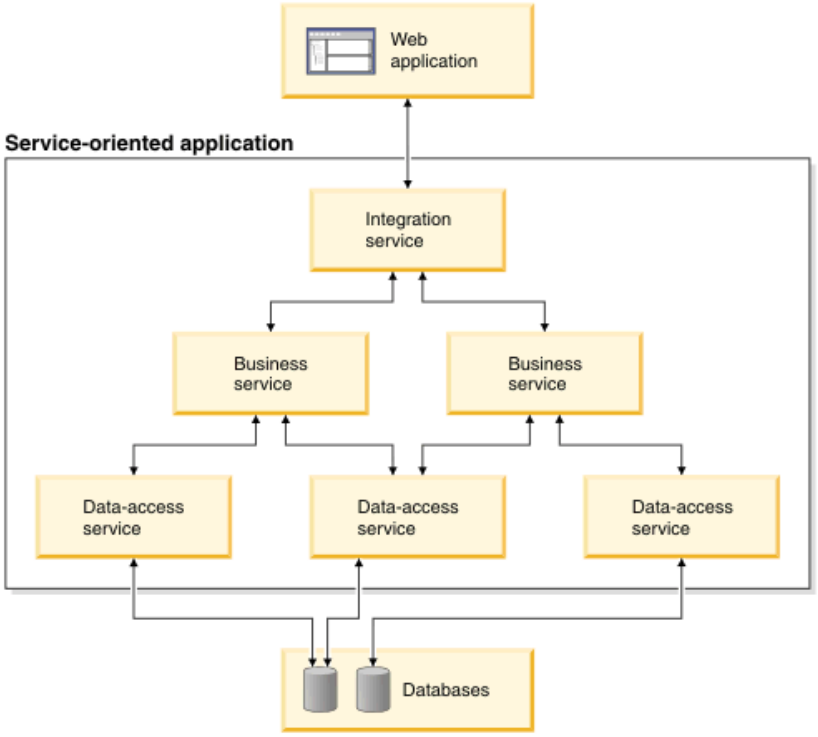


Figure 3.2: Service Oriented Architecture Example[2]

happen the other way around. For create/update operations, when the integration service passes information onto the business services, that information will be processed and sent to the data-access layer, where it will be persisted in the database. In addition to databases, it may also access message queues to send events (IBM 2019).

SOA has many variations and extensions, applying new patterns and good practices adapted to new realities. Micro-Services is one of them.

Micro-Services Architecture

Micro-Services, or Micro-Services Architecture, are a part of SOA where the services are more fine grained. These services should be independent of each other in order to be as loosely coupled as possible. Micro-Services Architecture is widely known for following the Single Responsibility Principle (SRP), since its function within the system is very well defined.

This approach is perfect for complex systems, given that they are typically heterogeneous, unbounded and dynamic. The architecture of these systems must employ a high level of abstractions, to be able to deal with the complexity of these systems.

A service that consumes another, doesn't need to know any internal logic of it, as long as a clear interface has been defined. The interface is what is responsible for the establishment of a contract that enables a consumer to be able to communicate. Each of these services must be autonomous. This means that, as long as none of the contracts has been broken, each service is responsible for itself. For example, in a product service, that has as its main responsibility, the management of the products on an e-commerce platform. As far as the contracts established are not broken (no longer be able to create products in the same way,

is a good example of this), the service may change the way it manages the products. This also means that consumers must be abstracted of how, in this case the product service, works internally and just use the contracts, as a service.

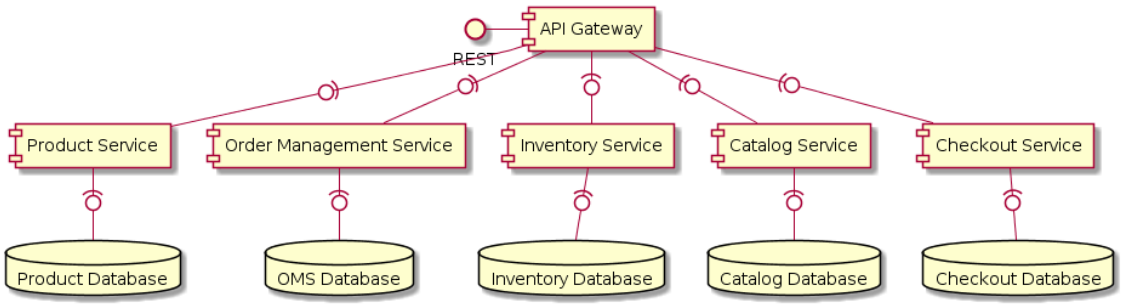


Figure 3.3: Microservices Architecture Example

In figure 3.3 is presented an architectural example of micro-services. This example follows the one previously mentioned of an e-commerce platform. In the figure, it is possible to verify that the responsibilities are divided into five services:

- **Product Service**, with the responsibility of managing the existing products on the platform;
- **Order Management Service**, which is responsible for processing the orders;
- **Inventory Service**, is who is responsible for managing the quantity of each product that is available;
- **Catalog Service**, managing which products are available and when;
- **Checkout Service**, with the responsibility of finishing the order on a front-office side and process the payment.

As it was mentioned, these services are completely loosely coupled since there are no dependencies between any of them. Every service is also entitled and responsible for maintaining its own database, where it will store the data necessary to its correct functioning. In a well implemented micro services architecture, it should also be possible to deploy each of the services independently, without impacting the correct functioning of the others.

Command Query Responsibility Segregation

CQRS is a pattern described by Greg Young where it is defended that the models used to update information and the models used to read information can be different (Fowler 2011). The most common use of a database is often a Create, Read, Update and Delete (CRUD) approach. This means that all create, read, update and delete operations use the same model. This approach often brings some disadvantages. It can turn the management of permissions and security into a more complex task, since each entity is target of both read and write operations. It may also lead to an information mismatch since some fields may not be required some information that is already existent in the database. Finally, it may create concurrency problems, when a set of services operate over a given amount of data. This might also create performance issues, since read and write operations can't be scaled independently (Masashi Narumoto 2017).

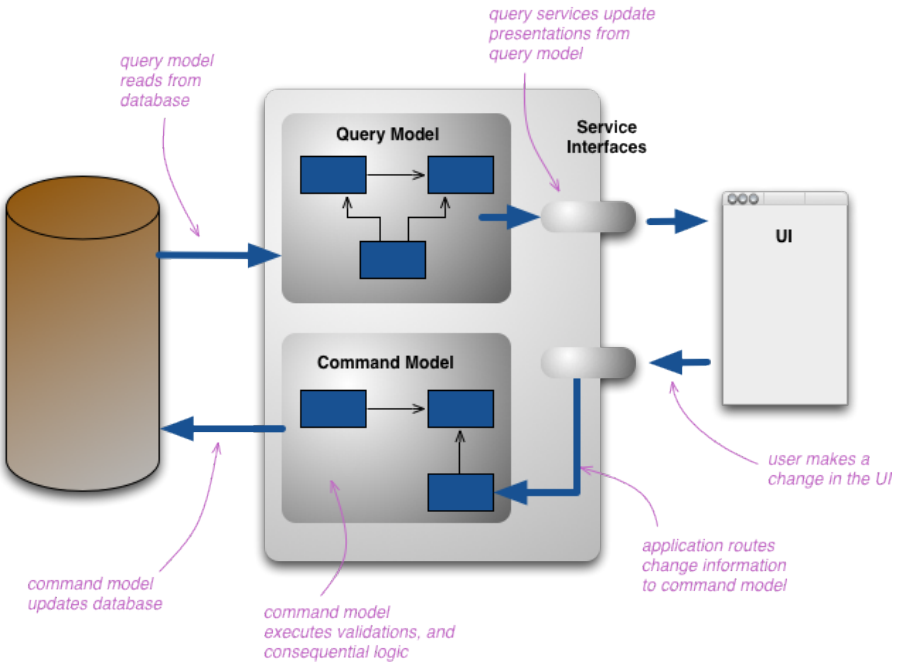


Figure 3.4: CQRS Typical Implementation[3]

This separation maps directly with the command and query concept. Write operations, like CRUD's create, update and delete actions, are commands. Read operations that solely retrieve information are considered queries.

This pattern separates these two operations by having two different interfaces. By doing this, the possibility of having different models is also enabled. On the other hand, CQRS can't be automatically generated by scaffold mechanisms as CRUD can (Masashi Narumoto 2017). CQRS is often implemented by having two different services, one responsible for the queries, other responsible for the commands. This allows to scale the system independently, which can be useful on services that have a high load of reads, but not many updates.

CQRS is recommended to be used only in specific parts of a system. In particular, parts where there is a high chance of concurrence and/or parts where Domain Driven Design (DDD) was used. It is also a good option for systems that need to handle high performance applications, because, not only as said earlier, it is possible to scale independently, but also because is possible to add different optimization to each side (Fowler 2011).

In the figure 3.4 there is a schema of a typical CQRS implementation. It is possible to see the two interfaces that are available, one for the queries, and other for the commands. In case of the command, the interface is connected to its model where it will validate the user/consumer's input and, then, proceed to the operation in the database. For the queries, the behaviour is very similar. After the user/consumer's request, it is sent to the model, which then retrieves the data from the database and processes it. Finally, the information is returned to the requester.

CQRS is a very powerful architecture that solves many specific problems. However, before deciding to use this pattern, a careful analysis should be carried out. The implementation of this pattern requires a change in mindset and many information systems fit in the traditional

[3]retrieved from https://martinfowler.com/bliki/CQRS.html

approach, and implementing CQRS would add unnecessary complexity to the project (Fowler 2011).

Event Sourcing

Event sourcing is an architectural style, where every change in the state of an entity triggers an event, and that the event is stored in the same sequence that they were applied. The whole purpose of doing this is to replicate the state of a certain entity by reprocessing all the events (Fowler 2005).

This architecture also passes the responsibility of calling all operations inherent to a business process to the services that actually do that operations. In contrary to Hypertext Transfer Protocol (HTTP) calls, where the source (which triggers the event) needs to know processes to call, in event sourcing (and most event-driven architectures), the source only needs to make sure that the event is published. The listeners are the ones who need to check if that particular event is something that needs to start any operation

The figure 3.5 presents an example for this situation. The user sends a request to the front-end server, which then publishes an event to the Kafka Broker. When this happens, both "First Service" and "Second Service" will receive the information about what happened, and then decide if they should trigger any process regarding the user action.

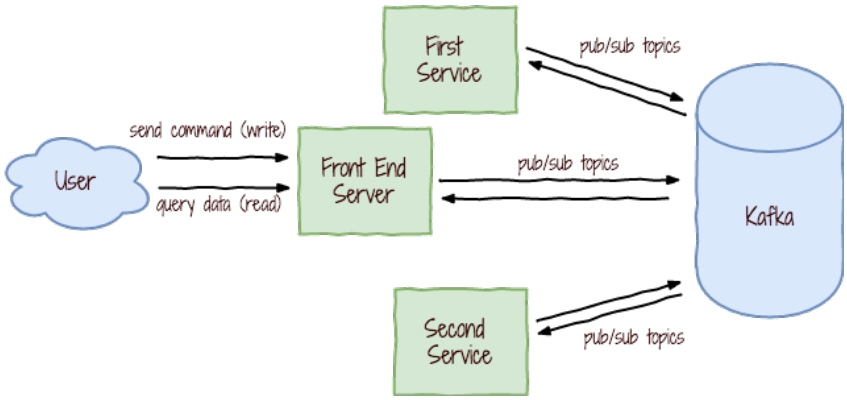


Figure 3.5: Event Driven Architecture using Kafka[4]

This architecture is very good for any application that needs to keep track of every change that occurred on a certain entity of their business. Bank systems are a good example where this architecture may be used. There must be a record of every transaction that is made in a bank account and with event sourcing, that is done, and the events produced will trigger the necessary processes to run.

3.2.2 Software Development Frameworks

In this section, an overview will be given over the current most used software frameworks. These frameworks are used as abstractions over more specific Operating System (OS) code,

[4]retrieved from `https://medium.com/kontenainc/event-sourcing-microservices-with-kafka-2568801527d8`

thus enabling its re-use. Furthermore, they provide several libraries, tools and compilers to facilitate the developer's work. Currently, the use of these abstractions is a standard to build and deploy applications within the industry. In the next sections, we will analyze the frameworks .NET Core, Java Spring and Node.js.

.NET Core

.NET is a software framework, developed and maintained by Microsoft that was released in 2002. This framework provides a large class library, Framework Class Library (FCL), that includes the main standard libraries, and language interoperability. This means that it is possible to use code written in other language of the .NET platform. This is possible because .NET runs in a software environment, rather than a hardware environment. This is a virtual machine called Common Language Runtime (CLR).

This platform also provides tools to create applications for several types of devices (phones, tablets, etc.). These applications can be console apps, Windows Forms Apps or (). When it comes to web development, .NET has a web application framework specifically for that. ASP.NET was released within the .NET Framework and was the successor of Active Server Pages (ASP), built on top of the CLR.

The possibility to use the CLR, offers the possibility to develop applications in the ASP.NET platform using different languages, within the range available. Languages like C#, Visual Basic and F# are all supported and be called directly in the code.

One of the main problems of the .NET Framework, was that it was only prepared to run on Windows Environments. These servers are often more expensive and slower than the Linux Servers. Java Spring, one of .NET's biggest competitors, doesn't have this restriction and can run on any environment and OS. To fight this problem, in 2016, Microsoft launched .NET Core 1.0 as an alternative to the traditional .NET Framework. Since then, the maintainers have kept on developing the platform. The version 3.0 was announced on Microsoft Build on May 2018 and it is expected to be released in 2019 (Microsoft 2019).

The development environment for any .NET project is the Visual Studio. This Integrated Development Environment (IDE) has lots of useful tools for development, including the integration with Azure, allowing direct deploys via Visual Studio. As the graphic of figure 3.6 presents, Visual Studio is the second most used development environment only bellow Visual Studio Code, which is also developed by Microsoft.

ASP.NET also offers automatic monitoring built in. This allows the maintainers to monitor the applications for some basic problems (like infinite loops or memory leaks) and immediately take action to solve the problem. This ensures the higher stability of the application (AltexSoft 2019).

However, the support of data-oriented development for .NET (and not just .NET Core) is provided by Entity Framework. Entity Framework is an Object-Relational Mapper (ORM) that connects the model of the application to relational databases. This framework is thought to offer low flexibility and not support all database designs. The fact of the development environment being Visual Studio also brings problems. The licensing costs of the IDE is about $45 per month/user for the base version[6]. Some people also complaint about

[5]data retrieved from https://insights.stackoverflow.com/survey/2018/

[6]prices retrieved from https://visualstudio.microsoft.com/pt-br/vs/pricing/ on 16/02/2019

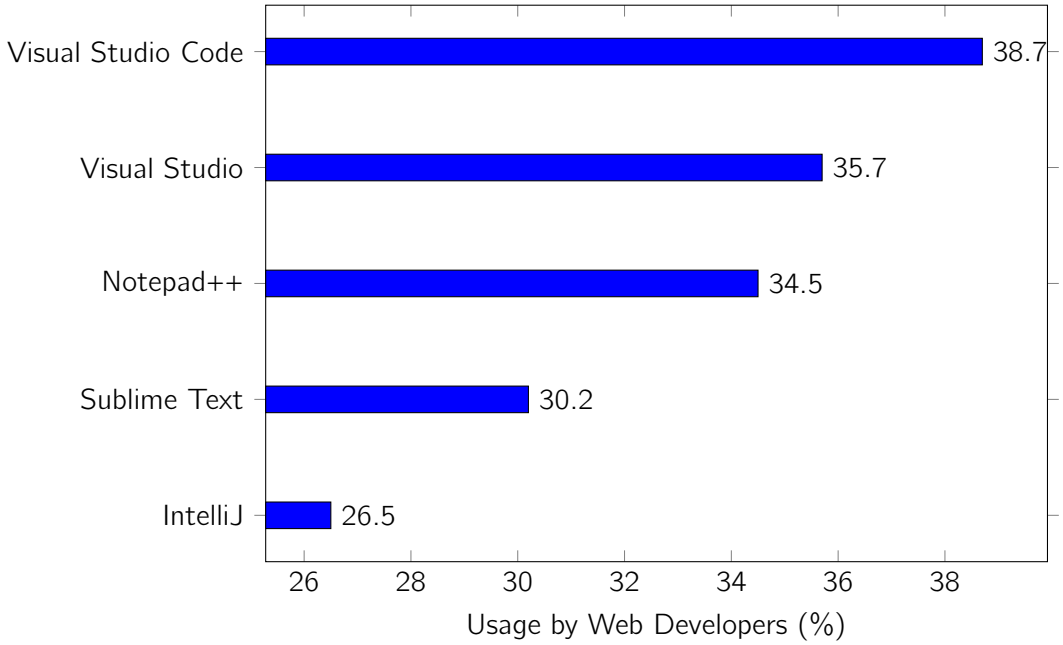


Figure 3.6: Top 5 most used IDEs by Web developers [5]

the technology being too connected to its vendor, Microsoft (AltexSoft 2019). As opposed to other frameworks that are much more community-driven, .NET still depends on decisions made by Microsoft, which may lead to offer no alternatives for integration that the ones that are distributed by them.

Java Spring

Java Spring is an open-source application framework that provides core features to the Java platform. The first version of this framework was released in 2002 by Rod Johnson as a beta, reaching to a stable version in 2004. Despite being an open-source platform, it is maintained by Pivotal.

Java Spring is a very flexible framework that supports both Extinsible Markup Language (XML) and annotations for configuring the Spring Beans. The beans are the backbone of the application that are instantiated, assembled and managed by the Spring Inversion of Control (IoC). The IoC is "a process in which an object defines its dependencies without creating them" (Thai 2018)

This framework provides support for several services, like Aspect Oriented Programming (AOP) for cross-cutting concerns, Authentication and Authorization, dependency injection and testing infrastructure, to write unit and integration tests. There is also a data access framework that abstracts the developer from many common difficulties that are faced. The most common data access frameworks for Java, such as Java Persistence API (JPA) and Hibernate are also supported (Pivotal 2019).

Spring also offers support for Java Development Kit (JDK) timers, logging frameworks and libraries that contain many useful functions for the developers to learn and use them to develop applications (DataFlair 2018).

Being built on top of Java, this means that it will run on the Java Virtual Machine (JVM). The JVM is a virtual machine that runs on top of the OS, abstracting its procedure calls and standardizing them for the programming language. This means that an application that was built using Java Spring, can run on any OS (Tyson 2018). This was a great advantage that Java had over its competitors, specially on the web development industry. Competitors like Microsoft's .NET didn't support cross-platform. However, in 2016, Microsoft presented .NET Core. A framework based on .NET but that could run on any machine.

Two of the main cons of the Spring Framework are its complexity and the long learning curve. This happens due to the 2400+ classes plus tools that the developers need to know to be able to develop their applications (Aswani 2014). This can make the project far more complex than what it needed to be. Also, there are no clear security guidelines. Cross-site scripting attacks or cross-site request forgery attacks avoidance are examples of documentation that is missing (DataFlair 2018).

Node.js

Node.js is a JavaScript (JS) runtime environment. It uses the Google Chrome V8 engine in order to run the code. As JavaScript was originally made to run in the client application, Node was born to extend the language to the back-end side (Patel 2015).

Node's main idea was to "use non-blocking, event-driven I/O to remain lightweight and efficient in the face of data-intensive real-time applications that run across distributed devices" (Capan 2019). That means that this technology didn't come to overcome every problem ever faced. Instead, it solves a particular need in the development world. For example, as Node is single threaded, it is not indicated for heavy Central Processing Unit (CPU) operations. The main advantage or using it is to build fast, scalable and lightweight applications. Another benefit of using Node.js is that both the front-end and back-end will be in a common language. This causes for a fast synchronization, which is especially good for event-based applications (AltexSoft 2018).

The way how it works is different from the traditional implementations. On that approaches, each new request would start a new thread in the system, using its Random Access Memory (RAM). When the available memory is all in use, the new connections will stay on-hold, waiting for a thread to be available. On the other hand, Node.js uses non-blocking I/O calls, which allows it to handle "tens of thousands of concurrent connections" (Capan 2019). The figure 3.7 demonstrates the difference between the thread management done in both the traditional applications and in Node.js.

All the points that were referred above, make Node.js as a perfect platform to be used in applications such as chat rooms, data streaming systems and stock trading platforms.

As every good thing has a but, Node is no exception. Despite being a quite new technology, the core of node modules is considered stable. However, many tool on npm (the Node.js package manager) are of poor quality and/or have not being properly documented (AltexSoft 2018). The fact of being open-source also has an impact. Despite the core modules being supervised by Joyent and other major contributors, the rest of the tools might lack in the code quality. There are also some performance problems when working with Node. As JavaScript was made to be asynchronous, it has a non-blocking Input/Output (I/O) model. This means that it can process many reads and writes that are queued in the background, without blocking the main thread. For this, Node uses callbacks. Callbacks are functions

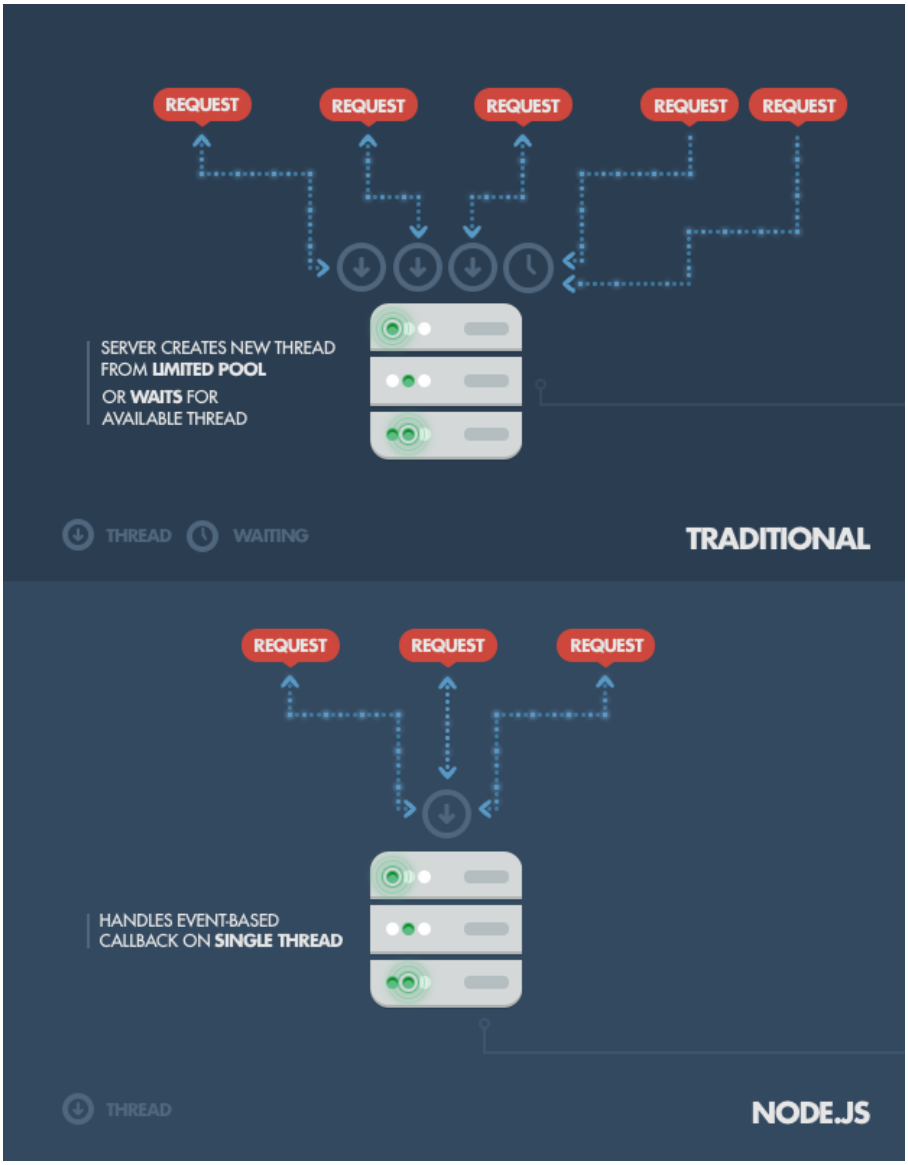


Figure 3.7: Node.js Thread management[7]

that run on background in a queue. If there is a situation where callbacks are nested within other callbacks, it will make the code difficult to read a maintain. This is commonly known as callback hell. The listing 3.1 presents an example of this problem.

[7] retrieved from https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' +
    height)
            this.resize(width, height).write(dest + 'w' + width + '_' +
    filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

Listing 3.1: Callback Hell example[8].

