



UNIVERSIDADE D
COIMBRA

Rita Miguel Fernandes Lemos Fernandes

CAMPAIGNS PLATFORMS A2P

Dissertation in the context of the Master in Informatics Engineering, Specialization in
Software Engineer advised by Professor Alberto Cardoso and presented to the
Faculty of Sciences and Technology / Department of Informatics Engineering
Under the guidance of Engineer Raul Fonseca along with Engineer Pedro Marques at Wit
Software

June, 2020

This page is intentionally left blank.

Abstract

Nowadays, devices are part of our daily lives and end up being indispensable in our routine, and in turn, there are more forms of communication in response to the needs of modern society.

Application-to-Person (A2P) campaign platforms enable businesses to communicate directly with consumers by sending promotional messages directly to their mobile messaging inbox (SMS, Whatsapp or RCS).

With the evolution of technology, nowadays, it is now possible for users to store their promotional coupons and loyalty cards in the virtual wallets of their devices, which would be a convenient alternative to physical passes.

The internship consists on development of a prototype A2P campaign platform, which allows validating mechanisms that enrich these campaigns, ensuring a better experience in their use. Functions were been developed under this prototype that includes a set of modules, respectively, SMS campaign engine for link distribution, loyalty module, vouchers, virtual wallet, profiling, segmentation, analytics, and campaign management module.

Keywords

"A2P", "SMS", "Campaign platforms", "Virtual wallet"

This page is intentionally left blank.

Resumo

Atualmente, os dispositivos fazem parte do nosso cotidiano e acabam por ser indispensáveis na nossa rotina, e, por sua vez, começam por existir mais formas de comunicação como resposta às necessidades da sociedade moderna.

As plataformas de campanhas A2P (Application-to-Person) possibilitam às empresas uma comunicação direta com os consumidores, através de envio de mensagens promocionais directamente para a *inbox* de *messaging* do telemóvel (SMS, Whatsapp ou RCS).

Com a evolução da tecnologia, hoje em dia, já é possível os utilizadores armazenarem os seus cupões promocionais e cartões de fidelidade nas carteiras virtuais dos seus dispositivos, o que seria uma alternativa conveniente aos passes físicos.

O estágio consiste no desenvolvimento de um protótipo de uma plataforma de campanhas A2P, que permite validar mecanismos que enriqueçam estas campanhas, garantido uma melhor experiências na utilização das mesmas. Foram desenvolvidas funcionalidades no âmbito deste protótipo que inclui um conjunto de módulos, respetivamente, motor de campanhas SMS para a distribuição de links, módulo de *loyalty*, *vouchers*, *wallet* virtual, *profiling*, segmentação, *analytics* e módulo de gestão de campanhas.

Palavras-Chave

"A2P", "SMS", "Plataformas de campanhas", "*Wallet* virtual"

This page is intentionally left blank.

Acknowledgements

First, I would like to thank Wit Software for the internship opportunity they gave me, and all the people involved in it in the past ten months.

Especially to my advisor, Raul Fonseca, for all the help available and for having guided me on the right path during the internship. Other special thanks to my tutor, Pedro Marques, for all the unconditional support and for the constant assistance in the development of this project that helped me to evolve at a professional level. I also want to thank all the employees at Wit Software who, in one way or another, helped me with any doubts that came up during the internship, I want you to know that I cherish all the time spent.

To my DEI advisor, professor Alberto Cardoso, for all the help in my constant doubts throughout the year, for having helped me deal with my worst moments and above all for having taught me that with effort and dedication, we managed to achieve our goals and that everything is rewarded.

Second, not least, I want to thank my closest friends, who are a fundamental part of my life, for always being on my side at all times. To Joana, friend and housemate, who gave me the support I needed in the last few months, especially in this pandemic situation that we all live in, in the company of each other that was essential to overcome the worst moments. To Matilde and Ana, the "mine" from Trás-os-Montes, for all the words of strength and friendship, helped me to overcome this stage with determination. Bruna, a friend for almost twenty years, for always helping me to face all phases of my life. To Maria Luís, for all the smiles that drew me and the strength, she gave me. To my friends, Francisco Quinaz and João Costa, who always gave me the strength to continue a phase that we started together.

Finally, I want to thank my parents, Maria Manuela and Miguel Ângelo, who, despite the distance, was always present at all times. For never letting me fall in my worst moments, for all the full strength they gave me, without their precious help, this would never have been possible, and for that, I will always be grateful. To my brother, and best friend, Eduardo Miguel, who is in another country, was still by my side in everything I needed, a phone call has never had such a bittersweet feeling.

To all of you, thank you, from the bottom of my heart.

This page is intentionally left blank.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Context | 1 |
| 1.1.1 | Motivation | 2 |
| 1.1.2 | Goals | 4 |
| 1.2 | Document Structure | 5 |
| 2 | State of the Art | 7 |
| 2.1 | Mobile Wallet App | 7 |
| 2.1.1 | Global Mobile Payment Statistics | 8 |
| 2.2 | Competitors | 10 |
| 2.2.1 | Wit Loyalty & Coupons Direct Competitors | 10 |
| 2.3 | Technologies | 14 |
| 2.3.1 | Back-end Technologies | 14 |
| 2.3.2 | Front-end Technologies | 15 |
| 2.3.3 | Support Tools | 15 |
| 3 | Approach | 17 |
| 3.1 | Methodology | 17 |
| 3.1.1 | Scrum Roles | 17 |
| 3.1.2 | Lifecycle | 17 |
| 3.2 | Planning | 18 |
| 3.2.1 | First Semester | 19 |
| 3.2.2 | Second Semester | 21 |
| 3.3 | Risk Management | 23 |
| 3.3.1 | Risks Identification | 24 |
| 3.3.2 | Risks Metrics | 27 |
| 4 | Requirements Analysis | 29 |
| 4.1 | User Stories | 29 |
| 4.2 | Functional Requirements | 30 |
| 4.2.1 | Agile Boards | 32 |
| 4.3 | Non-Functional Requirements | 34 |
| 5 | Software Architecture | 37 |
| 5.1 | Final Architecture High-Level Design | 37 |
| 5.1.1 | Loyalty & Coupons Backend | 38 |
| 5.1.2 | Management API Prototyping | 42 |
| 5.1.3 | Frontend API Prototyping | 43 |
| 5.2 | Loyalty App | 43 |
| 5.2.1 | System Context Diagram | 44 |
| 5.2.2 | System Containers Diagram | 45 |
| 5.2.3 | Reporting Module | 46 |

| | | |
|----------|--|-----------|
| 6 | Development and Final Product | 49 |
| 6.1 | Web Application | 49 |
| 6.1.1 | Authentication | 49 |
| 6.1.2 | Home Page | 50 |
| 6.1.3 | List campaigns | 50 |
| 6.1.4 | Insert/Update clients and Send SMS to Clients | 51 |
| 6.1.5 | Visualize number of customers using iOS and/or Android devices | 52 |
| 6.1.6 | Visualize number of points for each customer per brand and campaign | 52 |
| 6.1.7 | Visualize the five campaigns with the most downloads by customers | 53 |
| 6.1.8 | Visualize number of pass updates per brand | 54 |
| 6.1.9 | Visualize comparison between the number of downloads made by customers and the SMS sent to customers | 55 |
| 6.2 | Google Pay Interaction | 56 |
| 6.2.1 | Stamp card for Android users | 56 |
| 6.3 | Apple Wallet Interaction | 57 |
| 6.3.1 | Loyalty card for iOS users | 57 |
| 7 | Software Quality | 59 |
| 7.1 | Functional Testing | 59 |
| 7.1.1 | API Tests | 59 |
| 7.1.2 | Acceptance Tests | 60 |
| 7.2 | Non-Functional Testing | 60 |
| 7.2.1 | Performance Testing | 60 |
| 8 | Conclusion | 65 |

This page is intentionally left blank.

Acronyms

- A2P** Application-to-Person. 1, 2, 38
- API** Application Programming Interface. 4, 10, 11, 18, 59
- APNs** Apple Push Notification service. 41
- ELK** Elasticsearch Logstash Kibana. 46
- FCM** Firebase Cloud Messaging. 41
- HLD** High-Level Design. xiii, 37
- JWT** JSON Web Token. 43
- OTT** Over-the-Top. 1
- RCS** Rich Communications Services. 4
- SMS** Short Message Service. 1, 4
- SoTA** State of the Art. 7
- UI** User Interface. 11
- UX** User Experience. 59
- VPN** Virtual Private Network. 15, 61

This page is intentionally left blank.

List of Figures

| | | |
|------|---|----|
| 1.1 | Global average SMS open rate | 2 |
| 1.2 | The loyalty card problem | 3 |
| 1.3 | Consumers show desire for mobile Wallets | 4 |
| 2.1 | Global Top Mobile Payment | 8 |
| 2.2 | Passes examples from Loopy Loyalty | 13 |
| 2.3 | Passes examples from CherryPie | 13 |
| 3.1 | The Scrum Framework | 18 |
| 3.2 | Gantt Chart First Semester | 20 |
| 3.3 | Gantt Chart Second Semester | 22 |
| 3.4 | Real Gantt Chart Second Semester | 23 |
| 3.5 | Risk Matrix | 27 |
| 4.1 | Agile Board for User Story US01 | 32 |
| 4.2 | Agile Board for User Story US02 | 33 |
| 4.3 | Agile Board for User Story US03 | 33 |
| 5.1 | System architecture High-Level Design (HLD) | 37 |
| 5.2 | Interaction between the client and the server | 38 |
| 5.3 | Loyalty & Coupons View | 39 |
| 5.4 | Database tables | 40 |
| 5.5 | Apple Push Notification Overview | 41 |
| 5.6 | Firebase Cloud Messaging | 41 |
| 5.7 | Push Notifications View | 42 |
| 5.8 | API flow diagram [30] | 43 |
| 5.9 | Context Diagram of Loyalty App | 44 |
| 5.10 | Containers Diagram of Loyalty App | 45 |
| 5.11 | ELK Representation [34] | 46 |
| 5.12 | Filebeat Integration | 47 |
| 6.1 | Authentication in Loyalty App | 49 |
| 6.2 | Home Page | 50 |
| 6.3 | List campaigns | 50 |
| 6.4 | Insert/Update clients and Send SMS to Clients | 51 |
| 6.5 | Number of customers using iOS and/or Android devices | 52 |
| 6.6 | Number of points for each customer per brand and campaign | 52 |
| 6.7 | Top 5 campaigns with the most downloads | 53 |
| 6.8 | Number of pass updates per brand and per day/campaign | 54 |
| 6.9 | comparison between the number of downloads and SMS | 55 |
| 6.10 | Example of stamp card for Android | 56 |
| 6.11 | Example of loyalty card for iOS | 57 |

This page is intentionally left blank.

List of Tables

| | | |
|-----|--|----|
| 2.1 | Mobile Wallet Apps Analysis | 9 |
| 2.2 | Competitors Analysis | 12 |
| 2.3 | Back-end Technologies | 14 |
| 2.4 | Back-end Technologies | 15 |
| 3.1 | Risk 1 - Apple Wallet Study | 24 |
| 3.2 | Risk 2 - New Technologies Learning | 25 |
| 3.3 | Risk 3 - Google Pay does not exist in Portugal | 25 |
| 3.4 | Risk 4 - Integration on the Wit Buzz Platform | 26 |
| 3.5 | Risk 5 - API updates | 26 |
| 3.6 | Risk 6 - The COVID-19 pandemic | 27 |
| 4.2 | User stories | 30 |
| 4.4 | Functional Requirements | 31 |
| 4.6 | Non-Functional Requirements | 34 |
| 5.1 | Information for update process | 39 |
| 7.2 | API Tests Result | 60 |
| 7.4 | Acceptance Tests Results | 60 |
| 7.5 | Caption | 61 |
| 7.6 | View Results SMS Tree | 62 |
| 7.7 | SMS Summary Report | 62 |
| 7.8 | View results campaigns and clients management tree | 63 |
| 7.9 | Campaigns and clients management summary report | 64 |
| 1 | First Semester Planning | 71 |
| 1 | First Semester Planning | 72 |
| 2 | Second Semester Planning | 73 |
| 3 | Information to send by device | 75 |
| 4 | Responses | 75 |
| 5 | Information to send by device | 76 |
| 6 | Responses | 76 |
| 7 | Information to send by device | 77 |
| 8 | Responses | 77 |
| 9 | Information to send by device | 77 |
| 10 | Responses | 78 |
| 11 | Information to send by device | 78 |
| 12 | Download pass | 80 |
| 13 | Update pass and data client | 80 |
| 14 | List campaigns | 81 |
| 15 | Get all campaigns | 81 |

| | | |
|----|---|----|
| 16 | Get all brands | 81 |
| 17 | Registering a Device to Receive Push Notifications for a Pass | 82 |
| 18 | Getting the Serial Numbers for Passes Associated with a Device | 82 |
| 19 | Getting the Latest Version of a Pass | 83 |
| 20 | Unregistering a Device | 83 |
| 21 | Upload clients data | 84 |
| 22 | Get top downloads from passes | 84 |
| 23 | Get data from SMS logs | 84 |
| 24 | Get total number of costumers using iOS and/or Android from downloads logs | 85 |
| 25 | Get number of updates per day from updates logs | 85 |
| 26 | Get number of updates per campaign from updates logs | 85 |
| 27 | Get number of downloads performed and sms sent from logs | 85 |

This page is intentionally left blank.

Chapter 1

Introduction

The present report focuses on the work done by the author at Wit Software, S.A., during the annual internship in the Master's degree in Informatics Engineer in the Department of Informatics Engineer of the University of Coimbra. The work took place at Wit Software, S.A., under the guidance of Professor Alberto Cardoso of the Department of Informatics Engineer and Engineer Raul Fonseca along with Engineer Pedro Marques at Wit Software, S.A.

1.1 Context

One of the primary sources of revenue for telecommunications operators is the Short Message Service (SMS). Currently, operators are losing SMS traffic to other Over-the-Top (OTT) technologies, such as Whatsapp, Facebook, Skype, and Messenger. These technologies offer better services at a low cost to users. These OTT messaging services have seen real growth in the past two years. However, SMS has features that beat the alternatives of view rate and open rate, and for this reason, this project aims to develop features that make their revival. Also, another advantage over OTT services is that an internet connection is not required to receive messages.

Application-to-Person (A2P) is a term that describes the process of sending text messages from an application to a device. Companies can use A2P SMS to communicate with their customers, sending alerts, and carrying out marketing campaigns, among other features. In the context of A2P campaigns, loyalty features are built, such as virtual loyalty cards, among others. This loyalty features allow the existence of an interaction between the brand and the customer so that the customer is satisfied and, consequently, loyal to the brand. In addition to the customer loyalty features, in the context of the internship, a campaign analysis block for the brand is also developed.

Figure 1.1 represents the result of a survey conducted in 2018, with about 992 participants, where it is possible to observe the performance of the SMS.

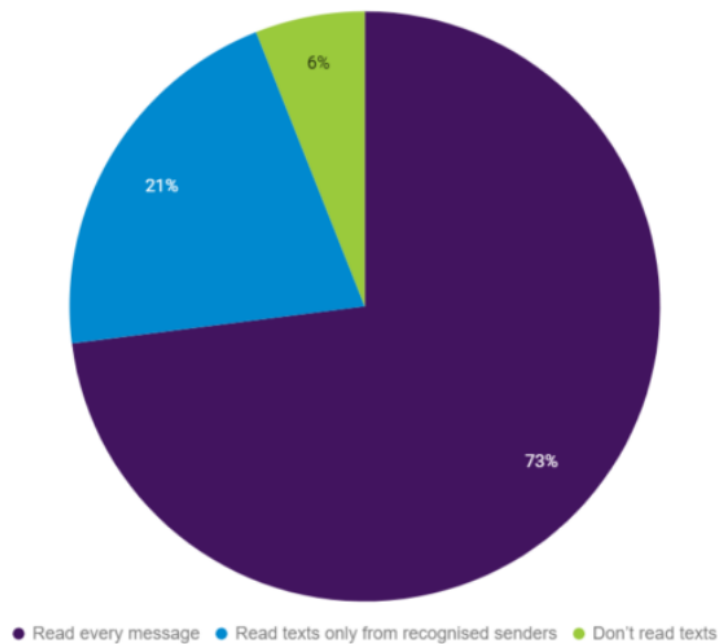


Figure 1.1: Global average SMS open rate [36]

A2P is a term used to describe a process of sending messages from a software app to a phone. These types of messages include shipping notifications from online stores, appointment reminders, promotional and loyalty program notifications, and two-factor authentication one-time passcodes for account security.

1.1.1 Motivation

Nowadays, as e-commerce evolves, the need to use the mobile phone is increasing more and more and consequently has made it easier to use various practical things. Virtual wallets have been replacing traditional wallets, with approximately 85% of smartphone users using in-store purchases, travel and events.

One of the most significant advantages of using virtual wallets for mobile payments is the ease and affordability the customer has. In addition to speeding up the entire payment process, it can help with retaining customers to the brand, thus preventing them from switching to a competitor.

Often using physical cards becomes a problem because, in addition to the likelihood of losing them is significant, we also forget about them, or in case of discount cards, we don't remember that we have them and can still use them.

Virtual wallet applications have been increasingly used, as they allow the use of various functionalities. Also, a great advantage is mobility since it is possible to use them on smartphones and tablets. Through the applications of virtual wallets, it is possible to make payments or store them and use virtual passes, such as discount coupons, boarding passes, concert tickets, among others.

Also, through the use of these applications, it's possible to be informed by notification of the use of discount coupons when they're within their expiration dates and/or we're near a store where we can use them. For example, if we have location enabled on our mobile phone and a specific pass registered into the wallet, this would allow us to send a notification that you're close to the stores and thus enjoy the services.

During the internship, we intend to develop features that include loyalty modules, support for vouchers, and wallet. These features will allow brands to send campaigns through previously spoken messaging to customers so that they can use the virtual wallet along with the received passes.

