Customer Service and Ticketing Chatbot: From Conception to Production

Gonçalo Lascasas Cereja Ribeiro Pereira

Thesis to obtain the Master of Science Degree in Information Systems and Computer Engineering

Supervisor: Prof. José Manuel da Costa Alves Marques

Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. José Manuel da Costa Alves Marques
Member of the Committee: Prof. Alberto Manuel Ramos da Cunha

December 2019
Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me, and without whom this project would not be possible. I would also like to thank the rest of my family, for the support throughout all these years, and specially my grand father, which I am sure he would be very proud.

I would also like to acknowledge my dissertation supervisors Prof. José Manuel da Costa Alves Marques and Eng. Joel dos Santos Teixeira for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times. To each and every one of you, thank you!
Abstract

Chatbots, also called conversational agents, are software systems which can interact with humans using natural language, through text or voice interfaces. They are facing a rise in popularity mostly due to recent advancements in AI and in this dissertation I present an analysis regarding the state of the art of this technology. It is possible to develop a chatbot really fast using a platform, although, each architecture brings limitations. With the help of Link Consulting I could design, prototype and develop a customer support chatbot with ticketing capabilities for the Portuguese railway company (CP). Machines are still far from truly understanding natural languages, reason why most chatbots are based on buttons, and the most important part must be ensuring users are able to obtain meaningful answers for their intentions. To evaluate this proposal, a user testing functionality with the five most important use cases was implemented inside the bot allowing a scalable way to test with real users. By developing everything from scratch I could integrate securely with services through APIs and obtain custom analytics apart from those gathered by Dashbot. The biggest limitations was being limited by messenger platform which is recurrently being updated, but overall CP Bot received great feedback from its more than 200 beta users. Bots will certainly be part of our lives but developers must ensure users have a pleasant experience, hopefully easier and faster than with alternative interfaces.

Keywords

Chatbot; Natural Language; Conversation Design; Customer Service; Ticketing.
Resumo

Os chatbots, também conhecidos por agentes conversacionais, são sistemas que interagem com os utizadores numa interface de língua natural, através de texto ou voz. Tem havido um crescimento na sua popularidade maioritariamente devido aos recentes avanços em IA e nesta dissertação apresento um analise completa do estado da arte desta tecnologia. É possível desenvolver um chatbot rápido através de plataformas mas cada arquitectura tem as suas limitações. Com a ajuda da Link Consulting foi possível desenvolver o CP Bot, em Português e Inglês, que entre outras funcionalidades permite ver e comprar viagens de comboio, receber alertas relevantes e obter suporte instantâneo e a qualquer hora do dia. A parte mais importante é garantir que os utilizadores conseguem obtem respostas relevantes para as suas intenções. Para avaliar esta proposta uma funcionalidades de teste foi implementada dentro do bot permitindo escalar o teste com utilizadores reais e melhorar a base de conhecimento. Por ter desenvolvido tudo de raiz foi possível integrar de forma segura com serviços externos e também analisar as mensagens passadas com parametros customizáveis para alem das metricas obtidas no Dashbot. A maior limitação foi sem duvida estar dependente da plataforma do messenger que tem sido constantemente atualizada, mas de forma geral o CP Bot recebeu um feedback bastante positivo dos mais de 200 utilizadores. Os chatbots farão certamente parte das nossas vidas mas os seus criadores têm de garantir uma experiencia agradavel aos utilizadore, de preferencia mais fácil e rápida que usando outra interface.

Palavras Chave

Chatbot; Língua Natural; Design Conversacional; Atendimento ao Cliente; Bilhética.
# Contents

1 Introduction ................................................................. 1
   1.1 Objectives ................................................................. 3
   1.2 Organisation of the Document ........................................... 4

2 Background ....................................................................... 5
   2.1 What are chatbots? .......................................................... 7
   2.2 Types of chatbots ............................................................ 8
   2.3 Conversation Design ....................................................... 9
   2.4 Chatbot Challenges ......................................................... 12
   2.5 Chatbot Architecture ...................................................... 12
      2.5.1 Natural Language Understanding ................................. 13
      2.5.2 Dialog Management ................................................ 14
      2.5.3 External API Calls ................................................... 14
      2.5.4 Natural Language Generation ................................... 14
      2.5.5 Chatbot Development Framework ............................... 14
      2.5.6 Chatbot Platforms ................................................... 15
   2.6 NLU Alternatives .......................................................... 16
   2.7 Payments in Chatbots ..................................................... 17
      2.7.1 Payments with Middleware ........................................ 17
      2.7.2 Payments with platform API ...................................... 18
      2.7.3 Payments through Mobile App .................................... 19
   2.8 Chatbot Evaluation ......................................................... 19
      2.8.1 Radziwill and Benton ................................................. 19
      2.8.2 Cas Jongerius ......................................................... 20
   2.9 Transportation Chatbots .................................................. 23
   2.10 Serverless Applications .................................................. 27
   2.11 Chatbot's Privacy ........................................................ 28
   2.12 Chatbot Analytics ......................................................... 30
## Implementation

### 3.1 CP contextualisation

### 3.2 Analysis

### 3.3 Architecture

#### 3.3.1 Channels

#### 3.3.2 Dialog Manager

#### 3.3.3 Dialogflow

#### 3.3.4 External API Calls

#### 3.3.5 CP

#### 3.3.6 Database

#### 3.3.7 Notification Manager

#### 3.3.8 Natural Language Generation

### 3.4 Design and Prototype

### 3.5 Development

#### 3.5.1 Dialogflow Training

#### 3.5.2 Build and Test

## Results

### 4.1 Automatic Metrics

#### 4.1.1 Functionality Tests

#### 4.1.2 Dialogflow Metrics

#### 4.1.3 Dashbot Metrics

### 4.2 User Testing

### 4.3 Good Practices

#### 4.3.1 Conception

##### 4.3.1.A Analyse

##### 4.3.1.B Design and Prototype

#### 4.3.2 Development

##### 4.3.2.A NLU Training

##### 4.3.2.B Build

##### 4.3.2.C Test

#### 4.3.3 Production
List of Figures

2.1 IM platforms monthly active users ......................................................... 8
2.2 Types of chatbots .................................................................................. 9
2.3 Example of structured elements and a webview ................................... 11
2.4 Chatbot Architecture ........................................................................... 13
2.5 Examples of Chatfuel’s plugins ............................................................... 15
2.6 Example of payment with Messenger API ........................................... 18
2.7 Example of AHP results ........................................................................ 20
2.8 Chatbot Maturity Levels ....................................................................... 22
2.9 OUIbot example interactions ................................................................. 25
2.10 Kayak interactions in Facebook Messenger ......................................... 26
2.11 Program hosting evolution .................................................................. 27
2.12 Conversation Funnel in Dashbot ........................................................... 30
2.13 Clustering in Botanalytics ................................................................... 31

3.1 CP Bot Architecture ............................................................................... 37
3.2 Chatfuel prototype message count ......................................................... 43
3.3 Examples of chatbot interactions ............................................................ 43
3.4 Development Iterations ......................................................................... 44
3.5 Final prototype message count ................................................................. 45
3.6 Helping entities ....................................................................................... 46
3.7 Date entities ............................................................................................ 46
3.8 FAQ entities and intent .......................................................................... 47

4.1 Top five of the table with all intents handled ........................................ 54
4.2 Session flow ........................................................................................... 55
4.3 Sessions metrics in Dialogflow ............................................................... 55
4.4 Live transcripts functionality in Dashbot ............................................. 56
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5 Audience Builder functionality in Dashbot</td>
<td>56</td>
</tr>
<tr>
<td>4.6 User test results on 17th October 2019</td>
<td>58</td>
</tr>
<tr>
<td>4.7 Examples of user test conversations</td>
<td>59</td>
</tr>
<tr>
<td>4.8 Proposed chatbot life cycle</td>
<td>60</td>
</tr>
<tr>
<td>4.9 Conversation design elements</td>
<td>62</td>
</tr>
<tr>
<td>A.1 Initial Conversation Flow</td>
<td>76</td>
</tr>
<tr>
<td>A.2 Near stations conversation flow</td>
<td>77</td>
</tr>
<tr>
<td>B.1 CP Bot Poster</td>
<td>80</td>
</tr>
<tr>
<td>B.2 Example of chatbot ad with deep links</td>
<td>81</td>
</tr>
<tr>
<td>B.3 User test results - part 1</td>
<td>82</td>
</tr>
<tr>
<td>B.4 User test results - part 2</td>
<td>83</td>
</tr>
</tbody>
</table>
List of Tables