Node is constantly chosen for solutions that require real-time updated data, like video-conference systems or online gaming. By using JS programming language, the learning curve for Node is kept to a minimum. This means that developers with front-end experience (which requires a lot of JavaScript) can start programming server-side without facing many difficulties. The fact of being a lightweight makes this a very good option for micro-service architectures. Despite being an open-source technology, it has a great corporate support. That support was initially provided by Joyent, but in 2015, with the creation of the Node.js Foundation, IBM, Microsoft, Paypal, SAP and Fidelity also offered their support by becoming founding members or the organization (AltexSoft 2018).

3.2.3 Swagger

Swagger is an open-source framework that allows the generation of automatic documentation of the available endpoints in an API. This is done without any human intervention. Swagger is capable to ask the API for an YAML or a JSON file that contains a detailed structure of the entire API. This file is essentially a resource listing using the OpenAPI specification. This file contains the following information (Swagger 2019):

- What are the operations supported by the API;
- What are the endpoint's parameters and return values;
- Authentication and Authorization information.

[8] retrieved from http://callbackhell.com/ on 16/02/2019

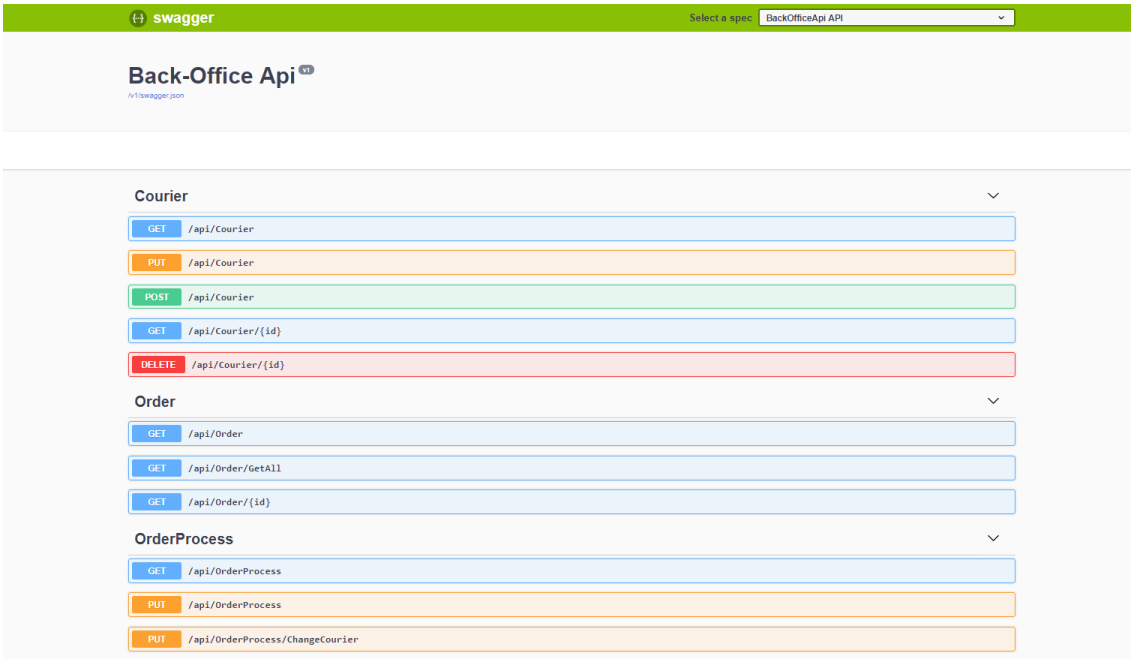


Figure 3.8: Back-Office API Endpoints on Swagger page

The figure 3.8 shows the listing of endpoints in one of the solution's APIs, Back-Office API. This allows the consumers to know from the start how to use the API. Opening one endpoint, as shown in figure 3.9, it is possible to verify the required parameters, the return value and even to call that endpoint, making it easier to make a direct request to the API.

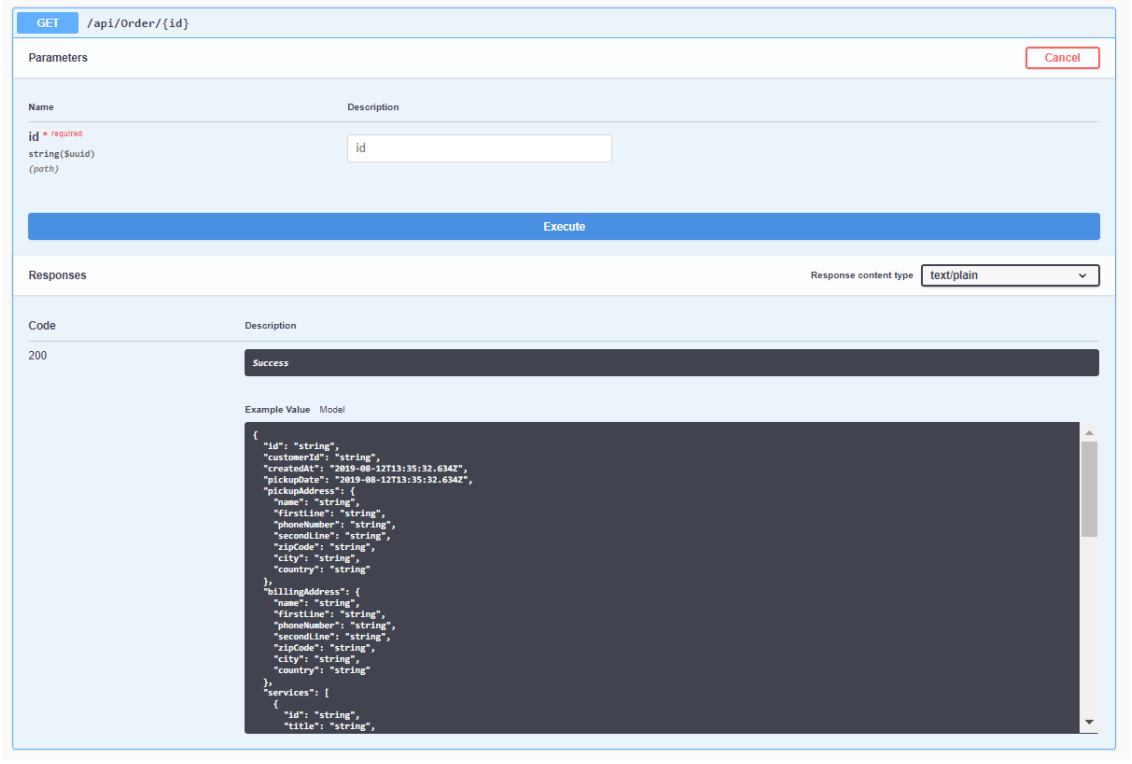


Figure 3.9: Open endpoint on Swagger

Swagger has support for many languages and frameworks, such as .NET, Java, Go, C++, etc.

Chapter 4

Value Analysis

Value is the most important concern when it comes to the development of an innovation and/or project. Not only in terms of monetary value, or cost, but specially the value that the product brings to the consumer.

According to Lawrence D. Miles, the value of a product or service is attained when measuring the products performance and cost, where the product's performance is its ability to satisfy a certain need (Miles 2015). This means that the value of a product increases when it attains the highest performance at the lowest cost. In mathematics, value can be defined by the following formula:

$$value = \frac{(Performance_x + Capability_x)}{Cost_x}$$

This chapter provides an overview of the value analysis theory that will allow to understand fundamental concepts of this area of research. These concepts are essential to comprehend the value proposition that will also be presented in this chapter.

4.1 New Concept Development

Innovation is not a standard process. It requires a great ability of creativity, opportunity identification and problem solving. The Fuzzy Front End (FFE) is the first stage in that process. This is the point where, in most cases, the opportunities are found and ideas for a solution appears. This stage is a very experimental, unpredictable and unstructured one. As it requires a great amount of creativity, there is no standard process that will lead to a certain result. The results of this will influence the second stage of the process, the New Product Development (NPD). This is the process of development and production of a solution to the opportunity that was found. Finally, the product would enter the commercialization phase, where it would be sold to the customers and start generating revenue. However, it was difficult to compare FFE practices across different companies. This happens because there is a lack of common terms and definitions for this method's key elements. Without it, it is almost impossible to create new knowledge, or when it was possible, it would be ineffective (Koen, G. M. Ajamian, et al. 2002).

To fill this gap and provide a common language on the front-end activities, Peter Koen and a group of researchers, developed a new theoretical construct, the New Concept Development (NCD) model (Koen, G. Ajamian, et al. 2001). In order to avoid the term FFE, as it implies that the activity is mysterious, uncontrollable and unmanageable, Peter Koen introduced

a new name for this stage, calling it Front End of Innovation (FEI) (Koen, Bertels, and Kleinschmidt 2014). The model is shown in figure 4.1.

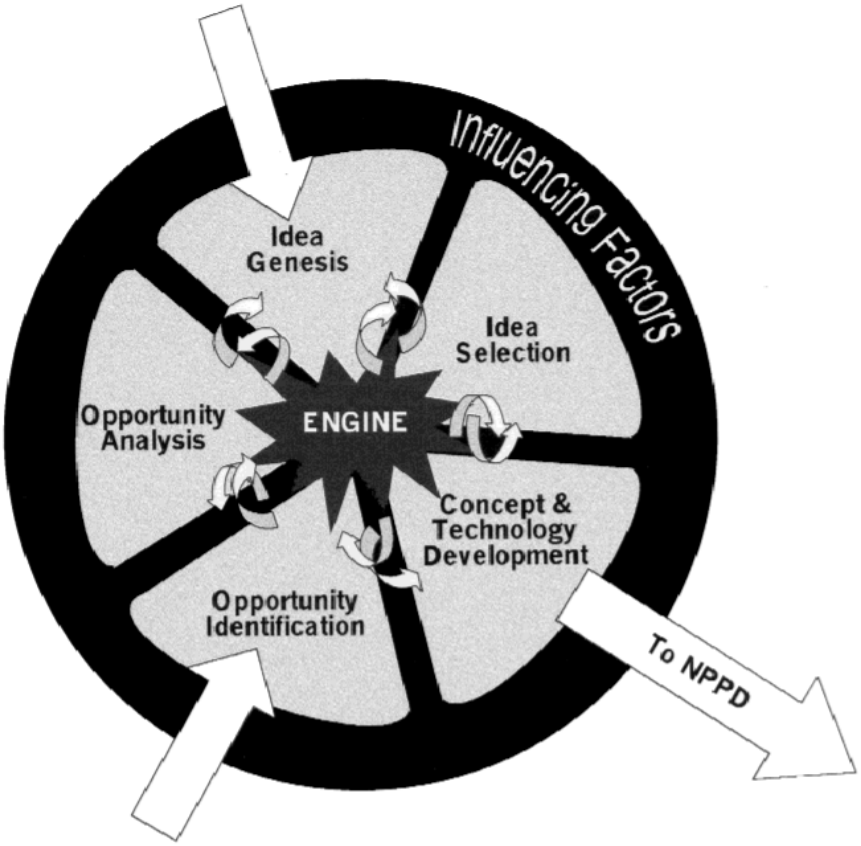


Figure 4.1: NCD model[1]

This model is composed by three distinct areas: the engine, at the center, the wheel, which are the five slices near the engine, and the rim, at the border of the model. The engine provides power to the innovation process. It is "fueled by the leadership and culture of the organization" (Koen, G. Ajamian, et al. 2001). The wheel, is composed by five key activity elements: Opportunity Identification, Opportunity Analysis, Idea Generation and Enrichment, Idea Selection and Concept & Technology Development. The third element, at the border of the model, consists on the external environmental factors that influence the engine and the activity elements. Examples of these factors may be the Business Strategy, Organizational Capabilities, the existent technology to be used and anything that is external to the organization (customers, competitors, trends, etc.). The arrows between the elements are there for the same reason why the model has a circular shape. To indicate that ideas flow and iterate between the five elements. The arrows pointing into the model represent the starting points that a project can have. It may start on either Opportunity Identification, or Idea Genesis. Projects will leave the FEI by entering the New Product and Process Development (NPPD) (Koen, G. Ajamian, et al. 2001).

[1] retrieved from `https://web.stevens.edu/cce/NEW/PDFs/Clarity_FEE.pdf`

4.1.1 Opportunity Identification and Analysis

These elements refer, respectively, to the phases where a given company or individual identify an opportunity and analyze it to have a better perception of it and more information regarding the problem. The Opportunity Identification is the most common activity to serve as a starting point for a project, since it reflects a need that is yet to be fulfilled.

It was the case of the project of this dissertation. As it was already explained in sections 1.2 and 2.2, despite the existence of multiple online stores selling numerous products, when it comes to services, the number of options decreases drastically, and service providers, are already trying to offer a solution, without technology support. This creates an opportunity to innovate in this sector. Besides that, the existence of a pilot SP available to work on the project also took a great part in the opportunity analysis.

4.1.2 Idea Genesis and Selection

The Idea Genesis is the first form of a solution to fulfill the needs found in the Opportunity Identification stage. It includes the process of birth, development and maturation of the opportunity into an concrete idea (Koen, G. Ajamian, et al. 2001). To do this, companies often engage with customers and other companies and institutions in order to understand the real needs of their possible users, and link with cross-functional teams for brainstorming sessions, in order to have a more clear and open view of the whole picture. This element is also a common entry-point for projects.

In most cases, there are many possible solutions to a certain need. The vastness of good ideas also creates a new problem, which is to choose which ideas the company should select to proceed to the development stage (Koen, G. Ajamian, et al. 2001). This turns out to be a very difficult task, since at this point there is a high degree of uncertainty and the information is very limited. Also, the Return on Investment (ROI) is still a very foggy and risky guess.

In the project in analysis in this thesis, there was already an idea for a solution. The first idea would be to create a solution to suppress the needs of the single pilot client that was already on-board, by developing a website where a customer could order laundry jobs and the SP could manage its orders, connecting it to a network of couriers. However, in the first meetings with the student, a new idea surged. The idea was to create a platform, instead of a simple website for just one partner. By being abstracted to the SP and the service that was being provided, the platform could easily scale. As the ideas matured, and using the decision process referred in the next sub section, it was decided to proceed with the last one, but taking into account that first partner would be a laundry, and focusing on those needs for the Minimum Viable Product (MVP).

Idea Selection using the Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP), is a multi-criteria decision method that uses mathematical techniques to aid in the decision makers to choose an option in a group of various alternatives. This process is based in the crossing of the alternatives with the existent criteria (Saaty 1990).

In order to select which of the ideas identified in the idea genesis phase, the AHP was the method that was used for this. Firstly, an hierarchy decision tree was built:

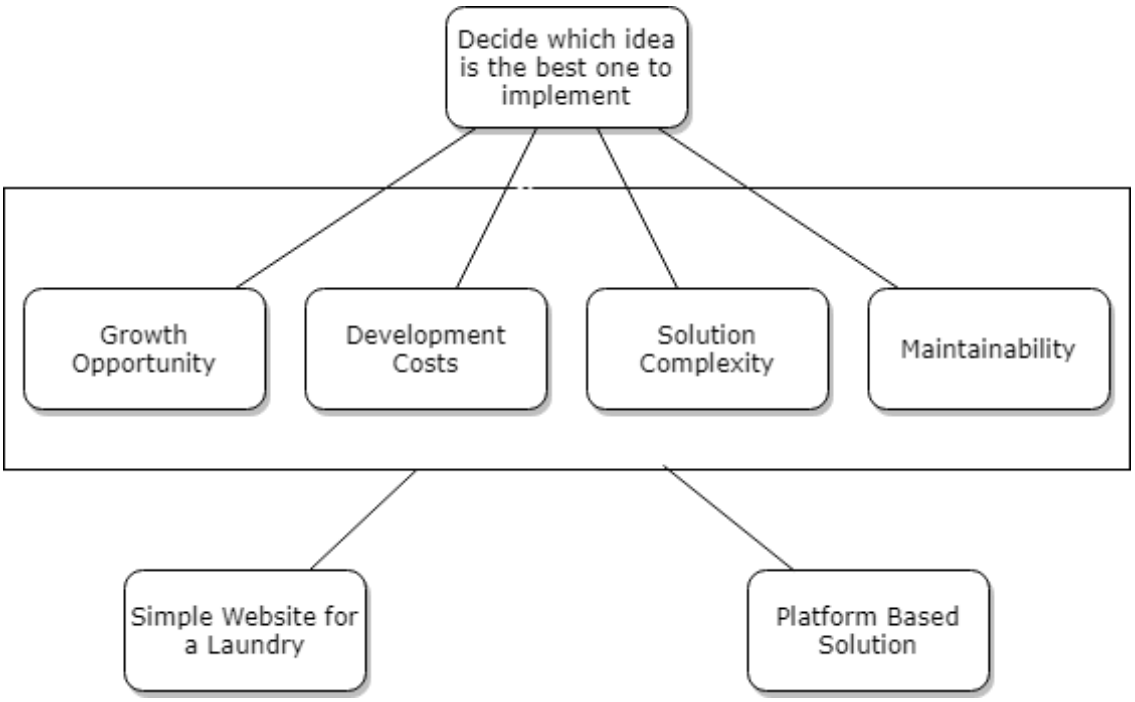


Figure 4.2: Hierarchy Decision Tree

The figure 4.2 presents the hierarchy decision tree for the idea selection. The first layer reflects the main objectives of this analysis, which is to decide which of the ideas identified in the idea genesis, should be selected. On the second layer, the criteria that will be used for this analysis are presented. These criteria are:

- **Growth Opportunity** - The possibility of the project reaching an higher level and gaining more partners;
- **Development Costs** - The costs in both time and money to develop a given solution;
- **Solution Complexity** - This measures the complexity of a solution for the user. In other words, if it will be difficult to use or not;
- **Maintainability** - This measures how easy, or not, the solution will be to add, remove or correct functionalities to the project.

By having these criteria into account, it is possible to evaluate which decision is the best one to accomplish the main objective. The table 4.1

Table 4.1: AHP Evaluation Table

Criteria	**Growth Potential**	**Development Costs**	**Solution Complexity**	**Maintainability**
Growth Potential	1	5	4	2
Development Costs	1/5	1	3	1/2
Solution Complexity	1/4	1/3	1	12/30
Maintainability	1/2	2	6	1
Total	2	8 1/3	14	3,666666667

The growth opportunity and maintainability are the most important criteria in the decision, since they affect both what the solution can become in the future. The development costs and complexity are, nonetheless, factors to take into account since they can affect the delivery date. However, these last two are not as important as the first ones.

This matrix presents the normalized matrix in the AHP evaluation method.

$$M = \begin{bmatrix} 0,5128 & 0,6000 & 0,2857 & 0,5455 \\ 0,1026 & 0,1200 & 0,2143 & 0,1364 \\ 0,1282 & 0,0400 & 0,0714 & 0,0455 \\ 0,2564 & 0,2400 & 0,4286 & 0,2727 \end{bmatrix}$$

Lastly, the table 4.2 presents the weights of each criteria that was used for this decision.

Table 4.2: Weights of each criteria

Criteria	**Weight**
Growth Potential	0,4860
Development Costs	0,1433
Solution Complexity	0,0713
Maintainability	0,2994

Hereupon, taking into account that the growth potential is the most important factor in this decision, the idea of developing a platform instead of a single website directed for a single laundry, was selected to fulfill the needs of this project, since a platform that supports multiple service providers, with different types of services is more likely to grow than a simple website for a laundry.

4.1.3 Concept & Technology Development

The last element of the NCD model is one of the most important. It serves as an exit door to development processes such as the NPPD. This activity consists in the "development of a business case based on estimates of market potential, customer needs, investment requirements, competitor assessments, technology unknowns, and overall project risk" (Koen, G. Ajamian, et al. 2001).

For this project, this phase consisted in the development of the thesis formalization. This document states the problem and objectives of the project, mentioning also several requirements, constraints and restrictions.

4.2 Value Proposition

Value Proposition is one of the most widely terms used by companies, nowadays. However, it is difficult to find a clear definition for what it really means. Companies have been defining their Customer Value Proposition (CVP) in many different styles, with their perception of the concept (Anderson, Narus, and Rossum 2006). According to this same study, These definitions lie on three major types: all benefits, points of difference and resonating focus.

- **All Benefits**: The most used by managers when asked to build a value proposition. This type simply consists in listing all the benefits that it is believed that the product/solution in question might provide. The more are listed, the better. However, this might lead the customer, to select which product to acquire just based on the price.
- **Points of Difference**: The second approach has the premise that the customer has alternatives other than the product that a given company is offering. As opposed to the previous type, points of difference focuses on the advantage of a product over its competition. It is focused on the question "why should a customer go with our product over our competitors'?" instead of "Why should I buy your product?". From a customer's point of view, this facilitates the process of choosing which is the best solution to his problem.
- **Resonating Focus**: Despite the advantages of the last point over all benefits, there is still room for improvement. The last approach focuses on two points that complement the previous type. This proposition "steadfastly concentrates on the one or two points of difference that deliver, and whose improvement will continue to deliver, the greatest value to target customers" (Anderson, Narus, and Rossum 2006). This means that instead of listing all the attributes and benefits of the product, and what differs the product over the competition, resonating focus defends that only the key aspects for the target customers should be mentioned. Secondly, this approach may contain a Point of Parity (PoP). A PoP is a mandatory element for a customer to even consider the company's product as a viable option.

Given the possible approaches for the value proposition, for the context of this thesis, it was chosen to follow the last approach, resonating focus. Since for this project, there will be three types of customers, it will be defined, also, three value propositions.

4.2.1 Value Proposition for Final Customers

The final customers are this project's most valuable stakeholder. If the platform doesn't meet the customers' needs, they will not use it, which means that there is no point for service providers to be present in the platform. Eventually, this could lead to the project not being viable.

The value proposition for customers is as it follows:

- Choose from a range of service providers;
- On-click service request;
- Have any goods needed picked-up/delivered at your address, at your time;
- Pay with your phone. No need for counting change.

Hereupon, the slogan that will be used to attract and present the value proposition to final customers will be:

Focus on what is important. We take care of the boring tasks.

4.2.2 Value Proposition for Service Providers

The service providers are the reason why the customers use the platform. Without service providers, there wouldn't be any customers and, by consequence, no business. The value proposition for the service providers is as it follows:

- Get online on a ready to use Platform;
- Be able to deliver and pickup at customer's address;
- Pay what you gain. You only pay fees over the services;
- Reduced costs when compared to a proprietary solution;
- Increase your range of customers.

The slogan that will be used to attract and present the value proposition to service providers will be:

Do what you do best. The logistics are on us.

4.2.3 Value Proposition for Couriers

The couriers are the ones with the responsibility of carrying the goods from the customer to the SP and vice-versa. It is a very important job since they are the only party to have direct contact to the final customer. The value proposition for couriers is as it follows:

- Flexible work hours;
- Be your own boss;
- Easy to get on-board;
- No usage costs;
- Increase your income.

With this, the slogan that will be used to attract and present the value proposition to couriers will be:

Create your own schedule and increase your income

4.3 Value for the Customer

When developing a new product or service, one of the most important things that companies must have in mind, is the value that it will bring to their customers. However, this exercise is a harder task than it appears. The reason for this is because value is a subjective concept. It is a trade-off between benefits and sacrifices. Those benefits and sacrifices might have a different impact on distinct customer segments. What might be one's benefit, might be indifferent for other. Even if we look to the individual customer in a single segment, this phenomenon may occur due to the individuality of each customer. Nevertheless, it is expected to occur less often. Value may also be relative to the current competition on the market. When a certain company launches a product that has similar functions to a

competitor's early released product, the customers are already used to that features, which makes it lose the novelty factor (Ulaga and Eggert 2004).

Value for the Customer (VC) is described by Tony Woodall as "any demand-side, personal perception of advantage arising out of a customer's association with an organization's offering, and can occur as reduction in sacrifice; presence of benefit (perceived as either attributes or outcomes); the resultant of any weighed combination of sacrifice and benefit (determined and expressed either rationally or intuitively); or an aggregation, over time, of any or all of these" (Woodall 2003). On this same study, Woodall divides VC into two sub-forms: Nature of Derived VC and Contingent VC. The first one refers to VC in its derived form. It focuses on excellence and more in derivatives as material value, functional value and emotional value. It is also called "Consumption Value" (Ulaga and Eggert 1991). Most of these are types of value don't change over the time. The second one, focuses on the value on a longitudinal timeline, from the period before the customer buys the product, passing for the transaction it-self and ending in the after use. The whole process is part of the experience in Contingent VC.

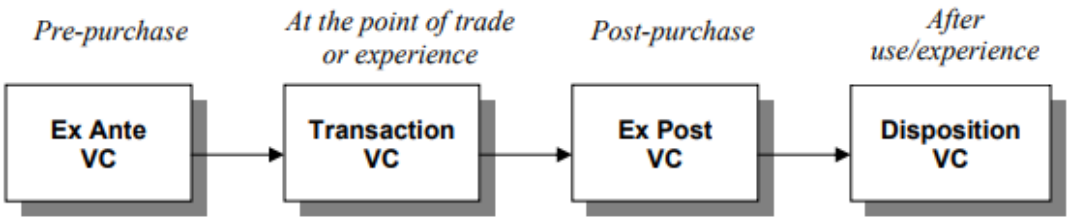


Figure 4.3: A Longitudinal Perspective on VC[2]

In the figure 4.3, it is possible to verify each of the temporal phases inherent to the Contingent VC. Each of these temporal stages have associated values, benefits and sacrifices.