Figure 1.2 shows the top two reasons why the use of physical loyalty cards is a problem - this statistic is provided by **Airship** in a 2016 study. [38]

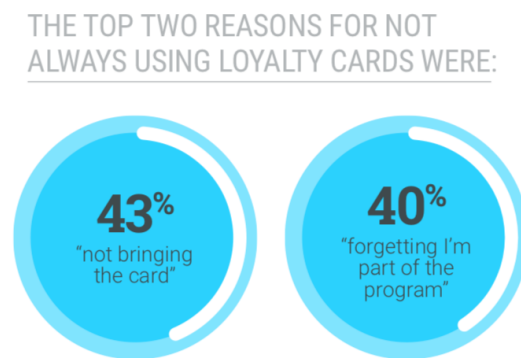


Figure 1.2: The loyalty card problem [38]

Figure 1.3 shows what consumers desire for mobile Wallets. These statistics were provided by **Airship**, a customer engagement platform, and fielded in 2016. [38]

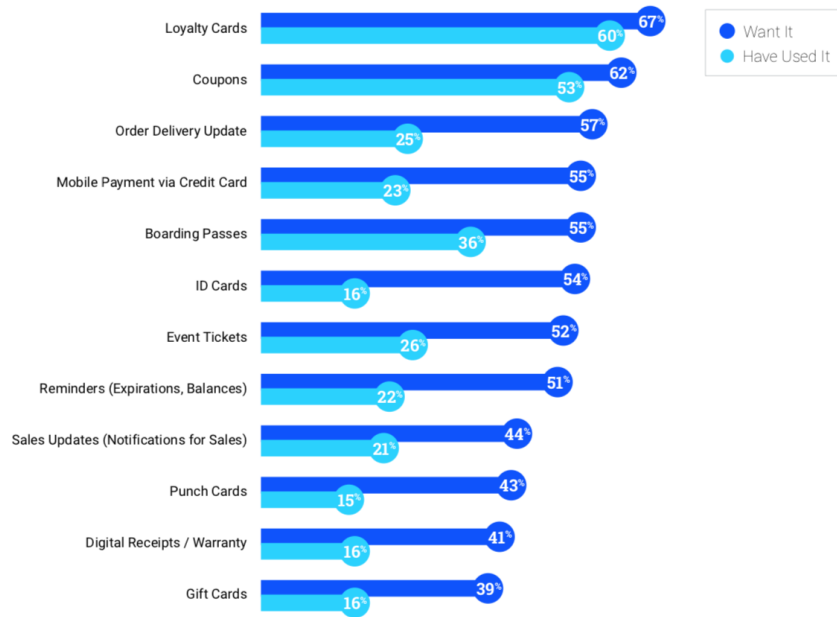


Figure 1.3: Consumers show desire for mobile Wallets [38]

1.1.2 Goals

The objective of this project is to develop a set of features that complete an A2P campaign platform, a campaign management platform for brands.

Project Goals

The main objective of the project is to develop an Application Programming Interface (API) for managing and distributing virtual passes to strengthen a better customer relationship, offering rewards, and consequently obtaining customer retention for brands. This internship focus on the creation of promotional campaigns and their distribution through push notifications, messages via SMS, WhatsApp, and Rich Communications Services (RCS), for iOS and Android devices.

The developed API was implemented in such a way that it is possible to integrate into different applications that allow the use of the developed tools, such as the creation and management of coupons, among others. Wit Software currently has a team developing a dashboard that allows brands (main customer) to manage their campaigns and distribute them among their customers, in addition to having access to the platform activity statistics, which in the future can be used as application integration.

1.2 Document Structure

This document is divided into 8 chapters, including the Introduction and the following:

- **State of the Art:** It presents state-of-the-art analysis, which defines the possible competitors for the product to be developed, as well as comparisons with each other, including Wit's solution.
- **Approach:** It presents the methodology used for project development, defining all its characteristics and the planning for the two semesters, representing all the changes made.
- **Requirements Analysis:** It presents an analysis of the requirements identified, more precisely, user stories, functional, and non-functional requirements.
- **Software Architecture:** It gives a review of the architecture solution provided, as well as the high-level architecture designed for the project.
- **Development and Final Product:** Presents and describes the layouts obtained during the internship project.
- **Software Quality:** It describes all the tests adopted to evaluate, measure, and guarantee the quality of the developed software.
- **Conclusion:** Provides a conclusion on project development so far, defining what may have gone wrong and well.

This page is intentionally left blank.

Chapter 2

State of the Art







This chapter will present an analysis of the current market and the applications that compete against the internship product. State of the art is the first significant phase in the development of a project to demonstrate research results. The importance of State of the Art (SoTA) is a research business base.

The survey of the SoTA is a crucial step in the development of a project because it allows assessing the viability of the project through the analysis of existing solutions, identifying the strengths and flaws they have, as well as analyzing the saturation level of the project. Based on this analysis, it is possible to determine the project risk level and its probability of success.

2.1 Mobile Wallet App

Different virtual wallet apps store payment card information and virtual passes on a mobile device. These virtual wallets have become a more convenient and affordable way for users to make in-store payments and virtual service coupons and can be used by merchants associated with the service provider of their virtual wallet.

These virtual wallets are available in different countries and operating systems; some of these are:

- Apple Wallet [6] 
- Google Pay [10] 
- Wallet Passes [32] 
- Pass2U [19] 
- Samsung Pay [23] 
- WeChat Wallet [33] 

2.1.1 Global Mobile Payment Statistics

According to a 2019 study, **WeChat Pay** is the world's largest mobile payment platform with over one billion users [1].

| Company | Active users | Latest figures from |
|-------------|--------------|---------------------------------------|
| WeChat | 1 billion+ | Tencent (Jan 2019) |
| Alipay | 1 billion+ | Alipay (Jan 2019) |
| Paypal | 250 million | PayPal (Sep 2018) |
| Apple Pay | 383 million | Loup Ventures; QZ (Feb 2019) |
| Amazon Pay | 50 million | Evercore ISI, Investopedia (May 2018) |
| Samsung Pay | 1 billion+ | Statista (Aug 2017) |
| Google Pay | 24 million | Statista (Aug 2017) |

Figure 2.1: Global Top Mobile Payment [1]















According to a 2019 study [1], 36.3% of smartphone users are expected to make a mobile in-store payment at least once every six months.

Analysis

The three most popular apps [27] globally are **Apple Wallet**, **Google Pay**, and **Samsung Pay**. For integration in this project, Apple Wallet and Google Pay will be used, respectively, for iOS devices and Android devices.

Below, in table 2.1, we can see a comparative analysis between some existing virtual wallet applications.

Table 2.1: Mobile Wallet Apps Analysis

| |  |  |  |  |  |  |
|------------------------------|---|---|---|--|---|--|
| Availability | Global | 130 countries | Global | Global | 25 countries | 3 countries |
| Scan to Pay | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| NFC Tap to Pay | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| Time Based Alerts | ✓ | ✓ | N/A | ✓ | N/A | ✓ |
| GPS Lockscreen Notifications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Push Messages | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| Support Bluetooth Beacons | ✓ | ✓ | ✓ | ✓ | N/A | N/A |
| Mobile Payment | Apple Pay | Google Pay | Google Pay | Apple Pay, Google Pay | Samsung Pay | WeChat Payment |
| Pay In-App | ✓ | ✓ | N/A | N/A | ✓ | API for Apps |
| Pay In-Store | ✓ | ✓ | N/A | N/A | ✓ | ✓ |
| Operating System |  iOS |  ANDROID |  ANDROID |  ANDROID  iOS |  ANDROID |  ANDROID  iOS |

After analyzing the mobile platforms previously presented, Google Pay and Apple Wallet were chosen to develop the project. Since both applications are installed natively on the devices and are the most used in Europe and America, they stand out quickly in the selection for the desirable requirements.

2.2 Competitors

This section presents an analysis of the current market and some services that compete with Wit solutions. Two types of competitors have been identified. The first are platforms that contain features that indirectly compete with WitBuzz.

Witbuzz is a web-based platform developed by a team of Wit Software, which allows us to create campaigns of different categories and send them to a set of selected customers, as well as observe all user activity on the platform. Campaigns can be engagement, promotion, retention, survey, gamified ads, and a custom ad. The project developed in the internship will later be integrated into WitBuzz, an A2P campaign management platform. The second competitors are the platforms that compete directly with the stage to develop.

Analyzing potential competitors is a smart and essential process because it allows you to understand the success strategy better, estimate the impact of risks, and understand its viability.

2.2.1 Wit Loyalty & Coupons Direct Competitors

This subsection presents platforms that compete directly with the project to be developed in the internship.

PassKit [20] allows us to manage and distribute virtual passes, such as coupons, loyalty cards, event tickets, boarding passes, and more. This platform enables users to store any pass in their virtual wallets.



Also, PassKit has the advantage of integrating with payment options, Apple Wallet, and Google Pay, respectively.



Loopy Loyalty is a web application launched by **PassKit**, to create and manage digital stamp cards for *Apple Wallet* and *Google Pay*. Design cards online, send push messages to customers, capture transaction history, and view customer insights [16].

The **CherryPie** is one of the solutions created by **PassKit** that includes all the tools for campaign management, clients, and messaging [28].



Voucherify is a cloud-based platform for managing promotions, helping to increase customers for service, and helping them retain. [31]



This Application Programming Interface (API) offers solutions for gift cards, in-cart discounts, coupon discounts, referral, loyalty, giveaway programs and landing pages.

Voucherify is an enterprise-ready digital promotion platform, and the following features are highlight:

- **Developer-friendly API:** Flexible REST API, SDKs, and webhooks reduce integration time with the rest of your marketing ecosystem
- **Strategic, 24/7 support:** The team offers customized onboarding, strategy consulting, technical guidance and round-the-clock support
- **Role-based access control:** Build custom workflows suited to each individual or team with fine-grained privileges and custom roles
- **Enterprise-grade infrastructure:** Launch mission-critical projects with a secure, GDPR- and CCPA-ready with a robust platform

This competitor is the most outstanding of all, is the most complete in its characteristics and also provides the necessary tools for a possible integration:

- Well-documented REST API
- Quick start guides and sandbox
- 10 SDKs and User Interface (UI) widgets
- Webhooks
- Monitoring and logs

CouponsTools allow to create and distribute coupons to users. This platform has digital coupon management, vouchers management, distribution, customer engagement and interaction, validation, data capture, and statistics that enables businesses to expand and increase sales. [18]



iVend Loyalty is an offer management application that allows users to accumulate points or purchases that they can enjoy for discount coupons, gifts, or other rewards. [7]

Besides, this application gives brands access to their customers' purchase preference histories and, therefore, can offer them proposals and rewards through notifications.

Although **iVend Loyalty** is part of an **iVend Retail** management pack, it works independently and can be integrated with other services.

Talon.One is a straightforward integration with a flexible API, and people can create promotional campaigns that boost their ROI. [26]



Analysis

The complex analysis is made between the competitors and the Wit solution, thus obtaining the various possible solutions and even later to be able to integrate the modules with one of the existing APIs.

Below, in table 2.2, we can see the products that we intend to purchase, marking them with a checkmark which is possible to buy in the corresponding service, and with a red cross mark those which aren't available.

| | Wit Loyalty & Coupons | Voucherify | Talon.One | iVend Loyalty | CherryPie | Couponstools | PassKit | Loopy Loyalty |
|--|-----------------------|------------|-----------|---------------|-----------|--------------|---------|---------------|
| Coupons | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Loyalty Cards | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Gift Card Management | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Event Tickets | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Boarding Passes | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ |
| Membership Management | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Discount Cards | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Support for native ios and android wallets | ✓ | ✗ | ✓ | - | ✓ | ✗ | ✓ | ✓ |
| Integrations and third-party guides | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✗ |
| Cross-channel delivery | ✓ | ✓ | ✓ | - | ✗ | ✓ | ✓ | ✓ |

Table 2.2: Competitors Analysis

Figure 2.2 shows different examples of stamp cards using **LoopyLoyalty** software.

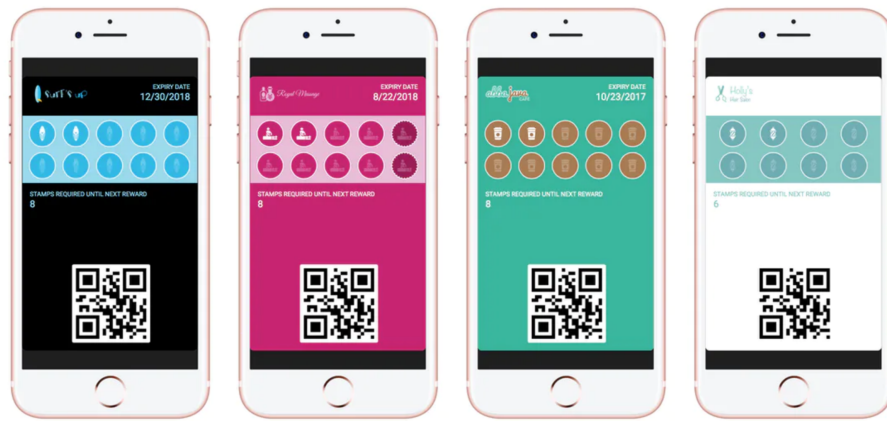


Figure 2.2: Passes examples from Loopy Loyalty

Figure 2.3 shows different examples of passes types created by **CherryPie**.

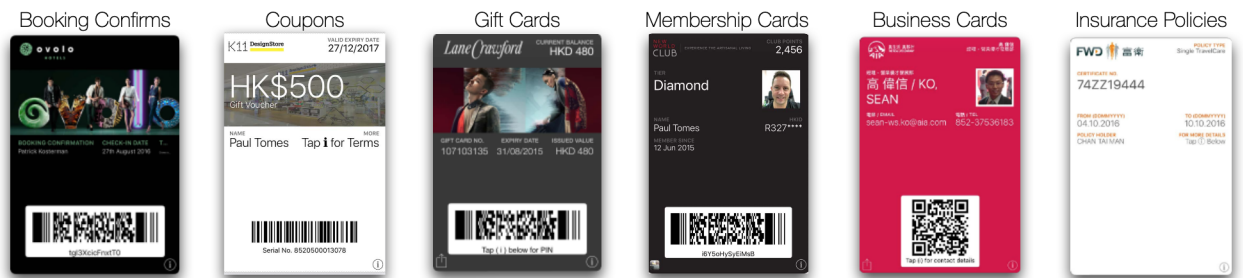


Figure 2.3: Passes examples from CherryPie
[12]

The difference between Loopy Loyalty and Cherry Pie is the type of pass that both can create. Loopy Loyalty only develops passes like stamp cards, while Cherry Pie develops virtual passes of different types, as shown in the figure above.

2.3 Technologies

This section will describe an analysis of the technologies used for project development, including the implementation of the Rest API, database, and other tools.

2.3.1 Back-end Technologies

For the development of this project, two back-ends were implemented, and several languages, frameworks, and libraries were used.

Apache Maven is a project management tool based on the POM (project object model) that provides a complete project building life cycle. The programming language used was Java. [13]

Table 2.3: Back-end Technologies

| Technologies | Description |
|---------------------|--|
| Java | The Java™ Programming Language is a general-purpose, concurrent, strongly typed, class-based object-oriented language. It's typically compiled to the bytecode instructionset and binary format defined in the Java Virtual Machine Specification [13] |
| Maven | Apache Maven is a software project management and comprehension tool [17] |
| Spring Boot | It is a framework for Java developers that provides a quick and straightforward way to configure and run web-based applications [25]. |
| PostgreSQL Database | It is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads [37]. |
| Kibana Dashboard | It is a free and open user interface that lets you visualize your Elasticsearch data and navigate the Elastic Stack. [14] |
| Elasticsearch | It is a highly scalable open-source full-text search and analytics engine. [2] |
| Logstash | It is a log aggregator that collects data from various input sources, executes different transformations and enhancements and then ships the data to various supported output destinations. [8] |
| FileBeat | It is a log data shipper for local files. Filebeat agent will be installed on the server, which needs to monitor, and filebeat monitors all the logs in the log directory and forwards to Logstash. [15] |

2.3.2 Front-end Technologies

The Loyalty App Front-end was also developed for interaction with the brand.

Table 2.4: Back-end Technologies

| Technologies | Description |
|--------------|--|
| Typescript | The TypeScript is a typed superset of JavaScript that compiles to plain JavaScript. [29] |
| React | A JavaScript library for building user interfaces |
| CSS | A mechanism for adding style to a web document. |
| Bootstrap | Front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins [5] |

2.3.3 Support Tools

To support the development of the internship project were used some tools, some of which are used within Wit Software and support the Scrum structure.

- **Redmine:** A web platform for project management that includes tools that allow checking the status of the project through agile boards after identifying user stories and functional requirements, in addition to allowing you to create graphics, such as the Burndown Chart.
- **IntelliJ IDEA:** It is an IDE written in Java that allows you to develop computer software.
- **Visual Studio Code:** It is a code editor redefined and optimized for creating and debugging web applications used in the development of the Loyalty App front-end.
- **Postman:** It is a software development tool used to perform API tests.
- **Ovearleaf:** It is an online LaTeX editor used for this internship report.
- **GitLab:** GitLab is a software repository manager based on git, which allows you to store the implementation code on Wit's servers.
- **Apache Jmeter:** JMeter is a tool that performs load tests on static or dynamic resources.
- **Forticlient:** Simplifies remote user experience with built-in auto-connect and always-up Virtual Private Network (VPN). [9] features.

This page is intentionally left blank.

Chapter 3

Approach

This chapter describes the company's approach to software development that is followed in the internship project. Also, the identified risks and the planning of tasks to be performed during the internship are presented.

3.1 Methodology

This section describes an overview of the methodology used for software development. The methodology adopted was **Scrum**, a lightweight, agile, and simple to understand methodology, which is a framework where people can solve complex problems while creatively and productively delivering products with the highest possible value.

3.1.1 Scrum Roles

- **Product Owner:** The product owner is responsible for maximizing product value, and the work of the development team, is the sole person responsible for managing the Product Backlog [11]. In this project, the project owner Engineer Raul Fonseca.
- **Scrum Team:** The Development Team comprise professionals working to deliver a potentially usable "Done" product increment at the end of each Sprint [11]. Raul Fonseca, Pedro Marques, and I are the scrum team in this project.
- **Scrum Master:** "The Scrum Master is responsible for promoting and supporting Scrum, helping everyone understand Scrum theory, practices, rules, and values. The Scrum Master is a servant-leader for the Scrum Team" [11]. In this project, the scrum master is Pedro Marques.

3.1.2 Lifecycle

The **Scrum** process focuses mainly on project management, where it assumed that the team is self-organizing, this methodology based on dividing the project into several goals so that incrementally the results can be shown regularly to the client. Thus, some problems are identified at a stage that will not significantly affect project development.

At the beginning of a project, after the Product Owner meets with stakeholders to better understand the needs, a set of requirements is defined that the project must address,

called **Product Backlog**. After the Product Backlog is defined, each requirement is given priority and which ones have the highest value. The priority requirements are then sent to the **Sprint Backlog** so that they take part in work cycles called **Sprints**, in which the duration of these does not exceed four weeks.

At the beginning of the cycle of each sprint, there is a planning meeting called Sprint Planning, where the Sprint Backlog and the defined time to complete the sprint are presented. During the execution, daily meetings are held, from 10 to 15 minutes, to review what was done the previous day and rectify what will be done next. After completing the sprint, review and retrospective meetings are held. In the **Sprint Review**, it is made a presentation of what has already been developed and alignment the elements of the product backlog in the next sprint. The **Sprint Retrospective** attends to raise difficulties faced during that period and find solutions on how to reduce some possible obstacles in the next sprint.