2.1 NLU platforms comparison ................................................. 17
2.2 Travelling Chatbots Comparison ........................................... 24
3.1 Chatbot execution times ..................................................... 49
4.1 Evaluation Metrics .......................................................... 53
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHP</td>
<td>Analytic Hierarchy Process</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>CP</td>
<td>Comboios de Portugal</td>
</tr>
<tr>
<td>DPO</td>
<td>Data Protection Officer</td>
</tr>
<tr>
<td>FaaS</td>
<td>Function-as-a-Service</td>
</tr>
<tr>
<td>FAQ</td>
<td>Frequently Asked Questions</td>
</tr>
<tr>
<td>GDPR</td>
<td>General Data Protection Regulation</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IM</td>
<td>Instant Messaging</td>
</tr>
<tr>
<td>IP</td>
<td>Infraestruturas de Portugal</td>
</tr>
<tr>
<td>NLG</td>
<td>Natural Language Generation</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>NLU</td>
<td>Natural Language Understanding</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>TTS</td>
<td>Text To Speech</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td><strong>UX</strong></td>
<td>User Experience</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>VM</strong></td>
<td>Virtual Machine</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Objectives ......................................................... 3
1.2 Organisation of the Document ................................. 4
Ever since the rise of the internet, it has influenced the way humans interact with each other. It allows us to reach out to a much greater number of people on a global scale. Starting two decades ago, social media has played an important role in changing the way people communicate [1]. Since 2015, the number of monthly active users of the four largest messaging apps surpassed the four largest social networks [2]. Users are rapidly turning to social media to request and receive customer service. However, a majority of these requests can’t be addressed timely or not even addressed at all.

Chatbots are software agents in which users can request data or services using natural language interfaces, text and voice. They are typically designed and developed for integration with mobile messaging applications, and represent a potential shift in how people interact with data and online services. Users want access to personalised information on demand, preferably 24/7, and although chatbots are not a new technology, the increase in popularity has strengthened due to the recent advancements in AI. Natural Language Processing is the AI sub-category that allows chatbots to “learn with experience” instead of matching user input with expected patterns. Chatbot adoption is also increasing with the launch of developer API by most popular messaging platforms such as Facebook Messenger, Kik, Slack, Skype, Line, and Telegram.

1.1 Objectives

Many building alternatives are at our disposal, but creating a complex chatbot is not easy, because each architecture has its own limitations. Implementing a customer support chatbot with ticketing capabilities introduces technical complexities that prevent the use of conventional architectures. To provide dynamic data, such as ticket availability, latency of responses should also be taken into account. Facebook Messenger and Slack have timeout limits of 20 and 3 seconds, respectively, and even platforms such as Dialogflow have similar policies, in this case 5 seconds.

My main objectives with this thesis are:

- Understand how well chatbots integrate with existing systems.
- Implement a goal-oriented chatbot that understands its users using Natural Language Processing (NLP).
- Evaluate implemented chatbot through effectiveness, efficiency and satisfaction metrics.
- Compare intent and entity detection rates for EN, pt-PT and pt-BR agents in Dialogflow.
- Understand advantages and limitations of this technology.
- Produce a list of guidelines or good practices for chatbot’s designers and developers.
1.2 Organisation of the Document

This thesis is organised as follows: Chapter 2 starts with some background detailing what are chatbot, how they work and some of its history. It will also focus on the alternative architectures, along with methods for dialog systems evaluation, details regarding conversational design and current challenges in chatbot development. Chapter 3 introduces an agile chatbot's life cycle, which starts with its conception, then an iterative development phase, and ending with a chatbot ready for production. Then, I explain how it relates with the with my proposal, CP Bot, including architectural decision. Chapter 4 details the method the evaluation process of this proposal, providing results from both user testing and some automated metrics. Finally, Chapter 5 presents some personal conclusions about this project, including limitations and what should be done in future work.
2 Background

Contents

2.1 What are chatbots? .......................................................... 7
2.2 Types of chatbots .......................................................... 8
2.3 Conversation Design .................................................... 9
2.4 Chatbot Challenges ...................................................... 12
2.5 Chatbot Architecture ................................................... 12
2.6 NLU Alternatives ......................................................... 16
2.7 Payments in Chatbots .................................................. 17
2.8 Chatbot Evaluation ...................................................... 19
2.9 Transportation Chatbots .............................................. 23
2.10 Serverless Applications ............................................... 27
2.11 Chatbot’s Privacy ....................................................... 28
2.12 Chatbot Analytics ...................................................... 30
2.1 What are chatbots?

Different terms can be found on the Internet to refer to chatbots, the most common are conversational agents, dialog systems and virtual assistants. In a vertical chatbot, users can expect it to be specialised in a certain topic, such as sending a message to a specific organisation. While when interacting with virtual assistant, such as Alexa or Siri, they expect to address all types of user requests, representing an horizontal, or generalist chatbot. In general, chatbots goal is enabling users to interact with systems using natural language, just like engaging in a conversation with a person [3].

There are chatbot applications in a great variety of domains and in multiple messaging platforms. Users can find thousands of chatbots, for example, in BotList (https://botlist.co) and even search by categories. In particular, chatbots for customer service are seen as promising, mostly because users want support throughout the day, anytime and anywhere. Also, organisations benefit from automating conversations that would require human intervention.

Chatbots appeared when the British mathematician Alan Turing questioned if machines could think along with a way of testing it, and conceptualised this problem as an “imitation game” (now known as Turing Test) [4]. It requires 3 players, an AI machine (A), and two humans, one being the player (B) and the other is the interrogator (C). The interrogator has the goal of identifying which of the players, A or B, is the human. If he does not guess correctly, it means that the machine was able to mistake the human, and we can consider it can think.

The Turing Test, although never imposing limitations on how the machine was build, for the time that was implemented was a very notable work, especially for the AI community. That was the motivation that triggered the study and development of smarter chatbots. The Loebner prize competition has been used to evaluate machine conversation chatbots. This Loebner Prize is a Turing test, which evaluates the ability of the machine to fool people that they are talking to human. [5]

The first chatbot was developed in 1966, named “Eliza”, and it mimicked human conversation by matching user prompts to scripted responses [6]. On 1995, “Artificial Linguistic Internet Computer Entity”, A.L.I.C.E, was a bot with a knowledge of AIML documents. It was able to match user input with patterns and respond with corresponding template message. Chatbot development kept evolving, and in 2001 “Smarterchild” was already widely distributed across SMS networks and it was able to quickly access data and consequently have a customised conversation [7].

Until the 21st century most chatbots were rule-based, making them very limited to understand user input. In 2006 IBM launched “Watson” which was specifically designed to compete on “Jeopardy!” an American TV show featuring a quiz competition. In 2011, Watson was able to defeat two of the show former champions [8]. It is being continuously developed and currently is IBM’s AI platform for business.

After IBM, chatbots started to appear as Virtual Personal Assistants. Apple launched Siri in 2010, Google Now was presented in 2012, which was followed by Amazon’s Alexa and Microsoft Cortana in
2015. At this point chatbots were already able to delegate requests to web services, set reminders, send recommendations and retrieve information to user questions using search engines like Google and Bing.

In April 2016, Facebook launched their Messenger platform and exactly two years after Messenger platform was released, Facebook announced at their annual conference, F8, more than 300,000 active bots, with 8 billion messages exchanged between people and businesses every month [9]. They also stated, in Q4 2017 earnings report, that Whatsapp had 1.5 Billion monthly users and 60 Billion messages sent per day. Recently, they launched their business API for automated messaging with customers, initially being made available a selection of companies, which will gradually be offered to a larger group of customers [10].