The table 4.3 presents a mapping between each of the temporal phases of the Contingent VC, and some of its associated values, benefits and sacrifices, for the project in analysis in this thesis. As was already exposed in exposed in section 4.2, there are three different clients of this system: the final customer, the service provider, and the courier. Since the main problem to be solved by this project will be a problem of the service providers, the mapping was done for the benefits and sacrifices of that specific client.

Focusing on the benefits and the sacrifices, it is verified that there are many benefits for the customer and just a few sacrifices, most of them related to the price and training costs.

The Ex Ante and Ex Post perceptions of a product will prepare the customer for the experience that he/she will have during this process. They will both take part in the customer perception of the quality of the service. In the case of the first temporal stage, it will affect the customer's perception of the service/product's value. The perceived value is the impression that is passed to the possible customer of the value of what is being presented. However, this process depends a lot in the customer's previous experiences and expectations (Groth and Dye 1999).

[2]retrieved from https://cdn.ymaws.com/www.ams-web.org/resource/resmgr/original_amsr/woodall12-2003.pdf

Table 4.3: Contingent VC Temporal Stages Values mapping

Temporal Stage	**Values**	**Benefits**	**Sacrifices**
Ex Ante VC	Expected value Desired value	Product characteristics Features	Price
Transaction VC	Transaction value Exchange value Acquisition value	Support Security	Acquisition costs
Ex Post VC	Delivered value Received value Postpurchase value	Functional benefits Operational benefits Logistical benefits	Training Costs Installation costs
Disposition VC	Use value Redemption value	Service support Reliability Convenience	Maintenance costs

Chapter 5

Requirements Engineering

This chapter has the purpose to present the process in which the requirements, both functional and non-functional, were gathered. It will start with an overview over the existent user groups, and then presenting the functional requirements.

5.1 User Groups

In this section, the existent actors and their actions will be presented. This will provide a better understanding regarding the users and their responsibilities in the platform.

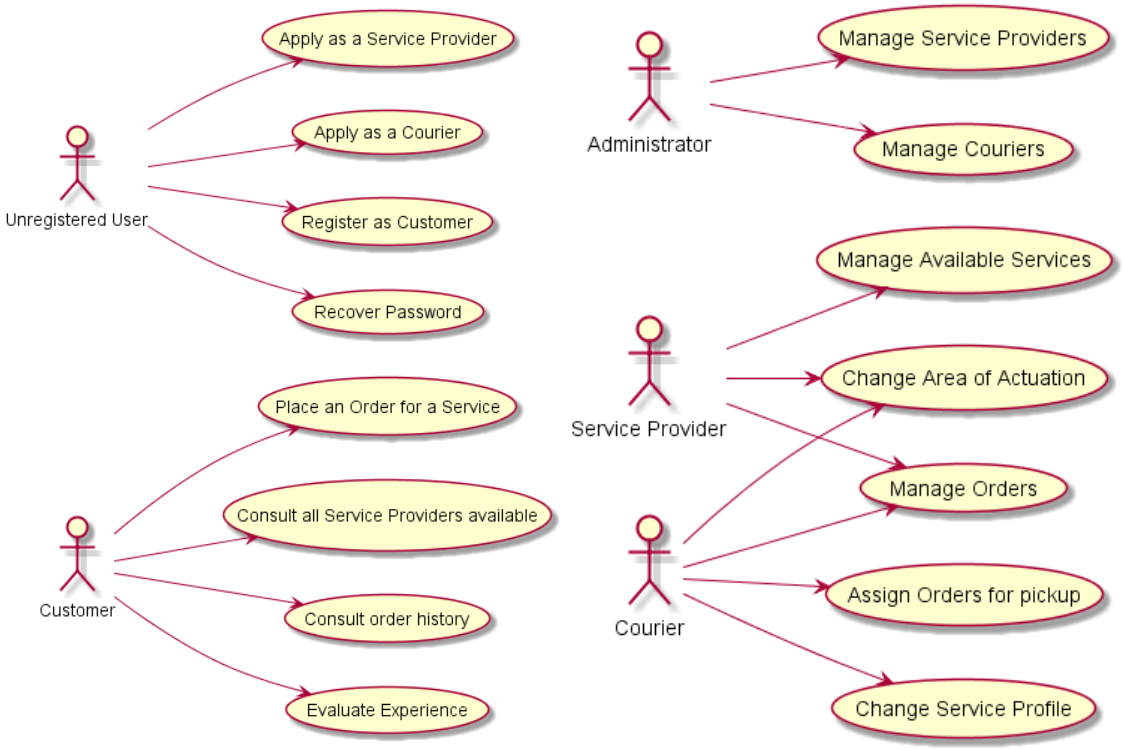


Figure 5.1: Use Case diagram

Considering that there will be different responsibilities for the users of this platform, there is a need to define what are they and which user groups do the define. Hereupon, the main users of the platform will be:

- **Unregistered User** - A user who has just reached the platform. Hasn't authenticated yet;
- **Customer** - A user which is a customer and is in look of a service;
- **Courier** - A user which is a courier and will be the people which will make the transportation of the necessary goods and materials associated to a service;
- **Service Provider** - A user which is a service provider and will be the responsible to provide the service requested by the customer;
- **Administrator** - The administrator of the platform. Responsible to manage service providers, couriers and the platform as a whole.

The figure 5.1 presents a use case diagram where it is possible to see all the actors and the functionalities that each one can access. On the left side there are the actors which access the core front-office functionalities such as placing an order or register. These actors are the customer and an unregistered user. On the right side we have the management and back-office actors that are responsible for the operational side of the platform. The actors for these actions are the service provider, the courier and the platform administrator.

5.2 Functional Requirements

In this section, the functional requirements of the system will be presented. Each of the requirements will then be split in smaller user stories throughout the project.

5.2.1 FF01 - Register as Customer

User Groups: Unregistered User

Requirement Description

Besides being able to consult some marketing material and information about the platform, the most common action that an unregistered user can make is registering. If the user chooses to register as a customer, some data will be needed to make it happen. The mandatory information for the registration be successful are:

- Name
- E-Mail
- Password

The user may complement with other information (address, phone contact, edit information) in the "My account" page.

After the registration, an email will be sent to confirm the email address. In that email, there will be an url for the confirmation. While the account has not been activated, no orders may be done.

5.2.2 FF02 - Apply as a Service Provider

User Groups: Unregistered User

Requirement Description

An unregistered user may also choose to apply as a service provider. In this case, some data will be needed to make it happen. The mandatory information for the registration be successful are:

- Name of the service provider
- Name of the representative
- Address
- Service type (Laundry, mechanics, warranty...)
- Area of Actuation (Coordinates of the location of the service provider and the desired radius)
- Email
- Password

The user may complement with other information (phone contact, edit information) in the "My account" page.

After the registration, this application for service provider will be sent to the platform administrators which will evaluate the viability of the SP. In a first version, the management of the applications for service provider is not in the scope of this project.

The service provider will only be available in the platform after the administrator's approval.

5.2.3 FF03 - Apply as a Courier

User Groups: Unregistered User

Requirement Description

An unregistered user may also choose to apply as a courier. In this case, some data will be needed to make it happen. The mandatory information for the registration be successful are:

- Name
- Profile type (motorcycle, van, car, etc.)
- Area of Actuation
- Email
- Password

After the registration, this application for courier will be sent to the platform administrators which will evaluate the viability of the courier. In a first version, the management of the applications for courier is not in the scope of this project.

The courier will only be available in the platform after the administrator's approval.

5.2.4 FF04 - Log In

User Groups: Unregistered User

Requirement Description

In order to authenticate before the platform, the unregistered user needs to log into an account. This is done with the email and password set upon the registration. If the password was changed after the registration, the latest password must be used.

5.2.5 FF05 - Recover Password

User Groups: Unregistered User

Requirement Description

In case of a lost/stolen password, an user may request for a password restore. To do this, the user must provide the email associated to the account and an email will be sent with a link to restore the password.

5.2.6 FF06 - Place an order for a service

User Groups: Customer

Requirement Description

When a customer selects a service that he/she desires, it will trigger the main process of the platform. Starting on the Service Display Page (SDP), the user needs to input some information regarding the service itself (in case of laundry, the platform will need information about the size of the size of the article, among others). After filling up all information regarding the selected service, the customer will be taken to the checkout page. In the checkout page, the customer will need to fill more information:

- Pickup address
- Delivery address
- Billing address
- Preferred pickup hours

- Preferred delivery hours

After filling up all the information, the customer will be led to the payment page where he/she will select one of the existing payment options. This process will be done by a third party.

When the payment is completed, the order will be registered in back-office, an email with the order confirmation will be sent to the customer, and the order will be available to be consulted to the customer, the service provider that was selected and the couriers, to assign the orders. At this point, the customer also needs to print the shipping guide, that will follow the article to the article.

5.2.7 FF07 - Check all available service providers

User Groups: Customers, Unregistered User

Requirement Description

When entering the platform web page, the customer (registered or not) will be able to check all the existing service providers and their services. All the services will be presented under its service provider on the Service Listing Page (SLP). On this page, only basic information will be shown, as a description of the service and an estimate of prices.

5.2.8 FF08 - Check order history

User Groups: Customers

Requirement Description

The customer will be able to check their order history. This menu will be available as a tab of the "My Account" page.

This page will show all orders made, date, and status (check section 5.5 for more information regarding the order states). The customer can also open the orders to get more information of what was ordered.

5.2.9 FF09 - Evaluate Experience

User Groups: Customers

Requirement Description

Every service provider will have a rating, depending on the performance for each service. This rating will be made by the customer after each order. The customer will need to rate some aspects regarding their experience with the platform, the service provider and the

courier. In the first version, only service providers will have a rating. The customer will need to answer questions (classify 0-10) as:

- How was your experience regarding the order process?
- Did the Service Provider fulfill your expectations?
- What about the experience with the courier?

5.2.10 FF10 - Manage available services

User Groups: Service Providers

Requirement Description

As the platform's service providers' business grows, so will grow the offering of services available. This menu allows the service providers to create new, change and delete existent services. To be noted that a change of a service will actually create a new one, in order for the current orders for that service don't be changed.

5.2.11 FF11 - Manage orders

User Groups: Service Providers, Couriers

Requirement Description

In this menu, the service provider will be able to check all orders (current and past) and move them between a series of steps:

1. New
2. Picked on customer address
3. Delivered to Service Provider
4. Service Started
5. Service Finished
6. Picked on Service Provider Address
7. Delivered to Customer
8. Canceled

The section 5.5 provides further information regarding the order states and its transitions.

If the shipping guide is lost for some reason, the service provider can also print that document.

5.2.12 FF12 - Change Area of Actuation

User Groups: Service Providers, Couriers

Requirement Description

Both the couriers and service providers will need to select the Area of Actuation (AoA). Only customers within this area of actuation will be able to select the services of the service provider in question. Likewise, couriers also need to select an area where they can make transportation. For an order to be fulfilled, there needs to be a courier that has both the customer and the service provider in their AoA.

5.2.13 FF13 - Change service profile

User Groups: Couriers

Requirement Description

There may be services that require different kinds of vehicle for the goods to be transported. In this menu, the courier can change his vehicle type. There will be three options:

- Scooter
- Car
- Van

In this case, a Van can transport any kind of goods, and a scooter will only be able to transport small goods.

5.2.14 FF14 - Assign orders for pickup

User Groups: Couriers

Requirement Description

The courier will be presented all the unassigned orders of in the defined AoA. Here, the courier will be able to assign himself to the orders, meaning that he/she accepts to make the transportation and picking up the goods the schedule requested by the customer. When this assign is done, an email notification is sent to the courier's email and the customer as well.

5.2.15 FF15 - Manage Service Providers

User Groups: Administrators

Requirement Description

In this menu, the platform administrators will be able to create, remove and consult information about the existing Service Providers.

5.2.16 FF16 - Manage Couriers

User Groups: Administrators

Requirement Description

In this menu, the platform administrators will be able to create, remove and consult information about the existing Couriers.

5.3 Non-Functional Requirements

In the previous section, the functional requirements were identified. Those requirements help to understand **what** the application will do. On the other hand, non-functional requirements specify **how** the application will do it.

In this section, an overview will be given regarding these requirements. They are to be considered throughout the development of the features identified in the section 5.2.

5.3.1 Usability

The system needs to be accessible from anywhere. This means that web apps need to be responsive for users being able to access it from mobile devices.

5.3.2 Security

The SnapTasks Portal is the one which will be responsible for managing the order requests and by consequence, it will also be responsible for managing the payments. It is mandatory to ensure the security of the payments, to diminish fraudulent orders.

To avoid man-in-the-middle attacks, all communications between services must be done using HTTPS.

All personal data must be handled taking into account the current legislation.

The system as a whole must have, at least, basic security systems, as:

- **Authentication:** the system must be sure of who is making a certain request. This can be managed using username/password login;
- **Authorization:** the system must ensure that whoever is making a certain request, has permissions to do so. This can be managed by assigning roles to users, where a role grants access to a number of functionalities.

5.3.3 Availability

The SnapTasks Portal must have the minimum downtime possible, for not losing customers because of having the website down. The back-office tools should also be down the least time possible. However, this scenario has a lighter impact, since the customers are still able to access the portal and place orders.

5.3.4 Scalability

The delivered solution must be capable to be scaled dynamically. This means that the access of many different users at the same time does not affect the website performance.

To attain this, a cloud service, such as *Azure* is to be used, allowing to increase and decrease the hardware resources depending on the needs.

5.3.5 Resilience

The platform must be resilient to handle wrong input from the users or sudden failures of dependent services.

Those errors should be handled and the user who has made the request should be informed about what happened.

5.3.6 Performance

The system must be fast and efficient in order not to lose a customer because the application was not responding. For this to happen, most of the processing should be done in back-office, using asynchronous processes.

This requirement will be more important as the platform has more users.

5.3.7 Internationalization and Localization

The platform will be used firstly in Portugal, therefore, it should have its front-office in Portuguese.

However, as the platform grows, there may be the need to have SnapTasks Portal in different languages and cultures. The application should be prepared for that case.

5.3.8 Maintainability

The platform code should be easy to maintain. This means that both bug fixing and adding new features should be as easy as possible, following the industry's best practices.

5.4 Process View

To have a better understanding of how the whole platform will work, we need to have a clear view of the operational process which this system supports. This can be achieved by analyzing the flow that an order takes, since it is placed until the service is finished and all goods are returned to the customer.

It starts when a registered customer accesses the system. He/she will select a service he/she needs, proceed to checkout, make the payment and place the order. At this point, the order is created in the system and can be accessed by the courier to assign himself, compromising to make the delivery from the customer's pickup address to the service provider address. This assignment is only for the first leg of this order (from the customer to the SP). At the date selected by the customer, the courier will go to the customer's address to pickup the goods associated to the order and then deliver them to the service provider. The service provider now has the goods and is ready perform the ordered service. After the service is finished, the order status is changed to *Service Finished*, allowing the couriers to know that the goods associated to that order are ready to be returned to the customer. Therefore, they also can assign themselves to that transportation, compromising that they will do the second leg of the order (from the SP to the customer). In this case, the goods can be picked up at any time, because the service provider has an open store. After picking up the goods at the SP's address, the courier will deliver the items to the customer, finishing the process.

The figure 5.2 presents a more graphical view of this process and shows how the user groups interact.

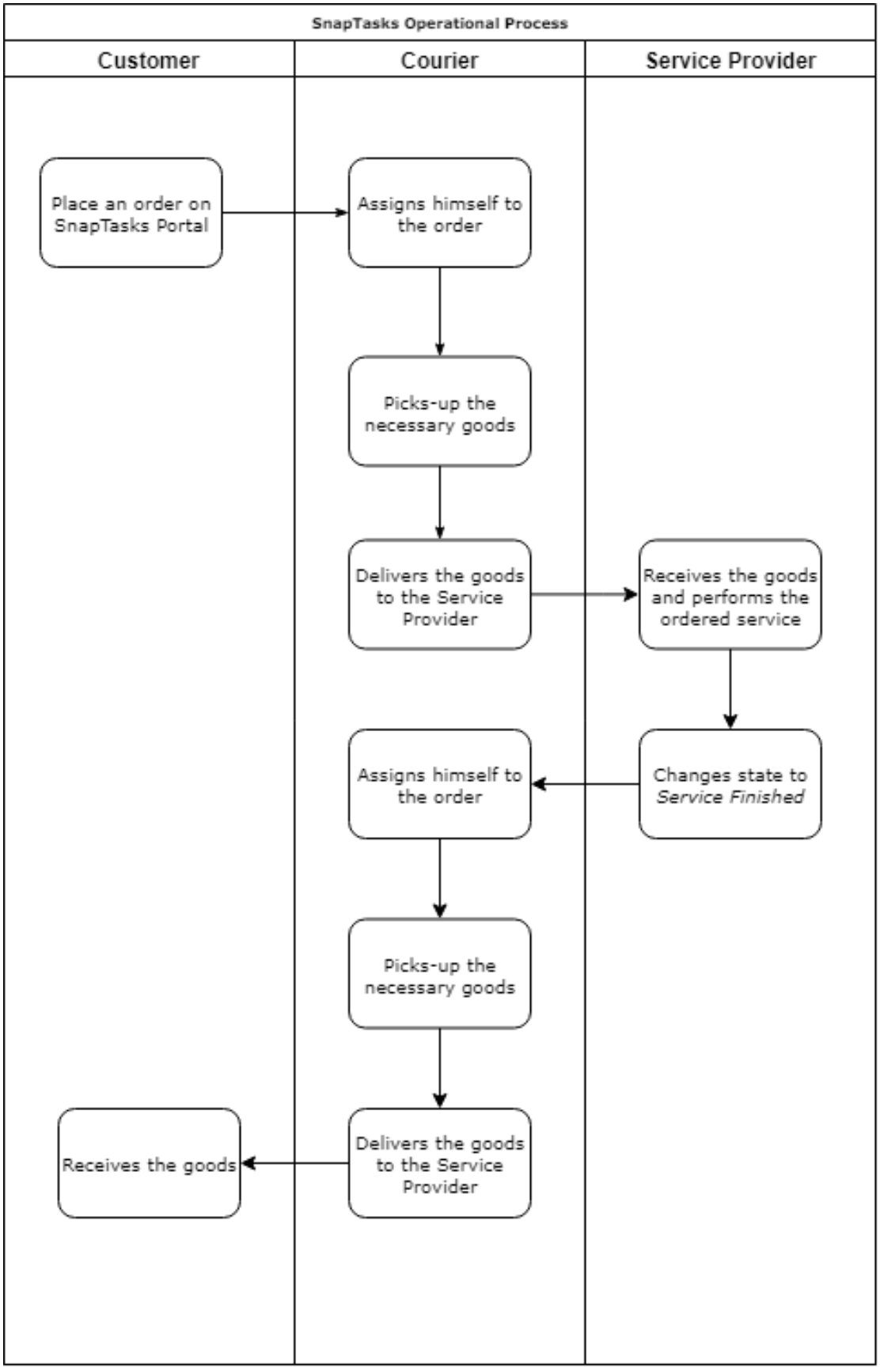


Figure 5.2: Process diagram

5.5 Order Process States

In the previous section, it was possible to understand that the order goes through several steps for the service to be fulfilled. These steps are represented by the states in which an order can be. Each state represents something that is occurring/occurred, or an action that is needed.

The possible states for a given order are the following:

- New;
- Picked on customer address;
- Delivered to Service Provider;
- Service Started;
- Service Finished;
- Picked on Service Provider Address;
- Delivered to Customer;
- Canceled.

These states reflect the operational status. Each order must go through these states sequentially. For example, an order cannot be delivered to the customer before the service is finished, or, an order's service cannot start before the goods are delivered to the service provider.

The figure 5.3 shows how the order states connect to each other and represents the happy path workflow.

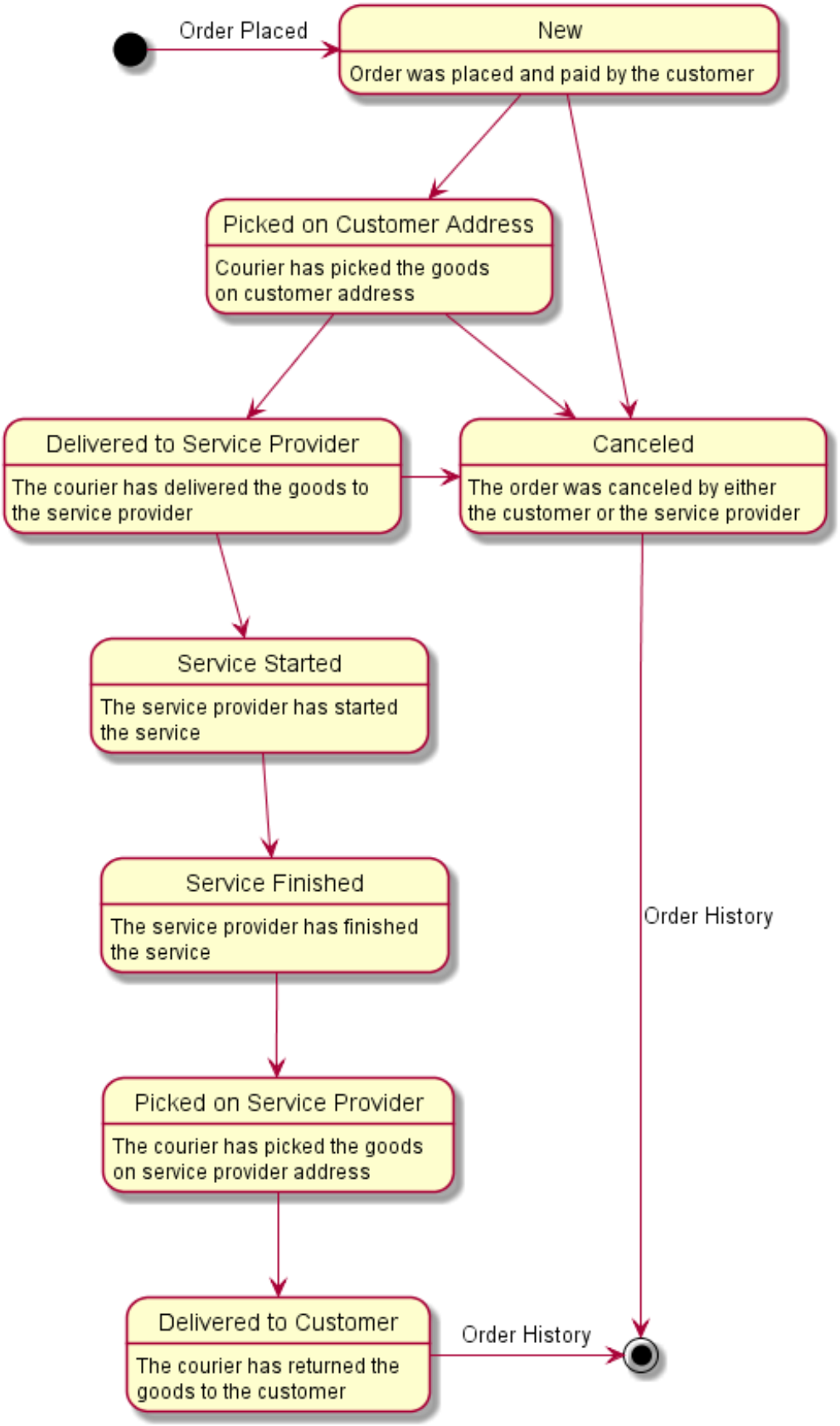


Figure 5.3: Order State diagram

Chapter 6

Analysis of the Existent Solutions

After the identification of the functional and non-functional requirements of this project, in the previous chapter, it is necessary to have a better understanding about which of the architectures and technologies, identified in the section 3.2, are the best choice to be used in the context of this dissertation. It is important to perceive the difference between each of them and compare its pros and cons. In this chapter, the advantages and disadvantages of using each of the technologies or architectures in the context of the project of this thesis.

6.1 Architectural Approaches

The first topic to be discussed, is also one of the first topic to discuss when developing a new project from scratch. The architecture of a system is like the skeleton a human being. It sustains the whole body while actually doing nothing by itself. A clear definition of the architecture that will be used in the system may prevent many problems since it provides a clear vision of the system as a whole.

In the section 3.2.1, a few architectures were presented to the reader. In this section, the best and the worst of each of the architectures will be presented and then compared using Cost-Benefit Analysis Method (CBAM). Developed by the Software Engineering Institute (SEI), this method is an "architecture-centric method for analyzing the costs, benefits and schedule implications of architectural decisions" (Ionita, Hammer, and Obbink 2005).