Below in figure 3.1, we can observe the various phases of this methodology. [35]

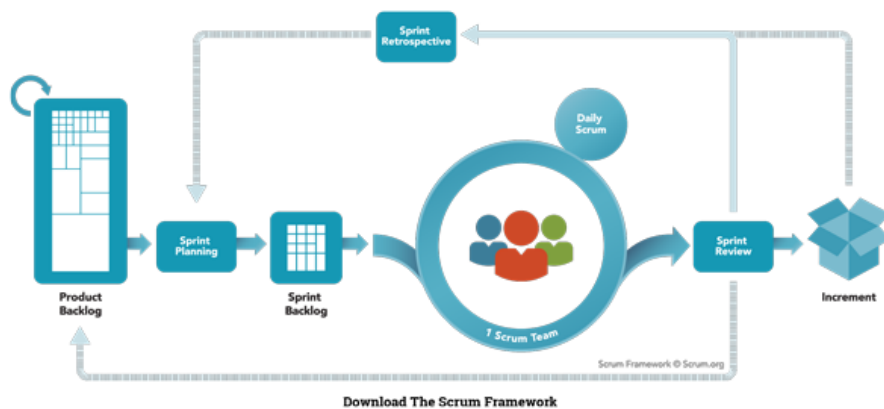


Figure 3.1: The Scrum Framework [35]

3.2 Planning

This section presents the planning for both semesters for the completion of the internship. The plan represents an orientation of the internship proposal. Some of these proposals have changed, influencing the order of tasks, taking into account their priority.

As seen previously, an API for integration in a native **iOS** application and a native **Android** application will be developed, focusing on the iOS app in the first semester and continuing in the second semester for the Android app.

Since we have adopted a Scrum methodology, the use of Gantt Charts for plain is not practical, as it is not static sprint tasks are taken directly from the Product Backlog. Keeping a Gantt Chart up-to-date requires much effort since sprint identification needs to be well planned and could change regularly. As a representation of the work done in the first semester and the work done in the second semester, agile boards were created. Chapter 4 shows the tasks followed by the agile boards for each one.

In appendix 8, we can see the full plan, respectively, the defined sprints, the tasks, and the dates.

3.2.1 First Semester

For the first semester, the planned tasks were as follows:

- State of the art survey
- Research activities on the main techniques and concepts A2P campaigns
- Requirements analysis
- Identification and design of the functional modules to be prototyped and their architecture
- Definition of the development plan
- Prototyping the functionality of some of the A2P platform's internal modules
- Management API prototyping
- Preparation of prototypes for internal demonstrations
- Intermediate internship documentation

The initial plan was subject to minor changes and the first tasks were kept, respectively. The research needed to achieve state of the art, including an analysis of competitors and technologies, and requirements gathering to define the stories of the user and functional and non-functional requirements. Next and focusing only on the iOS branch, an analysis was made of all the documentation provided by Apple that needs to be respected for the development of virtual passes, so the client-server architecture was taken into consideration, thus creating a high-level architecture. After this phase, the development of the necessary software to complete the defined sprints was started, thus obtaining a final demonstration.

During the internship, sprints planned to create a *Gantt chart* as a representation of the work done and their estimated hours. Below in Figure 3.2, we can see the Gantt chart for the first semester.

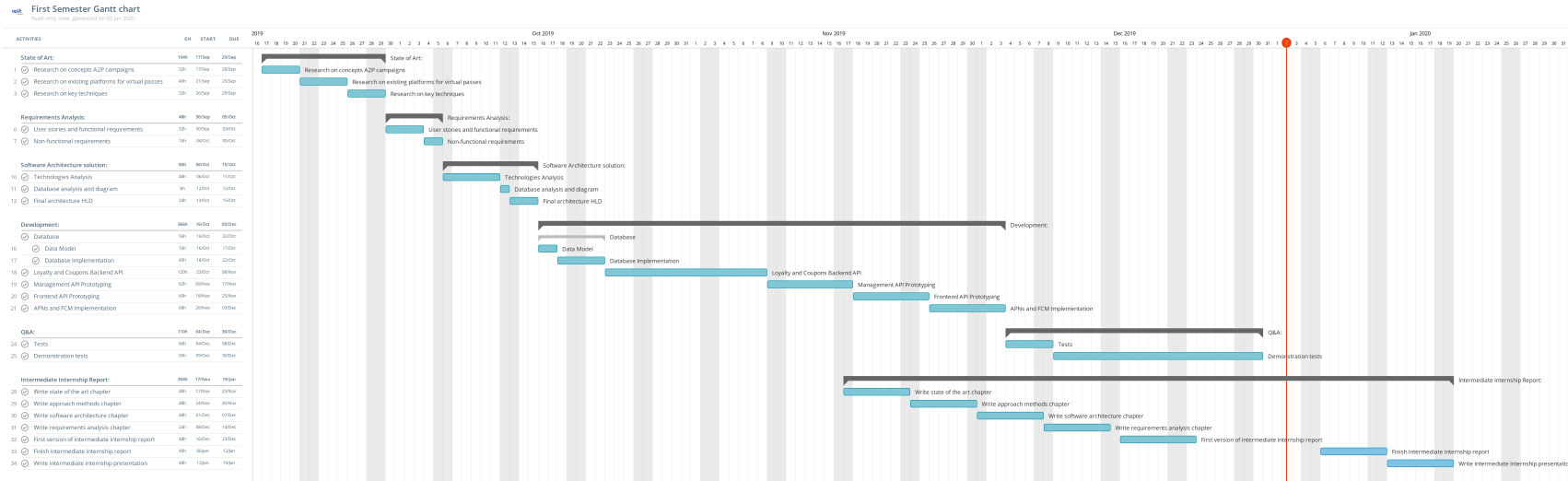


Figure 3.2: Gantt Chart First Semester

3.2.2 Second Semester

For the second semester, the planned tasks are as follow:

- Continued prototyping of the functionality of some of the A2P platform's internal modules;
- Continued prototyping of the management API;
- Frontend API prototyping;
- Prototyping of demonstrative use case;
- Functional tests;
- Usability tests;
- Evaluation of non-functional requirements;
- Final internship documentation.

The tasks defined in the internship proposal, by Wit for the second semester, were duly changed. These were identified as a continuation of the tasks performed in the previous semester, but now for the Android branch.

As previously presented, in the second semester, a second version of the modules developed in the first semester would be extended but now for Android devices, which in turn would be integrated into a platform that Wit has been growing in recent years, the Wit Buzz Platform.

During the internship, sprints planned to create a *Gantt chart* as a representation of the work done and their estimated hours. Below in Figure 3.3, we can see the Gantt chart for the second semester.

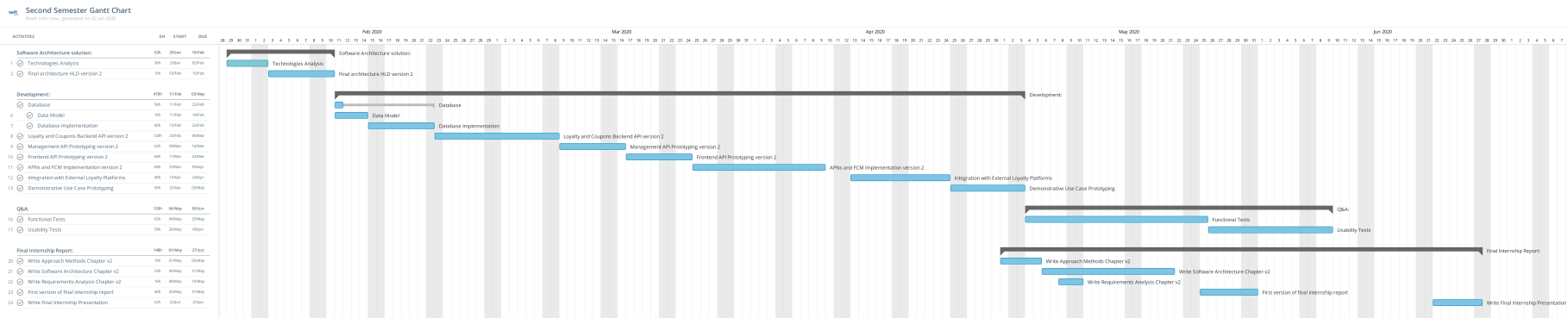


Figure 3.3: Gantt Chart Second Semester

Although the tasks proposed for the beginning of the semester were maintained and completed, some crucial changes were made during the remainder of the period. Below, in Figure 3.4, we can see the management of the real plan followed in the present semester.

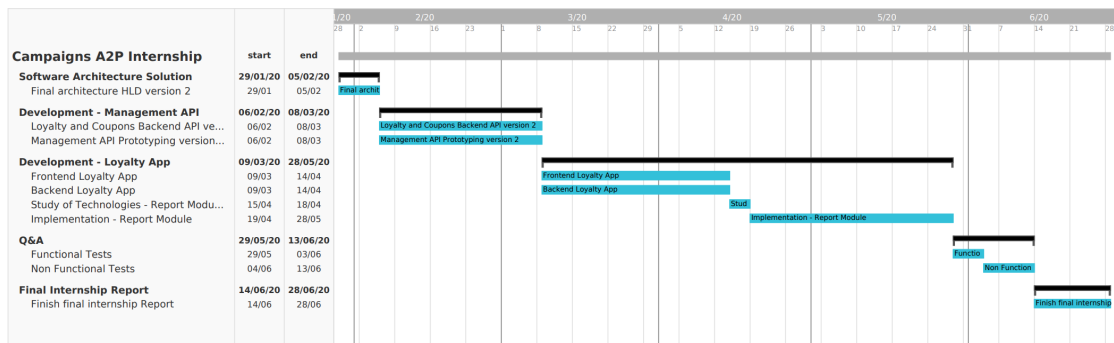


Figure 3.4: Real Gantt Chart Second Semester

The main change was the integration in the Wit Buzz Platform, after better analyzing how this integration would be done, it was concluded that it would be a very complex integration and consequently it would require a lot of effort at the User Experience level on the part of the intern. This effort and focus on UX would mean a significant departure from the objectives initially defined for the internship. Instead, a reference Web App was developed, the Loyalty App Platform, where all the defined functionalities were implemented, to carry out functional and non-functional tests, and finally, get a demo. With the development of the web app, we were able to prove the correct functioning of the Platform and thus create more confidence about the success of the future integration of the Loyalty App Platform with Wit Buzz.

3.3 Risk Management

In a software development project, there is a risk that can be avoided or minimized if detected early. This section presents an analysis of the risks identified during the project and the mitigation plans to reduce the impact they may have.

A risk can be defined as a condition that, eventually, can result in a negative effect on the development of a project. Risk management serves to identify, analyze, plan, and monitor risks. When this management is done, a reduction in the impact of the project is expected.

For risk management, there are four steps that must be followed:

- **Risk Identification:** In this phase, the intern looks for factors that may hinder the development or success of a project.
- **Risk Analysis:** Once a risk has been identified, it must be analyzed according to some attributes, respectively, the impact of the risk and the probability of its occurrence.

The following measures are considered for the impact of risk:

- **High** It can affect project performance and significantly affect the schedule. Project success may not be achievable.
- **Medium** It may slightly affect project performance and schedule. Project success can be achieved.

- **Low** It may have little effect on project performance and schedule. Project success can be easily achieved.

The following measures are considered for the likelihood of the risk occurring:

- **High** probability of happening is greater than 70%
 - **Medium** probability of happening is between 30% and 70%
 - **Low** probability of happening is below 30%
- **Risk Avoidance and Mitigation:** In this phase, the objective is to mitigate or eliminate the occurrence of risks; for this, it is necessary to establish a mitigation plan.
 - **Risk Monitoring:** At this stage, risks are monitored by updating them during the development of the project.

3.3.1 Risks Identification

This subsection represents the risks identified during the internship.

| | |
|------------------------|---|
| ID | R01 |
| Title | Apple Wallet Study |
| Description | The API to develop involves integration with an internal Apple application, it takes some time to understand how the application is structured. |
| Verified | Yes |
| Mitigated | Yes |
| Impact | High |
| Probability | Medium |
| Mitigation Plan | - Study of existing documentation, provided by Apple. |

Table 3.1: Risk 1 - Apple Wallet Study

| | |
|------------------------|--|
| ID | R02 |
| Title | New Technologies Learning |
| Description | The intern had to learn to deal with new technologies that she was not familiar with (<i>Typescript</i>). Difficulties in using these technologies can cause delays in the development of tasks. |
| Verified | Yes |
| Mitigated | Yes |
| Impact | High |
| Probability | Medium |
| Mitigation Plan | <ul style="list-style-type: none"> - Help from the Wit tutor and advisor; - Train with the resolution of online tutorials on technologies. |

Table 3.2: Risk 2 - New Technologies Learning

| | |
|------------------------|---|
| ID | R03 |
| Title | Absence of Google Pay |
| Description | Google Pay App doesn't exist in Portugal |
| Verified | Yes |
| Mitigated | Yes |
| Impact | High |
| Probability | High |
| Mitigation Plan | <ul style="list-style-type: none"> - Use apk file from Google Pay; - Contact the Google team to create accounts and provides some features. |

Table 3.3: Risk 3 - Google Pay does not exist in Portugal

| | |
|------------------------|---|
| ID | R04 |
| Title | Integration on the Wit Buzz Platform |
| Description | The integration in the Wit Buzz Platform is very complex to implement and requires a lot of user experience effort on the part of the intern. It can lead to delays in project development. |
| Verified | Yes |
| Mitigated | Yes |
| Impact | Low |
| Probability | High |
| Mitigation Plan | - Development of a web app to carry out tests and obtain a demonstration, and if necessary for integrations in future projects; |

Table 3.4: Risk 4 - Integration on the Wit Buzz Platform

| | |
|------------------------|--|
| ID | R05 |
| Title | API updates |
| Description | Changes in APIs (External services) can compromise the application |
| Verified | Yes |
| Mitigated | Yes |
| Impact | High |
| Probability | Low |
| Mitigation Plan | - Check frequently for possible API updates; |

Table 3.5: Risk 5 - API updates

| | |
|------------------------|--|
| ID | R06 |
| Title | The COVID-19 pandemic |
| Description | The appearance of COVID-19 led to a state of emergency in the country, which forced all Wit employees to leave the office and work remotely at home for three months. There may be consequences, such as deadlines, increased difficulties, and poor health. |
| Verified | Yes |
| Mitigated | Yes |
| Impact | High |
| Probability | High |
| Mitigation Plan | - In addition to the dailys between the intern, the tutor, and the advisor, it was essential to always be available for any necessary contact. Wit provided all support, from technical support to health support. |

Table 3.6: Risk 6 - The COVID-19 pandemic

3.3.2 Risks Metrics

To better understand the impact on the project, a risk analysis is made through a risk matrix. A risk matrix represents the probability of an unwanted event and a measure of the consequence of that event. [42]

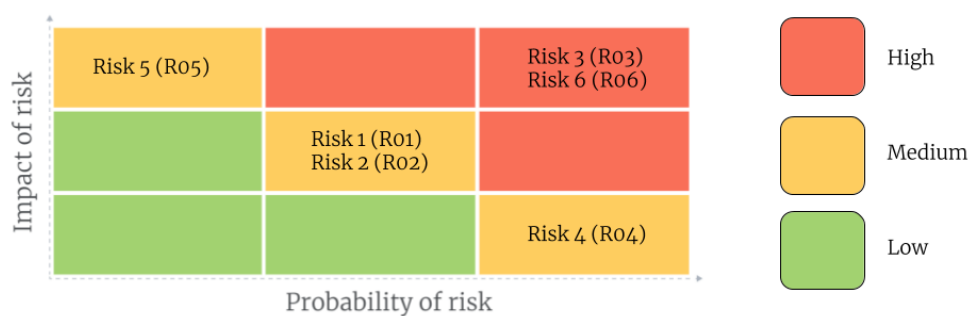


Figure 3.5: Risk Matrix

This page is intentionally left blank.

Chapter 4

Requirements Analysis

In this chapter, we will analyze the requirements needed for the software development. The requirements analysis process is presented as one of the most important steps in the development of a project, allowing to identify the intended objectives and also a way to evaluate the success of the project.

4.1 User Stories

User stories are a major artifact in agile software development to describe functionality from the perspective of the end-user. User stories are usually written as "As a <type of persona>, I want to <action> so that <outcome>". The *User Stories* are defined to formalize the functional requirements.

| ID | "As a" | "I want to" | "In order to" |
|------|------------------------------|--------------------------------------|---|
| US01 | User with iOS device | use the iOS Wallet | store my loyalty cards and coupons, for convenience purposes |
| US02 | User with iOS/Android device | receive updates when pass is changed | be able to use it in campaigns and/or events |
| US03 | User with Android device | use the android wallet (Google Pay) | store my loyalty cards and coupons, for convenience purposes |
| US04 | Brand | use the Management API | create Loyalty and coupons campaigns that links to my existing Loyalty and Coupons platforms, so I can grow my business |
| US05 | Brand | use the Loyalty App Platform | visualize all the campaigns ever created |
| US06 | Brand | use the Loyalty App Platform | visualize all the statistical flow data reports |
| US07 | Brand | use the Loyalty App Platform | insert new customers |
| US08 | Brand | use the Loyalty App Platform | update existing customers |
| US09 | Brand | use the Loyalty App Platform | send messages to existing customers with new campaigns and when they are updated. |

Table 4.2: User stories

4.2 Functional Requirements

Functional requirements are statements of how the system should react to specific inputs and how they should behave in certain situations. These specify a function that the system must be able to perform. In this subsection, a list of functional requirements is presented, as well as their prioritization.

To prioritize functional requirements, we used the **MoSCoW** [41] method, a prioritization technique that assesses the importance and/or urgency in solving a task, assigning one of four categories:

- **Must Have:** Represent non-negotiable product needs that are mandatory for the team

- **Should Have:** Represent important initiatives that are not vital, but add significant value
- **Could Have:** Nice to have initiatives that will have a small impact if left out
- **Will Not Have:** Represent initiatives that are not a priority for this specific time frame

| FR ID | Description | Priority |
|-------|--|-------------|
| FR01 | User regist a pass for updates | Must Have |
| FR02 | Unregist a pass | Must Have |
| FR03 | Get the serial numbers for passes associated with a device | Must Have |
| FR04 | Get the latest version of a pass | Must Have |
| FR05 | Send push notifications to device | Must Have |
| FR06 | Create a virtual pass | Must Have |
| FR07 | Download a virtual pass | Must Have |
| FR08 | Update existing virtual pass | Must Have |
| FR09 | Brand authentication on the Loyalty App | Must Have |
| FR10 | Create campaigns created by the respective authenticated brand | Must Have |
| FR11 | List all campaigns created by the respective authenticated brand | Must Have |
| FR12 | Use of a reporting framework | Should Have |
| FR15 | Show main page and main menu | Must Have |
| FR16 | Upload customers list | Must Have |
| FR18 | Send a text message to the clients | Must Have |
| FR19 | Report on the number of customers using iOS and/or Android devices | Must Have |
| FR20 | Report of the top five campaigns with the most downloads by the client | Must Have |
| FR21 | Report on the top five clients with the most points per campaign | Must Have |
| FR22 | Report on the number of pass updates, by the campaign, and by date | Must Have |
| FR23 | Report on the comparison between the number of downloads made by customers and the SMS sent to customers, by the campaign, and by date | Must Have |

Table 4.4: Functional Requirements

4.2.1 Agile Boards

As mentioned in the previous chapter, agile boards were created to better manage the tasks to be carried out. Agile Boards are for teams that plan sprint tasks.