The use of chatbots is having an exponential growth, mostly because Instant Messaging (IM) platforms gave developers the ability to create automated agents to interact with users in their channels. Figure 2.1 represents monthly active users of most popular messaging platforms in 2017.

![IM platforms monthly active users](image)

**Figure 2.1:** IM platforms monthly active users

### 2.2 Types of chatbots

Chatbots can be divided into different types depending on their purpose and the way responses are generated, as demonstrated in Figure 2.2 [11]. Closed domain chatbots are easier to develop because they have a somewhat limited space of possible inputs and outputs. Customer support or shopping assistants are typical examples, and their purpose is to help users get useful information or achieve other goals. Users can still ask anything, but the system isn’t required to handle all cases and the users don’t expect it to.
Open domain bots have existed since famous Eliza was introduced to participate in Loebner’s contest. They are mainly for entertainment purposes and focus on giving smart responses to maintain the conversation. The infinite number of topics and the fact that a certain amount of world knowledge is required to create reasonable responses makes this a hard problem. One way to assess them is by measuring average length of conversations, and obviously, if conversations are short means that the chatbot is not entertaining enough.

A retrieval-based chatbot uses a repository of predefined responses and some kind of parameter to pick the most appropriate response based on the message and context. The parameter could be simple, as in a rule-based expression match, or complex when using Machine Learning classifiers. These models don’t generate any new text, they just pick a response from all available templates.

Chatbots with generative-based responses are harder to create because they need to generate new responses from scratch. These models are typically based on Machine Translation techniques, but instead of translating from one language to another, they “translate” from an input to an output set [12].

### 2.3 Conversation Design

Early ages of computer interfaces used conversation design, which was not user-friendly because rule-based programs in command lines implied that the user to remember all command they would like to execute. Graphical User Interface (GUI) came along with the purpose of improving user experience. Well designed GUI allow first-time users to interact with systems even without reading a manual.

Designing for web or mobile apps have long been standardised, with established usability practices such as Nielsen’s usability heuristics or Gestalt principles for organisation of visual elements. However,
they are not very helpful on conversational interfaces that barely use User Interface (UI) elements. Conversation is a complex system that has been studied by linguistics and conversation analysts, and has some challenges such as unexpected inputs, follow-up questions, changes of topic, and recovery from error. A wide range of topics can be addressed in these interfaces [3], including:

- The choice and use of different media for input and output.
- How to promote engagement and retention.
- How to make the customer experience more personal and more pleasant.
- The use of personas and branding.
- How to measure the quality of the interaction.
- Design guidelines.
- Whether the system simulates human conversation to the extent that it could be mistaken for a human.

Chatbots can send richer message types that can be used in conjunction with ordinary text messages in order to enrich bot experience, although available design elements depend on the messaging platform being used.

Many messaging platforms provide structured messages forms, which can further enrich the flow of conversation. These structures can contain buttons, quick replies, or show different alternatives in a rich representation. The advantages of structured messages are manifold [13]:

1. They constrain the conversation into a limited number of expected outcomes, reducing the possibility of users feeling trapped in a dead end where they have to "guess" their way out.
2. They push the user to use the service, suggesting how the conversation can continue. They also reduce the need for the users to "explore" the interface, making it easier to learn and use.
3. Buttons and quick replies reduce interactions to a single tap instead of requiring complex typing.
4. The service can be implemented more easily.

Figure 2.3 shows some examples of structured conversational elements and a webview to pick a date. Webviews can be used in longer interactions where users might want to edit their input and for content that's specially visual. A common use case for them is to perform payments, keeping information secure, as it goes directly from the website to the payment provider. If required, these interactions should be brief and combined with more conversational interactions.

According to Microsoft [14], the following factors influence a bot's success, even though none of these questions relates to how smart the bot is:
• Does the bot easily solve the user’s problem with the minimum number of steps?

• Does the bot solve the user’s problem better/easier/faster than any of the alternative experiences?

• Does the bot run on the devices and platforms the user cares about?

• Is the bot discoverable? Do the users naturally know what to do when using it?

Chiara Martino, Chatbot Conversation Designer, defines her job as a Linguist who deals with the NLP component of a chatbot, who design its conversational flows and who writes the most appropriate copy for each these flows. To give users the answers they are looking for, and engage them in a pleasant conversation, the conversation designers’ workflow is divided into three main steps [15]:

1. **Understanding** what the user needs is of primary importance, although matching the correct intent is not an easy task. Human languages are formally complex and highly dependent on context, a request can be expressed in so many different and ambiguous ways, that is highly unlikely to map them all.

2. **Answering** the perceived user intention by designing possible flows, and guiding the user towards the answers. In this step conversation designers plan tree-structured paths based on a careful analysis of the domain and of the specific topics dealt by the chatbot.

3. **Entertaining** phase focus on defining chatbot’s personality, which depends on the brand it is representing and on the target users. This includes defining the level of formality, if it should be funny or serious, or deciding if it should give brief and practical solutions or to entertain the user with jokes.
2.4 Chatbot Challenges

The idea of automating and scaling one-to-one conversations using technology appeals to lots of brands and services. Yet, it is very challenging to create chatbots that master natural languages like humans conversations, which encompass much more than just text. Humans can read between the lines, leverage contextual information and understand double layers like sarcasm. Bots quickly forget what they are talking about, which is like talking to someone with little short-term memory.

Building an intelligent chatbot is not devoid of challenges. From making the chatbot context-aware to building the personality of the chatbot, there are challenges involved in making a great and intelligent chatbot. Conversational designers play an important role at defining how each conversation is scripted, including follow-up questions, self-correcting messages, or sudden change of conversion topic. These challenges, among others, lead to chatbots failing to deliver user experiences that are as seamless, delightful and efficient as they were envisioned to be.

Chatbots that try to answer every possible question usually fail due to lack of training. Instead, they should focus on training intensively social cues and closed domains utterances to ensure great user comprehension. Fallback responses, which may happen in every point of the conversation should also be differentiated through the context. To illustrate, imagine the user is requesting a train schedule but writes “what are the next trains in Caxcais”, the chatbot might not identify the misspelling of “Cascais” but it could reduce user frustration by replying an invalid station fallback, asking the user to try a different station while keeping the context of the conversation.

Previous research suggest that when designing chatbots, attention to details is required to make conversations feel natural and human-like. Moreover, the provision of inappropriate social cues to humanise a chatbot might create unrealistic user expectations and lead to misunderstandings, particularly when these cues overplay the chatbot’s actual capabilities [16]. They also conclude that a customer service chatbot that sends dynamically delayed responses will yield a higher level of perceived humanness than a customer service chatbot that sends near-instant responses. In addition, users will also be more satisfied with the interaction with a customer service chatbot that implements dynamically delayed responses.

2.5 Chatbot Architecture

Chatbot’s conversation is an endless cycle of receiving user’s utterances, then understand its meaning, and finally produce the most appropriate response. A chatbot is composed by a frontend and a backend connected by webhooks calls. Calling a webhook differs from calling an Application Programming Interface (API) by being possible to send data without being requested, improving system’s efficiency.

Frontend clients, or channels, are gateways for clients to access chatbots, and it is possible to reach
bigger audiences by integrating with multiple frontends, usually Instant Messaging (IM) platforms, but also possible through apps and websites. The backend of a chatbot is very similar to a typical Dialog System with natural language understanding, dialog management, natural language generation [17]. It needs to analyse and structure received messages, and identify user’s intent to generate the best response possible. To do so, it might require accessing databases or calling external APIs. Chatbot’s typical architecture is present in Figure 2.4. Although not included below, chatbots might have Automatic Speech Recognition (ASR) and Text To Speech (TTS) modules to interact in spoken language.

![Figure 2.4: Chatbot Architecture](image)

### 2.5.1 Natural Language Understanding

Natural Language Understanding (NLU) is responsible for extracting semantic information and meaning from natural language in order to create grammatical data structures that can be processed by the Dialogue Management unit in the next step [18]. NLU must be trained wisely and with a multitude of examples to correctly identify message’s intent and entities.