6.1.1 Micro-Services

Micro-services, as was already mentioned in section 3.2.1, micro-services architecture is a architectural style where applications are broken in autonomous "small pieces", the so called services, with a very well defined responsibility.

One of the biggest advantages of the micro-services architecture is that each service is independent of the others. It is possible to scale or deploy just one service at a time without affecting the others. This facilitates the development because, since the services are independent of each other, they can be developed in different languages. However, the communication protocol must remain the common. This makes the service more targeted to what is its responsibility. All this makes the system more robust as a whole. As the services are loosely coupled, if one services fails, only the functionalists that need that particular service will be affected as opposed to the monolithic, where if anything fails, everything fails. Lastly, it is easier to coordinate the development with several teams when the system uses a

micro-services architecture, since every development will be localized in a single application and there will be less conflicts when merging code (Hubbell 2019).

On the other hand, if the initial definition of the responsibility is not well defined, it is much harder to make a structural refactor on a system with this architecture than it is with a monolith. Besides, any interface changes need to be coordinated with the teams that work on the other services. The benefit of being able to deploy independently creates another problem. There needs to be a deployment process for each application. If each service has the need to have its own deployment process, this can affect the complexity of maintaining that process. Testing may also be another problem. If the service is being tested has dependencies of other services, it may be difficult to test it. Either the service has correct data to be accessed directly, or the dependencies are mocked. Finally, as the communication between service is made through the network, this may affect the overall performance of the system (Hubbell 2019).

As micro-services are a part of SOA, the analysis will be kept in this more specific architecture.

6.1.2 Event Sourcing

As was already mentioned in section 3.2.1, Event Sourcing is an architectural pattern where every change on an entity, generates and persists an event. This event will be listened by the interested parties which will or will not do something about it, as its logic dictates.

The best thing about this pattern is the preservation of the state history. This element is very useful for certain businesses as banks, where every change may be audited. In Event Sourcing, every change in the state of an entity is kept in record. Using an event driver to communicate asynchronously between services offers a extremely loosely coupled alternative to the traditional HTTP requests, since in this case, the service doesn't even know about the existence of the service where it is getting information from (Michelson 2011). In case a business process needs to do several things in response to a change on an entity, the responsibility to trigger those actions are no longer in the source of that change, but on the services that are listening the event. They are the ones responsible to take (or not) action over a certain event.

Nonetheless, as it is true for every pattern, this approach also brings problems. Tools for development are still rare and as it is a relatively new alternative, there are not many use cases to take as an example (Queiroz 2017). Also, for development environment, this requires all services that have dependencies to be synchronized, which may bring problems for integration tests.

6.1.3 CQRS

In a nutshell, CQRS is a pattern that segregates the reads and the writes into two different components. This is explained in more detail in section 3.2.1.

CQRS offers great benefits for systems that have a big discrepancy between the number of reads and writes, since, as they are separate components of the system, they can be scaled independently, offering a bigger flexibility to the system. In case of a big concurrency happening in the system, it is also possible to optimize each component for the action that

they perform. Also, it offers the possibility to have different representations for read and write operations.

However, using CQRS pattern offers several risks. The use of different models may cause inconsistency problems between them. This inconsistency is even bigger for systems that have a big overlap between reads and updates, which is true for most applications. If the application doesn't have a lot of traffic to handle, the usage o CQRS may bring more problems than benefits. As this platform is expected to grow, and to have its traffic increasing over time, this is something that may bring some problems.

Its complexity is considerable and it is focused in a specific problem to be solved.

6.1.4 Summary

In the previous points, several benefits and risks were presented regarding each of the possible architectures. The table 6.1 presents a summary of the relevant pros and cons of each one.

Table 6.1: Summary of Pros and Cons of each evaluated architecture

Architecture	Pros	Cons
Micro-Services	Services are Independent Different languages are supported Low coupling Resilience Easy coordination between teams	Need for a clear definition of each service Interface changes need to be coordinated Many deployment processes Testing dependencies Performance
Event Sourcing	State history persistence Extremely low coupling Event triggers all needed processes	Few use cases implemented All dependencies must be synchronized for tests
CQRS	Flexibility on reads/writes Independent scaling and optimization Different representations for reads and writes	Possible inconsistency problems Complexity

Considering the project of this thesis, it was decided, all things considered, to go with the micro-services approach. This decision was based on the wide number of use cases that exist for this architectural style, its proved resilience, which is also a non-functional requirement of this project (check section 5.3). Since there is no requirement for an extremely low coupling and independent scaling between read and write operations, the complexity of CQRS and the risk of implementing an architecture with fewer use cases implemented as event sourcing, don't compensate the pros of these architectures, meaning that they are not the best option for the problem we are trying to solve.

6.2 Software Development Frameworks

The purpose of using a Software Development Framework is to have, right out of the box, a series of tools, abstractions and libraries that help on the development process. It is very important to use a framework that fulfils the needs of the project and facilitates its implementation.

In the next sections, several possibilities will be analyzed in order to correctly understand which one fits this project the best. The approach to do this will be similar to what was done in the section 6.1, weighing the pros and cons of each one.

6.2.1 .NET Core

.NET Core, as explained in section 3.2.2, is a cross-platform software framework that provides several tools to create applications for different purposes.

In this project, we aim to develop an e-commerce platform that provides different functionalities for logistics in service provision. Since this framework allows the creation of both APIs and web applications, it means that it is possible to develop the project in this framework.

The fact of being cross-platform, means that the costs can be reduced since the services can run on Linux, which is a cheaper option when compared to Windows Server. This framework also provides a seamless integration with azure which enables the deploy of services directly from Visual Studio.

The automatic monitoring saves us the time of having to implement the basic monitoring features such as response time, application exceptions and throughput.

Regarding the issues of this framework (also identified in section 3.2.2) holds on the licensing costs for the usage of some functionalities in Visual Studio, the low flexibility o Entity Framework, and the fact that Microsoft technology depends a lot on the company's decisions.

Taking into account that this is an academic project, the existence of academic licenses removes the first issue. On the other hand, the other two cons are still unsolved.

6.2.2 Java Spring

As presented in section 3.2.2, Java Spring is an open-source software framework that, similarly to .NET Core, allows the development of both web applications and APIs. Since these are the essential application types of this project, it is also possible to be developed using this framework

Java Spring is a very flexible framework that provides tools for several cross-cutting concerns, such authentication, authorization and logging. It also provides support for AOP, which enables the implementation of those cross-cutting concerns in a clean way.

However, this framework has a high learning curve due to the large number of classes and tools that the developers need to know to develop efficiently, which can make the project more complex which, consequentially, makes the development to take longer. Additionally, there is a lack of security guidelines, as was already mentioned in the section 3.2.2.

Considering the nature of the project (e-commerce and with a fixed delivery date), these cons are very difficult to overcome.

6.2.3 Node.js

Node.js is a software framework that was designed for applications that require real-time updated data, such as video-conference rooms or online gaming, as mentioned in section 3.2.2. As the last two frameworks, it also allows the development of both APIs and web applications, which allows the project to be developed in this framework.

This is a very lightweight framework, since it uses non-blocking, event-driven I/O, that provides tools to build fast and scalable applications. It uses JavaScript as its development language, which keeps the learning curve to a minimum for a team that already has to use JavaScript in the front-end, since this language is largely used for front-end purposes.

However, since this framework is largely developed by the community, there are many tools that are of poor quality, specially in non-core modules. Also, as Node.js is single threaded, it is not indicated for applications that require heavy CPU operations.

As this project does not have any requirement regarding real-time data, the biggest advantage of using Node.js ends up having no use. The low learning curve is also a pro that does not apply to our case, since no one in the development team has experience in JavaScript

6.2.4 Summary

In the previous sections, the pros and cons of each framework and its viability for this project were analyzed. The table 6.2 presents a summary of the relevant benefits and risks of each one.

For the project of this dissertation, and taking into account the benefits and risks of the three evaluated frameworks, it was decided to go for the .NET Core framework. This decision was mainly for the cross-platform possibility and its reliability. As the main cons are concerning licensing costs and there are student licenses available for free, the downside of this framework is fairly reduced.

When compared to the Java Spring, the main problem is the Spring high complexity and learning curve. Also, the lack of security guidelines for this framework is a threat for an e-commerce platform, as this project is. With the existent deadlines, the benefits of this framework doesn't pay off the risks.

In the case of Node, its lightweight and fit for micro-services architectures would be a great plus. However, with the problems of the existent tools, and the unfamiliarity by the development team of this technology, it was understood, that .NET Core would be a better fit.

Table 6.2: Summary of Pros and Cons of each evaluated software development framework

Framework	Pros	Cons
.NET Core	Cross-platform Development in Visual Studio Multi-Language Automatic Monitoring	Licensing Costs Low flexibility of Entity Framework Dependency on Microsoft
Java Spring	Flexibility Support for AOP Provides several useful libraries	High learning curve Complexity Lack of security guidelines
Node.js	Lightweight Same language as front-end Low learning curve Good option for micro-services	Many tools are of poor quality Callback hell Single-thread Quality of code on non-core modules

Chapter 7

Design and Architecture

The importance of a correct architecture was already mentioned in the sections 3.2.1 and 6.1. For that reason, in this chapter, we will start with an architectural study, where the bet architecture for the system was thought and designed. Although the architecture of a system is a very important step while designing a system, the inner design of the component must not be forgotten. That design will also be presented in this chapter. Furthermore, the domain model of the platform, will also be presented.

7.1 Architectural Study

In order to have the most correct architecture for the platform in analysis in this dissertation, a study took place, where the system as a whole was examined to see how it would be possible to model an architecture that would fit our system. The name of the platform is SnapTasks.

The first step was to identify how many different front-end applications the system would need. This was done taking into account the existent users groups that exist and their needs. As it was already presented in section 5.1, there are four main actors: the final customers, the service providers, the courier and, finally, the administrators. The decision of what applications to have was mostly based on the actors that exist, and their needs.

These will be the client applications of the platform:

- **SnapTasks Website**: the main website of the platform. This is where the users will create their accounts, customers will place their orders and know the available services;
- **Back-Office (BO) Management**: this application will be responsible for all BO and administration operations. Service providers will track their orders in this application;
- **SnapTasks Mobile Application**: similarly to the main website, this will provide essentially the same functions, but on a mobile Android app. Taking into account the graph in figure 3.1, it is almost mandatory to have a mobile application, from the beginning;
- **Courier App**: this application allows couriers to check the orders that need pickup and assign themselves to them. This application will also provide the location where the courier needs to pickup/delivery and integrate with the Global Positioning System (GPS).

As these applications will be mandatory in our application, one possible architecture would be to keep things to a minimum and have only one API that would be responsible to provide all the necessary functionalities to the client apps. Furthermore, it would also be needed a

database to persist all the data. Having the minimum number of components means that, in short-term, there will be less development work, the monitoring will also be easier (Lumetta 2018). This architecture is presented in figure 7.1.

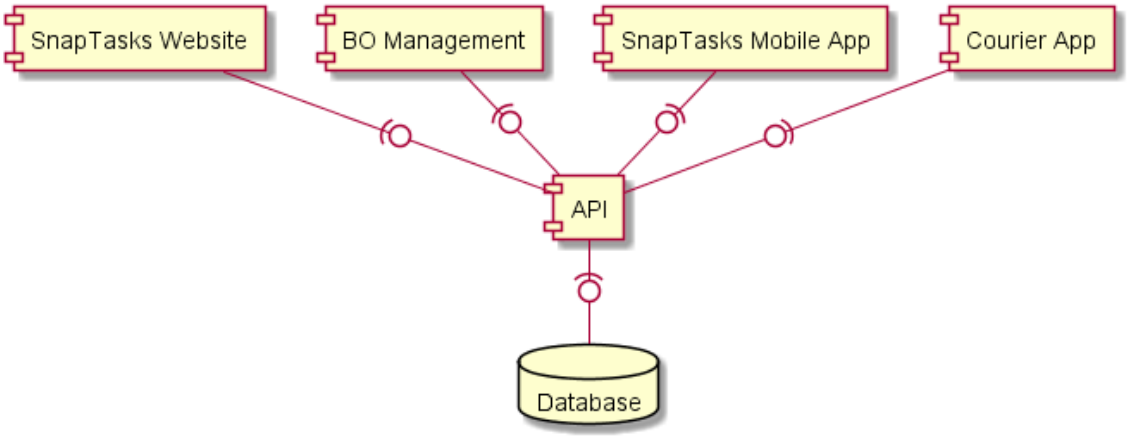


Figure 7.1: Possible architectural solution (Monolith)

These architectures are known as monolith. This is because all the logic of the system is centred in a single component, without any rule. Monolithic architectures bring more evil than good. With every logic located in the same place, the code can easily become tightly coupled and become very hard to read (Gibson 2015). On a logical thinking, it doesn't make sense to provide functions that a certain application will never need.

In the proposal presented in the figure 7.1, this happens with the Front-Office (FO) and BO applications. Since each application type needs very different kinds of functionalities. The first ones will need more customer focused operations such as get the available services and placing an order. The second ones need more operational functionalities, like managing service providers or add more services to the platform.

For that reason, it was decided to split the API into two different ones. The first for front-office operations and the second for back-office operations. In this solution, it is possible to segregate the functionalities and restrict its access to the applications that actually, need them. The figure 7.2 presents an illustration of the possible separation of these functionalities.

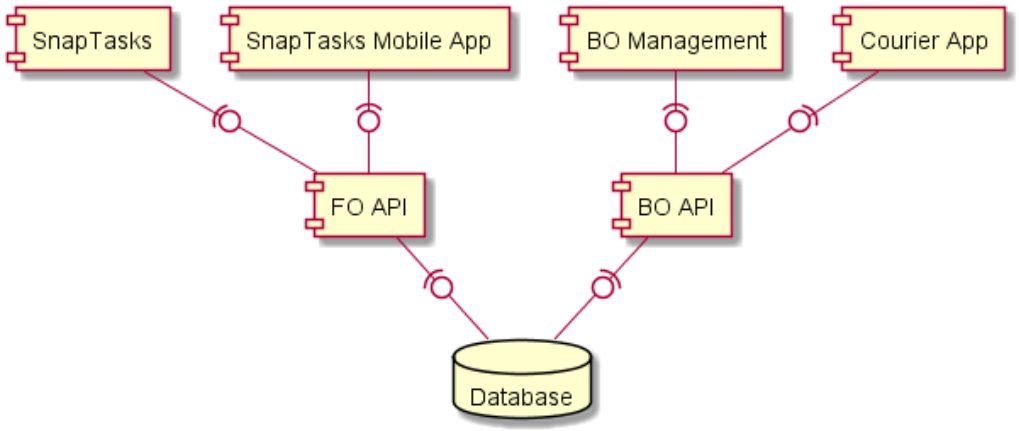


Figure 7.2: Separation of FO and BO APIs

In this architecture, the separation of allowed not only to separate the concerns of each API, but also, to have less load on the services. In case of an unexpected increase of orders, it is possible to temporarily boost the resources of just the FO API, instead of having to upgrade the whole system.

On the other hand, as the application is expected to grow both in terms of number of functionalities and in terms of load, it would be better to have the code even more segmented and decoupled. To achieve this, and as was already explained on section 6.1.4, a micro-services architecture was designed.

To design this architecture, it was needed to identify which services were needed and what was their responsibility. From that analysis, the following services were identified:

- **Pricing Service**: this service is responsible for the creation, calculation and get operations of prices. Each new price that is entered will create a new registry on the database, in order to keep track of the prices and provide a link of the prices that were used in the orders;
- **Order Management Service**: this service is responsible for all the operations that concern the orders. This includes the overall management (create, update), transitions between steps and other logic that is inherent to this entity;
- **Service Provider Management Service**: similarly, this service is responsible for the operations that regard both the service providers and their services. Management operations like create, deactivate and update are under the scope of this service;
- **Courier Service**: the last service is relative to the couriers. Following the previous logic, this service is also responsible to manage the couriers and provide operations that regard the couriers.

All services will be bellow both FO API and BO API, which means that those APIs can access the services, but the services can't access the APIs. Furthermore, as the services are only responsible to make operations that only affect the entity in question, there should be no reason for a micro-service to access another micro-service. Lastly, each service is provided a database and the service is responsible to manage it. No service can access the database of another service. The figure 7.3 presents the proposed architecture.

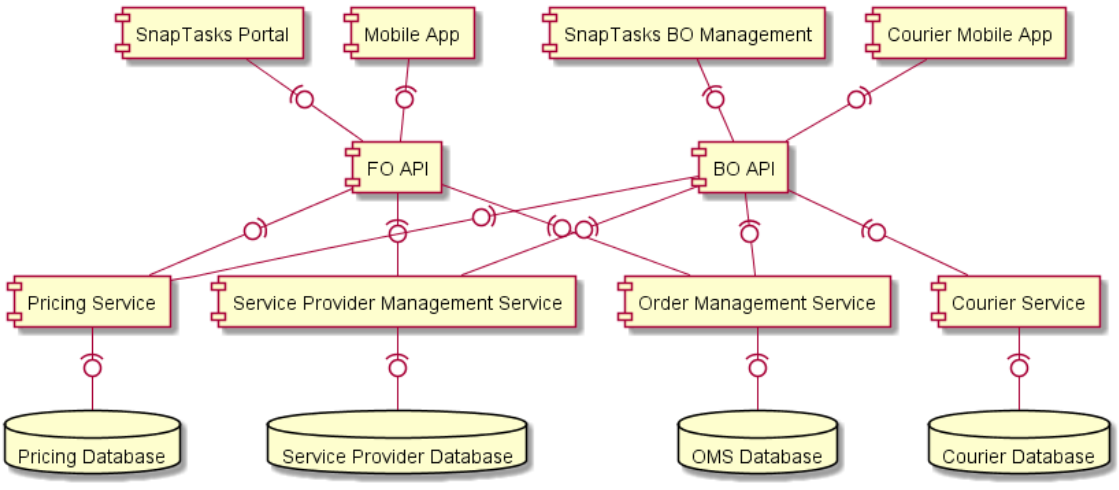


Figure 7.3: Proposal of a Micro-Services Architecture

7.2 Domain Model

The design of the domain model is one of the most critical phases in software engineering. This phase not only allows the team to have a better understanding of the business that the application is contained, by helping to identify key concepts and ideas of the business domain, but can also serve as a baseline for the development of a new solution. The domain model is a visual representation of the entities that concern the problem that our application is trying to solve.

With the purpose of having a better understanding of the problem it is being aimed to solve, a domain model was designed. This domain model presents the main concepts and entities that are present in the domain. The figure 7.4 represents the visual component of the most important concepts on the business domain.

With this visual representation of concepts, it's easier to understand what are the key entities that concern the problem. However, there is still a need to deep dive into these concepts to know what they represent.

The **order** is one of the most central concepts in the diagram because it is one of the most crucial components of the solution. The whole purpose of the platform is to create and process orders. They are placed by the final **customer** by interacting with the platform front-end. When the customer places his/her order, it is possible to define where and when he/she desires that the pickup of goods is to be made. This order also has a **cost**, which is the value that the customer paid for it, which includes both the value for each service, and the logistics fee, for the transportation of goods (including promocodes, if that is the case).

The **service providers** are another fundamental part of the platform. They are the ones who provide the **services** that are available on the platform without whom, there would be no orders to be placed. The SP also has a **representative**, which is the point of contact between the provider and the platform. To improve the service quality and avoid orders from customers who are too far away from the SP, it is required for the service provider to define an **area of actuation**. This AoA defines a circular area with its center in the SP's address, where it is possible to order from that service provider.

The services that are provided can be of several **service types** (laundry, mechanics, warranty, etc.), however, for the first version of this application it will only allow laundry services. Nevertheless, the platform is prepared to provide any kind of services. The platform is also prepared to have multiple services gathered in a single order, but as an MVP, only one service can be ordered in the front-end.

Each service has its own **prices**. The price represents the breakdown of what the user will pay por a given service.

The goods that concern a given order are transported from the customer's and the service provider's **address** by the **courier**. This actor is also responsible to assign him/herself to the orders that he/she will deliver. The address contains the necessary information to reach the customer and the service provider, such as name, address info, zip code, city, etc.

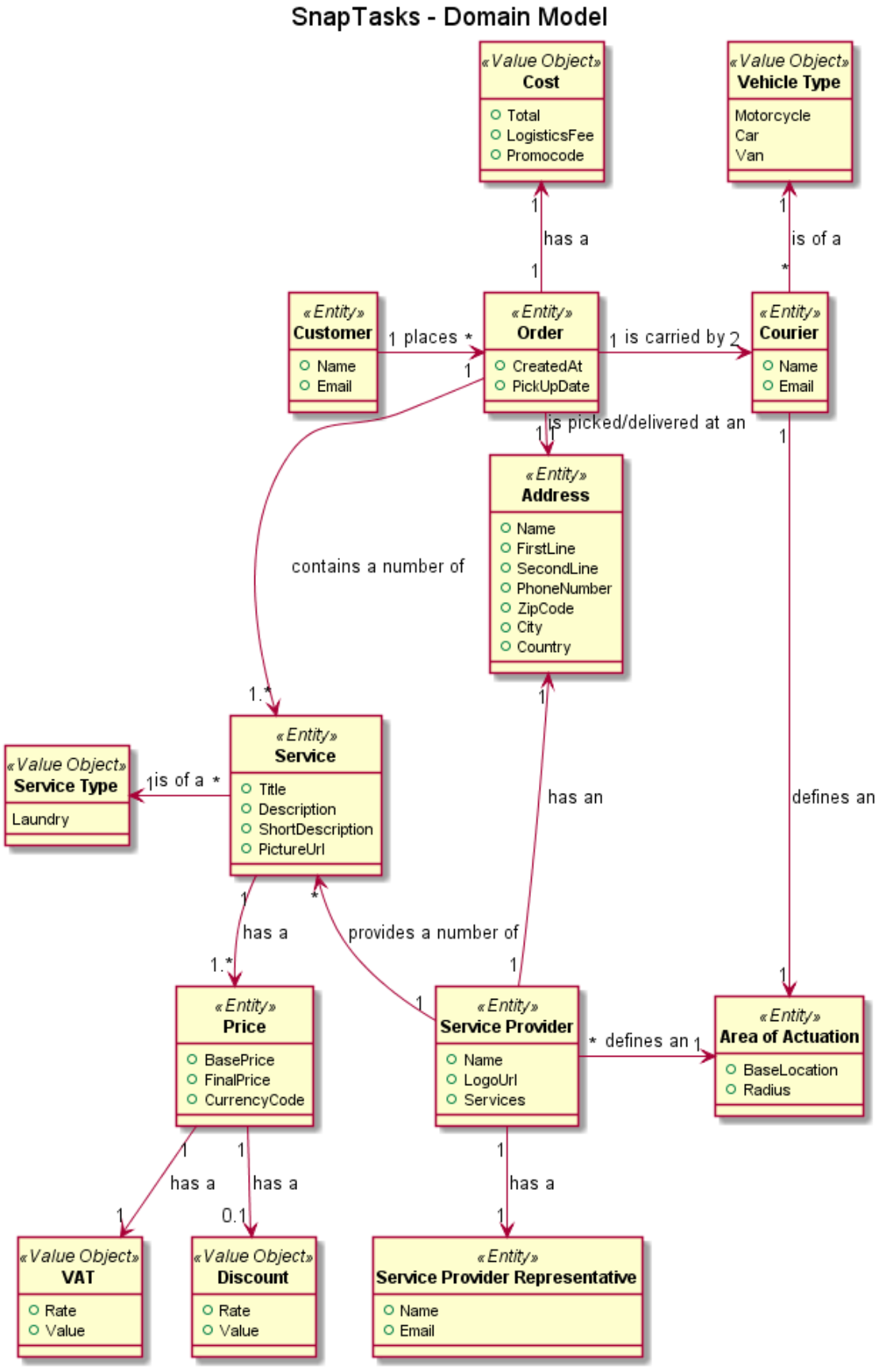


Figure 7.4: Snaptasks Domain Model

7.3 Internal Project Structure

In the section 7.1 the macro architecture of the solution was presented. On the other hand, the internal project structure of each component is still to be presented. The present section is the one where that will be done.

For this solution, each component had an internal layer architecture. This is one of the most common architecture patterns in the industry, and is also known as n-tier architecture pattern. This pattern organizes the application into horizontal layers, each one with its own responsibility, performing a specific role for the application. Despite the nonexistence of a rule for how many layers to exist in a given application, the standard consists in four layers: presentation, business, persistence and database (Richards 2015).

The layered architecture pattern has as one of the most powerful features the separation of concerns among components. The components within a given layer will only need to deal with logic that is that layer's responsibility. For example, the components that are inside the business layer, will only need to deal with business logic. Same applies for the components of the persistence layer, which will only need to have persistence related logic. This makes the components' responsibility easy to define and, by consequence, makes it easier to develop, test and maintain them.