The represented agile boards are divided into five columns, respectively:

- **In progress:** Tasks that are under development.
- **Implemented:** Tasks already implemented
- **Ready for QA:** Tasks waiting to be tested
- **In Dev QA:** Tasks being tested to ensure quality.
- **Done:** Tasks that are completed by the end of a sprint.

In figure 4.1 we can see the agile board corresponding to the first user story (US01): As a user with iOS device, I want to use the iOS Wallet to store my loyalty cards and coupons, for convenience purposes.

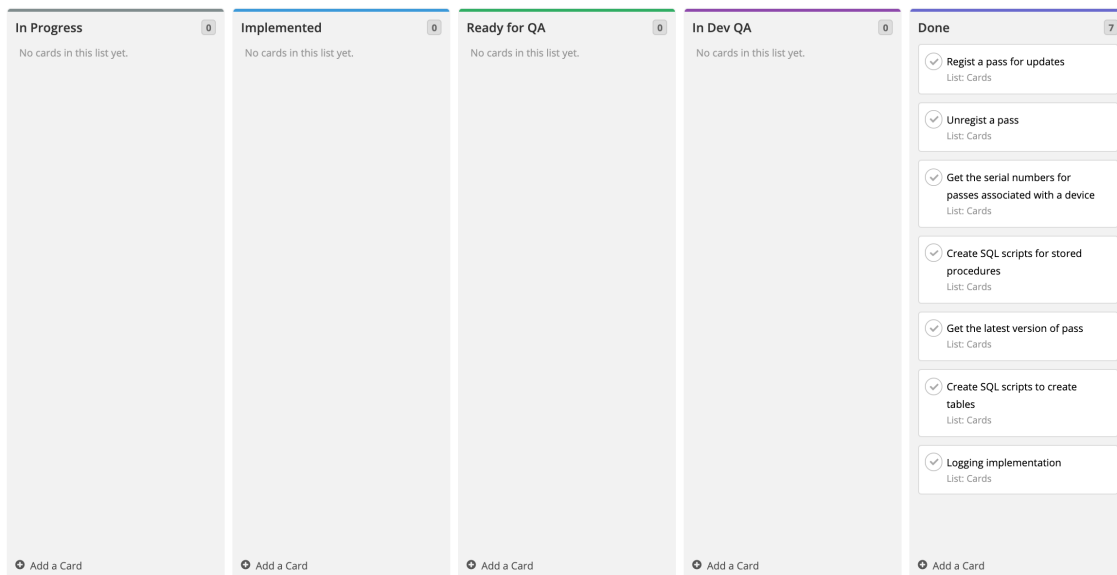


Figure 4.1: Agile Board for User Story US01

In figure 4.2 we can see the agile board corresponding to the second user story (US02): As a user with iOS device, I want receive updates when pass is changed.

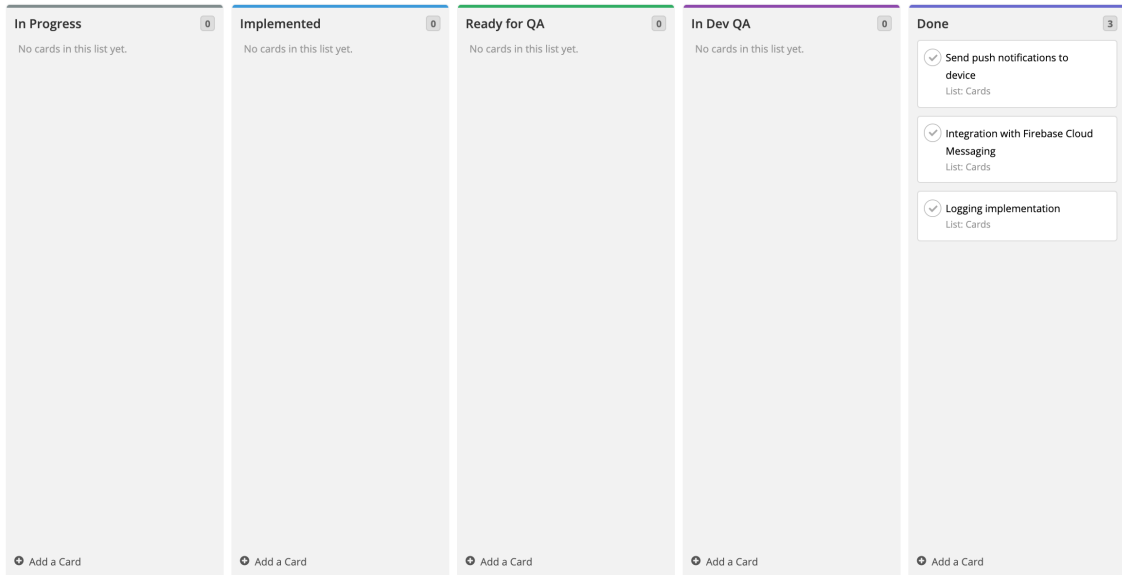


Figure 4.2: Agile Board for User Story US02

In figure 4.3 we can see the agile board corresponding to the third user story (US03): As a brand, I want to use the WIT Buzz Platform to create Loyalty and coupons campaigns that link to my existing Loyalty and Coupons plarforms, so I can grow my business.

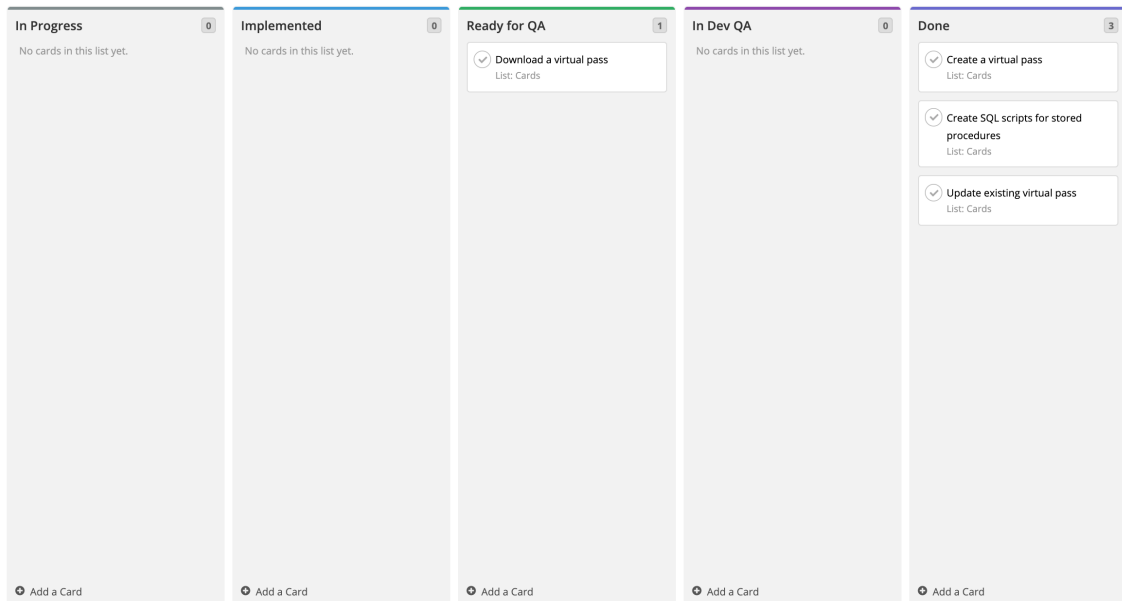


Figure 4.3: Agile Board for User Story US03

4.3 Non-Functional Requirements

The following section presents the quality attributes, have been identified two: scalability and availability.

| ID | Quality Attribute | Description | Priority |
|-------|-------------------|--|-----------|
| NFR01 | Scalability | Server maintains responsiveness under heavy usage | Must Have |
| NFR02 | Scalability | Server supports a large amount of users uploading data | Must Have |
| NFR03 | Availability | System has an acceptable uptime | Must Have |
| NFR04 | Security | Apple certificates and Google API Key encrypted | Must Have |

Table 4.6: Non-Functional Requirements

In terms of **scalability**, the system must be able to handle multiple requests at the same time, even if the number of users and, consequently, the number of requests increases without impairing performance. Also, the platform is intended to be able to scale horizontally. In the attribute of **availability** quality, the system must ensure a minimum operational level of performance of 99.5%. [22] If users are unable to access the service and interact over time, it will be unavailable or suffer downtime.

The tables below shows more specifically how the system should react to the previously mentioned non-functional requirements.

| | |
|-------------------------|---|
| ID | NFR01 |
| Stimulus Source | Users |
| Stimulus | High number of requests |
| Artifact | Server |
| Response | Server responds to requests successfully |
| Measure Response | All requests are met with responses. Response times must less than 3 seconds. |

| | |
|-------------------------|------------------------------|
| ID | NFR02 |
| Stimulus Source | Users |
| Stimulus | Users sends data from device |
| Artifact | Database |
| Response | Server stores data |
| Measure Response | User related data is stored. |

| | |
|-------------------------|---|
| ID | NFR03 |
| Stimulus Source | Server or Database |
| Stimulus | Crash or Malfunction |
| Artifact | System |
| Response | - Informs system owners of a crash. - System is unavailable. |
| Measure Response | System repairs respect an uptime of 99.5% (downtime lower than 3.6 hours/month) |

| | |
|-------------------------|--|
| ID | NFR04 |
| Stimulus Source | System |
| Stimulus | System creates Google passes and Apple passes |
| Artifact | System |
| Response | - Inform system of successful or unsuccessful operation. |
| Measure Response | Transfers of passes data between system components are encrypted |

This page is intentionally left blank.

Chapter 5

Software Architecture

In this chapter, an analysis of the software architecture, which is a crucial phase before any development, will be presented. The architecture will define all the modules used to build a system and how they relate and facilitate decision making and change management.

Before building software architecture, it's crucial to choose the right technologies for development.

5.1 Final Architecture High-Level Design

Figure 5.1 represents all interactions between system components. We can see three different modules, respectively, Management API, Loyalty & Coupons Backend API, and Frontend API.

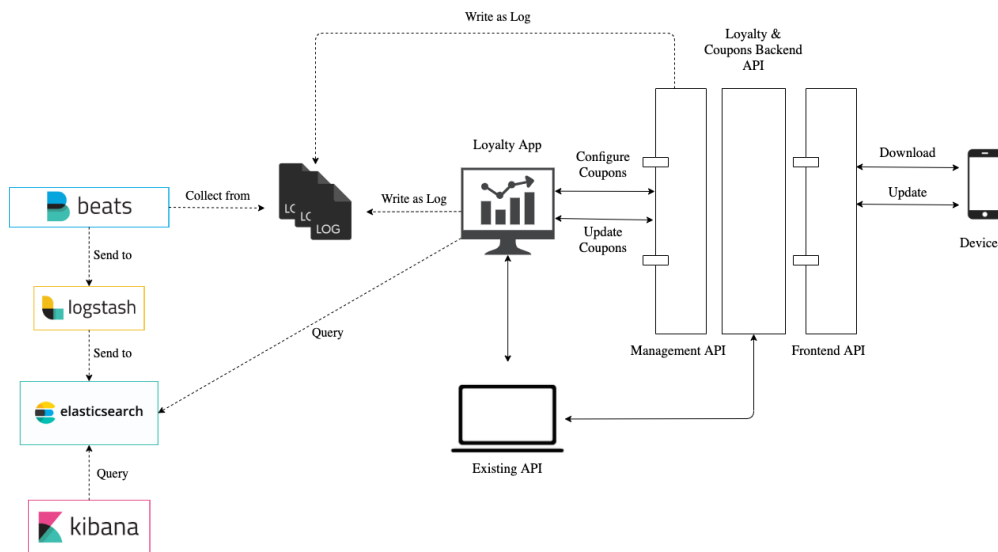


Figure 5.1: System architecture High-Level Design (HLD)

The following subsections describe the structure of the components.

5.1.1 Loyalty & Coupons Backend

In this section, we will describe an analysis of Apple's existing architecture for the upgrade of passes by communicating with the built-in Wallet App. The main features will be analyzed to differentiate what already exists and what has implemented.

Pass updates are an integral part of the Application-to-Person (A2P) campaign strategy and allow the user to take real-world actions. It's possible to update a pass when something in the real world changes, such as the date of an event, the delay of a flight, or even a brand promotion [39].

Overview of the Communication

Updating a pass requires interaction between the smartphone, Apple servers, and the API implemented by the author. To make this possible, must follow the documentation provided by Apple by following steps. 5.2.

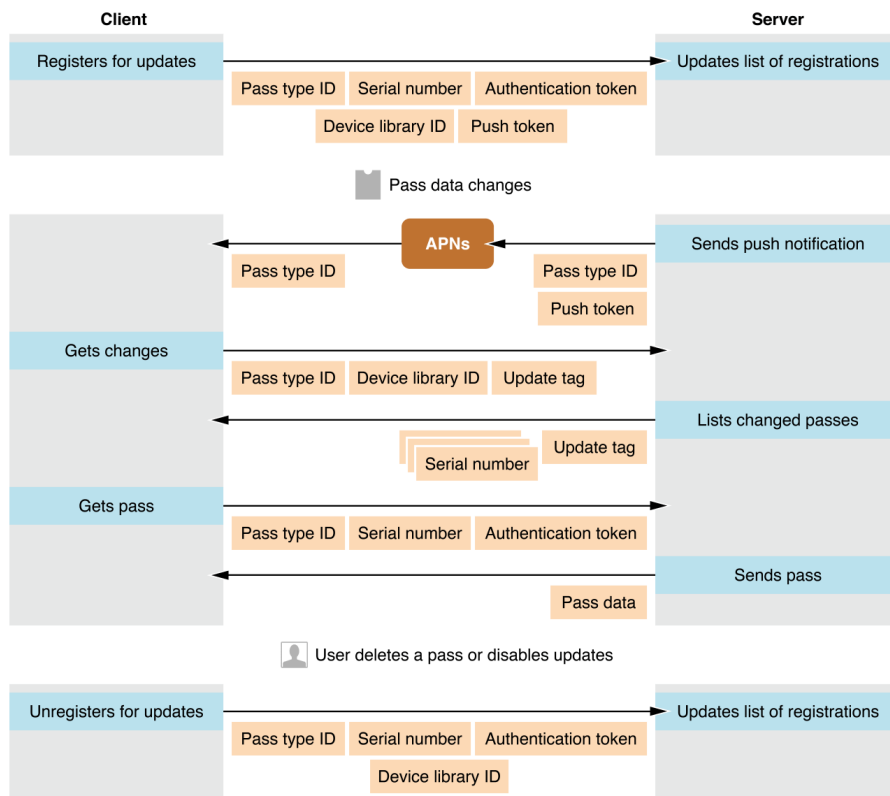


Figure 5.2: Interaction between the client and the server [39]

Table 5.1: Information for update process

[39]

| Information | Source | Purpose |
|-------------------------------------|--------------------------|--|
| Web service URL | Defined in the pass | Tells Wallet how to contact the web server |
| Pass Type Identifier, Serial number | Defined both in the pass | Together, uniquely identify the pass |
| Device library identifier | Defined in the device | Identifies the device and authorizes requests |
| Authentication token | Defined in the pass | Authorizes requests |
| Push token | Defined in the device | Allows the server to send push notifications to the device |
| Update tag | Server defines | Describes state |

Figure 5.3 represents a schema of how Apple Wallet interacts with the server and what was implemented in the backend.

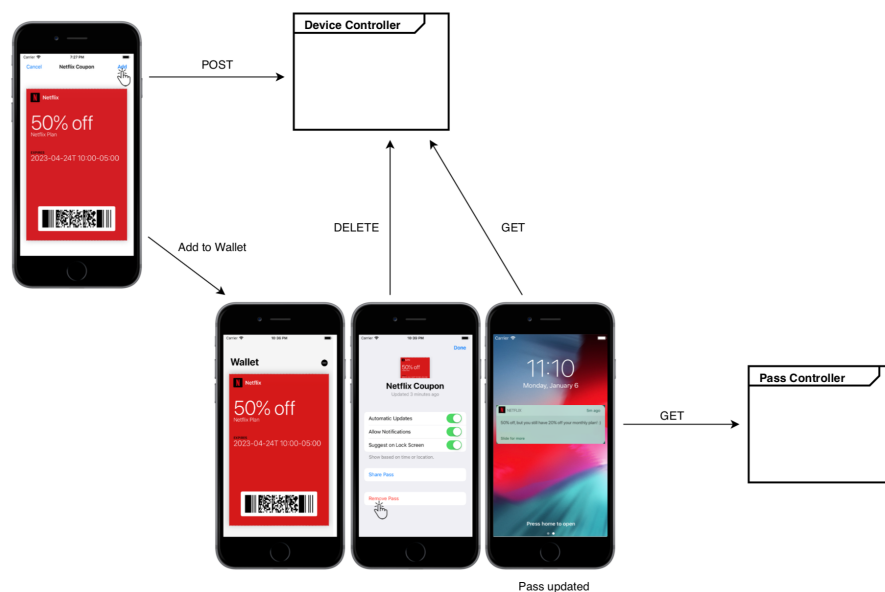


Figure 5.3: Loyalty & Coupons View

When the user receives and opens their pass through a message or email, it can be added to the Apple Wallet. After adding to Apple Wallet, it sends a POST request to the server to register the pass on the phone.

With the pass stored in Wallet, it can be removed, if the user removes it, i.e., the Wallet sends a DELETE request to the server to unregister the pass on the mobile phone and consequently delete the database record.

For the user to receive update notifications, the pass needs to be registered, and the notifications active. After a new update is detected, a push notification is sent, and the Wallet sends a GET request to the server to obtain the latest version of the pass.

The appendix 8 depicts the detailed description of the REST API implemented in this module [21].