If an user asks “Can you please show me the trains tomorrow from Lisbon to Porto?”, the NLU should identify that the user’s intent is to get the schedule, but also identify the origin, destination and date entities that were mentioned. The context should also be updated in order to recall what is the current conversation topic, either directly with in a NLU engine or locally in Dialog Management. NLP can be considered as a conjugation of NLU and Natural Language Generation (NLG), and the three main concepts in NLU are explained in detail below:

- **Entities** represent concepts that are often specific to a domain as a way of mapping phrases to approved phrases that catch their meaning.

- **Intents** represents a mapping between what a user says and what action should be taken.

- **Context** are strings that represent the current context of the user expression. This is useful for differentiating phrases which might be ambiguous and have different meaning depending on what was spoken previously.
2.5.2 Dialog Management

Also known as action planner, this module must maintain conversation context and plan the right sequence of service actions for incoming messages. Depending on user intention it must decide if it is necessary to retrieve additional information to generate a proper response, such as accessing an database or call an API. Finally, it either forwards a final response to NLG or the structured information required to generate it.

2.5.3 External API Calls

Integration with external services is useful when users request specific information or tries to invoke operations that cannot be processed locally. These include calls to databases stored in the cloud, typical APIs to retrieve information, and even payment gateways like Paypal. There is no limit for external calls but they should be performed quickly because the user is waiting and some platforms can even block replies after a certain period of time.

2.5.4 Natural Language Generation

NLG receives as input a structured representation of information required to generate a response to the user. This conveys information about who is speaking, the dialogue history, and the context [17]. For chatbots available in many channels, this module must take into account which richer message elements are available in each platform. This module should implement the desired personality and be consistent in its answers. Additionally, it can even implement multiple personalities to interact with different users.

As previously seen, the responses can be either retrieved from a knowledge base or generated, where both have pros and cons. Due to the repository of handcrafted responses, retrieval-based methods can avoid grammatical mistakes. However, it may be unable to handle unseen cases for which no appropriate predefined response exists. Generative-based models models are hard to train, and quite likely to make grammatical mistakes, especially on longer sentences, and typically require huge amounts of training data.

2.5.5 Chatbot Development Framework

A Chatbot development framework is a set of predefined functions and classes which developers use for faster development. It provides a set of tools that help write the code better and faster. They are used by coders to build chatbots from scratch using programming languages. There are two popular alternatives, Howdy's Botkit and Microsoft Bot Framework (MBF). Both these frameworks:

- Are open source.
• Can integrate with popular messaging platforms.
• Have good documentation.
• Have an active developer community.

Building with MBF allows automatic message conversion to any frontend but it has the downside of adding an extra module in chatbot architecture and can lead to higher latency, specially if the bot is not hosted in Azure. BotKit has libraries for many messaging frontends, connecting backend directly to the user, allowing the use of new features when IM platforms are updated.

2.5.6 Chatbot Platforms

Chatbot frontends are online ecosystems where chatbots can be built and deployed. They allow interacting with third-party services and can integrate with multiple frontends in a few clicks. The user interface is user-friendly, generally drag-and-drop based and is more suitable for beginners or non-technical users to develop bots without coding, making it easy and cheap to launch a bot. In comparison to coding from scratch it brings some limitations, like setting complex logic rules [12].

Manychat and Chatfuel are the most popular chatbot platforms with active communities, and with free plans, but PRO plans might be required for big audiences, to access limited features and to remove their branding. Figure 2.5 includes most of Chatfuel plugins to build the conversation flows, which include payment support, user input validations, from many others.

![Figure 2.5: Examples of Chatfuel's plugins](image-url)
2.6 NLU Alternatives

At the moment, the most popular NLU services available are Dialogflow, Lex, LUIS, Watson and Wit.ai, and they belong to Google, Amazon, Microsoft, IBM and Facebook, respectively. All of them are cloud platforms based on machine learning, provide a Software Development Kit (SDK) for a variety of programming languages and support different natural languages. Table 2.1 compares them based on the following taxonomy [19]:

- **Usability** indicates the ease of use of the user interface provided by the NLU platform. It can assume the following values:
  - *high*, if simple and intuitive.
  - *medium*.
  - *low*, if difficult to use for a developer and without supporting documentation.

- **Languages** represents how many languages are supported.

- **Webhook SDK Availability** indicates whether and how a developer can integrate the chatbot with other software, independently from the ones in the cloud platform.

- **Pre-build Entities** reports how many pre-build entities are offered in English agents. Such examples are detecting locations, numbers or dates.

- **Pre-build Intents** indicates how many pre-build intents are provided for English agents. They are usually for specific tasks, like currency conversion or weather forecast but can also cover small talk topics.

- **Default Fallback Intent** indicates whether the platform has a fallback mechanism for intents. This allows the proper classification of intents that are not recognised as part of existing intents.

- **Context Support** reports if whether the NLU platform can manage the context in a conversation or it must be left to the developer’s code.

- **Integrations** represents how many third-party integrations are available in the cloud platform. They are usually popular messaging platforms and personal assistants but can also be SMS services or web widgets.

- **Price** is the pricing for using the NLU platform.

Rasa NLU is an open source alternative that does not provide the advantages of cloud-based solutions, such as managed hosting and scalability, but on the other hand, it leverages adaptability, and full data control because no third-party is required to process valuable training data. It processes incoming
messages in a so-called pipeline. There are loosely coupled modules for entity extraction, for intent classifications, pre-processing, and others [20].

Rasa Stack is a combination of Rasa NLU and Rasa Core that provides all machine learning tools to build contextual AI assistants. It is used by thousands of developers and even in fortune 500 companies such as Allianz, AXA and BMW [21]. Although it can used to build assistants in any language, it requires a vast amount of training data to work properly.

### 2.7 Payments in Chatbots

Adding payments capabilities to messaging apps can increase their usage and the chatbots hosted on them, and at the same time generate revenue. Integrating payments to chatbot ecosystem is a very important step to compete with mobile applications. WeChat showed how adding payments to the chat experience can have vast implications on user engagement. In China, the chat app accounts for roughly 35% of the total mobile internet time spent. This allows users to make P2P payments as well as pay for goods without leaving the app. Below I will present a variety of ways to integrate payments in chatbots.

#### 2.7.1 Payments with Middleware

These payments are performed using payment provider API, such as Paypal and Stripe. The chatbot sends a request to the API endpoint and it returns a link which the user will click through a button. After that, the customer is redirected to the middle layer in a webview on top of the conversation and carried indirectly in the messaging app. This webview will contain either a login or a form to enter payment details, and in the end it returns the chatbot payment results. The chatbot analyses the response and

<table>
<thead>
<tr>
<th></th>
<th>Dialogflow</th>
<th>Lex</th>
<th>LUIS</th>
<th>Watson</th>
<th>Wit.ai</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usability</strong></td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Languages</strong></td>
<td>15, plus variants</td>
<td>Only English</td>
<td>10</td>
<td>13</td>
<td>50</td>
</tr>
<tr>
<td><strong>Webhook/SDK Availability</strong></td>
<td>Both</td>
<td>SDK</td>
<td>Both</td>
<td>SDK</td>
<td>SDK</td>
</tr>
<tr>
<td><strong>Pre-build Entities</strong></td>
<td>58</td>
<td>98</td>
<td>14</td>
<td>26</td>
<td>29</td>
</tr>
<tr>
<td><strong>Pre-build Intents</strong></td>
<td>10</td>
<td>15</td>
<td>22</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>Default Fallback Intent</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Context Support</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Integrations</strong></td>
<td>17</td>
<td>4</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>Price</strong></td>
<td>Free, up to 180 req/min</td>
<td>Free, for one year (with limits)</td>
<td>Free, up to 10k req/month</td>
<td>Free, up to 10k req/month</td>
<td>Free</td>
</tr>
</tbody>
</table>

Table 2.1: NLU platforms comparison
informs the user if the payment was successful. Using this method to support payments has the benefit of being supported in any messaging app with multi-factor authentication.