The figure 7.5 presents the generic layer structure used for the services that compose Snap-Tasks solution. Despite the layer structure being the same for all components, the biggest difference resides in the *Data* layer. Given that only the bottom services have access to databases, only those services will have a *Repository* and *Database Object Database Object (DBO)* sub-layer, since without the need to access a database, there is no need for repository logic and DBO. Instead, those services will most likely need to access other services in order to retrieve data. That logic is contained in the *Gateway*.

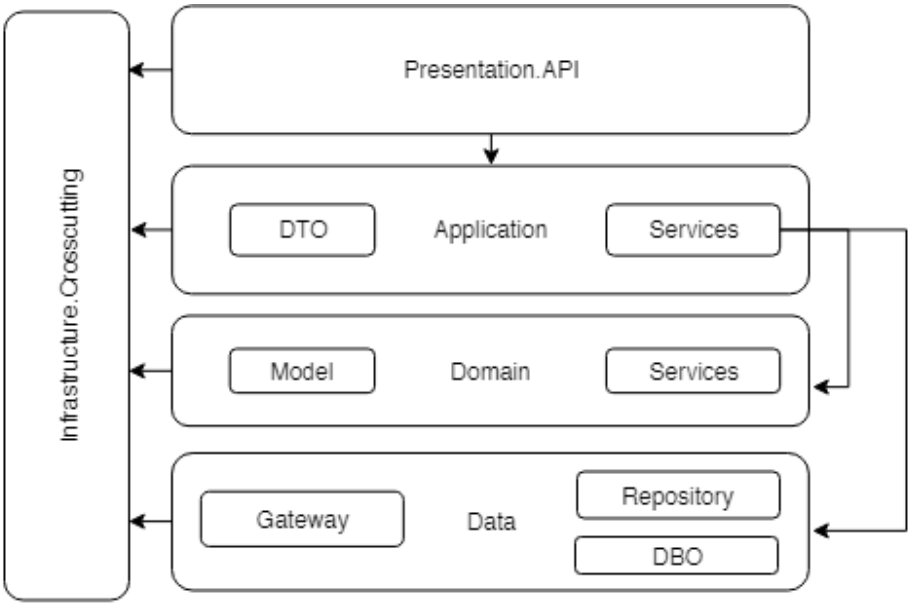


Figure 7.5: Generic representation of the internal project structure

7.3.1 Presentation

The top layer is named **Presentation.API** or **Presentation.Web**, depending if the component is an API or an application with Graphic User Interface (GUI) respectively, is responsible to handle the application's incoming requests. This layer is also responsible to ensure that the incoming data is valid to proceed to the next layer. When the application has a GUI, this layer is also responsible to handle the user interaction with the interface. The figure 8.6, of section 8.4.1, presents the SLP of one SP.

In the case of the APIs, it is possible to check the available endpoints in the API's Swagger page (more information about swagger on section 3.2.3).

7.3.2 Application

The second layer in the chain, *Application*, is the one that is responsible for most of the business logic. This layer is divided into two sub-layers, **Application.Dto** and **Application.Services**. The first one is where the Data Transfer Objects (DTOs) are defined. The segregation of this to a separate package, allows to generate a Dynamic Link Library (DLL) that can be used by other projects that need to use this application's DTOs.

The second sub-layer, is the one responsible for the business logic itself. The methods of this layer will be called by the presentation layer, retrieve data from the data layer below, apply the necessary logic and then retrieve it to the presentation layer.

7.3.3 Domain

The *Domain* layer is the one where the domain-related logic is contained. It is divided into two separate sub-layers, **Domain.Model** where the domain model is specified, and **Domain.Services**, where domain-related logic (for example, logic that concerns only one object) are implemented. Most of these methods are implemented using as extension methods, and can be used as an extension of the object itself.

7.3.4 Data

The bottom layer is responsible to know where and how to get and persist data in order to the application to work properly. The *Data* layer can be composed by the **Data.Repository** and **Data.Dbo**, when the application needs to access a database, and by a **Data.Gateway**, if it needs to access another application, being that application within the SnapTasks ecosystem, or not.

The *Data.Dbo* sub-layer is where the DBOs are defined. These objects are a representation of the database itself and are needed in order to the used ORM, Entity Framework Core, to work properly. These objects are also used by the other sub-layer to access the database. In the *Data.Repository*, all the logic needed to access the database is contained.

In terms of the *Data.Gateway*, it contains all the necessary logic needed to access an external service. As opposed to when we access a repository, we also have ownership of the DBOs needed to access, when accessing an external service, that ownership belongs to the

application we are calling. The *Data.Gateway* needs to use the DTOs provided by that service (also check section 7.3.2).

7.3.5 Infrastructure

Last, but not least, there is the *Infrastructure* layer. This layer is responsible for the cross-cutting concerns that are required by all the other layers. That includes:

- Logging
- Authorization
- Authentication
- Mappings
- Caching

This layer ends up being slightly different from the others because it is not above or below any of the other layers. Instead, it is on the side, and may be accessed by any of the other layers. However, the *Infrastructure* layer can not access any of the remaining layers of the project.

Chapter 8

Implementation

This chapter approaches the most technical component of this thesis. Here, it will be approached how the first version of the application was developed, considering both functional and non-functional requirements, defined in chapter 5. Starting from how the code is structured in the different components, proceeding to the explanation of one of the main features of the project, the price calculation and how the code quality was measured. Finally, the application's user interfaces will be presented.

8.1 Code Structure

For the development of the project of this dissertation, the IDE used was Visual Studio 2019. Each of the components identified in section 7.1 resulted in a solution in Visual Studio and, also, a different repository in the Version Control System (VCS). In this case, the VCS used was *Git*.

Inside the IDE, the code was structured taking into account what was presented in section 7.3. Each of the layers was mapped into a different *Class Library* project. This allowed to have the code to be compiled separately which eases the management of dependencies between layers. After compilation, each project will result in a different DLL. This allows for the compiled code to be used in different solutions by either having a direct reference to the compiled DLL, or by creating a *nuget* package and later installing it in the desired projects.

The figure 8.1 presents the code structure on Visual Studio. For this example, the component used was the *Order Management Service*.

8.2 Price Calculation

The price calculation strategy is one of the most important components of an e-commerce website. It can be the thing that makes a customer come back to use the platform or, otherwise, it can be the thing that makes the customer to never come back. Furthermore, the calculation and breakdown of the price is also very important for law compliance. If, for example, the prices' Value Added Tax (VAT) parcel is not correctly calculated, the company may be paying less taxes than it should. This can result in fines that the company must pay, unnecessarily.

The price is an entity by itself rather than being a property of the service, as shown on figure 7.4. Each price has is immutable and have its own identifier. When an order is placed, this

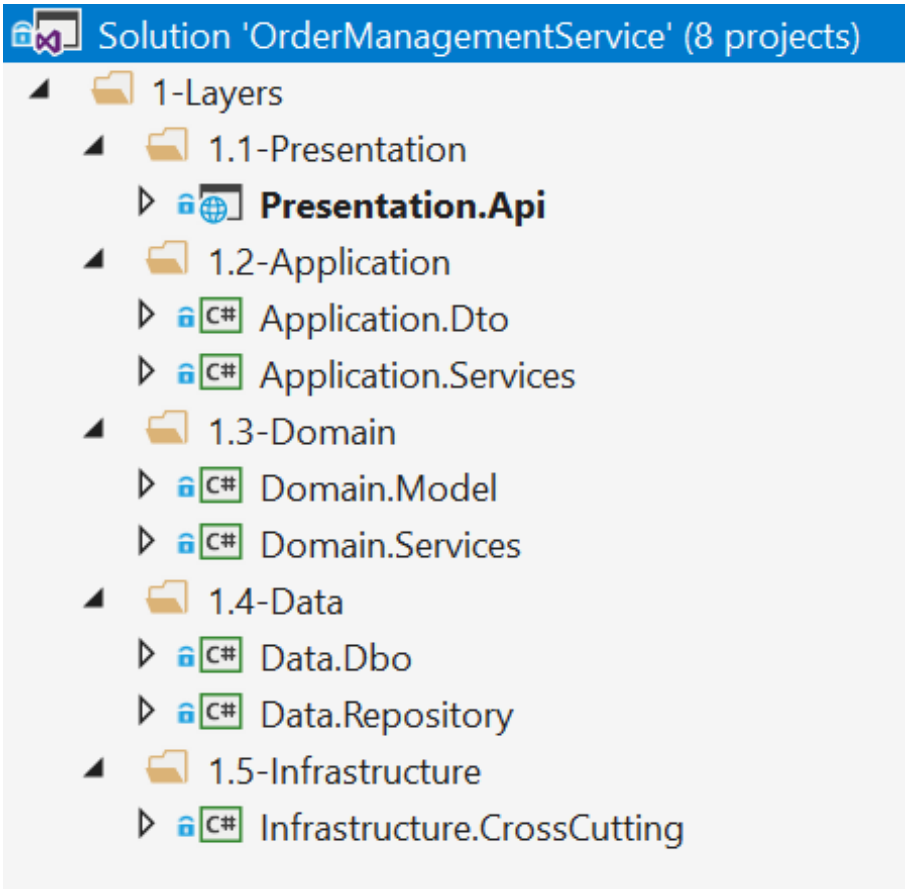


Figure 8.1: Code structure on Visual Studio

identifier is also persisted to keep a record of the price that was paid for each service. Also, this allows to have multiple prices for each service, if there is ever such a need. This is very useful for a use case where there are different prices for the same service, for example, with a Very Important Person (VIP) customer segment, where customers in this segment have access to lower prices than a regular customer or have different prices for each country.

On a technical level, when the user needs to update the price of a given service, the user makes that request to *SnapTaks BO Management*, in the service details. This application uses BO API's endpoint to make the operation which afterwards makes a call to Pricing Service. This component is the one that contains all logic regarding prices and for that reason is where the logic for this operation is contained. The service will calculate the price, taking into account the VAT and discount rates. If there is already a price in the database for the given service, that price will be disabled and the new price will be persisted as the price in use for that service.

The sequence diagram in figure 8.2 represents the update price flow between the several components of the platform.

The price composed by the **base price** and **final price**. The first one is the price that is defined by the SP. This value does not take into account neither VAT or discounts. These parcels are only accounted for in the final price, which is the price that will be paid by the customer for the given service. The formula below presents how the final price is calculated, using both VAT and the discount rate.

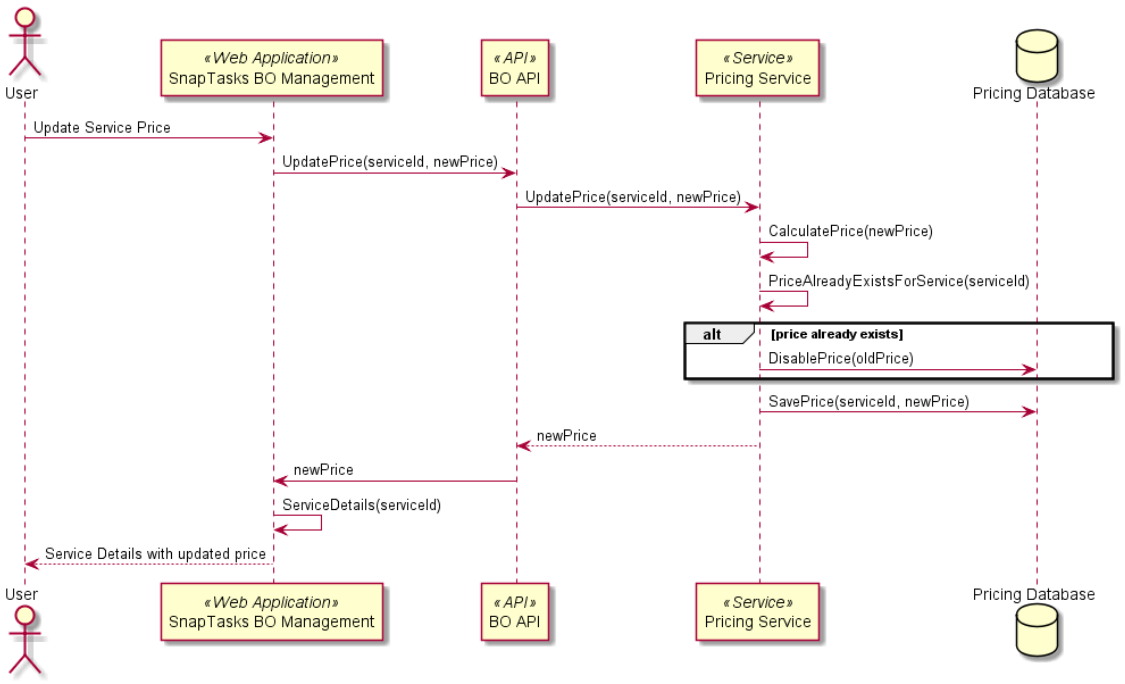


Figure 8.2: Sequence diagram for price update

$$finalPrice = basePrice * (1 + \frac{discountPercentage}{100}) * (1 + \frac{vatPercentage}{100})$$

The discount is applied first, since it is to be applied to the price before taxes, otherwise the platform would be applying discounts over the tax value. Since the tax value is always to be paid to the taxable country, this would result in a wrong price calculation.

In the case that the customer has a promocode, that promotion will be applied to the final order value, instead of being applied to each service price. The formula below presents how the total of an order is calculated.

$$total = ((\sum_{1}^{n} finalPrice) + logisticsFee) * (1 - \frac{promocodePercentage}{100})$$

To reach this value, the final prices of all services are summed. Then, the logistics fee (value for the courier to do the transportation) is added to that value. Finally, if the customer used a promocode, that rate is reduced from the overall price.

The platform will apply a commission model in order to retain profit. In a first phase, this commission has a flat rate of 10% over the total of each order. However, this percentage may vary over the time and be different from service provider to service provider.

$$commission = total * (\frac{commissionPercentage}{100})$$

8.3 Code Quality

Many of the non-functional requirements identified in section 5.3 can be measured by analyzing the code quality. This can provide an overview of requirements such as maintainability, resilience and performance.

Code quality is the definition of code that is of good code (with a high-quality standard) and bad code (with a low-quality standard) (Bellairs 2019). The main problem with this definition remains on the concept of good and bad. Since this is very subjective, there is no golden rule to check if a given piece of code has the quality or not.

The desired quality for one team, organization, or even application, can be totally different from another. It all clings on what is the main objective of that code. While in more enterprise companies, that develop applications focused on the final user, need to have a code base that is maintainable, reusable and easily changed, companies that develop critical systems need their code to be resilient, failure proof and fast. These are non-functional requirements for the applications developed by such companies.

To analyze the project's code quality, it was decided to use a code reviewing tool, in order to have an unbiased analysis. The chosen tool for this purpose was Codacy.

Codacy is a static code analysis tool which aims to automate the code review process. It offers features to notify about security issues, code coverage, code duplication and code complexity (Codacy 2019). After analyzing the code for potential problems, this tool assigns a grade to the project.

This grade ranges from **A** to **F**, being **A** the highest possible grade. This helps to have a better understanding of the quality of the project, taking into account issues of several categories:

- Code Style
- Compatibility
- Error Prone
- Performance
- Security
- Unused Code

The calculation of the final grade is based on the number of found issues per Thousand Lines Of Code (KLOC). Steve McConnell said in his book "Code Complete" that the industry average should be about 15 - 50 issues per KLOC, when the code has some level of structured programming behind it (McConnell 2004).

With this, the objective for this project was to have a code base with a grade greater or equal to **B**. This objective was accomplished in every service of the SnapTasks platform. The figure 8.3 presents the result of the code evaluation.

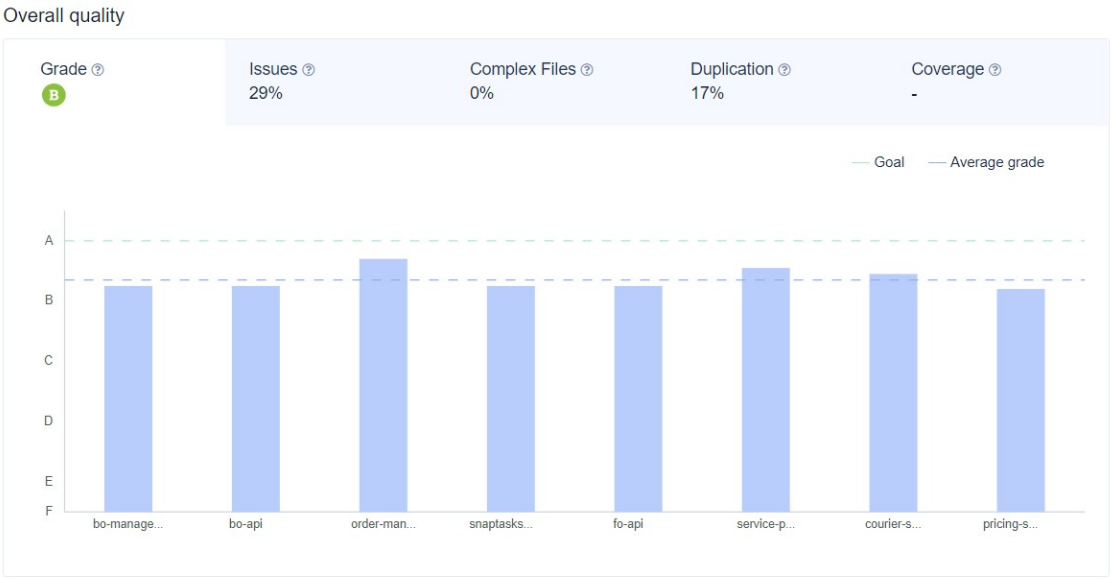


Figure 8.3: Codacy grade to SnapTasks projects

8.4 User Interfaces

The user interface is one of the most critical topics for the success of a commercial software project. The user needs to find the application easy to use, attractive and consistent on all pages. The User Interface (UI) is mostly important because it can turn visitors into buyers, as it facilitates the interactions between them and the system (Berezhnoi 2019).

The UIs are also very important because it combines both the functional requirements of the system (what it will do) and the non-functional requirements (how it will be done), specified in sections 5.2 and 5.3, respectively. The user interfaces can be considered the face of a given feature, since it is what the user will see, when interacting with the system.

In this section, the most important pages of the two end-user applications (SnapTasks Portal, section 8.4.1, and SnapTasks BO Management, section 8.4.2) will be presented, grouped by the application they belong.

8.4.1 SnapTasks Portal

The SnapTasks Portal is the front-office application which the final customer will use. It provides features such as place order, check order history and check available services. Since this is the application where the customers will access, it is the one where there is a bigger need for a good UI.

The most important pages of this application are the **Login Page**, the **Service Provider Listing Page**, the **Service Listing Page**, the **Service Display Page**, the **Checkout Page** and the **Order History Page**. These pages are described below.

Login Page

The login page is the page where the user will make the authentication, which will then allow him/her to access more actions, such as order history and place orders. The login may be done in two different ways: the first is to use an account that was previously created on the SnapTasks platform, using e-mail and password. the other is to use one external authentication service. Currently, there are two different authentication services: Facebook and Google. The figure 8.4 presents the UI of the login page.

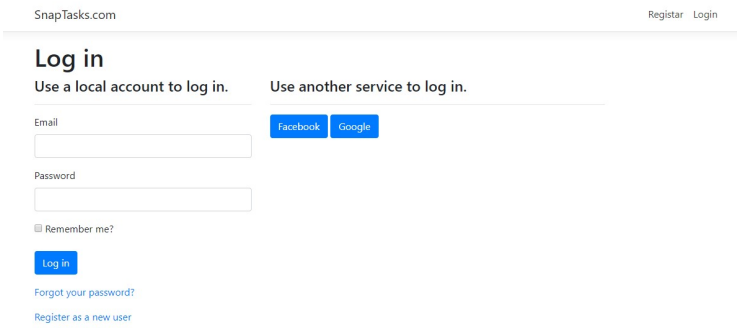


Figure 8.4: SnapTasks login page

Service Provider Listing Page (SPLP)

The Service Provider Listing Page (SPLP) is one of the most important pages in the SnapTasks Portal, since it serves as the home page of the application. The main purpose of this page is to present the service providers available on the platform. To that responsibility are added the ones of a home page, which are mainly to catch the attention of the end user and make it appealing to browse the website. The figure 8.5 presents the GUI of this page.

Figure 8.5: SnapTasks service provider listing page

Service Listing Page (SLP)

The SLP is accessed when the user selects a service provider in the SPLP. This page presents the list of services that are provided by the selected SP. Besides the name of the service, it also presents a description of the service and its price. The figure 8.6 presents the UI of this page.

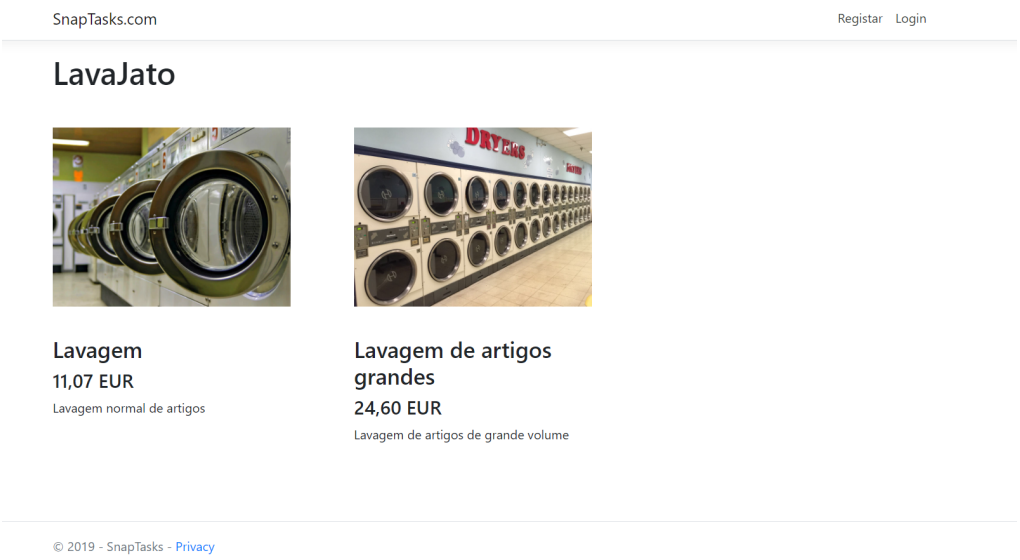


Figure 8.6: Service Listing Page of a Service Provider

Service Display Page (SDP)

When the user selects a service in the SLP, he/she is redirected to the SDP. This page has as its main purpose to display the details of the selected service. It provides information such as a more detailed description, the base price, the VAT and discounts (if any) and the final price. This page also provides the option for the user to place an order for that service. The UI of that page is presented in figure 8.7.

Figure 8.7: SnapTasks service display page

Checkout Page

The checkout page is the page where the customer places an order. It provides the information of the selected service, informs the total value of the order and asks for the information needed to fulfill the order (date of pickup, billing and pickup address and payment information).

For the payment processing, SnapTasks integrates with a third-party payment provider, Stripe, that processes the payments in a secure way. Stripe also provides tools for testing purposes, as several test credit cards.

This page is presented in figure 8.8.