Database

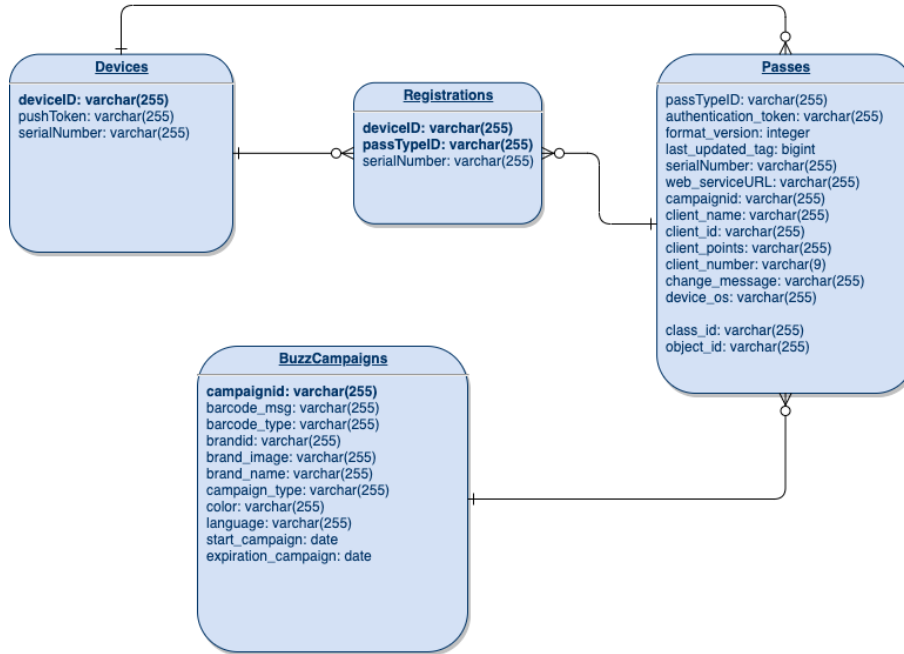


Figure 5.4: Database tables

There are two entities, *devices* and *passes*, and one relationship, *registrations*.

Device table: "Its device library identifier identifies a device, and it also has a push token" [39].

Passes table: "A pass is identified by a pass type ID and serial number. This table includes a last-update tag (such as a timestamp) for when the pass was last updated and typically includes whatever data needed to generate the actual pass" [39].

Registrations table: "A registration is a relationship between a device and a pass. It's needed to be able to look up information in both directions: to find the passes that a given device has registered for, and to find the devices that have registered for a given pass. Registration is a many-to-many relationship: a single device can register for updates to multiple passes, and a single pass can be registered by multiple devices" [39].

Buzz Campaigns table: This table stores the necessary data for all promotional campaigns created by the brands. A campaign can have multiple associated virtual passes and also various customers. But only one customer can have a virtual pass for their campaign.

Push Notifications

Apple Push Notification service (APNs) is a service that sends remote notifications to devices. Remote notifications allow posting information and data to devices. For your device to receive notifications, they must be enabled even when the app is not running.

Notifications serve to keep the user informed of new updates to the pass that are already stored in the Apple Wallet. As soon as the pass information is updated, a notification is sent to the device.

Figure 5.5 shows a delivery path scheme for remote notification.

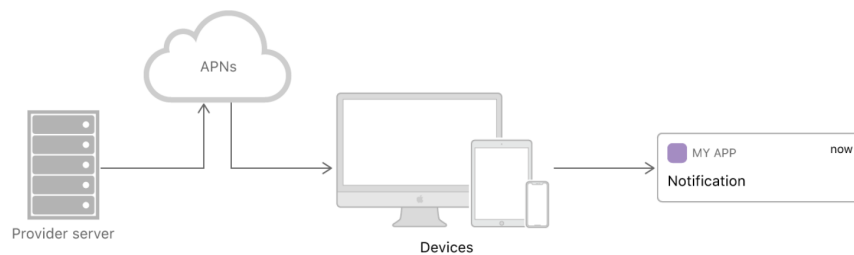


Figure 5.5: Apple Push Notifications Overview
[24]

Firestore Cloud Messaging

Firestore Cloud Messaging (FCM) is a **Google** solution that lets you send messages across platforms. With FCM, it is possible to send a notification to a device with information and data that is available for synchronization. [40]

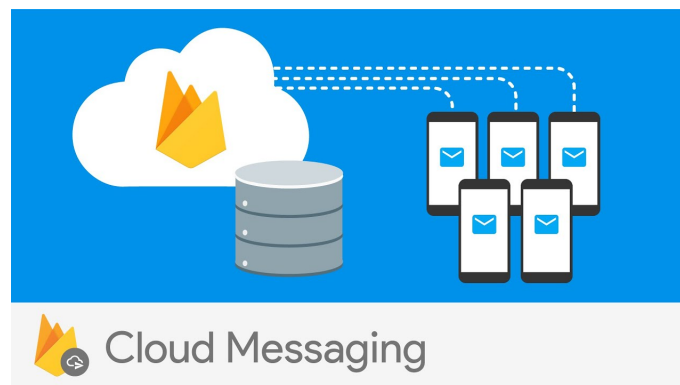


Figure 5.6: Firestore Cloud Messaging
[40]

It is possible to use FCM instead of APNs for server-to-client communication when integrating with Apple Wallet. As a backend implementation will be performed for both platforms (**GooglePay** and **Apple Wallet**), the use of FCM becomes more advantageous so that both components can use the same remote notification service.

Figure 5.7 is a schematic of how Apple Wallet interacts with the server when a pass is changed.

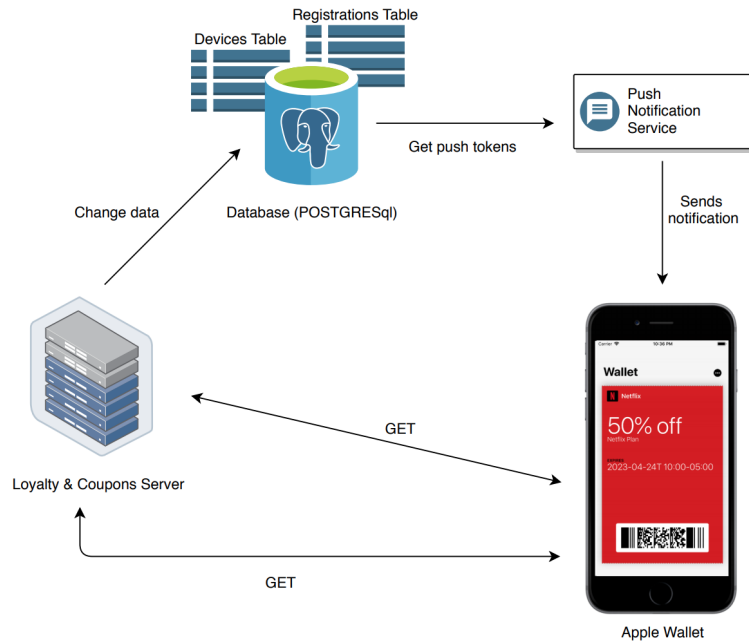


Figure 5.7: Push Notifications View

When a pass is changed, the user is alerted by push notifications. If a pass is not registered in Apple Wallet, remote notifications cannot be sent.

After you receive the notification, Apple Wallet sends a GET request to the pass serial number server that has changed at a specified time. The server sends the most recent update tag with the serial numbers stored in the database. As soon as Apple Wallet receives the information, it makes a GET request, again, from the server to obtain the latest version of that pass.

5.1.2 Management API Prototyping

This API component was implemented to communicate with WitBuzz and handle all Wallet pass configuration as configuration mechanisms for creating passes to obtain various available pass templates, creating a virtual pass that does not yet exist in the database, the coupon update (communication with the *WitBuzz Dashboard*) and the possibility of downloading a pass.

The Management API was implemented with the Spring Boot structure in Java language and consisted of two controllers, respectively *Passes Controller*, and *Apple Wallet Controller*. The *Apple Wallet Controller* is only intended for communication, as the name says, with the *Apple Wallet App*. This communication with the wallet requires an HTTPS connection to the server.

Passes Controller, is for communication with the *Google Pay API*. *Google Pay API* uses a concept of objects and classes, where an object digitally represents loyalty cards for a single user, as well as other different types of virtual passes; and a class allows you to manage common data for all users. That is, a loyalty card that a user may have on the *Google Pay App* is represented by a *LoyaltyObject* that refers to a *LoyaltyClass*.

Figure 5.8 represents a typical communication flow used to insert a class, save an object,

and update an object to a loyalty card.

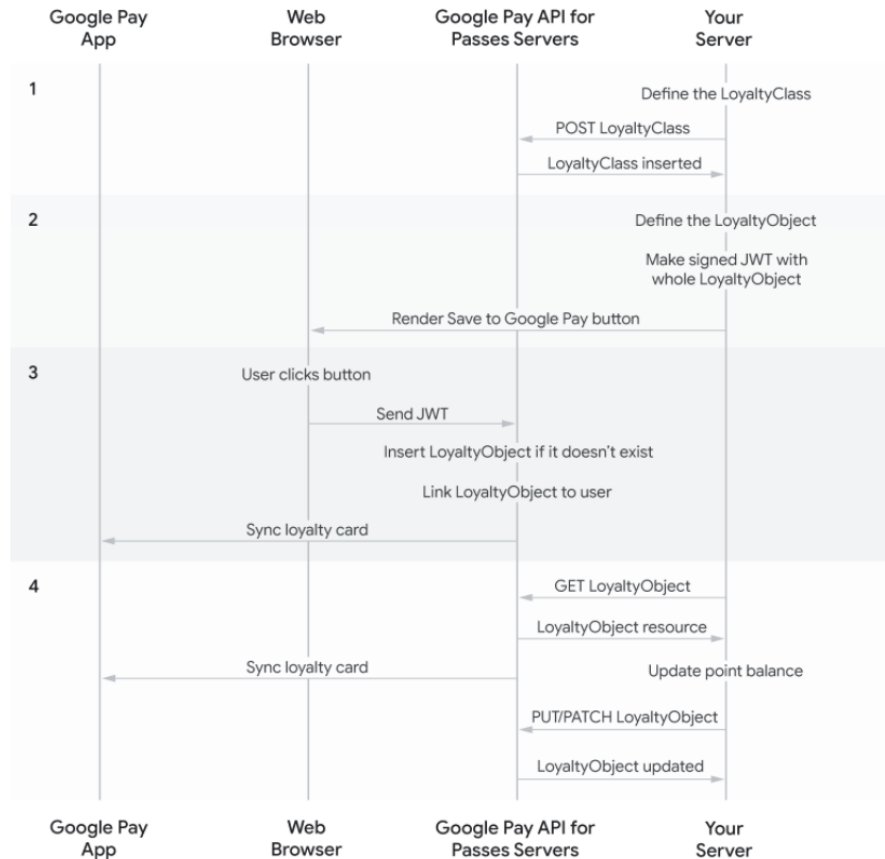


Figure 5.8: API flow diagram [30]

Unlike the Apple Wallet API, the Google Pay API for creating virtual passes generates a JSON Web Token (JWT) for an object of a specific class. After the customer receives an SMS to download his virtual pass, that JWT is then generated that will allow the user to store his loyalty card in the respective virtual wallet. The user is directed to a landing page that allows saving a `LoyaltyClass` object, and that object to be saved is rendered on the landing page based on the JWT.

To assist the management of virtual passes for Android, a platform, the Google Pay Merchant Center, was used to verify the information of the generated passes.

5.1.3 Frontend API Prototyping

This API component has implemented to communicate with its device for some of the features available, such as receiving a message/email campaign with the ability to download to the **Wallet App** and also receiving notifications of new pass updates.

5.2 Loyalty App

In this section, the description of the architecture of the Loyalty App Platform is made. The Loyalty App was developed with the objective of obtaining a reference integration that

shows that the platform works as intended. Additionally, it has the advantage of serving as a demonstration of the present stage. In the initial plan, as mentioned in section 3.2.2, the integration with the Wit Buzz Platform would be developed, a platform to be developed by Wit, as this integration was not possible, due to the large UX effort on the part of the intern, it was then decided to create the Loyalty App Platform that will replace the Wit Buzz Platform only for the modules needed in the campaign flow.

The Loyalty App is composed of a Back-end and a Front-end, which allows a brand to carry out various functionalities, such as viewing campaigns previously created, uploading a list of customers and their own information, and, consequently, sending SMS to your customers. It is also possible to view some analysis reports on the flow of campaigns.

5.2.1 System Context Diagram

A context diagram is a starting point for representing and documenting a software system that allows you to define expectations and obtain an overall picture of the system. In figure 5.9, we can see the context diagram of the Loyalty App Platform.

As previously seen in the representation of User Stories, there are two types of users, respectively, brands and their customers. A brand must authenticate itself in the Loyalty App and use its features for convenient purposes, such as managing its campaigns and its customers, through external services and platforms (Apple Wallet and Google Pay).

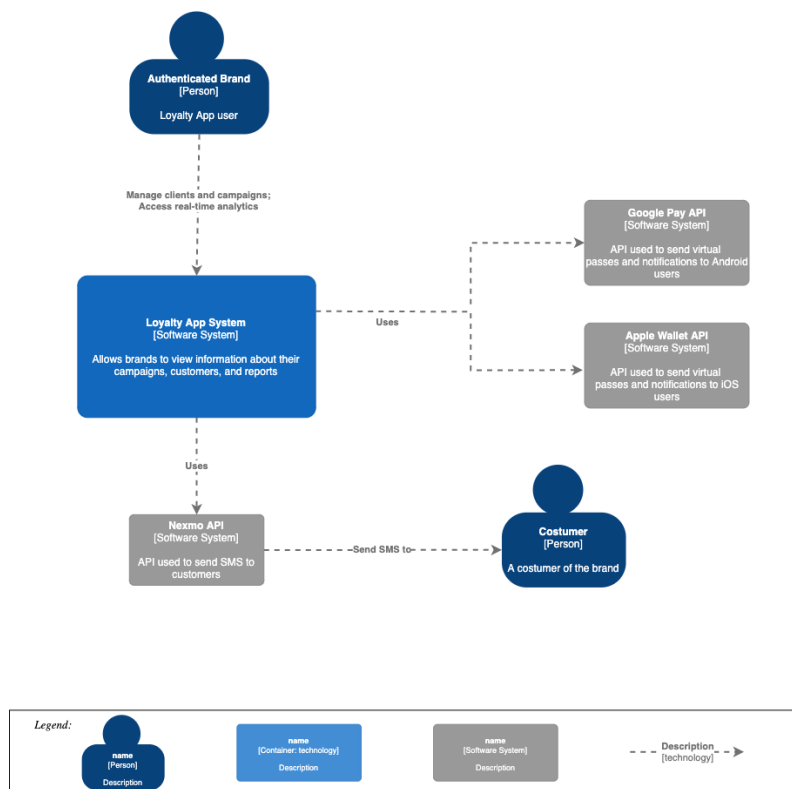


Figure 5.9: Context Diagram of Loyalty App

The context diagram is also made up of three external APIs, respectively:

- **Nexmo SMS API:** Nexmo is a paid service that integrates SMS and voice messages through a simple API.

- **Google Pay API:** The Google Pay for Passes API allows user interaction through loyalty cards, gift cards, offers, event tickets, flight boarding passes, and transit passes. It is used so that users can store their virtual passes on the Google Pay app. These saved items are stored as objects.
- **Apple Wallet API:** A REST service protocol is used to communicate with the server about pass changes and to look for newer versions of these when changed.

5.2.2 System Containers Diagram

After presenting the context diagram and understanding how the system can fit into the global IT environment, a container diagram is represented. A container is a separately executable or implementable element that can be used to execute code or store data. The container diagram shows the high level of the software architecture, where the main technologies are represented, and how the containers communicate with each other. In figure 5.10 shows the container diagram of the Loyalty App system.

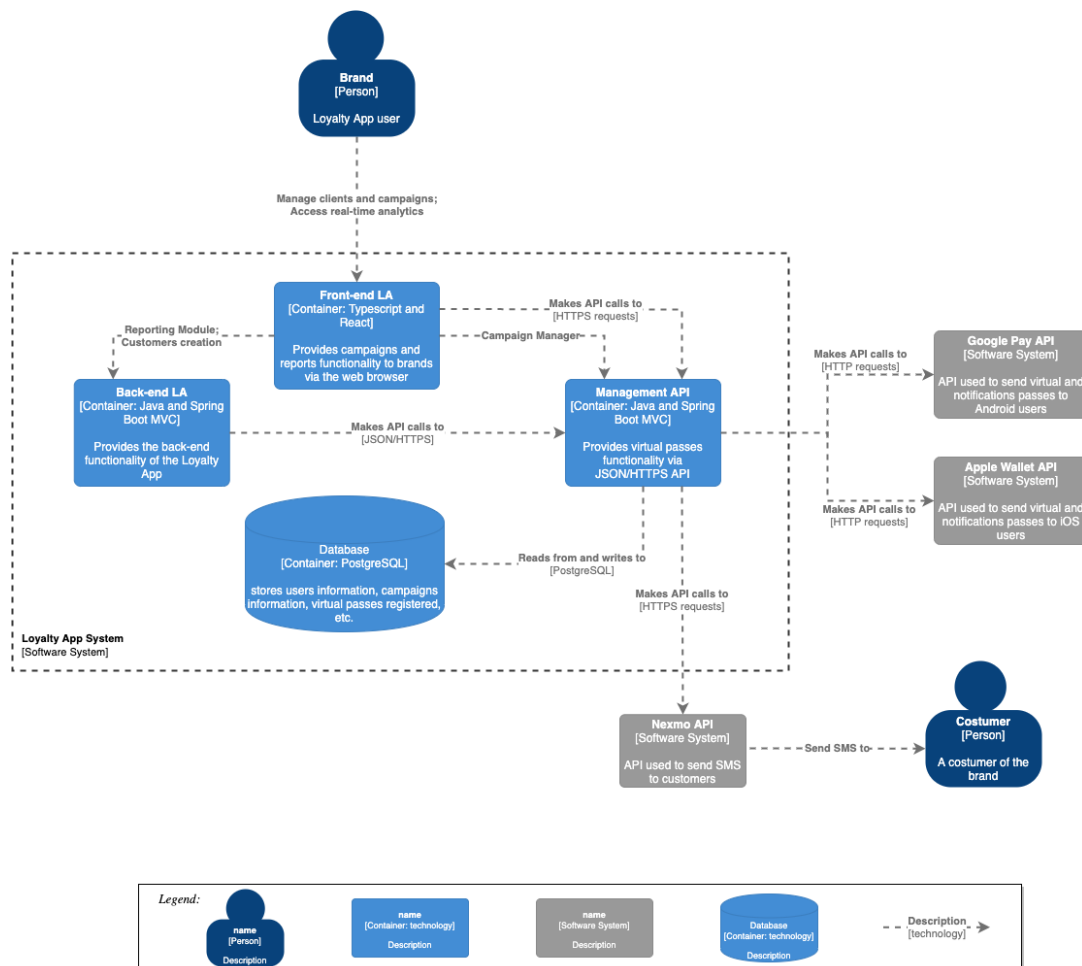


Figure 5.10: Containers Diagram of Loyalty App

The system was decomposed into four containers, each in charge of a specific set of functionalities.