2.7.2 Payments with platform API

Some IM providers, like Facebook Messenger, Telegram and WeChat, also support payments inside their channels. In Messenger platform, as demonstrated in Figure 2.6, developers create a generic template containing the product or service, the price and a Buy button that opens a dialog when tapped. This dialog will display all necessary information to pay instantly, and for convenience can include optional fields, such as phone number or shipping address. When the Pay button is tapped, chatbot backend receives tokenized credential and requested user information, and should confirm the purchase with a receipt template. This feature is still in beta on messenger platform and is only supported in the USA [22]. Regarding Telegram Bot Payment API, it does not process payments from users and instead relies on different payment providers and works in more than 200 countries.

The biggest advantage of payments using platform API is the fast checkout process with support for a multitude of payment methods. But, as they only act as a messenger between the bot developer, the paying user, and their chosen payment system. Since platforms don’t process the payments, no sensitive data can be accessed. Due to this structure, they are not able to handle complaints.

![Figure 2.6: Example of payment with Messenger API](image)
2.7.3 Payments through Mobile App

These kind of payments are becoming popular in Portugal, where user provide their phone number and in few seconds receive a notification to accept or reject the payment inside their App. This has the disadvantage of forcing the user to leave its messaging platform, although allow faster payments than seeing or generating a credit card. In Portugal, MB Way is the most used payments app with more than 1.5 million users and developers can integrate with their system by executing the following steps [23]:

- Register in developer area of their website to access technical details
- Implement and test technical interfaces
- Request a technical certification of the implementation

After the implementation, the process works by invoking an asynchronous request with user phone number, merchant information and details about the purchase to their API. The user then accepts or rejects the operation, and confirm payments using bio-metrics or a secure pin. Finally, MB Way responds whether or not the operation succeeded.

2.8 Chatbot Evaluation

2.8.1 Radziwill and Benton

Radziwill and Benton, in 2017, examined the academic and industry literature to provide a comprehensive review of quality attributes for conversational agents, and to identify appropriate quality assurance approaches. They extracted quality attributes from 32 papers and 10 articles, and grouped them based on similarity. The complete list of the quality attributes is present in their paper, but in general, they were aligned with the ISO 9241 concept of usability: “The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments.” [24]. The chatbot’s effectiveness relates to the completeness and accuracy with which users can achieve their goals during the conversation. The efficiency refers to how well resources are applied to let the users achieve those goals. Lastly, the satisfaction refers to how the user feels about the interaction with the chatbot.

They present a goal-oriented approach for quality assessment of a dialog system that can be used to compare different agents, as well as, two versions of the same system to see if quality has improved over time. They also concluded that the quality attributes table, present in their paper, should be used as a checklist for a chatbot implementation to make sure they address key issues. The method presented is based on the following steps [24]:

1. Prioritise quality attributes with appropriate metrics for the use case and create an hierarchical model, Figure 2.7 was the one used in their investigation.
2. Create pairwise comparisons between category attributes, expressing the relative merit of each quality attribute as compared with the others.

3. Within each category create a new priority matrix, in this case would be one for each of the four top-level quality attribute categories.

4. Use the measured values of the lowest-level attributes, to create one priority matrix for each of them.

5. Use Analytic Hierarchy Process (AHP) [25] to accomplish the comparison. The consistency score should be reasonable, below 20%. High value consistency values indicate some discrepancy in the individual assessments, for example, prioritising all elements on one level of your hierarchy low with respect to each other. Their AHP results are present below, which weighted more heavily the OLD chatbot given the categories and attributes priorities that were set.

<table>
<thead>
<tr>
<th>Select Between Old and New Chatbots</th>
<th>Weight</th>
<th>OLD</th>
<th>NEW</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessibility</td>
<td>100.0%</td>
<td>66.2%</td>
<td>33.8%</td>
<td>18.4%</td>
</tr>
<tr>
<td>MeaningIntent</td>
<td>54.5%</td>
<td>39.1%</td>
<td>15.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>SocialCues</td>
<td>47.7%</td>
<td>35.7%</td>
<td>11.9%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Performance</td>
<td>6.8%</td>
<td>3.4%</td>
<td>3.4%</td>
<td>0.0%</td>
</tr>
<tr>
<td>UnexpectedInput</td>
<td>32.1%</td>
<td>24.6%</td>
<td>7.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Escalation</td>
<td>28.1%</td>
<td>21.1%</td>
<td>7.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Affect</td>
<td>4.0%</td>
<td>3.5%</td>
<td>0.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Entertaining</td>
<td>9.4%</td>
<td>1.6%</td>
<td>7.8%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Personality</td>
<td>7.8%</td>
<td>1.3%</td>
<td>6.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Humanity</td>
<td>1.6%</td>
<td>0.3%</td>
<td>1.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>SpecificQs</td>
<td>4.1%</td>
<td>1.0%</td>
<td>3.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>ThemedDiscussion</td>
<td>1.9%</td>
<td>0.3%</td>
<td>1.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Transparent</td>
<td>0.4%</td>
<td>0.2%</td>
<td>0.2%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Figure 2.7: Example of AHP results

### 2.8.2 Cas Jongerius

Cas Jongerius, in July 2018, submitted his thesis on *Quantifying Chatbot Performance by using Data Analytics*, presenting an evaluation that can be distinguished between questionnaire-based methods, that reflect Quality of Service (QoS) and Quality of Experience (QoE), and through automatic metrics
that purely reflect the data. He recommends to combine QoS and QoE evaluations in order to get a complete view of the system, although a prior attribute selection must be executed, as some of metrics don’t apply to all chatbots. In both, the quality attributes are scored on a 1-5 Likert scale, allowing to get individual scores by category and a global score by averaging them [26]. The format of a typical five-level Likert item, for example, could range from 'very unsatisfied' to 'very satisfied'.

Quality of Service metrics are based on an expert review, requiring an objective perspective and a deeper understanding of the chatbot, in order to capture its performance. For this evaluation he refers to the quality attributes present in Radziwill and Benton paper. Quality of Experience measurements are based on user’s opinion and reflect user’s satisfaction and the perceived performance of the system. By letting normal users review chatbots with a collection of other evaluation criteria, it’s possible to measure how they perceive and experience the interaction with the chatbot. Multiple researchers have created methods to assess user’s perceived performance of a chatbot, although many of them use characteristics that overlap. A selection of the relevant chatbot characteristics should be performed, and a combined list is presented below: [26]

- Impression
- Command / Robustness
- Effectiveness
- Navigability
- Ease of Use
- Aidability
- Comprehension / Clarity
- Naturalness
- Friendliness / Personality
- Visual look
- Process feedback

Automatic Metrics focuses purely on metrics that can be automatically measured by analysing the conversations of chatbots. By specifying the different attributes and their possible values, a solid foundation is created for performing consistent data analyses. In combination to the source of the messages (User, Chatbot or Mixed) can be scored on three different levels: [26]
• Dialogue act level is the lowest level for which a metric score can be determined. It is usually the same as one chat message but users can send multiple dialog acts at once or split it up over two or more messages.

• Conversation level is a collection of all the individual dialog acts in a conversation into one set. A metric score can be calculated by summarising the metric scores of all the dialogue acts in the corresponding conversation.

• Entity level is the highest level, where all the conversations of an entity are collected into one set. A metric score can be calculated by summarising the metric scores of all the dialogue acts from the corresponding entity.

One thing to have in mind, is that the automatic metrics and the perceived performance score can be influenced by numerous attributes. The modality of a chatbot can be text-based, voice-based or visual-based, and advanced chatbots can make use of multiple modalities. The device represents the device on which the users interact with the system, like a browser on a laptop or a mobile app. The style represents chatbot’s purpose, which can be either goal-oriented (GO) or non-GO. Lastly, maturity concerns the advancement of the chatbot. Often Q&A dialog system can only recognise keywords, while more advanced ones make use of NLP. The more advanced the NLP and learning techniques, the higher the maturity level. According to Snijder [26], AI expert, the maturity of chatbots can be subdivided into five levels, present in Figure 2.8.