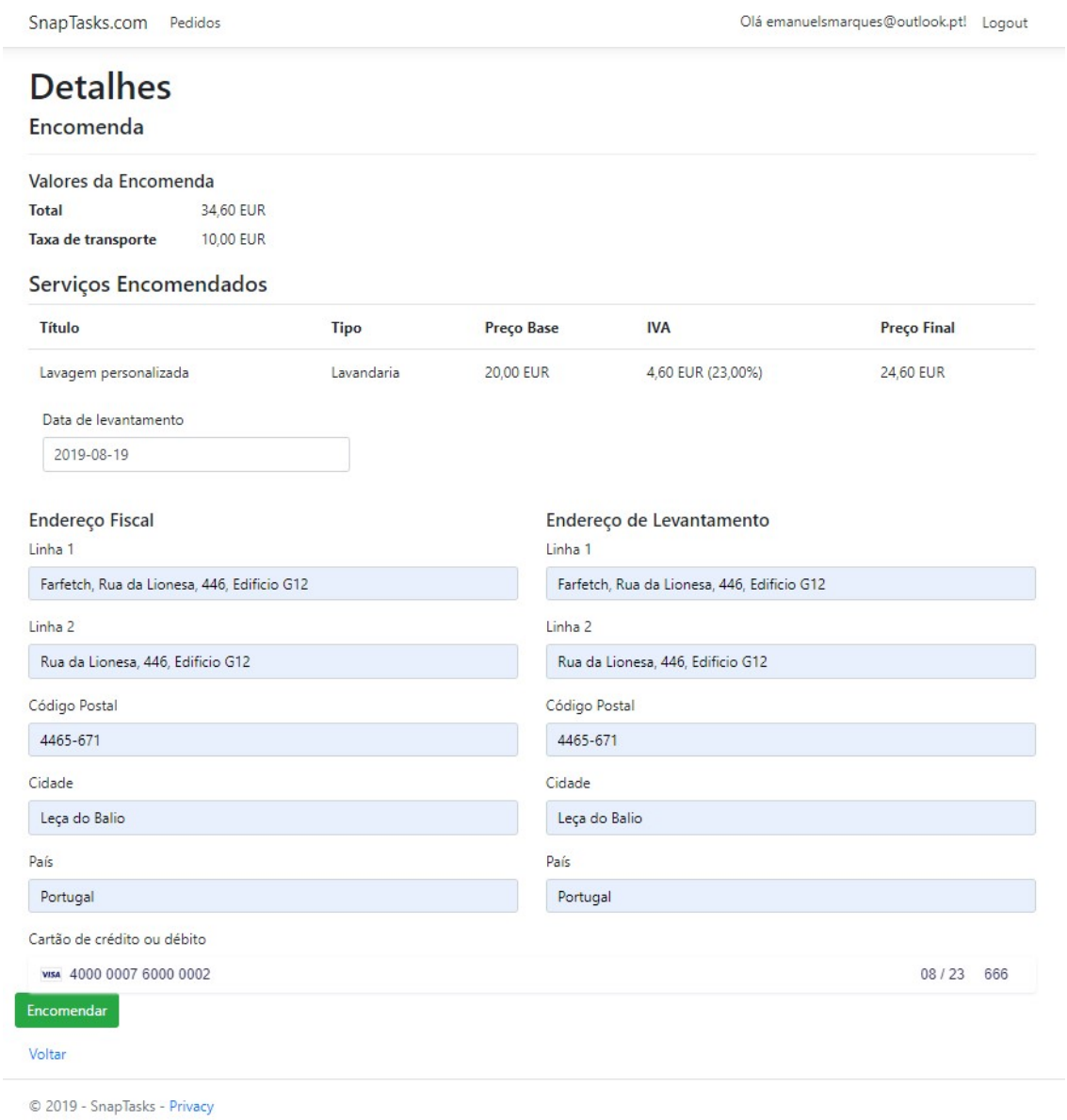


Figure 8.8: SnapTasks checkout page

Order Details Page

To access further information regarding an order, the user can open the details, in the order history page. This will redirect the user to the order details page. This page presents all the relevant information that concerns the selected order. The page presents the ordered services, the billing and pickup address, the values of the order (both final price and total. Check how these fields are calculated in section 7.2), the order status and the pickup date.

The UI of this page is presented in figure 8.9.

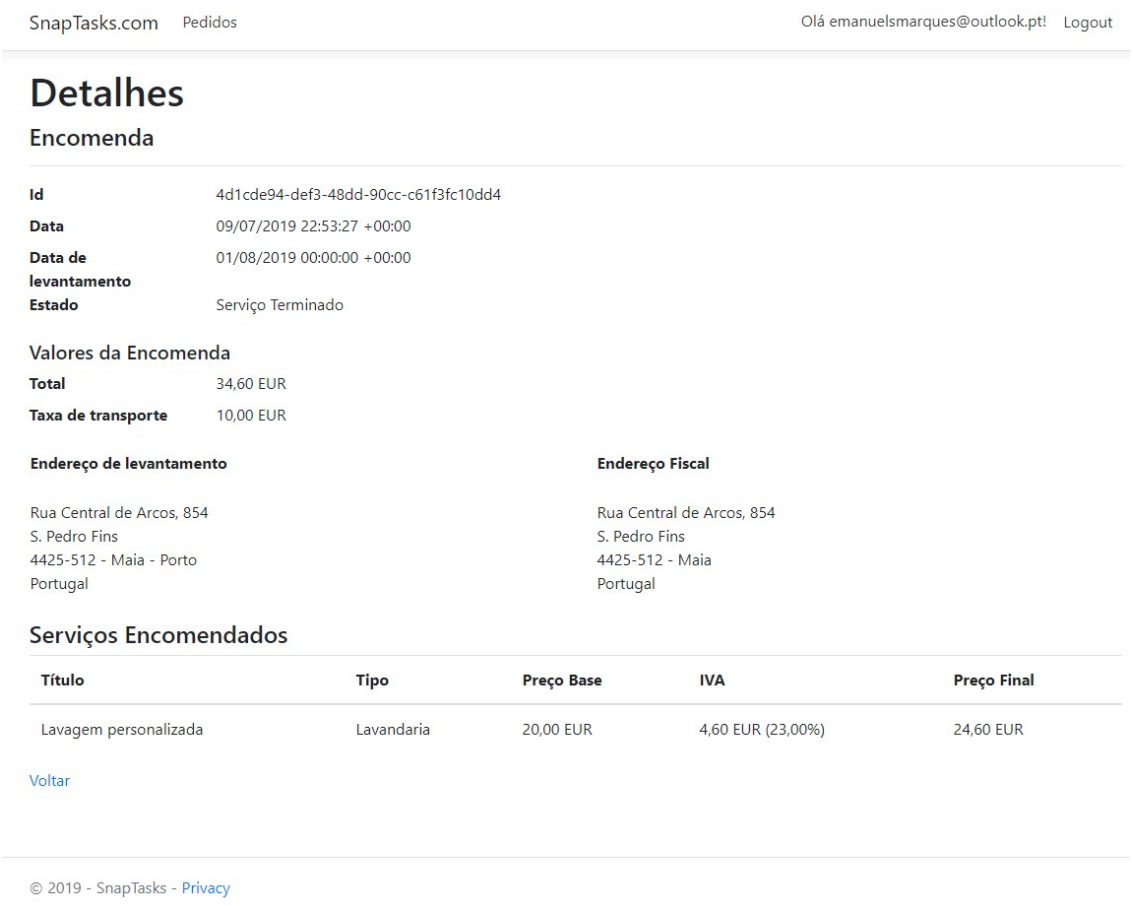


Figure 8.9: SnapTasks order details page

Order History Page

The order history presents the customer all the orders he/she had placed in the past. This page shows basic information about the order, such as the identifier, the creation date, the total value and the current status. With this page, it is possible to have an awareness of all the orders that were placed and their status. The figure 8.10 presents the UI of this page.

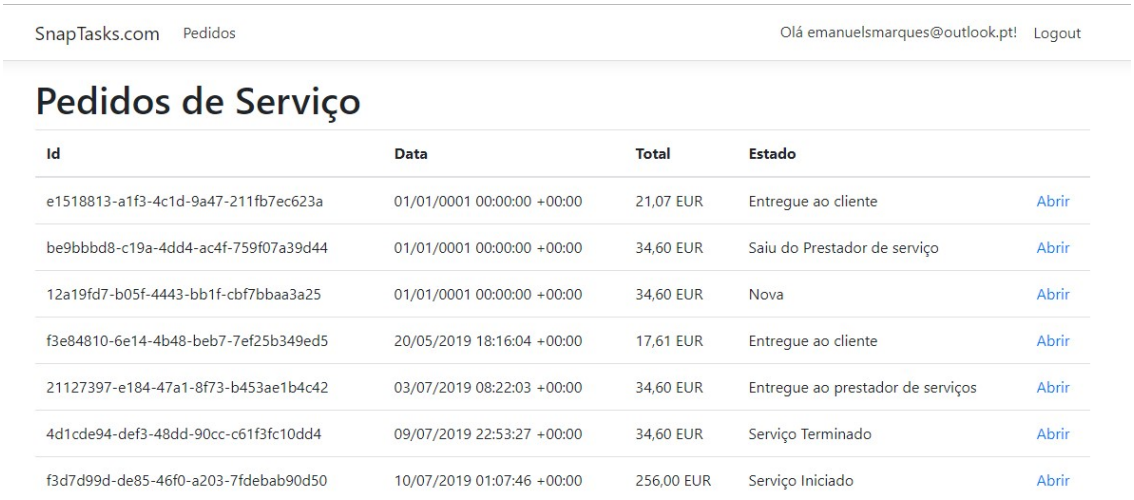


Figure 8.10: SnapTasks order history page

8.4.2 SnapTasks Back-Office Management

SnapTasks BO Management is an application whose purpose is to provide back-office and management tools. This application is designed to be used mostly by the operations teams who have the responsibility to manage the service providers, the couriers and the orders.

Since this application is not to be used by final customers, the UI that is used is not as critical as in the SnapTasks Portal. However, it is still very important that the use of features is intuitive and easy to learn, since the less time something takes to be done, the cheaper it will be.

The pages of this application are grouped into three main categories:

- **Service Provider Management Pages**, which are the pages where the SPs and their services will be listed, created, edited and disabled;
- **Order Management Pages**, where the placed orders can be listed, accessed and processed;
- **Courier Management Pages**, whose pages have the purpose of creation, edit and disable couriers.

These pages are further detailed and presented in the next sections.

Service Provider Management Pages

The Service Provider Management pages are the group of pages that provide tools to manage the service providers and their services. It is possible to create, disable, and edit both the service providers and its services. For the last case, it is also possible to update the price of a service. These pages can be accessed by the administrators of the platform, who can create and edit any SP and its services, and by the service provider's account, who can just edit their information and services.

The figure 8.11 presents the page that lists the current active service providers in the platform and the figure 8.12 shows the details of a service provider, and which services it provides.

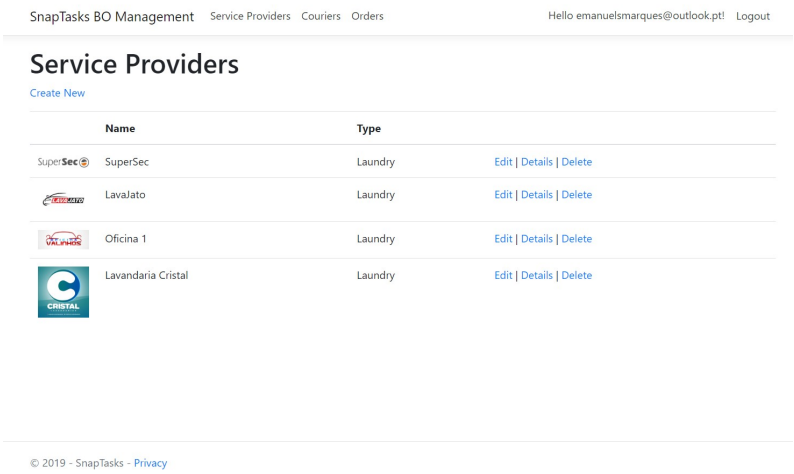


Figure 8.11: List of service providers page

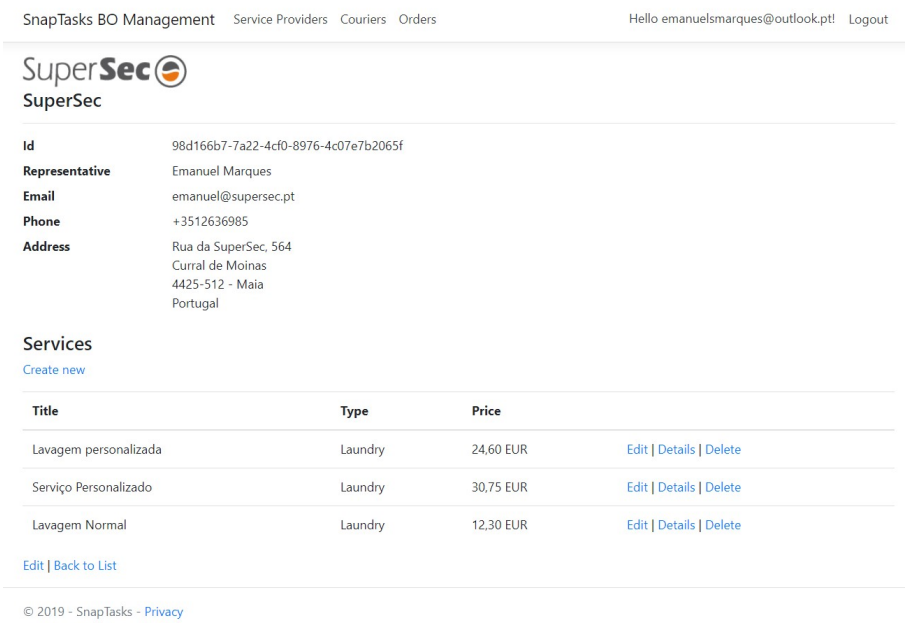


Figure 8.12: Service Provider details page

Courier Management Pages

Similarly to the previous pages, this group contains the features to create, edit and disable couriers. This is essentially an administration feature since only platform administrators have access to it.

The figure 8.13 presents the page that lists all the available couriers.

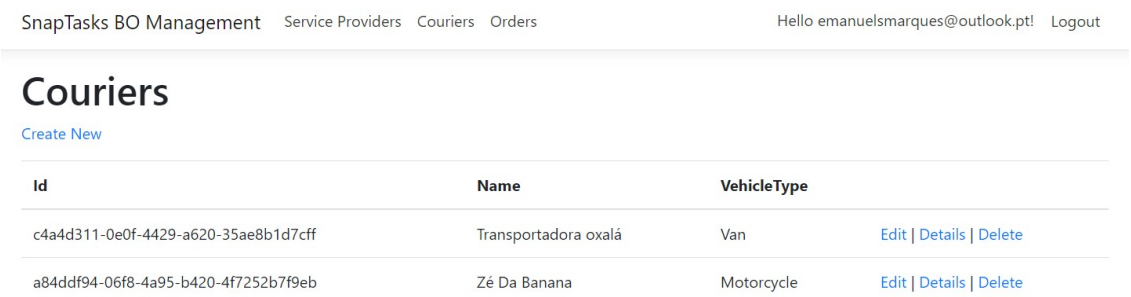


Figure 8.13: Courier list page

Order Management Pages

The order management pages are the pages where order information will be available and order processing features (like changing step, or courier assignment) will be available. Here is also possible to view the details of a given order and check the order history.

The figure 8.14 presents the page that lists the placed orders to be processed in the platform and the ones that were already processed. The figure 8.15 shows the details of a given order.

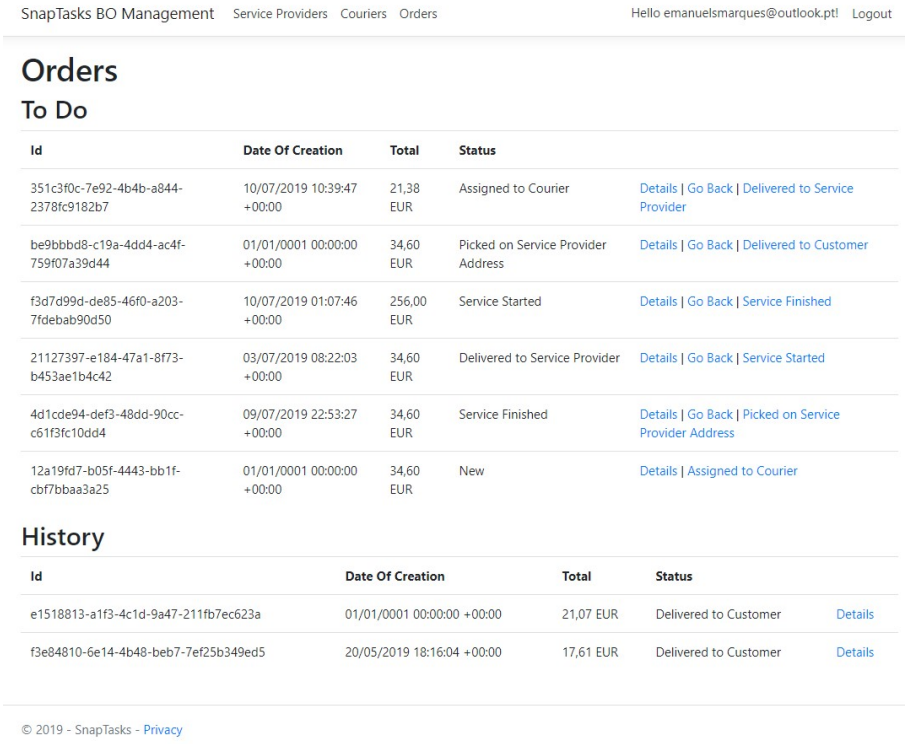


Figure 8.14: Order Processing Page

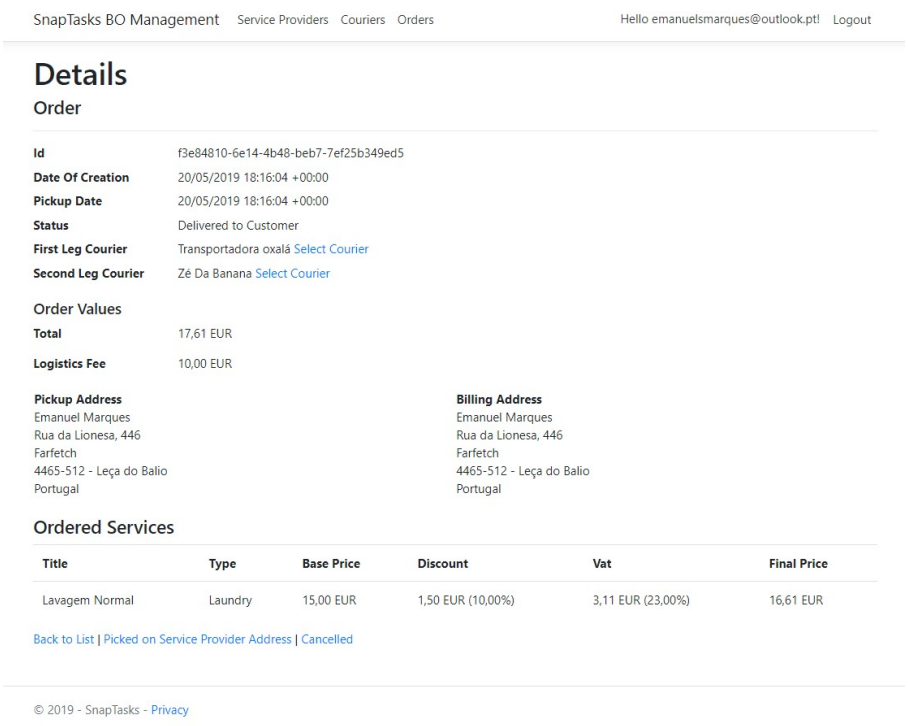


Figure 8.15: Order details page

Chapter 9

Evaluation of the Solution

After developing a new solution or project it is necessary to review if the final product has met the initial expectations. To accomplish this, the solution needs to be evaluated using a methodology that defines the system's functional and non-functional requirements. The designed model represents the ideal solution. The system is evaluated in several aspects of the dimensions, taking into account if the requirement was or not fulfilled, and with what quality.

In this chapter, it will be presented the methodology that was used and how it was used.

9.1 Evaluation Methodology

To evaluate the quality of the solution, it was chosen to apply a Quality Evaluation Framework (QEF). QEF is used to identify the relevant quality factors to a certain business and to help on the definition of the metrics to objectively measure the quality factors that were identified (Heidari and Loucopoulos 2014). A model (available on appendix B) was made to represent the ideal system. This model is divided into three different dimensions:

- Functional
- Security
- Efficiency

The functional dimension is divided into two factors: the functional requirements and the user interaction. The first lists all needed use cases to be accomplished for the project to be a success. These will be measured by either the user has access to the functionality or not. In the case of FF11 and FF12, an intermediate point is acceptable, where it is available for just some of the actors. The second factor refers to the usability of the application. This will be measured mostly by questioning the users for their experience.

The second dimension, security, refers to the concerns relatively to, as the name suggests, security of the platform. In this case, the only concerns that exist are associated to the security it, therefore it is its only factor. Similarly to the previous dimension, it will be measured by having or not fulfilled each of the requirements. There are intermediate positions for RN01 and RN02 when they are not put into practice every time.

Lastly, the third dimension that will be evaluated is the efficiency. This dimension was split into two factors. The first concerns the overall structure of the page and the user can access its contents in an intuitive and direct way. The second one relates to the navigation

resilience. In this factor, the requirements regarding the error handling and progress bars, showing progress for long tasks, will be evaluated. Both the factors of this dimension will be measured by having or not fulfilled the requirement. The only exception is the requirement EN02, where it is acceptable to have up to two long tasks not showing a progress bar.

For this project to be successful, the quality factor in the QEF should be above 90% and have no dimension with a quality below 80%.

9.2 Evaluation Results

The model presented in section 9.1 was applied to the final version of the prototype developed in the scope of this thesis. This version was finished in August 6th, 2019 and the evaluation was made in September 2nd, 2019.

The requirements of the dimensions were weighted taking into account the priority and importance for the correct functioning of the platform, being 10 the most important and 2 the least important.

The **functional dimension** is the biggest dimension since it contains all the use cases required by the platform. At this dimension there were three requirements that were postponed to a second version of the platform:

- FF02 - Apply as a Service Provider
- FF03 - Apply as a Courier
- FF09 - Evaluate Experience

These requirements lost priority in comparison to others because in a first delivery, the prototype will run with a test service provider and limited number of couriers. The same happens for the FF09, where in a first phase, the experience will be reported directly to the existing service provider.

This dimension was concluded with an estimated evaluation of 95.64%.

The second dimension, **security**, is one of the most important aspects in an e-commerce platform. It may attract or withdraw attention from possible customers and directly affects the reliability of the website. This dimension was completed in 100%.

Lastly, there is the **efficiency dimension**, which includes both the structure and navigation resilience. Since the platform still lacks beta testing in a real scenario (having real customers, couriers and service providers), there still room for improvement here. For this reason, the requirement *EN03 - Application runtime does not have errors, and unexpected errors should be well treated* was only partially accomplished. With this, the evaluation result of this dimension was 83.33%.

Looking at the solution as a whole, the system had an evaluation of 95%. This is a result that was better than expected. In the beginning of the project, the objective was to have a evaluation of at least 90%. The system attained 5% above than the required. It was also required to have no dimension with its evaluation below 80%. As it was previously explained, the lowest score was 83%.

In the last page of the appendix B, it is possible to check the evaluation matrix.

9.3 Usability Surveys

To understand how the platform meets the users needs, two surveys were run: one for the final customer perspective, and other for the service provider perspective. In the next sections, both surveys will be analyzed.

9.3.1 Customer Perspective Survey

The first survey has as its purpose understand how the final customer feels about the usability of the SnapTasks portal. It includes a tutorial to place an order and explains how to use some of the main features. In the appendix D it is possible to check the original inquiry, and in the appendix F, it is possible to consult the summary of the answers. This inquiry was made to seven people of different backgrounds.

How do you rate the ease of registration and log in?

This question has as its main purpose the evaluation of the log in and registration process.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very Hard* and 5 is *Very Easy*.

This question had a result of **5**, out of 5.

How do you rate the presentation of service providers?

This question has as its main purpose the evaluation of the home page, which is also the SPLP.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very Hard to Understand* and 5 is *Very Easy to Understand*.

This question had a result of **4.57**, out of 5.

How do you rate a service provider's presentation of services?

This question has as its main purpose the evaluation of the SLP of a given service provider and how those services are presented to the user.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very Hard to Understand* and 5 is *Very Easy to Understand*.

This question had a result of **4.43**, out of 5.

Regarding the detail given on the page of a service, was the information sufficient?

This question has as its main purpose the evaluation of the SDP of a given service and how the details of a service are presented to the user.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very Insufficient* and 5 is *More than Sufficient*.

This question had a result of **4**, out of 5.

Was the price of a given service clear to you?

This question has as its main purpose the evaluation of how clear the price of a given service was to the user.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Not clear at all* and 5 is *Very clear*.

This question had a result of **4.71**, out of 5.

How do you rate the ease of placing an order?

This question has as its main purpose the evaluation the order placing and checkout experience.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very hard* and 5 is *Very easy*.

This question had a result of **4.86**, out of 5.

Was the data presented about the order placed (after purchase) clear?

This question has as its main purpose the evaluation the order details page.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Not clear at all* and 5 is *Very clear*.

This question had a result of **4.86**, out of 5.

How do you rate the information given in the order history?

This question has as its main purpose the evaluation of the order history page and how the order history is presented to the user.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very Insufficient* and 5 is *More than Sufficient*.

This question had a result of **4.29**, out of 5.

Summary

In this survey, the results of all questions were above 4, which is an excellent result. The graph in the figure 9.1 presents a summary of the acquired results.