- **Front-end Loyalty App:** This container will provide campaign features such as viewing these, registering new customers, and viewing statistical reports through the

web browser.

- **Back-end Loyalty App:** This container describes the back-end functionality of the Loyalty App reporting module.
- **Management API:** This container will provide all the functionality for the flow of virtual passes between customers, from features such as updates and downloads on passes, among others. In section X, this container is presented in more detail.
- **Database:** This container, as the name implies, is the database for storing all the essential content, such as brand information, and consequently your campaigns and your customers. Some data from customer devices entered into the system is also stored.

The database was used to store all information about campaigns, customers, records of virtual passes, and the virtual passes of each customer. PostgreSQL was used, as previously said, it is an object-relational database management system (ORDBMS) with an emphasis on extensibility and compliance standards. PostgreSQL can handle workloads ranging from small single-machine applications to large applications, where it is used simultaneously by multiple users.

The reporting module consists of Elasticsearch Logstash Kibana (ELK) and Filebeat, which in turn will obtain data information through log files generated by the management API and the back-end loyalty app. The logs are generated as the functionalities are released and, consequently, the filebeat collects the information and sends it to the logstash, and it sends it to elasticsearch. Through DSL Queries, it is possible to obtain an answer to the desired analyzes. In the next section, it is possible to have more information about the reporting module.

5.2.3 Reporting Module

To develop the reporting module, ELK Stack was used. ELK is an acronym that represents three open-source projects, respectively, Elasticsearch, Logstash, and Kibana.

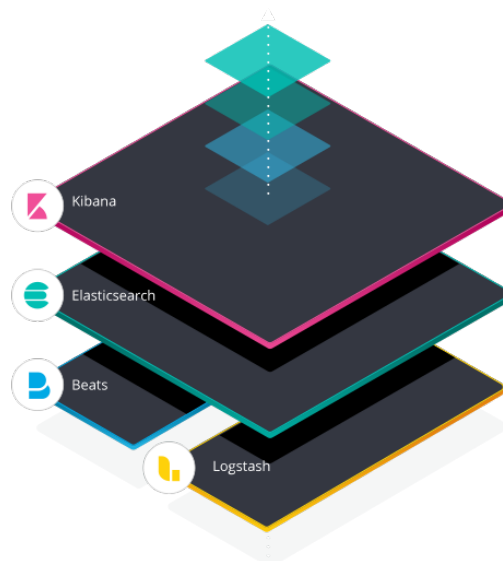


Figure 5.11: ELK Representation [34]

Elasticsearch is a distributed open-source search and analysis engine. It stands out for its simple REST APIs, speed, and scalability.

Logstash is a server data processing pipeline that consumes data from several sources simultaneously and transforms and sends them to a "stash," in this case, Elasticsearch. The logstash consumes, transforms, and sends data dynamically, regardless of format or complexity.

The Logstash event processing pipeline goes through three phases, respectively, inputs, filters, and outputs. Inputs will generate events; some can be through files, syslogs, networks, and beats. Filters change events, some of which include grok, mutate, drop, clone, GeoIP. Outputs send events elsewhere, some of the most commonly used ones include elasticsearch, file, graphite, and statsd.

Kibana is an open-source front-end application that allows users to search and view their data with graphs and tables in Elasticsearch, among other features. The Kibana dashboard provides real-time analytical visualizations on large volumes of data in support of use cases, such as [34]:

- Log and log analysis
- Infrastructure metrics and container monitoring
- Application performance monitoring (APM)
- Analysis and visualization of geospatial data
- Security analysis
- Business analyst

Filebeat is a lightweight sender for collecting and sending log files. The filebeat monitors the specified log files, collect the log events and sends them to Elasticsearch or Logstash for indexing. In figure 5.12, it is possible to observe how Filebeat works for the development of the reports module.

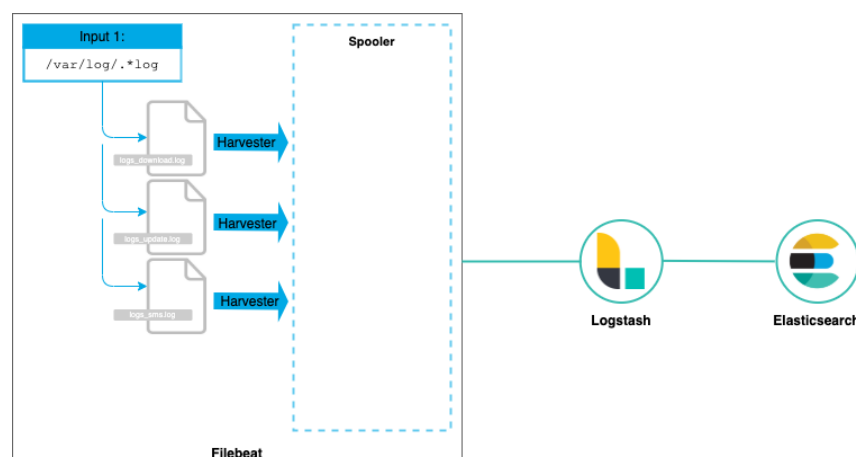


Figure 5.12: Filebeat Integration

Filebeat consists of two main components, respectively, inputs and harvesters. They work together to complete files and send event data to the specified output. Inputs are responsible for managing harvesters and finding sources for reading. For this module, the input type is a log, so the input will find all files in the unit that correspond to the defined paths

and start a harvester for each file. Harvesters are responsible for reading the content of a single file, which is read and sent to the content for output.

This reporting module is implemented in the Loyalty App (Backend and Frontend) and also in the Management API, to be able to observe all the selected metrics on the flow of the modules previously developed, as well as:

- Number of customers using iOS and/or Android devices;
- The number of points for each customer per brand and campaign already created by the brand, selecting a reach of 5 customers;
- The five campaigns with the most downloads by customers;
- The number of pass updates per brand, with the option to select a campaign for that brand and a range of days;
- The comparison between the number of downloads made by customers and the SMS sent to customers, with the option also to select the campaign and an interval of days.

Chapter 6

Development and Final Product

This chapter represents the project developed during the internship period, thus showing the web application Loyalty App.

6.1 Web Application

This section presents the layouts of the Loyalty App developed with some features on the flow of campaigns between customers and the reporting module.

6.1.1 Authentication

As previously seen in the definition of User Stories, there are two types of users, the brand and its customers. The Loyalty App is aimed at brands so that they can manage their campaigns and their customers. As the Loyalty App was developed for a reference integration, a simple authentication was implemented where it is possible to access the list of registered trademarks by selecting one and thus using the web app and its features.

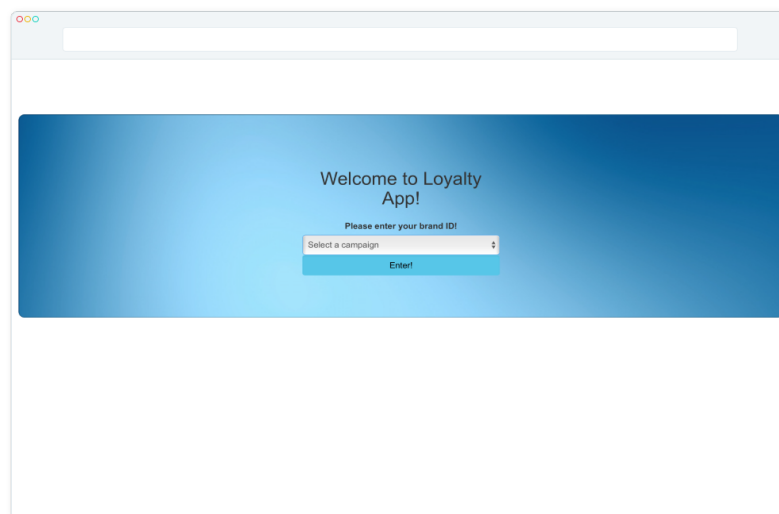


Figure 6.1: Authentication in Loyalty App

6.1.2 Home Page

After the brand authentication, the main page is displayed where we can access the desired features.

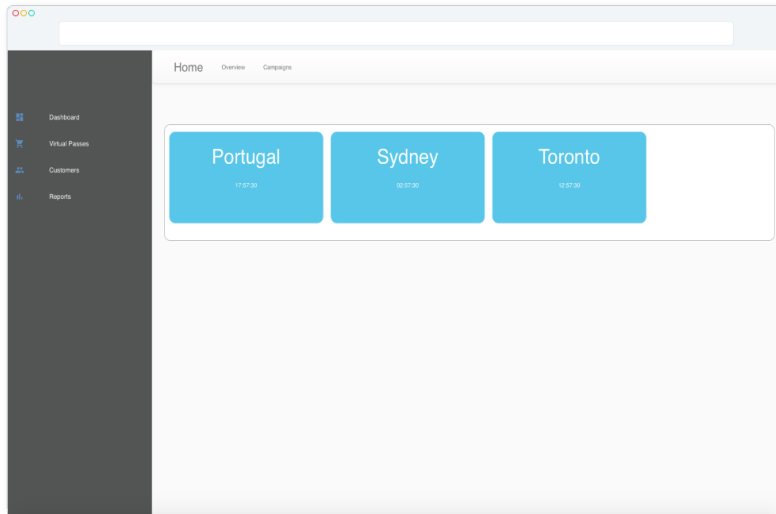


Figure 6.2: Home Page

6.1.3 List campaigns

One of the essential features is the list of campaigns previously created by the brand and stored in the database with the information of each one. The listing of the campaigns thus allows the brand to select the desired one to send to its customers.

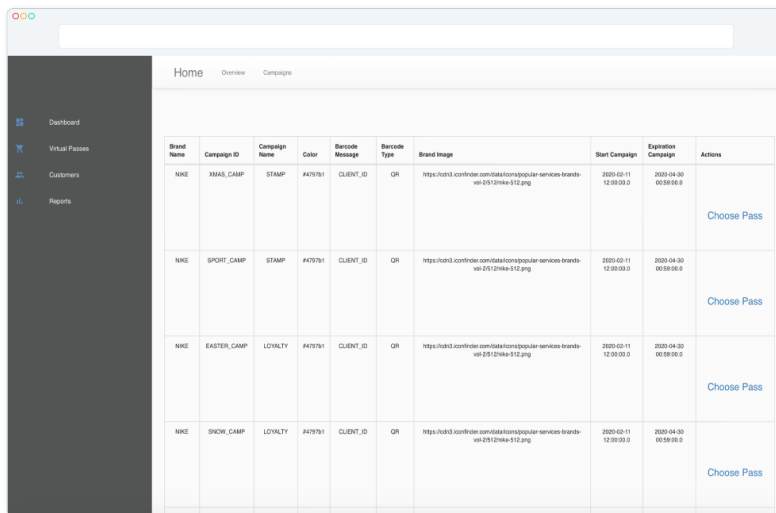


Figure 6.3: List campaigns

6.1.4 Insert/Update clients and Send SMS to Clients

As soon as a campaign is selected, the brand can then upload the list of its customers, in a file with the .csv format, with their information, such as name, ID, mobile phone number, points or stamps for the virtual pass and a message to send future notifications. If customers are already registered and stored in the database, this information is updated; otherwise new customers are inserted and consequently stored in the database. After uploading the list, it is then possible to send a message to customers with the necessary information to download the virtual pass in their mobile wallet applications.

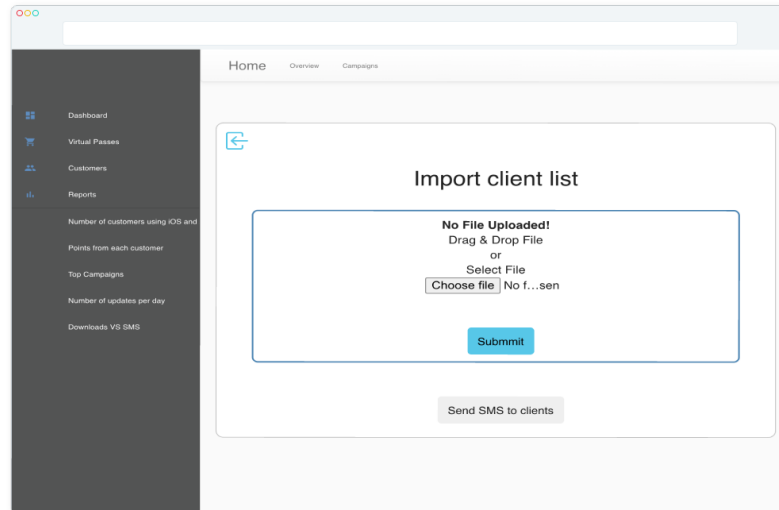


Figure 6.4: Insert/Update clients and Send SMS to Clients

6.1.5 Visualize number of customers using iOS and/or Android devices

With all the information recorded through log files, such as information about downloads, sent SMS, and updates, it is possible to obtain an analysis of these data. Through downloads made by customers after receiving their messages, we can obtain information on the number of customers using iOS and/or Android devices. Remember that for iOS devices, customers use the Apple Wallet app and for Android devices use the Google Pay App.

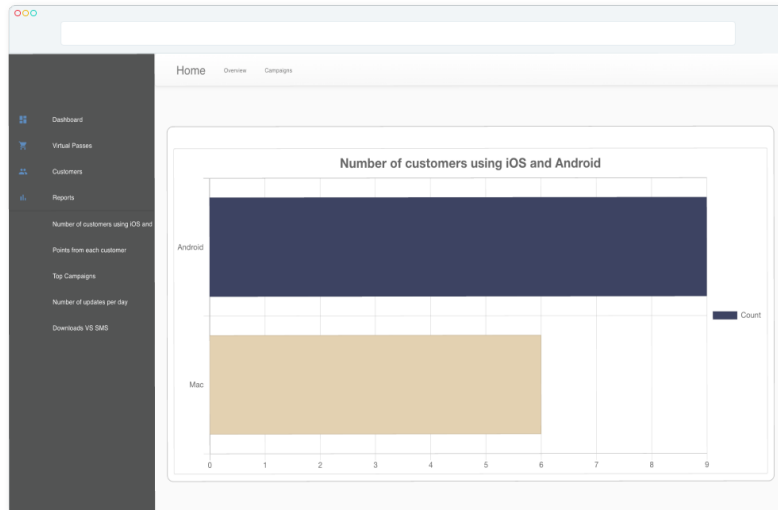


Figure 6.5: Number of customers using iOS and/or Android devices

6.1.6 Visualize number of points for each customer per brand and campaign

Through updates made by customers, it is possible to obtain information about the ten customers who have the most points in their virtual passes for a respective brand campaign.

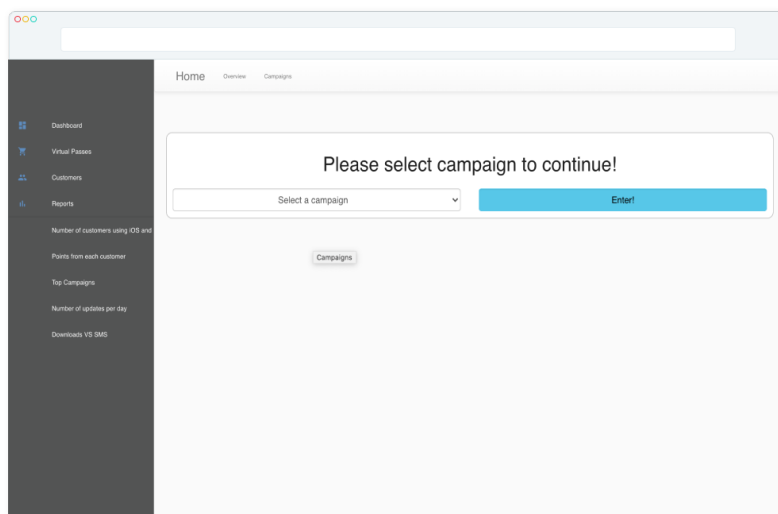


Figure 6.6: Number of points for each customer per brand and campaign

6.1.7 Visualize the five campaigns with the most downloads by customers

Through the downloads made by the customers for all campaigns created by the brand, it is possible to obtain the five promotional campaigns that stand out the most, being able to analyze the one that reaches positively in the customers. Campaigns can be created according to the brand you want at the time you want, and thus it is possible to know also when is the best time and what yields the most for your services.

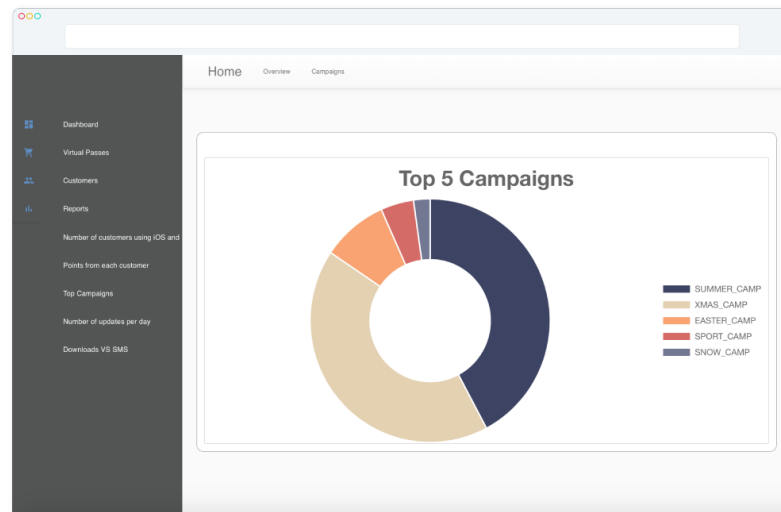


Figure 6.7: Top 5 campaigns with the most downloads

6.1.8 Visualize number of pass updates per brand

To view the number of updates made to customers' virtual passes, the update logs for the authenticated brand is analyzed. It is also possible to filter by campaign or for a selected period.