![Figure 2.8: Chatbot Maturity Levels](image)

The metrics present in his research focus mainly on text-based chatbots, for both GO and non-GO,
web browser chatbots on maturity level 3, because these chatbots are the most common ones and could profit most from the findings [26]. Numerous automatic metrics that they discuss are inspired by previous research that was based on a text-analysis program called Linguistic Inquiry and Word Count (LIWC) [27] to capture people’s social and psycho social states. A majority of the important automatic metrics he investigated about are presented below:

- **Sentiment** analysis is the discipline that analyses evaluations, sentiments, opinions, emotions, and attitudes from written language [28]. It has discovered that a chatbot that reads and responds to moods of human participants improve the user’s satisfaction [29].

- **Response Time** of chatbot should be less than a customer service representative, although nearly instant answers could evoke a negative artificial feeling from the users [16]. IBM reported, in 2017, that besides the advantage that chatbots are available 24/7 and generally support multiple concurrent conversations at once, the response time of chatbots is often a lot quicker than that of a human [30].

- **Word Count** represents the straightforward number of words in a message. There is no scientific research that relates the length of chatbot sentences to the perceived performance of the dialog system. Although, in 1997, Jakob Nielsen claimed that shorter texts better fit the reading behaviour of online users [31].

- **Turn Count** simply represents the number of dialogue acts that were sent in the conversation, and this metric can only be measured for the conversational and entity level.

- **Readability** represents the ease with which text can be understood, and multiple researchers have created algorithms that aim to score text’s readability. Even though these formulas are not fully accepted they do provide some insights into the complexity of a piece of text [26].

### 2.9 Transportation Chatbots

Many companies are already implementing chatbots in an attempt to improve they customer experience. Regarding train services I could only find OUIbot, developed specifically for French railways (SNCF) and only available in French. The other chatbots I found in transportation sector are targeted at flight searches. All of them only support English language and present the results using carousels. In table 2.2, some popular chatbots implementing customer service in the transportation sector are compared in terms of:

- **Channels**, which indicates different frontends in which they can interact with the chatbot. Typically, companies prefer messaging platforms and personal assistants over web plugins.
• **Context Management**, is required to provide a smart conversation, allowing to understand changes on user intent, as well as avoiding asking previously answered questions.

• **Booking**, representing how the booking process is handled after customer selects a trip. This includes introducing passenger details and payment processing, and can be done in the chatbot, via a redirection to the company’s [website](http://oui.sncf/bot) or through an [app](http://oui.sncf/bot).

• **Response Time**, which indicates how many seconds it takes for the chatbot to respond to the query “Show me tomorrow flights from Lisbon to Porto”. To test the response time in OUIbot, the query was “Montrez-moi demain les trains de lyon à monaco”.

• **Persistent Menu**, although currently only present in Messenger and in custom mobile applications, it’s a useful feature when the user gets stuck and wants to start over.

• **Human Handoff**, representing if the user can request a human representative to answer requests that are not automated.

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Context Management</th>
<th>Response Time (using NLP)</th>
<th>Booking</th>
<th>Persistent Menu</th>
<th>Human Handoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expedia</td>
<td>Alexa, Messenger, Skype</td>
<td>No</td>
<td>20s 26s 13s (Fallback)</td>
<td>Website</td>
<td>No</td>
</tr>
<tr>
<td>Hipmunk</td>
<td>Slack, Messenger, Skype</td>
<td>Yes</td>
<td>17s 24s 23s</td>
<td>Website</td>
<td>Yes</td>
</tr>
<tr>
<td>Kayak</td>
<td>Alexa, Google Assistant, Messenger, Slack</td>
<td>No</td>
<td>40s 29s 16s 7s</td>
<td>Website</td>
<td>Yes</td>
</tr>
<tr>
<td>OUIbot</td>
<td>Website</td>
<td>Yes</td>
<td>3s</td>
<td>Website</td>
<td>No</td>
</tr>
<tr>
<td>Mezi</td>
<td>App</td>
<td>No</td>
<td>67s</td>
<td>App</td>
<td>Yes</td>
</tr>
<tr>
<td>Skyscanner</td>
<td>Alexa, Messenger, Skype</td>
<td>Offline Offline No</td>
<td>Offline Offline 32s</td>
<td>Website</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table 2.2: Travelling Chatbots Comparison**

OUIbot is a custom made chatbot solution with proprietary design and it is not integrated with any IM platform. It is available at [http://oui.sncf/bot](http://oui.sncf/bot) and their train schedules are represented in a carousel. By hovering at the price it transforms into a button to obtain more details regarding the selected trip and an option to buy it. Examples of this interaction can be seen in Figure 2.9. Their bot is not present inside the OUI mobile app although users can authorise it to send personalised train times, when it knows they need it at a specific time. By logging into OUI account, it is also possible to save time on bookings and create alerts when low fares are available.
Expedia enables travellers to easily search for and make a hotel booking, or manage select elements of travel bookings, including hotel or flight confirmations or flight cancellations. They are present in Alexa devices, Facebook Messenger and Skype. Sadly, their Alexa skill can not understand most of the requests and the Skype version is very limited compared to the one in Messenger, as it can only help users that choose options from an initial menu and then follow the guided conversation. In both, after two fallback messages it suggests users to go back to menu or request a customer service representative. It was not able to recognise Porto (neither Oporto), so the response time was measured with Milan as destination. Before showing the results of the request it re-prompted the date of the flight, which should have been recognised.

Hipmunk is a platform that aims to help users plan trip and book reservations. Hello Hipmunk is their virtual travel agent powered by AI that is available on Facebook Messenger and Skype. It can understand flight preferences, the context of the conversation, and even give advice to user regarding best time for a particular trip. I could also notice that it keeps search results in some cache which can improve efficiency of certain requests.

Kayak introduced their Alexa skill, Facebook Messenger and Slack chatbots in 2016, which lets users...
search for flights, hotels and rental cars through text or voice. It can also give ideas on where to go within a budget, show when is the best time to travel, and can alert users regarding travel status. Trying to search for flights using Kayak Alexa skill was not easy as it could never identify the flight date, and airports were wrongly identified or not identified at all. In addition, it does not allow searches for one-way trips nor does it allow to modify search parameters. Through Messenger users can interact in two different ways as demonstrated in Figure 2.10. The fastest, works by applying NLP to user utterances, followed by a list of options in a carousel, with buttons to book the trips in their website. Alternatively, users can provide flight details using a webview, that is purely a zoomed website version in computers, but through mobile version of Messenger provides a very intuitive website.

![Figure 2.10: Kayak interactions in Facebook Messenger](image)

Mezi is a recent acquisition by American Express that introduces itself as a personal travel assistant. More than 60% of all flight requests are served by Mezi chatbots and it provides an end-to-end flight, hotel and dining booking experience. The user must download and install their app, either on Android or iOS, and at the time of this writing, it is not available in some specific countries stores, like the Portuguese App Store. Before using their in-app chat interface, it requires a valid US or Indian phone number, which is followed by nine questions regarding user preferences, for example, most often origin airport, favourite airlines, and an average of travels per year. Then, Mezi users can perform requests by typed messages or by being guided through buttons, calendars, and other types of UI present in the app. It takes longer to show results because it requires additional flight preferences.

Skyscanner was the world's first travel search brand to create a voice search skill for the Amazon Alexa, and launched their Facebook Messenger bot in May 2016. In addition to giving you the ability to type a destination, departure airport, and dates to search for flights, it is also trained to give inspirational destination suggestions based on user search trends and cheapest pricing. Sadly Messenger bot was shut down and Alexa says "sorry, i am having trouble accessing your Skyscanner flight search skill
right now", although their Skype chatbot works really good in English and the NLU behind it can even understand contextual messages as well as recognising misspellings.

### 2.10 Serverless Applications

There are multiple ways to host computer programs, and with no doubt they have been shrinking to smaller and more manageable services. Virtual machines came after shared/physical machines to provide dedicated OS per app. Containers were then introduced to manage the OS and server, allowing to run software more efficiently on the same hardware. Finally, serverless architecture provide highly scalable applications without the management overhead. They take advantage of the elasticity and distributed computing power, and cost according to consumed resources. Function-as-a-Service (FaaS) consists on building programs by composing stateless functions that run in response to a trigger. Figure 2.11, taken from a Google Next’18 conference, represents this evolution into serverless computing.