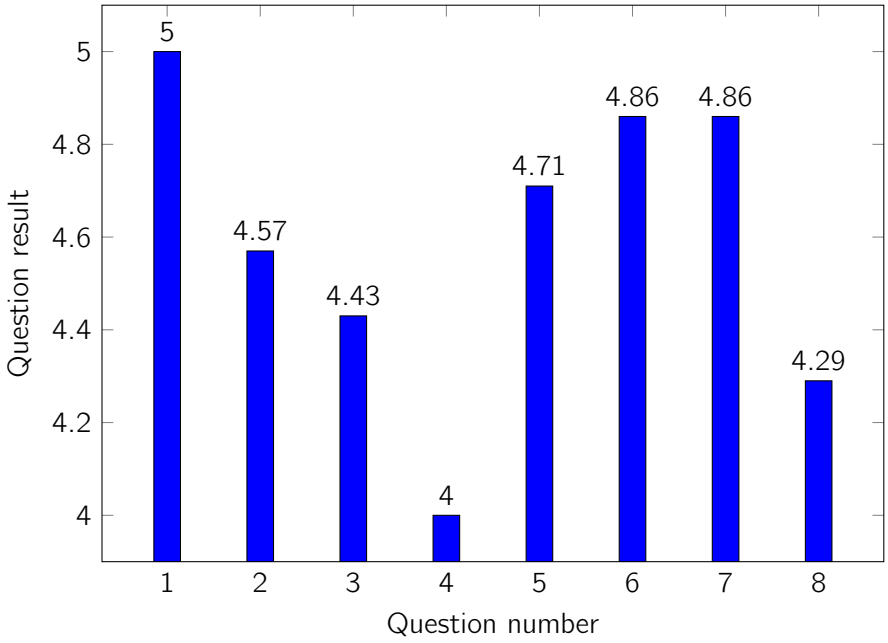


Figure 9.1: Summary of results of the Customer Perspective Survey

9.3.2 Service Provider Perspective Survey

The second survey has as its purpose understand how service provider feels about the usability of the SnapTasks BackOffice Management app. It includes a tutorial of how to use some of the main features. In the appendix E it is possible to check the original inquiry, and in the appendix G, it is possible to consult the summary of the answers. This inquiry was made to five people that work in service provision business.

How do you rate the ease of log in?

This question has as its main purpose the evaluation of the log in process.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very Hard* and 5 is *Very Easy*.

This question had a result of **4.4**, out of 5.

How do you rate the presentation of service providers?

This question has as its main purpose the evaluation of the presentation of the several service providers.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very hard to understand* and 5 is *Very easy to understand*.

This question had a result of **4**, out of 5.

How do you rate a service provider's presentation of services?

This question has as its main purpose the evaluation of the presentation of the services of a given service provider.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very hard to understand* and 5 is *Very easy to understand*.

This question had a result of **4**, out of 5.

Regarding the detail given on the page of a service provider, was the information sufficient?

This question has as its main purpose the evaluation of the presentation of the information regarding a given service provider. This information includes name, address, representatives and services.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very insufficient* and 5 is *More than sufficient*.

This question had a result of **3.6**, out of 5.

How do you rate the ease of updating the price of a service?

This question has as its main purpose the evaluation of the process of price update on a given service.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very insufficient* and 5 is *More than sufficient*.

This question had a result of **3.2**, out of 5.

Was the presentation of the service data clear?

This question has as its main purpose the evaluation of the presentation of details of a given service.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Not clear at all* and 5 is *Very clear*.

This question had a result of **4**, out of 5.

Was the price presentation clear?

This question has as its main purpose the evaluation of how clear the price presentation was to the user.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Not clear at all* and 5 is *Very clear*.

This question had a result of **4.2**, out of 5.

How do you rate the information given in the order history?

This question has as its main purpose the evaluation of how clear the information present in the order history was.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Not clear at all* and 5 is *Very clear*.

This question had a result of **4**, out of 5.

How do you rate the information given in the details of an order (click "Details")?

This question has as its main purpose the evaluation of how clear the information present in the order details was.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Not clear at all* and 5 is *Very clear*.

This question had a result of **4**, out of 5.

How do you rate the ease of changing an order status?

This question has as its main purpose the evaluation of how easy it was to change the order status.

This question asks the user to evaluate in a scale of 1 to 5, where 1 is *Very hard* and 5 is *Very easy*.

This question had a result of **3.6**, out of 5.

Summary

In this survey, the results of all questions were above 3, which is an good result, despite being lower than the customer result. The graph in the figure 9.1 presents a summary of the acquired results.

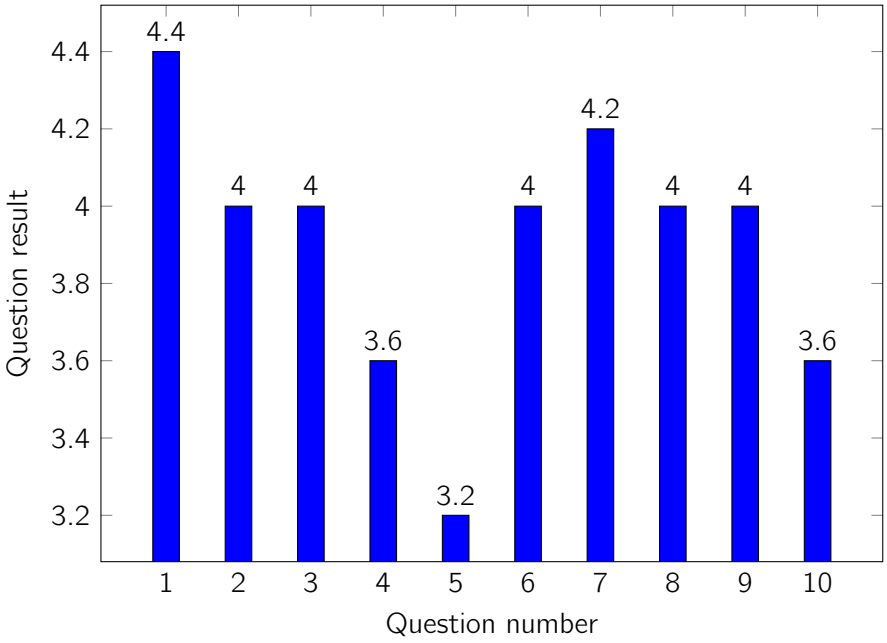


Figure 9.2: Summary of results of the Customer Perspective Survey

9.3.3 Results

The results of both inquiries help us understand the points that should be improved in the future. There is room for improvement specially in the SnapTasks BackOffice Management app. Nevertheless, the results of both surveys were good, since all results were above 3 and taking into account that we are analysing the MVP of the platform.

In the SnapTasks portal (Customer Perspective Survey), the results were even better, having all results above 4, proving that the customers are already used to these type of platforms and could easily start using them for service provision.

Chapter 10

Conclusion

In the final chapter of this dissertation, it is made a critical balance of the project as a whole. The objectives that were successfully fulfilled, the ones that were carried over a second version, the results and future work. The chapter is divided into two sections. The first is focused in approaching the achievements and results of the project. It will be discussed the implemented requirements, the attained objectives and the overall results. In the second section, it is presented the points that passed to a second version of the platform, an overview of possible improvements and enhancements for the growth of the platform. These recommendations aim to make the platform robust and maintainable enough to be a production level product.

10.1 Achievements and Results

The main objective of this thesis was to create a solution to the existent need of a platform that supports e-commerce applied to services where there is a logistical operation for it to be fulfilled. There were several requirements in terms of functionalities and technological challenges which aimed for the solution to be maintainable, dynamic and that supported multiple service providers if needed. The research made showed that there are similar problems that that already have an implemented solution (Uber Eats) and the analysis of its operational implementation helped to understand how our problem could be solved.

To solve the problem, a prototype was developed that responds to the needs that were described in the chapter 1.2. It provides the main functionalities needed to have the platform running, like allowing the management of service providers and their services, the management o couriers and the possibility to place and process an order in an easy and intuitive way. The developed solution uses a micro-services architecture allowing its growth in the future and facilitating the development by different people/teams. The APIs have a *Swagger* page where it is possible to check its endpoints, the arguments, the response body, the error codes and to make a direct request to the API.

The biggest challenge in the design of the platform was to design the best architecture for the platform. It was needed to decide which services made sense to use and which made not. The outcome was an architecture with two web applications, two APIs, four services, each one with its own database, all of the communicating seamlessly. Furthermore, the usage of an external payments provider also proved to be a challenge, since each third-party provider has its own rules that one needs to follow in order to integrate with it.

In terms of quality, the platform was evaluated in two different ways. The first one was applying the QEF model to the final version. This evaluation had a result of 95% which is 5% above the objective that was initially stated. These results proved that the requirements of the project were completed almost entirely. The second evaluation was to use *Codacy* to verify the code quality of the platform components. This evaluation also met the objectives, reaching the B grade from a scale F-A, being A the best grade, and F the worst. To get this grade it was needed a big attention to detail in terms of code quality.

The results of the surveys made to possible final customers and service providers also proved the viability of this platform. The first one proved that customers are used to how e-commerce platforms work and are willing to use an e-commerce platform for service provision. The second one, highlighted some improvements that are to be done in the back-office app, but, nevertheless, also had very positive results, proving that this kind of solution can be used for service provision management.

Lastly, there is still some limitations before using the platform in production. It should go through a beta testing phase where real service providers, with real services and real couriers were added to the platform. This will almost certainly raise issues that were not thought during the development of the project. One can say that no application/feature is really tested until it reaches the production environment, since it is very common to find things that were not discussed before.

10.2 Future Work

In the movie *The Social Network*, where the development of Facebook is depicted, it is said that *"a software product is never finished. The way fashion is never finished"*(adapted). Despite being a movie, the sentence is very much real. No software product is ever finished since there is always something to improve. And even if there is not, as the world changes, our code also needs to change.

This platform is no exception to this rule. There is still room for improvements in the existent features. There are still three features that were not developed in the first version. The user interfaces should also be a target of improvements both visually, where they could be redesigned to provide a unique interface that characterizes the brand, and in terms of usability, where User Experience (UX) studies should be done to understand what is the best layout for the users of the platform

The platform could also make use of a messaging service, specially for changes that happen to the order. The notification of customers about the progress of their orders, of service providers when there are new orders and couriers about orders ready for pickup are examples of use cases where these technologies could fit very easily.

The platform could also use a system of ticketing for issues that may happen, therefore creating a customer service team that aimed to help the customers for problems they may have.

Lastly, the platform could also integrate with billing software to facilitate the service providers' billing and tax calculations.

Bibliography

AltexSoft (2018). *The Good and the Bad of Node.js Web App Development*. url: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/ (visited on 02/16/2019).

– (2019). *The Good and the Bad of .NET Framework Programming*. url: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming/ (visited on 02/14/2019).

Anderson, James C., James A. Narus, and Wouter van Rossum (2006). "Customer Value Propositions in Business Markets". In: *Harvard Business Review*. url: https://fenix.tecnico.ulisboa.pt/downloadFile/1407993358860263/HBR-value-propositions[1].pdf.

Aswani (2014). *Advantages of Spring Framework With Limitations*. url: http://www.aksindiblog.com/spring-framework-advantages-disadvantages.html (visited on 02/16/2019).

Bellairs, Richard (2019). *What Is Code Quality? And How to Improve It*. url: https://www.perforce.com/blog/sca/what-code-quality-and-how-improve-it (visited on 08/07/2019).

Berezhnoi, Roman (2019). *What Is UI design and why is it important?* url: https://f5-studio.com/articles/what-is-user-interface-design-and-why-is-it-important/ (visited on 08/19/2019).

Capan, Tomislav (2019). *Why The Hell Would I Use Node.js? A Case-by-Case Tutorial*. url: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js (visited on 02/17/2019).

Carson, Biz (2016). *Why Uber started Uber Eats*. url: https://www.businessinsider.com/why-uber-launched-uber-eats-2016-3 (visited on 01/31/2019).

Codacy (2019). *Automated code reviews & code analytics | Codacy*. url: https://www.codacy.com/ (visited on 08/10/2019).

Costa, Nélson (2018). *Glovo vs. Uber Eats: qual a melhor opção?* url: https://www.e-konomista.pt/artigo/glovo-vs-uber-eats/ (visited on 02/24/2019).

DataFlair (2018). *Advantages of Spring Framework With Limitations*. url: https://data-flair.training/blogs/advantages-of-spring/ (visited on 02/16/2019).

Fowler, Martin (2005). *Event Sourcing*. url: https://martinfowler.com/eaaDev/EventSourcing.html (visited on 02/14/2019).

– (2011). *CQRS*. url: https://martinfowler.com/bliki/CQRS.html (visited on 02/07/2019).

Gibson, Sam (2015). *Monoliths are Bad Design... and You Know It*. url: https://www.thoughtworks.com/insights/blog/monoliths-are-bad-design-and-you-know-it (visited on 02/16/2019).

Groth, John C. and Richard Dye (1999). "Service quality: Perceived value, expectations, shortfalls, and bonuses". In: *Journal of Service Theory and Practice* 9. url: https://www.researchgate.net/publication/240257672_Service_quality_Perceived_value_expectations_shortfalls_and_bonuses.

Hartmans, Avery (2018). *The early history of Jeff Bezos and Amazon*. url: https://www.businessinsider.com/jeff-bezos-amazon-history-facts-2017-4 (visited on 02/03/2019).

Heidari, F. and P. Loucopoulos (2014). "Quality evaluation framework (QEF) : modeling and evaluating quality of business processes". English. In: *International Journal of Accounting Information Systems* 15.3, pp. 193–223. issn: 1467-0895. doi: 10.1016/j.accinf.2013.09.002.

Hubbell, David (2019). *Top 4 Pros and Cons of Microservices Architecture*. url: https://www.spkaa.com/blog/top-4-pros-cons-microservices-architecture/ (visited on 02/14/2019).

IBM (2019). *Service-oriented architecture (SOA)*. url: https://www.ibm.com/support/knowledgecenter/en/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html (visited on 02/08/2019).

Ionita, Mugurel T., Dieter K. Hammer, and Henk Obbink (2005). "Scenario-Based Software Architecture Evaluation Methods: An Overview". English. In: *11th Asia-Pacific Software Engineering Conference* 3. issn: 1530-1362. doi: 10.1109/APSEC.2004.38. url: https://www.win.tue.nl/oas/architecting/aimes/papers/Scenario-Based%5C%20SWA%5C%20Evaluation%5C%20Methods.pdf.

Koen, Peter A., Greg M. Ajamian, et al. (2002). "Fuzzy Front End: Effective Methods, Tools, and Techniques". In: *The PDMA ToolBook for New Product Development*. url: http://www.stevens-tech.edu/cce/NEW/PDFs/FuzzyFrontEnd_Old.pdf.

Koen, Peter A., Greg Ajamian, et al. (2001). "Providing Clarity and a Common Language to the "Fuzzy Front End"". In: *Research - Technology Management*. url: http://www.stevens-tech.edu/cce/NEW/PDFs/Clarity_FEE.pdf.

Koen, Peter A., Heidi M. J. Bertels, and Elko Kleinschmidt (2014). "Managing the Front End of Innovation—Part I". In: *Research-Technology Management*. url: http://frontendinnovation.com/media/default/pdfs/fei-article-1.pdf.

Lansat, Myelle (2018). *Here's how much millennials spend on Uber and Lyft in major US cities every month*. url: https://www.businessinsider.com/millennials-spending-uber-lyft-major-cities-2018-7 (visited on 01/31/2019).

Leiner, Barry M. et al. (1997). "The Past and future History of the Internet". In: *COMMUNICATIONS OF THE ACM* 40.2. url: http://bnrg.eecs.berkeley.edu/~randy/Courses/CS294.S13/1.1x.pdf.

Lumetta, Jake (2018). *Monolith vs Microservices: Which is the Best Option for You?* url: https://nordicapis.com/should-you-start-with-a-monolith-or-microservices/ (visited on 02/16/2019).

Manzoor, Amir (2010). *E-Commerce: An Introduction*. Lambert - Academic Publishing. isbn: 978-3843370301.

Masashi Narumoto, et al (2017). *Command and Query Responsibility Segregation (CQRS) pattern*. url: https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs (visited on 02/07/2019).

McConnell, Steve (2004). *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press. isbn: 978-0735619678. url: http://aroma.vn/web/wp-content/uploads/2016/11/code-complete-2nd-edition-v413hav.pdf.

Michelson, Brenda M. (2011). "Event-Driven Architecture Overview". In: *Elemental Links Research*. url: http://elementallinks.com/el-reports/EventDrivenArchitectureOverview_ElementalLinks_Feb2011.pdf.

Microsoft (2019). *What is .NET?* url: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet (visited on 02/07/2019).

Miles, Lawrence D. (2015). *Techniques of Value Analysis and Engineering*. Lawrence D. Miles Value Foundation. isbn: 9780070419261.

Patel, Priyesh (2015). *What exactly is Node.js?* url: https://www.thoughtworks.com/insights/blog/monoliths-are-bad-design-and-you-know-it (visited on 02/17/2019).

Pivotal (2019). *Spring Framework*. url: http://spring.io/projects/spring-framework (visited on 02/08/2019).

Queiroz, Gabriel (2017). *Vamos falar sobre Event Sourcing*. url: https://medium.com/@gabrielqueiroz/vamos-falar-sobre-event-sourcing-276ae66106f7 (visited on 02/14/2019).

Richards, Mark (2015). *Software Architecture Patterns*. O'Reilly Media, Inc. isbn: 978-1491971437. url: https://www.oreilly.com/programming/free/files/software-architecture-patterns.pdf.

Saaty, Thomas L. (1990). "How to make a decision: THe Analytic Hierarchy Process". In: *European Journal of Operational Research*. url: http://scholar.google.pt/scholar_url?url=https://www.researchgate.net/profile/Mohamed_Mourad_Lafifi/post/Problem_with_sub-criteria_code_for_multiple_decision_makers/attachment/59d644d679197b80779a0076/AS:450351808684035%5C%401484383646513/download/How%5C%2Bto%5C%2BMake%5C%2Ba%5C%2BDecision%5C%2BThe%5C%2BA%5C%2BH%5C%2BP.pdf&hl=pt-PT&sa=X&scisig=AAGBfm1SzctvCBxumEt6eNNdop54EMWjdQ&nossl=1&oi=scholarr.

Schneider, Todd W. (2018). *Taxi, Uber, and Lyft Usage in New York City*. url: http://toddwschneider.com/posts/taxi-uber-lyft-usage-new-york-city/ (visited on 01/31/2019).

statista.com (2018). *Percentage of all global web pages served to mobile phones from 2009 to 2018*. url: https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/ (visited on 02/04/2019).

Swagger (2019). *What Is Swagger?* url: https://swagger.io/docs/specification/2-0/what-is-swagger/ (visited on 08/12/2019).

Thai, Ngueyen Nam (2018). *What is a Spring Bean*. url: https://www.baeldung.com/spring-bean (visited on 02/16/2019).

Tyson, Matthew (2018). *What is the JVM? Introducing the Java virtual machine*. url: https://www.javaworld.com/article/3272244/core-java/what-is-the-jvm-introducing-the-java-virtual-machine.html (visited on 02/08/2019).

Ulaga, Wolfgang and Andreas Eggert (1991). "Why We Buy What We Buy: A Theory of Consumption Values". In: *Journal of Business Research*. url: https://www.researchgate.net/publication/4965989_Why_We_Buy_What_We_Buy_A_Theory_of_Consumption_Values.

– (2004). "Relationship value and relationship quality: Broadening the nomological network of business-to-business relationships". In: *European Journal of Marketing*. url: https://s3.amazonaws.com/academia.edu.documents/40705690/Ulaga___Eggert_RS_Value___Quality_EJM_2006.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1549840502&Signature=L%5C%2FJNLRhrebj4PmP3SqI80TKBeRA%5C%3D&response-content-disposition=inline%5C%3B%5C%20filename%5C%3DRelationship_value_and_relationship_qual.pdf.

Woodall, Tony (2003). "Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis". In: *Academy of Marketing Science Review*. url: https://cdn.ymaws.com/www.ams-web.org/resource/resmgr/original_amsr/woodall12-2003.pdf.

Zhang, Yingchao et al. (2014). "Mapping Services to Activities in Service Oriented Architecture (SOA) Design: A Simulation-driven Optimizing Method Based on DODAF2.0". In: *8th International Symposium on Service Oriented System Engineering* 40. url: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6830899.

Zhu, ChangSheng et al. (2013). "Service Oriented Architecture Design of Energy Consumption Information System about Petroleum Enterprise". In: *International Conference on Computational and Information Sciences*. url: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6643228.

Appendix A

Thesis Formalization

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

isep Instituto Superior de Engenharia do Porto

DEI-ISEP
Mestrado Engenharia Informática

Ano Letivo 2018-2019

Tese/Projeto/Estágio
Proposta

Título: Logistics Platform for Service Provision

Problema

INSTITUTO POLITÉCNICO DO PORTO

Os consumidores pretendem cada vez mais comprar produtos e serviços através das plataformas digitais (websites, apps, etc.). No entanto, quando necessitam de comprar alguns serviços, ainda não é viável, principalmente devido à inexistência de uma solução tecnologica para estes casos.. Este era o cenário típico da compra de refeições, que está a ser colmatado atualmente pelo serviço UBER Eats. Da mesma forma, quando um consumidor necessita, por exemplo, de lavar a sua roupa, principalmente no caso de peças grandes, como carpetes, edredões, etc. não existe outra solução senão dirigir-se pessoalmente a uma lavandaria para deixar a peça, e passar novamente para recolher. Devido à dificuldade de muitos consumidores se deslocarem às lavandarias dentro do horário útil, algumas começaram a fazer recolhas e entregas ao domicílio, mas acarreta uma grande logística para gerir telefones, rotas de recolhas e entregas, etc. o que faz com que o serviço que o consumidor recorre, tenha um acréscimo no seu valor e nem sempre é prestado com a qualidade desejada.

Objetivos

Criar uma plataforma que interligue vários prestadores de serviços (não apenas serviço de lavandaria) que quiserem integrar esta plataforma, com uma rede de motoristas que irão fazer as recolhas e entregas, aos seus clientes. De forma a que um cliente possa fazer através de um website ou de uma aplicação móvel, uma marcação de recolha de roupa, poder escolher uma lavandaria da sua preferência se desejar, poder acompanhar o serviço, efetuar o pagamento, e avaliar toda a experiência. O motorista deve poder escolher fazer serviços para um ou mais prestadores de serviços, receber novos pedidos e poder fazer alterações manuais no planeamento do serviço. Deve ainda poder imprimir um talão de recolha e entrega ao cliente e um talão para anexar a cada encomenda. Cada prestador de serviços deve ter uma visão geral de todo o serviço que lhe é atribuído, ter a possibilidade de alterar o serviço distribuído automaticamente pelo sistema aos motoristas, e avaliar o serviço dos motoristas, e aceder às estatísticas dos serviços

realizados. Deverá haver ainda uma área de administração com vários níveis de acesso (admin geral, admin de lavandarias, admin de parceiros, admin de clientes).