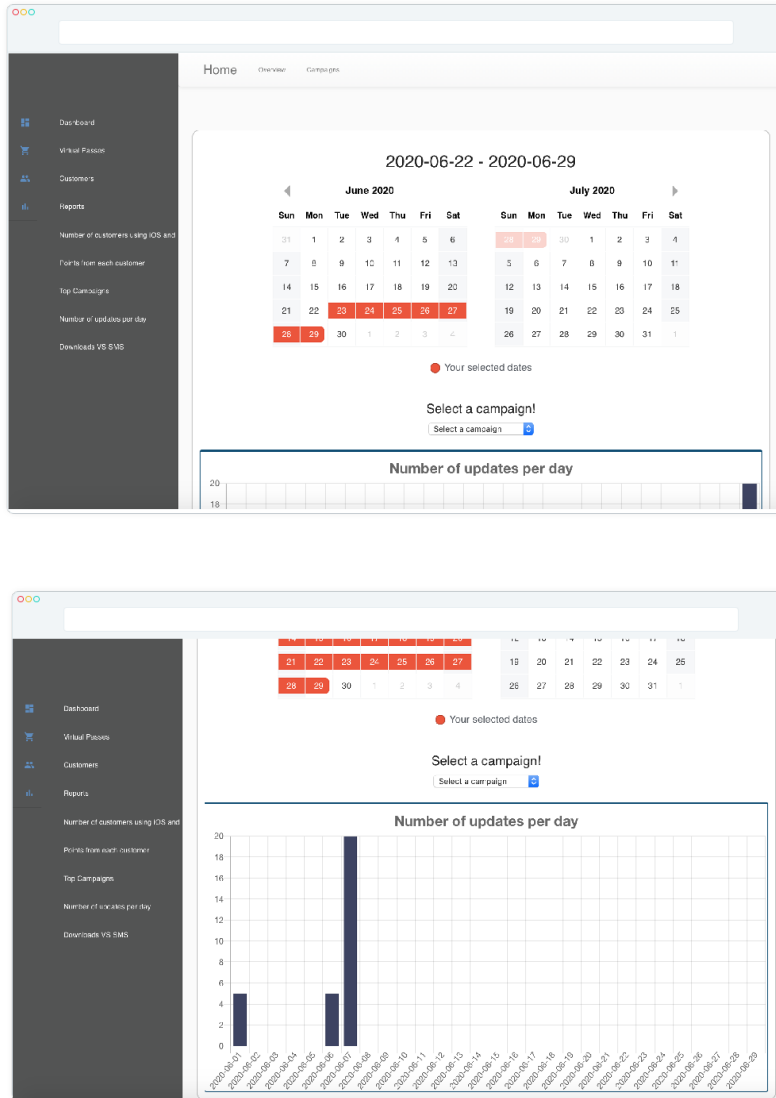


Figure 6.8: Number of pass updates per brand and per day/campaign

6.1.9 Visualize comparison between the number of downloads made by customers and the SMS sent to customers

To obtain an analysis of the number of SMS's sent and the number of downloads carried out, the information saved on these for the authenticated brand is considered. This analysis helps to understand the rejection value of customers about a pass, for example, after 100 messages have been sent to 100 customers, if there are only 50 downloads, it means that 50 customers have rejected that promotional pass. This analysis leads brands to be able to perceive that campaigns are more or less relevant to their customers. In this research, it is also possible to filter by campaign or for a selected period.

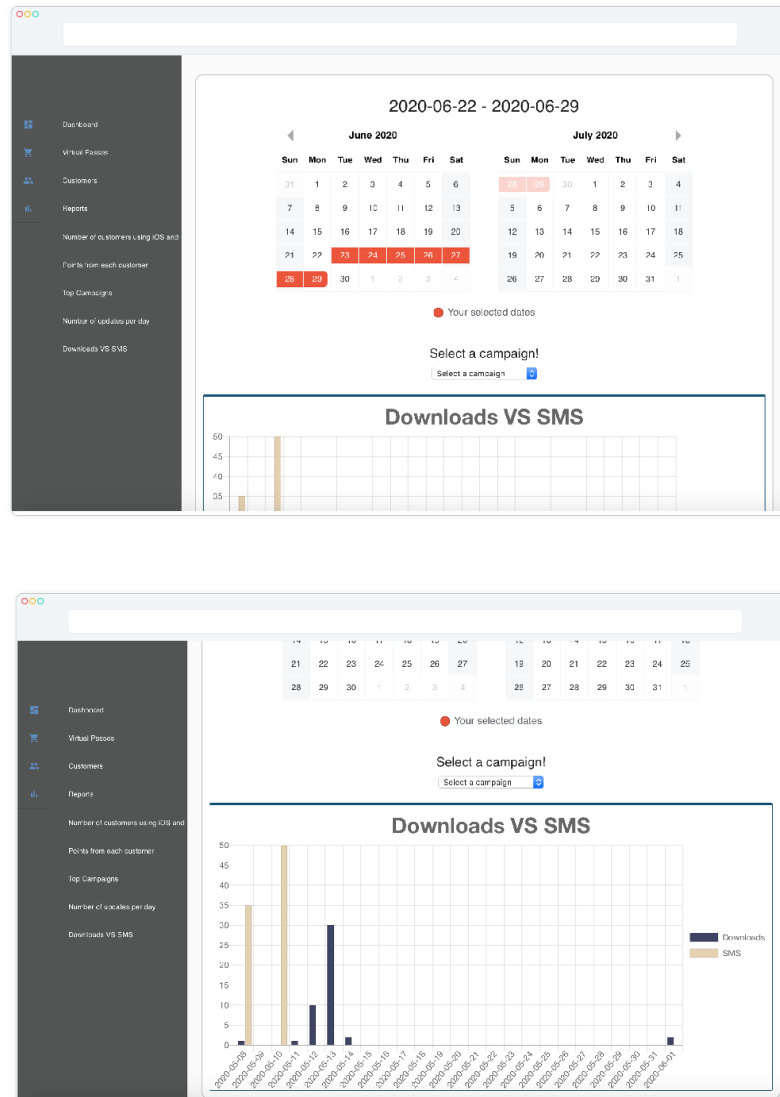


Figure 6.9: comparison between the number of downloads and SMS

6.2 Google Pay Interaction

In this section, we can see different loyalty passes generated for Android devices, which are eventually stored in the virtual wallet application, respectively, Google Pay App.

6.2.1 Stamp card for Android users

Stamp cards are loyalty cards that can be associated with any campaign for a brand, for example, and allow you to earn a profit after accumulating several stamps. In figure 6.10, we can see an example of a stamp card generated through the Management API developed in the stage, to store in the Google Pay App.

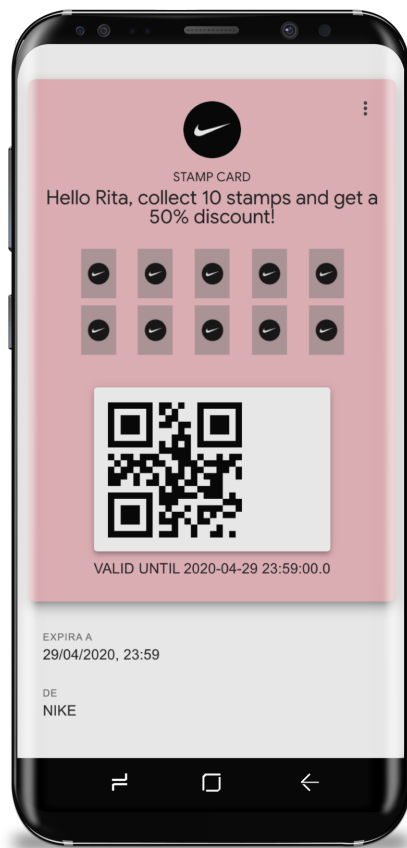


Figure 6.10: Example of stamp card for Android

6.3 Apple Wallet Interaction

In this section, we can see different loyalty passes generated for iOS devices, which are eventually stored in the virtual wallet application, respectively, Apple Wallet App.

6.3.1 Loyalty card for iOS users

Loyalty cards are loyalty cards that can be associated with any campaign for a brand, for example, and allow you to accumulate a maximum number of points to earn bonuses or prizes. In figure 6.11 we can see an example of a loyalty card generated through the Management API developed in the stage, to store in the Apple Wallet App.



Figure 6.11: Example of loyalty card for iOS

This page is intentionally left blank.

Chapter 7

Software Quality

The process of verification and validation of a developed product is one of the most important and fundamental steps in any software project. It is essential and expected that the final product would be delivered to the user without bugs and with the best possible User Experience (UX), with the goal of the customer being able to use it and be satisfied.

This chapter describes all the tests carried out on the final developed solution. The tests performed were divided into two sections, respectively, functional tests and non-functional tests.

7.1 Functional Testing

This section is for functional tests to validate and guarantee the quality of the software. Functional tests are a type of black-box test, focused only on the inputs and outputs of the software system without the internal knowledge of the implemented code.

7.1.1 API Tests

API is the link between different systems or layers of an application. Generally, applications consist of three layers, respectively, the service layer (API), the data layer, and a presentation layer (UI). The API test consists of making requests to a single or multiple API endpoints and validating your response, whether for performance, security, functional correction, or just status verification.

Testers are responsible for testing individual features or a set of features in a chain, and it is considered one of the most important tests during the development of a project. To perform API tests, it is necessary to send a specific call to the API, compare and record the results obtained, and expected. If an API does not work efficiently and effectively, it can never be adopted. Also, if this is interrupted because no errors were detected, there may be a risk of not only breaking down in a single application but also in a business process chain.

For this purpose, was used the Postman tool, an easy-to-use platform, with several integrations as support for Swagger formats and with execution, testing, documentation, and monitoring resources.

Table 7.2 shows the results of tests on the implemented features. The tests were executed during the development, and at the end of the implementation was made a final test. In appendix 8, it is possible to see the API tests in more detail.

| Controller | Total Tests | Failed | Pass |
|-------------------------|-------------|--------|------|
| Passe Controller | 30 | 0 | 30 |
| Apple Wallet Controller | 35 | 0 | 35 |
| Loyalty Controller | 32 | 0 | 32 |

Table 7.2: API Tests Result

7.1.2 Acceptance Tests

Acceptance tests are formal tests according to the user's needs, functional requirements, and business processes to determine whether or not the system meets the acceptance criteria. All acceptance tests described were performed by the intern.

Below we can see in Table 7.4, the necessary acceptable tests results.

| Controller | Total Tests | Failed | Pass |
|-----------------|-------------|--------|------|
| Management API | 9 | 0 | 9 |
| Web Application | 11 | 0 | 11 |

Table 7.4: Acceptance Tests Results

7.2 Non-Functional Testing

Non-functional tests are defined to verify the attributes of the system, such as performance, usability, reliability, among others. Non-functional tests help to reduce the risk and cost of production, improve knowledge of the product's behavior and technologies, and collect and produce metrics for research.

7.2.1 Performance Testing

The performance testing is a non-functional test performed to obtain responses in terms of system capacity and to determine system stability under various workloads. This test measures system quality attributes, such as scalability and reliability. This section presents a detailed analysis of the performance of some components of the system. To perform the performance tests were used, **Jmeter**, an application is open source software, 100% pure Java application. [3]

For each test, were recorded seven important values, respectively [4]:

Load Time: Measure the difference between the time when the request was sent and the time when the response has been fully received.

Latency: Measure the difference between the time when a request was sent and the time when a response has started to be received.

Average: The average elapsed time of a set of results.

Min: The lowest elapsed time for the samples with the same label.

Max: The longest elapsed time for the samples with the same label.

Throughput: Is measured in requests per second/minute/hour. The time is calculated from the start of the first sample to the end of the last sample. The formula is: Throughput = (number of requests)/(total time).

The following table describes functionalities selected to perform the server database performance tests, as well as the load time in milliseconds. The tests were performed for 60, 120, 180, 240, 300, and 360 campaigns; and 2000, 4000, 6000, 8000, and 10000 customers, both divided by 10 and 100 threads.

All of the following tests were performed with a VPN connection on the WIT Software Intranet.

Server (Virtual Machine):

<https://dev.wit-software.com/coupons/>
CentOS
2CPU
8GB RAM

Table 7.5 shows the performance results of the listing features required for campaign management. It is expected that the results for the listing of campaigns will be superior to that of brands since a brand can have 360 campaigns associated with it.

| # | Description | Threads | Average | Min | Max | Throughput |
|---|----------------|---------|---------|-----|-----|------------|
| 1 | List campaigns | 10 | 59 | 49 | 102 | 10.3/sec |
| | | 100 | 412 | 117 | 687 | 63.3/sec |
| 2 | List brands | 10 | 33 | 27 | 39 | 11.4/sec |
| | | 100 | 259 | 70 | 576 | 55.2/sec |

Table 7.5: Caption

Table 7.6 and table 7.7 shows the performance results of the functionality for sending SMS to customers.

A **POST** request was made to the <https://dev.wit-software.com/coupons/v1/send-message/brandID/campaignID> URL, which in turn sends a request to the previously mentioned Nexmo services, to the <https://rest.nexmo.com/sms/json> URL with the respective information.

| # | Description | Number of clients | Threads | Load Time (ms) | Latency |
|-------|---------------|-------------------|---------|----------------|---------|
| 1 | Send Messages | 2000 | 10 | 192 | 191 |
| | | | 100 | 292 | 292 |
| | | 4000 | 10 | 392 | 392 |
| | | | 100 | 434 | 434 |
| | | 6000 | 10 | 628 | 628 |
| | | | 100 | 1035 | 1035 |
| | | 8000 | 10 | 873 | 873 |
| | | | 100 | 700 | 700 |
| 10000 | 10 | 1082 | 1082 | | |
| | 100 | 637 | 637 | | |

Table 7.6: View Results SMS Tree

| # | Description | Number of clients | Threads | Average (ms) | Min | Max | Throughput |
|-------|---------------|-------------------|---------|--------------|----------|------|------------|
| 1 | Send Messages | 2000 | 10 | 192 | 192 | 192 | 5.2/sec |
| | | | 100 | 292 | 292 | 292 | 3.4/sec |
| | | 4000 | 10 | 198 | 84 | 392 | 4.5/min |
| | | | 100 | 434 | 434 | 434 | 2.3/sec |
| | | 6000 | 10 | 428 | 428 | 428 | 2.3/sec |
| | | | 100 | 1035 | 1035 | 1035 | 58.0/min |
| | | 8000 | 10 | 873 | 873 | 873 | 1.1/sec |
| | | | 100 | 700 | 700 | 700 | 1.4/sec |
| 10000 | 10 | 1082 | 1082 | 1082 | 55.5/min | | |
| | 100 | 637 | 637 | 637 | 1.6/sec | | |

Table 7.7: SMS Summary Report

A **POST** request was made to the <https://dev.wit-software.com/coupons/v1/templates/brandID/campaignID/typePass> URL, to create new campaigns. The tests for the creation of campaigns consisted of the creation of 600 to 3600 requests divided by 10 threads, executed individually.

A **PUT** request was made to the <https://dev.wit-software.com/coupons/v1/templates/brandID/campaignID> URL, to create new customers, and/or update customer data already registered in the database.

| # | Description | Number of campaigns/clients | Threads | Load (ms) | Time | Latency | Size in Bytes |
|---|---|-----------------------------|---------|-----------|------|---------|---------------|
| 1 | Insert campaigns in database | 60 | 10 | 384062 | | 380679 | 8280 |
| | | 120 | 10 | 474334 | | 474037 | 45405 |
| | | 180 | 10 | 513175 | | 510401 | 63107 |
| | | 240 | 10 | 701209 | | 691149 | 80810 |
| | | 300 | 10 | 814066 | | 775069 | 98512 |
| | | 360 | 10 | 1009910 | | 999423 | 116215 |
| 2 | Upload the list of clients to insert and/or update them in the database | 2000 | 10 | 27692 | | 27692 | 413165 |
| | | | 100 | 27773 | | 27773 | 413165 |
| | | 4000 | 10 | 88709 | | 88686 | 826247 |
| | | | 100 | 106342 | | 106341 | 826247 |
| | | 6000 | 10 | 189341 | | 189341 | 1239229 |
| | | | 100 | 189862 | | 189862 | 1239229 |
| | | 8000 | 10 | 295985 | | 295984 | 1652619 |
| | | | 100 | 296680 | | 296677 | 1652619 |
| | | 10000 | 10 | 452494 | | 452494 | 2067310 |
| | | | 100 | 444355 | | 444355 | 2067310 |

Table 7.8: View results campaigns and clients management tree

Analyzing the results in table 7.8, the functionality with the longest running time is the insertion of new campaigns, with 909910 milliseconds in just 360 simultaneous connections. This result was already expected since the amount of information is higher compared to that of customers.

The response time always will be equal or superior to latency and the larger data is, the larger difference between response time and latency will be.

When creating campaigns, a bottleneck was detected in the first test for 100 threads in 60 campaigns. A bottleneck appears in any system resource, such as hardware, software, or bandwidth, that places defining limits on data flow or processing speed. Bottlenecks affect some quality attributes, such as performance and scalability by limiting the amount of data throughput or restricting the number of application connections. These problems can occur at all levels of the system architecture, including the network layer, the Web server, the application server, and the database server.

| # | Description | Number of campaigns/clients | Threads | Average (ms) | Min | Max |
|---|---|-----------------------------|---------|--------------|--------|--------|
| 1 | Upload the list of clients to insert and/or update them in the database | 2000 | 10 | 27692 | 27692 | 27692 |
| | | | 100 | 27773 | 27773 | 27773 |
| | | 4000 | 10 | 88709 | 88686 | 826247 |
| | | | 100 | 106342 | 106342 | 106342 |
| | | 6000 | 10 | 189341 | 189341 | 189341 |
| | | | 100 | 189862 | 189862 | 189862 |
| | | 8000 | 10 | 295985 | 295985 | 295985 |
| | | | 100 | 296680 | 296680 | 296680 |
| | | 10000 | 10 | 452494 | 452494 | 452494 |
| | | | 100 | 444355 | 444355 | 444355 |

Table 7.9: Campaigns and clients management summary report

Chapter 8

Conclusion

The objective of this internship is to develop functionalities for the management and creation of virtual passes, for an A2P campaign platform, to guarantee a better experience in the use of these to users. Regarding the present semester, the functionalities developed were for the integration of a virtual wallet application on iOS, the Apple Wallet, which have been completed. For this to happen, it was necessary to research and study all the documentation provided by Apple and how an implementation would be made that could be integrated with the internal application.

The analysis of the state of the art, allowed understanding better the entire context of virtual portfolios, as well as virtual passes, and which functionalities exist in the market that could compete with the internship project. This analysis has become very important, as, in the second semester, the integration will be made with an existing application that allows using resources and adapting them to the context of the project.

After that, user stories were identified, which later helped to identify functional and non-functional requirements. The analysis of these requirements was quite significant in the implementation and representation of the project architecture.

Regarding the work of the first semester, it was successfully completed, although some tasks defined at the beginning of the internship have been modified. All specified requirements have been completed. Also, it was possible to obtain a final demonstration in real-time, with iOS devices.

The main personal goal is to take advantage of the internship offered by Wit Software and the Department of Informatics Engineering to enrich all knowledge of the development of a real project for the company and to complete the master's degree in informatics engineering specialized in software engineering with success.