Building applications on FaaS requires data management in both persistent and temporary storage, since each function execution is isolated, as well as mechanisms to trigger and scale function execution. As a result, cloud providers support FaaS with a “standard library” of services that combined can be use to build programs [32]. In Google Cloud Platform, these include Datastore (NoSQL database), Storage (large object storage), and more. This entire infrastructure is managed and operated by the cloud provider and developers simply register functions that uses these services and receive “pay-as-you-go” bills that scale up and down according to their storage and compute usage.
The same paper provides a view on simple cases where FaaS can offer benefits, and where it is incapable of providing an efficient solution. In short, they are attractive solutions for simple workloads of independent tasks, more specifically by the following three categories, divided based on the nature of interaction between function invocations:

1. **Embarrassingly parallel functions**: For applications where functions never need to communicate with other functions. They directly exploit auto-scaling features to scale up or down on demand because independent requests never need to communicate with each other.

2. **Orchestration functions**: These leverage the use of functions to orchestrate calls to existing autoscaling services, and a common use case is to prepossess events enabling efficient data manipulation at scale.

3. **Function composition**: These consist of applications that are compositions of small functions and typically include workflows of functions chained together via data dependencies.

They also refer some limitations of FaaS such as having functions with limited lifetimes, being impossible to ensure the same function instance handles user connections, multiple functions on same Virtual Machine (VM) sharing network bandwidth, along with no mechanism to access specialised hardware. Function’s execution time also depend if the function has run recently, also called as “warm” start and thus it reuses resources from past evocations. On the other hand, the “cold” scenario happens for the first execution or if it has been has been idle long enough. The limited lifetime of functions brings a higher latency in cold start scenarios, but on the other hand, it can be very cost-effectiveness by scaling down to zero instances for long periods without execution.

### 2.11 Chatbot’s Privacy

Privacy can be summarised as the right, claim and ability to limit, filter, and control the flow and use of personal data [33]. With the technological development, data is being collected everywhere, and people are expected to know the risks they expose themselves to by using such services. Terms of service are complicated and often require one click to agree, leaving people unaware of what type of data they share, with whom and what it is used for [34].

The General Data Protection Regulation, or GDPR, is the newly adopted data protection directive that was adopted by the European Parliament. It is active since 25th May, 2018 and applies to all companies either located in EU member states or companies that process data of European citizens. According to the regulation, personal data is any information that relates to an identified or identifiable living individual. This is harder than it sounds, because an email address that includes
name.surname@company.com, location data or an IP address might count as personal data too. Companies must nominate a Data Protection Officer (DPO) to ensure the execution of all regulations and to serve as a point of contact for users with data concerns. Subcontractors are subject to the same rules as companies who first collected user data, and companies must inform users of the object of data collection prior to information transfer. The DPO is then responsible to:

- Determine which pieces of data are collected, and which are personal and which ones are sensible
- Make sure the user is informed regarding data collection before the collection of his/her data
- Describe where the user data is kept and provisions to protect it
- Decide who can access the data

With these responsibilities mapping completed, the DPO should conduct audits and make the team is aware of personal data protection, as well as enforcing the following procedures:

- Make sure personal data is encrypted to avoid leaks in case of theft
- Ensure the company’s password policy is strong enough to secure the access of the data
- Possibly anonymise collected data to ensure personal data protection while maintaining the statistics
- Answer requests for information by the data subject
- Ensure the detection of data breaches
- Plan to communicate data breach incidents to the authorities and to data subjects

Chatbots in which the messages are sent to a third-party NLU or any other service, are responsible to notify them regarding users that requested their data deletion. If possible, bot should locally identify personal data included in messages and remove them before forwarding the message to external services. Regarding custom chatbot platforms, such as a website plugin or inside a mobile app, it is possible to ensure messages are only intercepted by the sender and the dialog manager, but if the bot resides inside an IM platform users should take into account that all of them, except WhatsApp, can see every message.

Facebook makes clear in their platform policy that they can "analyse the apps, website, content and data for any purpose, including commercial" [35]. They also outline that the data is processed for a number of reasons including:

- Improving their services from a technological/usability perspective
- Promoting safety and security, including verifying accounts are real
- Improving their advertising services.
2.12 Chatbot Analytics

Analysing the chatbot's performance is essential to optimise its response effectiveness and understand user behaviours. Based on past interactions it is possible to identify new or more optimised conversation paths. Training data and the logic behind the conversation should be continuously updated based on mishandled and unhandled intents. This is crucial to improve NLU failure rate over time and reduce human escalations. Live transcripts are a useful feature to monitor, pause and intervene when users present complex requests. Manually look into live transcripts becomes more difficult when the users increase and thus, allow to search transcripts by: Date, Intent, User, Session length or sentiment.

These analytics aim at obtaining more NLU training data, new feature requests, session flow tracking and where users abandon conversations. They should also segment audiences for re-targeting messages, and discover how to optimise the conversation funnel. In general, these analytic tools try to increase user satisfaction and engagement. Below, two of the most popular chatbot analytics solutions are presented.

2.12.1 Dashbot

With Dashbot, it is possible to track dozens of metrics such as, how many users interact with the chatbot, which time of the day has more request and retention rates. Conversation paths shows what kinds of questions users are asking at various points, and how the conversation progresses typically from there. They also provide a funnel feature to measure conversation goals, as shown in Figure 2.12, although it is not included in the free version. Dashbot supports most popular IM platforms, SMS, Twitter, and provides a cross channel dashboard.

![Figure 2.12: Conversation Funnel in Dashbot](image)

<table>
<thead>
<tr>
<th>Step Name</th>
<th>Mentions</th>
<th>Conversations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamemonk installed</td>
<td>1,632,150</td>
<td>6,779</td>
</tr>
<tr>
<td>Started game</td>
<td>86,508</td>
<td>1,594</td>
</tr>
<tr>
<td>Participated in game</td>
<td>1,373,304</td>
<td>850</td>
</tr>
</tbody>
</table>

May 1, 2017 to January 3, 2018
2.12.2 Botanalytics

Botanalytics is an AI powered conversational analytics platform provides deep insights on users interactions. It uses AI to enhance conversational success with sentiment analysis, intent detection, A/B testing and more. The dash dashboard shows message and session metrics over custom periods of time. They provide a graph with conversation paths segmented by intents, useful to discover improvement areas like bottleneck points that make user leave. They also show clustered messages to identify unhandled, as shown in Figure 2.13.

Their audience builder aims at saving money, getting greater ROI by sending advertisement only to specific users. In addition the segmentation feature allow to track specific phrases in conversations to observe funnel completion rates and conducting A/B tests with different funnels. Their free plan includes 5K messages per month in one single project with 30 day data history, for a single team member and one segmented event.

![Figure 2.13: Clustering in Botanalytics](image-url)
3 Implementation

Contents

3.1 CP contextualisation ........................................... 35
3.2 Analysis ............................................................ 35
3.3 Architecture ....................................................... 36
3.4 Design and Prototype ........................................... 42
3.5 Development ..................................................... 43
3.1 CP contextualisation

Comboios de Portugal (CP) is the largest railway company in Portugal, with more than 400 stations. They interact with customers through a phone line, an online chat, and a mobile application. They are also reachable in Facebook with around 230 thousand followers and 25 thousand in Instagram. Festive occasions with bigger affluence of people, and crowded stations recurrently make clients miss their trains in waiting lines because current booking alternatives are not efficient enough. To better understand the challenges behind selling CP tickets online, they can be divided into three types:

- **Long-Distance** includes both Intercity and Alfa Pendular trains. These correspond to trains with reserved seats associated with a personal document number, and the ticket verification performed inboard. They are already for sale in both website and mobile app.

- **Regional** trains can only be sold online if in conjugation with a Long-Distance train. In this service there are no marked seats and an online solution would have to make sure tickets get validated upon usage.

- **Urban** stations might have physical gates which currently only open with a valid VIVA transportation card.

A solution based on vouchers is being used for the second year to sell VIVA transportation cards with unlimited trips for either 1, 3 or 5 days. The purchase generates a 16-digit number which customers exchange for the physical card in any Metro vending machine.