Módulos curriculares (obrigatório na versão definitiva)

Ordenar por ordem de preferência:
- (obrigatório) Métodos de análise de problemas, pesquisa e escrita técnico-científica
- (e.g. 1º) Métodos de preparação e realização de experiências
- (e.g. 2º) Análise de resultados
- ( ) Demonstração de teoremas
- ( ) Especificação formal de algoritmos e verificação
- ( ) Equações diferenciais
- ( )
- ( )

Orientador (do DEI, doutorado/especialista, se já definido)

Nome: António Rocha

Email: ajo@isep.ipp.pt

Coorientador (se existir)

Nome:

Email:

Estudante (se já atribuído)

Nome: Emanuel Fernando Paiva da Silva Marques

Número: 1130553

E-mail: 1130553@isep.ipp.pt

INSTITUTO POLITÉCNICO DO PORTO

Appendix B

QEF Solution Evaluation

Dimension	Functionality
Factor	Functional

Requirement	Metric Evaluation	Wfk - Fullfilment (%)		
		0	50	100
FF01 - Register as Customer	An unregistered user canregister as a customer	No access to functionality	-	Full access to the functionality
FF02 - Apply as a Service Provider	An unregistered user can apply as a service provider	No access to functionality	-	Full access to the functionality
FF03 - Apply as a Courier	An unregistered user can apply as a courier	No access to functionality	-	Full access to the functionality
FF04 - Log In	An unregisterd user can log in	No access to functionality	-	All of the actors can acess
FF05 - Recover Password	An unregistered user can recover a lost password	No access to functionality	-	Full access to the functionality
FF06 - Place an order for a service	A customer can place an order for a wanted service	No access to functionality	-	Full access to the functionality
FF07 - Consult all available service providers	A customer or an unregistered user can consult all available service providers	No access to functionality	-	Full access to the functionality
FF08 - Consult order history	A customer can consult his order history on the platform.	No access to functionality	-	Full access to the functionality
FF09 - Evaluate Experience	A customer can evaluate his experience on the platform	No access to functionality	-	Full access to the functionality
FF10 - Manage available services	A service provider can manage its available services	No access to functionality	-	Access to the functionality
FF11 - Manage orders	A service provider or courier can manage the orders that are assigned to them	No access to functionality	Only part of the actors can use this	All of the actors can access this functionality
FF12 - Change Area of Actuation	A service provider or courier can change its area of actuation	No access to functionality	Only part of the actors can use this	All of the actors can access this functionality
FF13 - Change service profile	A courier can change its service profile (scooter, car,	No access to functionality	-	Access to the functionality
FF14 - Assign orders for pickup	A courier can assign himself to a given order for pickup/delivery	No access to functionality	-	Access to the functionality
FF15 - Manage Service Providers	An admin can manage (create, update, delete) service providers	No access to functionality	-	Access to the functionality
FF16 - Manage Couriers	An admin can manage (create, update, delete) couriers	No access to functionality	-	Access to the functionality

Dimension	Functionality
Factor	User Interaction

		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
FUI01 - Applications are intuitive	Interaction questionnaire to two type of users. The first user is a experienced user that already read the manual, the other is a unexperienced user that never read the manual. The questionnaire must be made to the three types of users (Final Customer, Service Provider and Courier) in a total of 6	0-1 positive questionnaires	2-3 positive questionnaires	4-6 positive questionnaires
FUI02 - All the applications present a same design experience	The design experience must be similar to the three solutions environments. Backgrounds, colors, design of buttons, font types, logos, icons and animations must be similar.	Low similarity	Only half of the categories present a similar design	High similarity
FUI03 - All the application present a same navigation experience	The navigation between screens present the same experience. Buttons to menu, advance, confirm and cancel	No	-	Yes
FUI04 - Applications have quick access to main functions	Executing the order placement functionality musn't take more than 4 steps (choose, pay, enter information...)	>4 steps	-	<= 4 steps

Dimension	Security
Factor	Security

Requirement	Metric Evaluation	Wfk - Fullfilment (%)		
		0	50	100
RN01 - Payments are made by a trusted payment provider	All payments are done by a trusted third party payment provider	No	Most are, but some stiil are not	Yes
RN02 - All Comunications are made with HTTPS	All comunications between client and server applications are made using HTTPS	No	Most are, but some stiil are not	Yes
RN03 - Each user type has its own permissions	Each role has its	No	-	Yes
RN04 - Login and Access security	Login exceptions (unknown username/password) must present user a message indicating the cause of unperformed login. Only registered users can make login.	No	-	Yes

Dimension	Efficiency
Factor	Structure, Navigation Resilience

		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
EN01 - Product has a good structure and allows users to access contents in a intuitive way to the main functions	Each application must have a main menu and only one sub-menu to each menu element.	No	-	Yes
EN02 - Application user interface is quick and fast responsible, with progress information whenever it's relevant	Functionalities that can take more than 1 sec must present a progress message to inform user. Functionalities that can take more than 10 secs must present also na alter message to inform user about the time complexity of the task.	No	There are up to two long tasks not showing a progress bar	Yes
EN03 - Application runtime does not have errors, and unexpectable errors should be well treated	Unexpected errors must be presented to the user with a clear message about the error.	No	There are some errors but the application is overall stable	Yes

q	D	Qi	Dimension	Qj	Wij (Factor Weight *j* in Dim *i*) [0,1]	Factor	rwjk (requirement weight *k* in Factor *j*) {2, 4, 6, 8, 10}	Requirement	wfk % requirement fulfillment *k*) [0,100]
95%	0,17	95,63636364	Functionality	94,54545455	0,80	Functional (Refering Use Cases)	10	FF01 - Register as Customer	100
							2	FF02 - Apply as a Service Provider	0
							2	FF03 - Apply as a Courier	0
							8	FF04 - Log In	100
							2	FF05 - Recover Password	100
							10	FF06 - Place an order for a service	100
							10	FF07 - Consult all available service providers	100
							10	FF08 - Consult order history	100
							2	FF09 - Evaluate Experience	0
							10	FF10 - Manage available services	100
							10	FF11 - Manage orders	100
							4	FF12 - Change Area of Actuation	100
							2	FF13 - Change service profile	100
							8	FF14 - Assign orders for pickup	100
							10	FF15 - Manage Service Providers	100
							10	FF16 - Manage Couriers	100
				100	0,20	User Interaction	10	FUI01 - Applications are intuitive	100
							10	FUI02 - All the applications present a same design experience	100
							8	FUI03 - All the application present a same navigation experience	100
							6	FUI04 - Applications have quick access to main functions	100
		100	Security	100	1,00	Security	10	RN01 - Payments are made by a trusted payment provider	100
							8	RN02 - All Comunications are made with HTTPS	100
							4	RN03 - Each user type has its own permissions	100
							8	RN04 - Login and Access security	100
		83,33333333	Efficiency	100	0,33	Structure	8	EN01 - Product has a good structure and allows users to access contents in a intuitive way to the main functions	100
				75	0,67	Navigation Resilience	10	EN02 - Application user interface is quick and fast responsible, with progress information whenever it's relevant	100
							10	EN03 - Application runtime does not have errors, and unexpected errors should be well treated	50

Appendix C

Business Model Canvas

Appendix D

Customer Perspective Inquiry

SnapTasks - Inquérito de usabilidade :: Cliente

No âmbito da minha tese de mestrado em Engenharia de Software, venho aqui pedir o seguimento deste tutorial e consequente resposta às perguntas colocadas.

O SnapTasks é uma plataforma que faz a ponte entre prestadores de serviços e clientes. Tipicamente, os serviços aqui prestados obrigam uma operação logística para que o serviço seja feito. Um exemplo clássico é o serviço de lavandaria, em que o cliente pede um serviço, um estafeta vai recolher a roupa a casa e, assim que o serviço estiver feito, devolve o material ao cliente.

*Obrigatório

Registo e Log In

Vá a https://snaptasks.azurewebsites.net/, crie uma conta e faça log in. Pode utilizar a integração com Facebook ou Google se for mais conveniente.

1. **Como avalia a facilidade de registo e log in?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil	◯	◯	◯	◯	◯	Muito fácil

Passe para a pergunta 2.

Prestadores de Serviços

Explore os prestadores de serviço existentes e os seus serviços.
Na página principal (https://snaptasks.azurewebsites.net) aparecerão os prestadores de serviço disponíveis. Clicando num, será possível ver os serviços prestados por esse prestador de serviços. Por último, ao clicar num serviço, é possível ver detalhes desse mesmo serviço.

2. **Como avalia a apresentação dos prestadores de serviço?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil de entender	◯	◯	◯	◯	◯	Muito fácil de entender

3. **Como avalia a apresentação dos serviços de um prestador de serviços?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil de entender	◯	◯	◯	◯	◯	Muito fácil de entender

4. **Relativamente ao detalhe dado na pagina de um serviço, a informação foi suficiente?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito insuficiente	◯	◯	◯	◯	◯	Mais que suficiente

5. **Para si, o preço de um dado serviço era claro?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Nada claro	◯	◯	◯	◯	◯	Muito claro

Checkout

Escolha um serviço e faça a sua encomenda.

Utilize os dados abaixo para o pagamento.

Numero do cartão: 4000006200000007
Validade: 05/23
CVC: 123

6. **Como avalia a facilidade de fazer a encomenda?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil	◯	◯	◯	◯	◯	Muito fácil

7. **A apresentação dos dados da encomenda realizada (após a compra) foi clara?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Nada clara	◯	◯	◯	◯	◯	Muito clara

Histórico de pedidos

Vá ao seu histórico de pedidos (https://snaptasks.azurewebsites.net/Order) e explore os menus.
Aqui é possível ver encomendas realizadas, ver o seu estado e detalhes.

8. **Como avalia a informação dada?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Muito insuficiente	◯	◯	◯	◯	◯	Mais que suficiente

Feedback adicional

Aqui poderá fornecer feedback adicional sobre a plataforma

9. **Quais foram os pontos mais fortes que viu na plataforma?**

10. **Quais foram os pontos menos fortes que viu na plataforma?**

Com tecnologia
Google Forms

Appendix E

Service Provider Perspective Inquiry

SnapTasks - Inquérito de usabilidade :: Prestador de Serviços

No âmbito da minha tese de mestrado em Engenharia de Software, venho aqui pedir o seguimento deste tutorial e consequente resposta às perguntas colocadas.

O SnapTasks é uma plataforma que faz a ponte entre prestadores de serviços e clientes. Tipicamente, os serviços aqui prestados obrigam uma operação logística para que o serviço seja feito. Um exemplo clássico é o serviço de lavandaria, em que o cliente pede um serviço, um estafeta vai recolher a roupa a casa e, assim que o serviço estiver feito, devolve o material ao cliente.

*Obrigatório

Registo e Log In

Vá a https://snaptasks-bo-management.azurewebsites.net/, crie uma conta e faça log in.

1. **Como avalia a facilidade de registo e log in?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil	◯	◯	◯	◯	◯	Muito fácil

Passe para a pergunta 2.

Prestadores de Serviços

Explore os prestadores de serviço existentes e os seus serviços.
Na página Service Providers (https://snaptasks-bo-management.azurewebsites.net/ServiceProvider) aparecerão os prestadores de serviço disponíveis. Acedendo a um, é possivel fazer várias operaçoes como:
- Adicionar serviços
- Remover serviços
- Alterar informações sobre o prestador de serviços

2. **Como avalia a apresentação dos prestadores de serviço?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil de entender	◯	◯	◯	◯	◯	Muito fácil de entender

3. **Como avalia a apresentação dos serviços de um prestador de serviços?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil de entender	◯	◯	◯	◯	◯	Muito fácil de entender

4. **Relativamente ao detalhe dado na pagina de um prestador de serviço, a informação foi suficiente?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito insuficiente	◯	◯	◯	◯	◯	Mais que suficiente

Serviços

Estando na página de um prestador de serviço, aceda aos detalhes de um dos serviços disponibilizados. Aqui e possível ver informação detalhada sobre esse serviço, alterar esse mesmo serviço, e atualizar o preço.

Para atualizar o preço, clique em Change Price, no fundo da pagina. Será apresentado um formulário para a atualização do preço.

5. **Como avalia a facilidade de atualizar o preço de um serviço?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil	◯	◯	◯	◯	◯	Muito fácil

6. **A apresentação dos dados de um serviço foi clara?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Nada clara	◯	◯	◯	◯	◯	Muito clara

7. **A apresentação do preço foi clara?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Nada clara	◯	◯	◯	◯	◯	Muito clara

Histórico de pedidos

Vá ao seu histórico de encomendas (https://snaptasks-bo-management.azurewebsites.net/Order) e explore os menus.
Aqui é possível ver encomendas realizadas, ver o seu estado e detalhes.

Avance encomendas ao clicar no nos botões de estado (ex. Assigned to Courier)

8. **Como avalia a informação dada?** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Nada clara	◯	◯	◯	◯	◯	Muito clara

9. **Como avalia a informação dada nos detalhes de uma encomenda (clicar em "Details")?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Nada clara	◯	◯	◯	◯	◯	Muito clara

10. **Como avalia a facilidade de mudança de estado de uma encomenda?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil	◯	◯	◯	◯	◯	Muito fácil

Feedback adicional

Aqui poderá fornecer feedback adicional sobre a plataforma

11. **Como avalia esta plataforma como solução para as suas necessidades de negócio?** *

12. **Quais foram os pontos mais fortes que viu na plataforma?**

13. **Quais foram os pontos menos fortes que viu na plataforma?**

Appendix F

Customer Perspective Inquiry - Answers

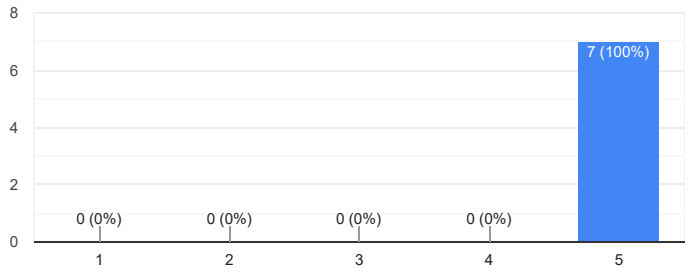
SnapTasks - Inquérito de usabilidade :: Cliente

7 respostas

Registo e Log In

Como avalia a facilidade de registo e log in?

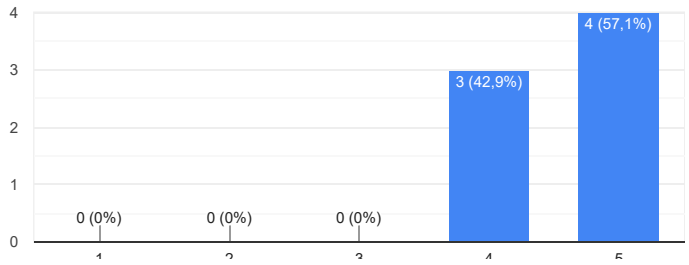
7 respostas



Prestadores de Serviços

Como avalia a apresentação dos prestadores de serviço?

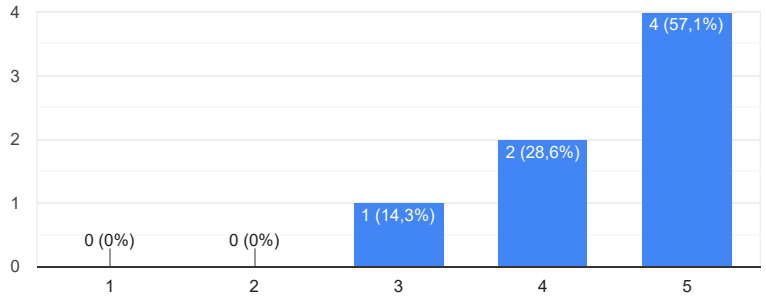
7 respostas



1 2 3 4 5

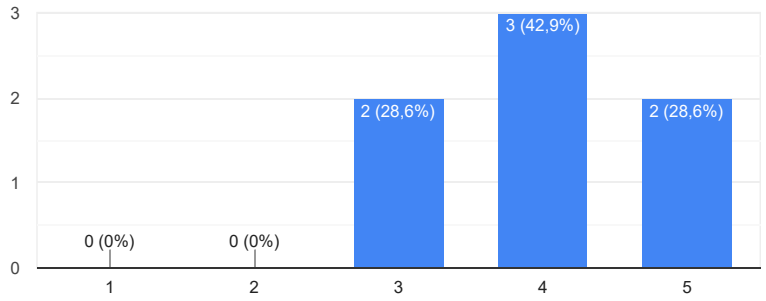
Como avalia a apresentação dos serviços de um prestador de serviços?

7 respostas



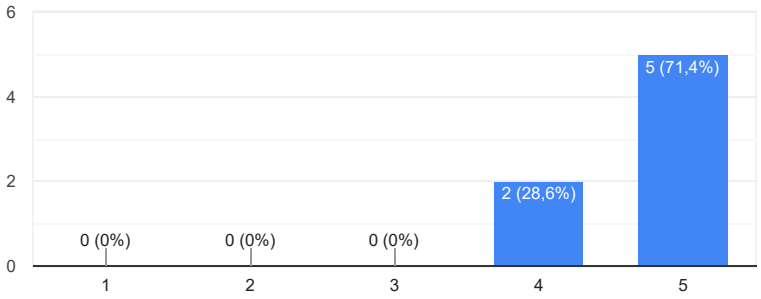
Relativamente ao detalhe dado na pagina de um serviço, a informação foi suficiente?

7 respostas



Para si, o preço de um dado serviço era claro?

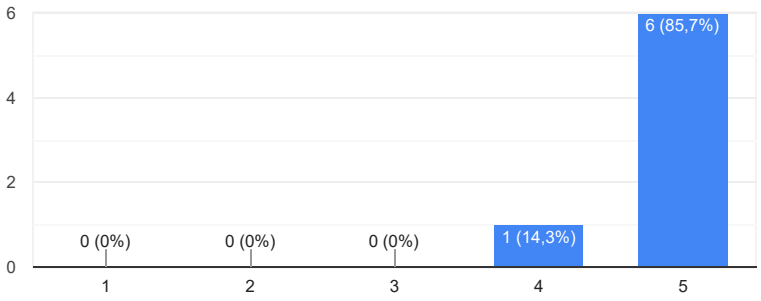
7 respostas



Checkout

Como avalia a facilidade de fazer a encomenda?

7 respostas



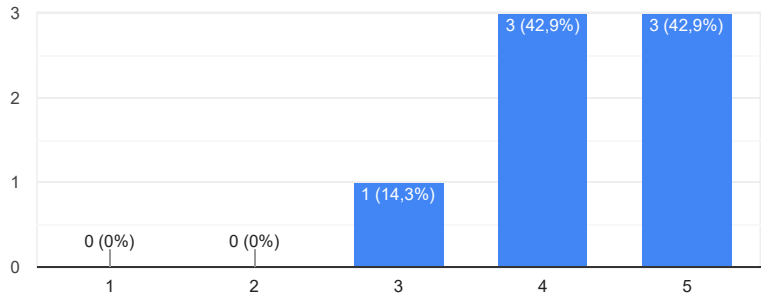
A apresentação dos dados da encomenda realizada (após a compra) foi clara?

7 respostas

Histórico de pedidos

Como avalia a informação dada?

7 respostas



Feedback adicional

Quais foram os pontos mais fortes que viu na plataforma?

3 respostas

Informação clara e interação intuitiva.

Facilidade e acessibilidade

As compras foram sempre efetuadas com sucesso.

Quais foram os pontos menos fortes que viu na plataforma?

2 respostas

Na página de pedidos de serviço só contém a seguinte informação: Id, Data, Total e Estado. Acho que era conveniente ter também o nome do serviço e uma descrição.

Simplicidade do site

Appendix G

Service Provider Perspective Inquiry - Answers

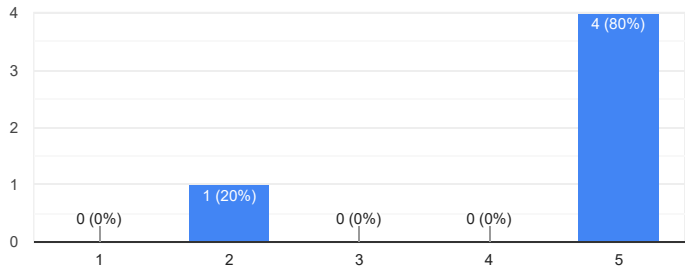
SnapTasks - Inquérito de usabilidade :: Prestador de Serviços

5 respostas

Registo e Log In

Como avalia a facilidade de registo e log in?

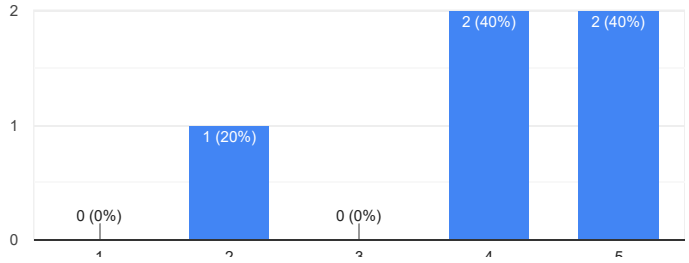
5 respostas



Prestadores de Serviços

Como avalia a apresentação dos prestadores de serviço?

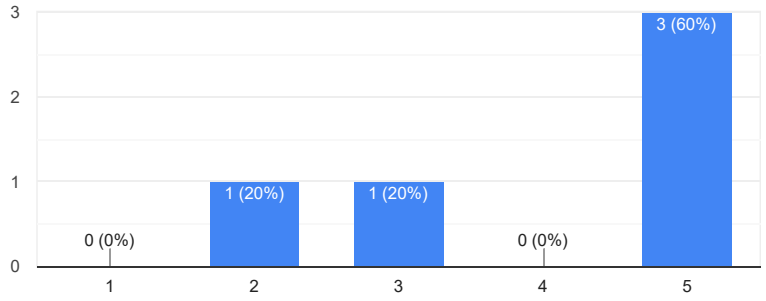
5 respostas



1 2 3 4 5

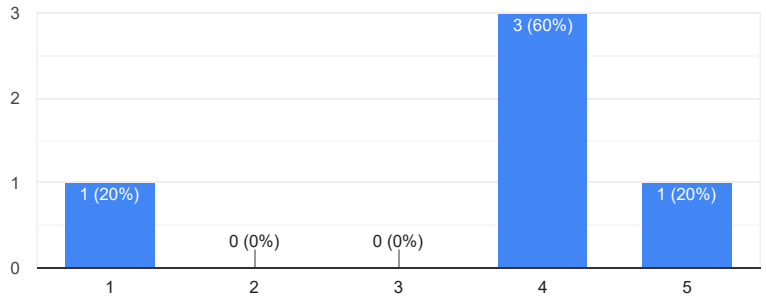
Como avalia a apresentação dos serviços de um prestador de serviços?

5 respostas



Relativamente ao detalhe dado na pagina de um prestador de serviço, a informação foi suficiente?

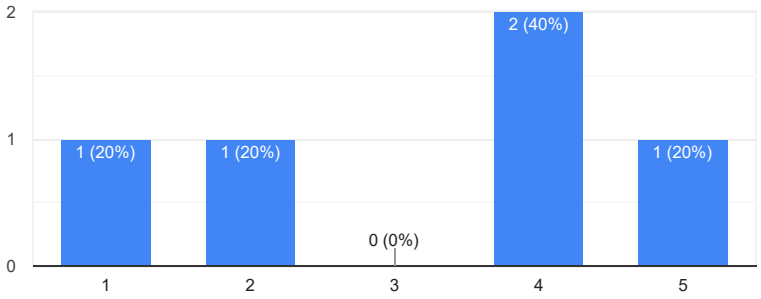
5 respostas



Serviços

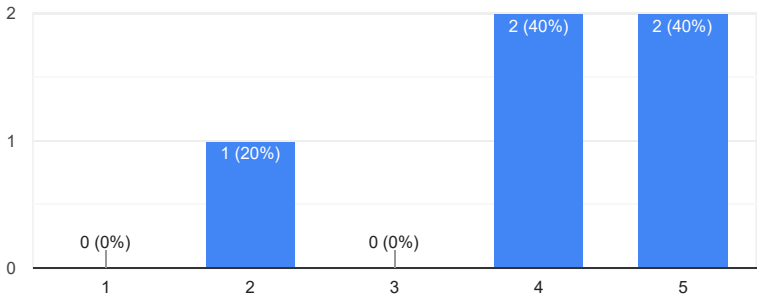
Como avalia a facilidade de atualizar o preço de um serviço?

5 respostas



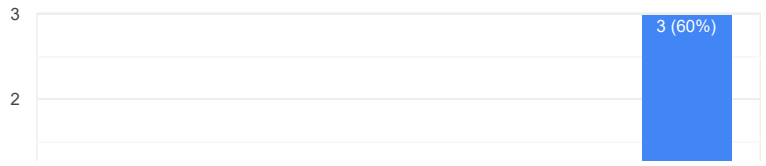
A apresentação dos dados de um serviço foi clara?

5 respostas



A apresentação do preço foi clara?

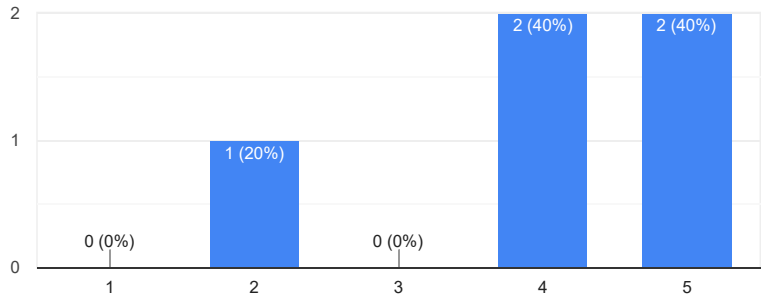
5 respostas



Histórico de pedidos

Como avalia a informação dada?

5 respostas

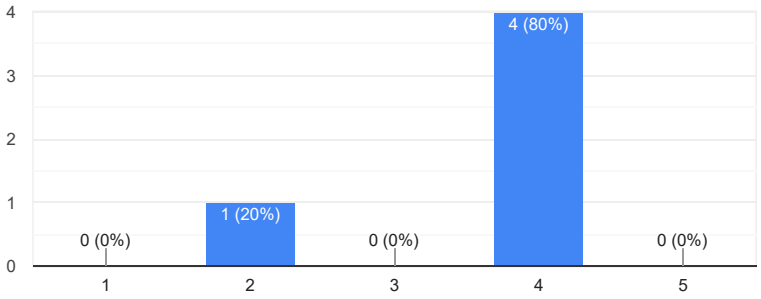


Como avalia a informação dada nos detalhes de uma encomenda (clicar em "Details")?

5 respostas

Como avalia a facilidade de mudança de estado de uma encomenda?

5 respostas



Feedback adicional

Como avalia esta plataforma como solução para as suas necessidades de negócio?

5 respostas

Esta aplicação pode-nos ajudar a prestar os nossos serviços a novos clientes de forma fácil e eficaz, fazendo com que o cliente não tenha que se deslocar às nossas instalações para poder usar os nossos serviços.

Excelente no sentido em que posso vender ainda mais

Top

pouco interessante. muitos aspectos a melhorar

Prática e transmite toda a informação necessária ao cliente

Quais foram os pontos mais fortes que viu na plataforma?

2 respostas

Simples e intuitivo

o conceito

Quais foram os pontos menos fortes que viu na plataforma?

2 respostas

A tradução à letra faz com que apareçam erros de português e termos pouco perceptíveis.

Na parte dos registos das encomendas, o facto da apresentação do id estar no inicio provoca confusão