In the second semester of this internship, the development of the intended prototypes was continued but now focused on features for integration in the Google Pay mobile application for Android devices. The same functionalities were developed for both apps (iOS and Android) but with different ways of development, ultimately obtaining what was intended.

After completing the tasks involved with virtual wallet applications, a web app, the Loyalty App, was developed, a reference integration with the aim of showing that all the tasks defined and completed work as intended. The Loyalty App, in addition, ended up having the advantage of serving as a demonstration for the project and may also be useful for future tests on other projects that may have been carried out.

In addition to the Loyalty App, including campaign features, it is also composed of a

reporting module developed with the purpose of visualizing interesting metrics related to the flow involved in the project and obtaining a conclusion on the relationship between customers and campaigns.

After the implementation of the project was completed, tests for the functional and non-functional requirements defined during the internship were successfully carried out, thus being able to validate the entire implementation.

During the internship, all the results acquired as a Wit Software employee are the result of a growth and learning process that had its lows as its lows, which led to studying and witnessing different approaches, such as the architectures involved and the technologies. In general, consequently, of all the work involved in this project, although some tasks were initially changed, we can proudly say that it was a mission accomplished!

During the internship, we ended up going through a challenging and complicated phase, the appearance of COVID-19 that led to a state of emergency worldwide, and that forced all Wit Software employees to leave their offices and work remotely for three months from home. Despite these difficult times, Wit Software was available in all kinds of help, from technical support to psychological support, always safeguarding the health and well-being of its employees.

Finally, the experience of working on this project with Raul and Pedro was undoubtedly gratifying, which allowed me to evolve personally as well as professionally, and that led to a small successful final prototype.

The A2P campaign platform is now a complete module, and although this is a successful job, there is always some additional work that can be done. As such, the project developed in the future may be integrated with Wit Buzz, the campaign management platform, and other features that have been developed by Wit Software.

References

- [1] 30 Global Mobile Payment Stats, Trends & Forecasts (2019 Update). <https://www.merchantsavvy.co.uk/mobile-payment-stats-trends/>.
- [2] An Overview on Elasticsearch and its usage - Towards Data Science.
- [3] Apache JMeter - Apache JMeter™.
- [4] Apache JMeter - User's Manual.
- [5] Bootstrap · The most popular HTML, CSS, and JS library in the world.
- [6] Como utilizar a app Wallet no iPhone, iPod touch e Apple Watch - Suporte Apple.
- [7] Complete Loyalty Program Software | iVend Retail. <https://ivend.com/ivend-loyalty/>.
- [8] ELK — Elasticsearch, Logstash, and Kibana - Z Little - Medium.
- [9] Forticlient - Next Generation Endpoint Protection.
- [10] Google Pay | Google Developers.
- [11] Guia do Scrum TM. Technical report.
- [12] Introduction to Mobile Wallet By PassKit. Technical report.
- [13] Java Programming Language. <https://docs.oracle.com/javase/7/docs/technotes/guides/language/>.
- [14] Kibana: Explore, Visualize, Discover Data | Elastic.
- [15] LOG Centralization: Using Filebeat and Logstash - Tensult Blogs - Medium.
- [16] Loopy Loyalty says. . . . <https://loopyloyalty.com/>.
- [17] Maven – Welcome to Apache Maven.
- [18] Mobile Marketing Software to create Digital Coupons, Vouchers and Loyalty Cards. <https://www.coupontools.com/>.
- [19] Pass2U | Apple Wallet and Google Pay Pass Solution.
- [20] PassKit - Extend your mobile reach with Apple Wallet and Google Pay Passes. <https://passkit.com/>.
- [21] PassKit Web Service Reference. <https://developer.apple.com/library/archive/documentation/PassKit>
- [22] Quality attributes in Software Architecture - Nikolay Ashanin - Medium.
- [23] Samsung Pay | Apps - The Official Samsung Galaxy Site.

- [24] Setting Up a Remote Notification Server | Apple Developer Documentation.
https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server
- [25] Spring. <https://spring.io/>.
- [26] Talon.One: The World's Most Powerful Promotion Engine. <https://www.talon.one/>.
- [27] The Most Popular Mobile Payment Apps.
- [28] *The User Guide CherryPie - PassKit*.
- [29] TypeScript - JavaScript that scales.
- [30] Typical API flow | Google Pay for Passes | Google Developers.
- [31] Voucherify: Promotion Management Software for Digital Teams.
<https://www.voucherify.io/>.
- [32] Wallet Passes | Passbook® Wallet for Android™.
- [33] WeChat Pay.
- [34] What is Kibana? | Elastic.
- [35] What is Scrum? <https://www.scrum.org/resources/what-is-scrum>.
- [36] What is the open rate for SMS in 2018? | Esendex Blog.
- [37] PostgreSQL: The world's most advanced open source database.
<https://www.postgresql.org/>, 2013.
- [38] Airship. The State of Mobile Wallet Marketing. Technical report.
- [39] Apple. Wallet Developer Guide: Updating a Pass.
https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/PassKit_PG/CH5-SW1, 2018.
- [40] Firebase. Firebase Cloud Messaging | Firebase.
<https://firebase.google.com/docs/cloud-messaging>, 2018.
- [41] Product Plan. What is MoSCoW Prioritization? | Overview of the MoSCoW Method.
<https://www.productplan.com/glossary/moscow-prioritization/>, 2019.
- [42] Ravi Chari. What is a risk model? <https://www.microtool.de/en/knowledge-base/what-is-a-risk-matrix/>.

Appendices

This page is intentionally left blank.

Appendix A

This appendix presents the sprints and tasks defined and implemented in the first semester and those to be implemented in the second semester.

First Semester

Table 1: First Semester Planning

| Sprint | Description | Tasks | Date |
|--------|--------------------------------|--|----------------------------|
| S01 | State of art | <ul style="list-style-type: none">• Research on concepts A2P campaigns• Research on existing platforms for virtual passes• Research on key techniques | 17/09/2019 - 29/09/2019 |
| S02 | Requirement Analysis | <ul style="list-style-type: none">• User stories and functional requirements• Non-functional requirements | 30/09/2019 - 05/10/2019 |
| S03 | Software architecture solution | <ul style="list-style-type: none">• Technologies analysis• Database analysis and diagram• Final architecture HLD | 06/10/2019 - 15/10/2019 |
| S04 | Development | <ul style="list-style-type: none">• Database<ul style="list-style-type: none">– Data Model– Database Implementation• Loyalty and Coupons Backend API• Management API Prototyping• Frontend API Prototyping• APNs and FCM Implementation | 16/10/2019 - 03/12/2019 |

Table 1: First Semester Planning

| Sprint | Description | Tasks | Date |
|--------|--------------------------------|--|----------------------------|
| S05 | Q&A | <ul style="list-style-type: none">• API testing• Demonstration tests | 04/12/2019 - 30/12/2019 |
| S06 | Intermediate internship report | <ul style="list-style-type: none">• Write state of art chapter• Write approach methods chapter• Write software architecture chapter• Write requirements analysis chapter• First version of intermediate internship report• Finish intermediate internship report• Write intermediate internship presentation | 17/11/2019 - 19/01/2020 |

Second Semester

Table 2: Second Semester Planning

| Sprint | Description | Tasks | Date |
|--------|---------------------------------|--|----------------------------|
| S01 | Software architecture solution | <ul style="list-style-type: none"> Technologies analysis Final architecture HLD version 2 | 29/01/2020 - 05/02/2020 |
| S02 | Development - Management API | <ul style="list-style-type: none"> Loyalty and Coupons Back-end API version 2 Management API Prototyping version 2 | 06/02/2020 - 08/03/2020 |
| S03 | Development - Loyalty App | <ul style="list-style-type: none"> Front-end Loyalty App Back-end Loyalty App Study of Technologies - Report Module Implementation - Report Module | 09/03/2020 - 28/05/2020 |
| S03 | Q&A | <ul style="list-style-type: none"> Functional tests Non-Functional tests | 29/05/2020 - 13/06/2020 |
| S04 | Final internship report | <ul style="list-style-type: none"> Finish final internship report | 14/06/2020 - 28/06/2020 |

This page is intentionally left blank.

Appendix B - REST API Documentation Apple Wallet

Devices Register for Updates

Method POST request to

`"/{version}/devices/{deviceLibraryIdentifier}/registrations/{passTypeIdentifier}/{serialNumber}"`

Description: After installing a pass, the iOS device registers with the server, asking to receive updates, the server saves the device's library ID and its push token.

Table 3: Information to send by device

| Parameters | Located In | Required |
|---------------------------|--------------|----------|
| Device library identifier | the URL | Yes |
| Push Token | JSON payload | Yes |
| Pass Type Identifier | the URL | Yes |
| Serial Number | the URL | Yes |
| Authentication Token | the header | Yes |

Responses

Table 4: Responses

| Conditions | Status |
|--|--|
| If the serial number is already registered for this device | returns HTTP status 200 |
| If registration succeeds | returns HTTP status 201 |
| If the request is not authorized | returns HTTP status 401 |
| Otherwise | returns the appropriate standard HTTP status |

Getting the Serial Numbers for Passes Associated with a Device

Method GET request to

`"/{version}/devices/{deviceLibraryIdentifier}/registrations/{passTypeIdentifier}{?passesUpdatedSince}=tag"`

Description

When a device gets a push notification, it asks your server for the serial numbers of passes that have changed since a specific point in time.

Table 5: Information to send by device

| Parameters | Located In | Required |
|---|------------|----------|
| Device library identifier | the URL | Yes |
| Pass Type Identifier (from the push notification) | the URL | Yes |
| Latest update tag | - | No |

Responses

Table 6: Responses

| Conditions | Status |
|----------------------------------|---|
| If there are matching passes | returns HTTP status 200 with a JSON dictionary with the following keys and values: <ul style="list-style-type: none"> • lastUpdated (string): The current modification tag • serialNumbers (array of strings): The serial numbers of the matching passes. |
| If registration succeeds | returns HTTP status 201 |
| If the request is not authorized | returns HTTP status 401 |
| Otherwise | returns the appropriate standard HTTP status |

Getting the Latest Version of a Pass

Method GET request to

`"/{version}/passes/{passTypeIdentifier}/{serialNumber}"`

Description: The device asks to server for the latest version of each updated pass. To prove that the request is valid, the device includes the pass's authorization token.

Table 7: Information to send by device

| Parameters | Located In | Required |
|----------------------|------------|----------|
| Pass Type Identifier | the URL | Yes |
| Serial Number | the URL | Yes |
| Authentication Token | the header | Yes |

Responses

Table 8: Responses

| Conditions | Status |
|----------------------------------|---|
| If request is authorized | returns HTTP status 200 with a payload of the pass data |
| If the request is not authorized | returns HTTP status 401 |
| Otherwise | returns the appropriate standard HTTP status |

Unregistering a Device

Method DELETE request to

`"/{version}/devices/{deviceLibraryIdentifier}/registrations/{passTypeIdentifier}/{serialNumber}"`

Description: The server disassociates the specified device from the pass, and no longer sends push notifications to this device when the pass changes.

Table 9: Information to send by device

| Parameters | Located In | Required |
|---------------------------|--------------|----------|
| Device library identifier | the URL | Yes |
| Push Token | JSON payload | Yes |
| Pass Type Identifier | the URL | Yes |
| Serial Number | the URL | Yes |
| Authentication Token | the header | Yes |

Responses

Table 10: Responses

| Conditions | Status |
|----------------------------------|--|
| If disassociation succeeds | returns HTTP status 200 |
| If the request is not authorized | returns HTTP status 401 |
| Otherwise | returns the appropriate standard HTTP status |

Logging Errors

Method POST request to

`"/{version}/log"`

Description: This endpoint is intended to help to debug the web service implementation. Log messages contain a description of the error in a human-readable format.

Response Returns HTTP status 200

Server Sends a Push Notification When Something Changes

Method GET request to

`"/{version}/devices/{deviceLibraryIdentifier}/registrations/{passTypeIdentifier}{?passesUpdatedSince}=tag"`

Description: When a pass needs to be updated, server sends a push notification to inform devices of the change. Push notifications are sent only to the devices that have registered for updates for that pass, and only when the pass has changed.

Table 11: Information to send by device

| Parameters | Located In | Required |
|---|------------|----------|
| Device library identifier | the URL | Yes |
| Pass Type Identifier (from the push notification) | the URL | Yes |
| Latest update tag | - | No |

This page is intentionally left blank.

Appendix C - Software Quality

API Tests

The following tests contain the API tests performed to ensure that the API does what is intended and returns the expected responses. All written tests aimed to analyze and evaluate the server's response.

Passes Controller

| T01 - Download | | | |
|----------------|-----------------------------------|--|---|
| Path | /download/{clientID}/{campaignID} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - <i>clientID</i> does not correspond to an existing client id | Code: 404 - Not Found |
| 2 | GET | - <i>campaignID</i> does not correspond to an existing campaign id | Code: 404 - Not Found |
| 3 | GET | - <i>device_OS</i> is Android | Code: 202 - Accepted |
| 4 | GET | - <i>device_OS</i> is iOS | Code: 202 - Accepted Message: .pkpass File |
| Last Result | OK | | |

Table 12: Download pass

| T02 - Update | | | |
|--------------|--------------------------------------|--|-------------------------|
| Path | /v1/templates/{brandID}/{campaignID} | | |
| # | Type | Conditions | Expected Responses |
| 1 | PUT | - <i>brandID</i> does not correspond to an existing brand id | Code: 404 - Not Found |
| 2 | PUT | - <i>campaignID</i> does not correspond to an existing campaign id | Code: 404 - Not Found |
| 3 | PUT | - <i>template</i> is empty | Code: 400 - Bad Request |
| Last Result | OK | | |

Table 13: Update pass and data client

| T03 - List all campaigns | | | |
|--------------------------|------------------------------|---|---|
| Path | /v1/templates/list/{brandID} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - brandID does not correspond to an existing brand id | Code: 404 - Not Found |
| 2 | GET | - campaignID does not correspond to an existing campaign id | Code: 404 - Not Found |
| 3 | GET | - All parameters matches | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 14: List campaigns

| T04 - Get all campaigns | | | |
|-------------------------|-------------------------|---|---|
| Path | /v1/campaigns/{brandID} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - brandID does not correspond to an existing brand id | Code: 404 - Not Found |
| 2 | GET | - brandID matches an existing brand | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 15: Get all campaigns

| T05 - Get all brands | | | |
|----------------------|------------|-----------------------------|---|
| Path | /v1/brands | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - There is no brand created | Code: 404 - Not Found |
| 2 | GET | - There is brands created | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 16: Get all brands

Apple Wallet Controller

| T06 - Register a device | | | |
|-------------------------|---|---|---------------------------------|
| Path | /v1/devices/{ <i>deviceID</i> }/registrations/{ <i>passTypeID</i> }/{ <i>serialNumber</i> } | | |
| # | Type | Conditions | Expected Responses |
| 1 | POST | - <i>passTypeID</i> does not correspond to an existing pass type identifier | Code: 400 - Bad Request |
| 2 | POST | - Empty <i>authenticationToken</i> parameter | Code: 401 - Unauthorized |
| 3 | POST | - If <i>serialNumber</i> is already registered for this <i>deviceID</i> | Code: 200 - OK |
| 4 | POST | - If registration succeeds | Code: 201 - Created |
| Last Result | OK | | |

Table 17: Registering a Device to Receive Push Notifications for a Pass

| T07 - Get serial numbers | | | |
|--------------------------|---|-----------------------------------|--|
| Path | /v1/devices/{ <i>deviceID</i> }/registrations/{ <i>passTypeID</i> ? <i>passesUpdatedSince</i> =tag} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - If there are no matching passes | Code: 204 - No Content |
| 2 | GET | - If there are matching passes | Code: 200 - OK Message: JSON dictionary with <i>lastUpdatedTag</i> and <i>serialNumbers</i> |
| Last Result | OK | | |

Table 18: Getting the Serial Numbers for Passes Associated with a Device

| T08 - Get latest version pass | | | |
|-------------------------------|--|---|--|
| Path | /v1/passes/{passTypeID}/{serialNumber} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - <i>passTypeID</i> does not correspond to an existing pass type identifier | Code: 400 - Bad Request |
| 2 | GET | - Empty <i>authenticationToken</i> parameter | Code: 401 - Unauthorized |
| 3 | GET | - If request is authorized | Code: 200 - OK Message: Payload of the pass data (.pkpass File) |
| Last Result | OK | | |

Table 19: Getting the Latest Version of a Pass

| T09 - Unregister device | | | |
|-------------------------|--|----------------------------------|---------------------------------|
| Path | /v1/devices/{deviceID}/registrations/{passTypeID}/{serialNumber} | | |
| # | Type | Conditions | Expected Responses |
| 1 | DELETE | - All parameters doesn't matches | Code: 401 - Unauthorized |
| 2 | DELETE | - All parameters matches | Code: 200 - OK |
| Last Result | OK | | |

Table 20: Unregistering a Device

Loyalty Controller

| T10 - Upload clients data | | | |
|---------------------------|--------------------------|--------------------------|---|
| Path | /uploadFile/{campaignID} | | |
| # | Type | Conditions | Expected Responses |
| 1 | POST | - File is empty | Code: 400 - Bad Request |
| 2 | POST | - All parameters matches | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 21: Upload clients data

| T11 - Get top downloads from passes | | | |
|-------------------------------------|----------------------------------|-------------------------|---|
| Path | /logs_download/search/{brand_id} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - Request is authorized | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 22: Get top downloads from passes

| T12 - Get data from SMS logs | | | |
|------------------------------|--------------------------|-------------------------|---|
| Path | /logs_sms/sms/{brand_id} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - Request is authorized | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 23: Get data from SMS logs

| T13 - Get total number of costumers using iOS and/or Android | | | |
|--|--------------------------|-------------------------|---|
| Path | /logs_sms/sms/{brand_id} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - Request is authorized | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 24: Get total number of costumers using iOS and/or Android from downloads logs

| T14 - Get number of updates per day | | | |
|-------------------------------------|--|-------------------------|---|
| Path | /logs_update/number_updates/{brand_id} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - Request is authorized | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 25: Get number of updates per day from updates logs

| T15 - Get number of updates per campaign | | | |
|--|---|-------------------------|---|
| Path | /logs_update/updates/{brand_id}/{campaign_id} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - Request is authorized | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 26: Get number of updates per campaign from updates logs

| T16 - Get number of downloads performed and sms sent | | | |
|--|----------------------------|-------------------------|---|
| Path | /logs_rejection/{brand_id} | | |
| # | Type | Conditions | Expected Responses |
| 1 | GET | - Request is authorized | Code: 200 - OK Message: JSON message |
| Last Result | OK | | |

Table 27: Get number of downloads performed and sms sent from logs