3.2 Analysis

The goal of this project was to implement a customer support chatbot for CP clients, aiming to improve user experience when interacting with the company. Its target users are mostly Portuguese customers but it also has an English agent aimed at tourist passengers, which usually are the ones that take longer to buy tickets in stations. According to a study from Marketest, 95,3% uses Facebook, 74,2% uses WhatsApp, followed by 70,8% on FB Messenger and 67,9% on Instagram from a total of 809 respondents [36]. WhatsApp didn’t have an API for developers at the time, and as a result, Facebook Messenger was the primary channel chosen for this chatbot.

Its main functionality is providing online booking of trips with the motivation of reducing queues in stations, both machines and office desks, and at the same time help the customers instantly and wherever they are. This proposal will simulate bookings for urban trains, which are delivered using a system of vouchers. After paying the user will receive a voucher associated with the purchase, which is then exchanged for a VIVA card in the stations.
To increase chatbot’s value it will support some intents other than providing timetables with respective booking options. These include the ability to find nearby stations, show active alerts or promotions and allow users to subscribe new alerts of multiple service categories. The bot will also be trained to provide useful information that is present in their FAQ (Lost and Found, Luggage transport, and more), as well as alternative ways to contact CP, which may be useful for users with complex requests that are not automated.

The requirements selected to implement in this project were

- Provide schedules and respective ticket selling
- Show next arrivals and departures on CP stations
- Automatically respond to Frequently Asked Questions (FAQ)
- Show active alerts
- Send alerts based on user’s preferences
- Provide help for reduced mobility customers
- Show alternative ways to contact CP
- Search nearby stations
- Understand basic small talk

### 3.3 Architecture

Before defining the final architecture, I tried Microsoft Bot Framework and Botkit Development Kit because they would ease the process of connecting to multiple channels, and already provide high-level functionalities which are less error-prone. They were easy to install and use, although I discarded Microsoft’s alternative because it would require an extra module in the architecture. Botkit’s alternative, being local, don’t have this limitation, although it maintains bot’s state and require the server to be always running even without requests for long periods of time. To overcome this, the final version of the Dialog Manager became a stateless application hosted in Google Cloud Functions which can scale up when needed but also saves resources by sleeping when there are no requests. It based on microservices that don’t store client data and being loosely coupled, it allows updating different parts of the architecture, instead of the whole application.

Only Wit.ai and Dialogflow support Portuguese language, but by testing both I found Google’s alternative more appealing with more features and a better usability. Also, Wit.ai only supports NLU for bots with a single language per project and require developers to host the backend on a third-party
provider. Based on these facts, Dialogflow was the NLU chosen for intent detection and entity extraction, and allowed hosting all architecture components in Google’s infrastructure, with improved security measures.

Figure 3.1 represents the current architecture behind CP Bot, and the sequence of numbers represents the order of steps that might be required to generate the select the response. Sending the message to Dialogflow can be skipped if the user clicks on buttons or quick reply. The same happens for calling External API whenever the necessary information to respond is present locally. Also, users can receive proactive messages depending on their subscription preferences.

**Figure 3.1: CP Bot Architecture**

### 3.3.1 Channels

Chatbots interact with users on channels, usually IM platforms but can also be accessible inside websites, mobile applications or even Twitter direct messages. There is no standard among these social
networks for prioritisation, routing, or error handling, and different channels are popular in different regions [37]. Currently, CP bot is only available through Messenger but might be available through other channels if needed.

### 3.3.2 Dialog Manager

This is the main module of the architecture and is responsible for handling every user message. It is connected to Dialogflow, the Database, and external APIs, which are orchestrated according to the conversation context and received message. Being a multi-language bot it must obtain all user information, including conversation language and context, from the Database. With user data obtained, it then forwards the user message to one of the following handler functions:

- Analytics
- Active Alerts
- Change Language
- Fallback messages
- FAQ
- GDPR
- Favourite Stations
- Near stations
- Next trains
- Plan Journey
- Questionnaire
- Subscriptions
- Tickets
- User test
3.3.3 Dialogflow

NLU is a mandatory module for this chatbot because of all the advantages that were presented in previous sections. At the moment, Dialogflow and Wit.ai are the only NLU engines that support Portuguese (pt-PT) language. Dialogflow was my final selection mainly because it allows to create multilingual agents and to import/export training data which will be very useful if the NLU engine needs to be exchanged in the future. At first, I found that pt-PT support was quite limited, by not recognising misspellings, or ignoring parts of the input that are irrelevant to identify the intent. During 2019, they introduced a spell checking which works quite well for small syntax errors, both grammatical and misspelled entities.

3.3.4 External API Calls

The chatbot require some external integration to obtain real-time information. Through them I can provide schedules, live departures times, get directions to stations, and even connect to online payment service providers. CP API is used to obtain requested schedules, and may soon allow real ticket emission. Currently it sends user’s origin, destination, and date of travel to receive respective schedule. Infraestruturas de Portugal (IP) API is required to access live departure times of trains in stations. PayPal has more than 250 million users from all over the world [38], and can also accept payments from guest users which pay with credit cards. QuickChart was only integrated while writing this thesis, and came from a necessity to analyse data from testing users.

3.3.5 CP

CP representatives might want to notify their customers about some relevant topic. These notifications could be the suppression or reinforcement of train services, promotions or other any other relevant alerts regarding their train services. Through an API endpoint they can integrate with the Notification Manager to send Portuguese and English versions of alerts to subscribing users. Currently, the bot supports the following subscriptions topics:

- Alfa Pendular
- Celta line (Oporto - Vigo)
- Coimbra urban trains
- Intercity
- Interrail
- Lisbon urban trains
• Lusitania line (Lisbon - Madrid)

• Oporto urban trains

• Promotions

• Regional and Inter-Regional

• Strikes

• Sud Expresso (Lisbon - Hendaye)

3.3.6 Database

The database being used in this project is Cloud Datastore, Google’s NoSQL database for web and mobile applications. It automatically handles sharding and replication, providing an highly available and durable database that scales automatically allowing to maintain high performance as traffic increases. It integrates via a RESTful interface and provides ACID transactions, SQL-like queries, indexes and more [39]. This database is responsible for maintaining which alerts are published, which fallback messages are accumulated for NLU training, and the following attributes for each user:

• Name

• Conversation contexts

• Favourite stations

• Topics subscribed

• Agent (Portuguese or English)

• Start and finish times for each task executed

• Fallback counter

• Last schedule searched

Additionally, it saves feedback submitted by users, which was shown as a button for those who performed in user testing. A new functionality of automated testings, still in development, will be based on collected conversation examples. Each functionality implemented will have a document with user messages and their respective answers, avoiding manually testing them.
3.3.7 Notification Manager

This module is an API endpoint responsible for sending proactive messages to users. When a CP representative adds a new alert, this is responsible to forward it to users depending on their subscriptions. This module can also be used to send follow-up messages to users that stopped interacting with the bot. To create new alerts the request must include the following parameters:

- Alert Topic
- Title (max 80 characters)
- Subtitle (max 80 characters)
- Image URL
- More Info URL
- Title (PT)
- Subtitle (PT)
- Image URL (PT)
- More Info URL (PT)
- (Optional) From Date
- (Optional) Until Date
- (Optional) Testing - Admin will verify both EN and PT versions before broadcasting

3.3.8 Natural Language Generation

This retrieval-based module is populated with templates of responses for all supported intents. Choosing the answer from a finite list it is possible to guarantee the messages won’t have grammatical mistakes and even allow to create a sense of personality. In case users interact with the chatbot through different channels, this module is responsible to generate the best reply according to each platform limitations. To illustrate this, a Slack user might receive the schedule of trains in a drop down menu while those interacting using Messenger receive them through quick replies.
3.4 Design and Prototype

I started by drawing low-fidelity expected conversation flows and then tried to convert them to a digital version, but without prior design experience, found it too much time consuming. These flows should be constantly updated to match new bot functionalities and updated response messages, although most of them remained in my notebook and served just as a guide to the development phase. Figure A.1 shows an example flow, which I built before the development stage, and Figure A.2 one which was only build at the moment of writing this thesis. Then, I decided to design the conversations using Chatfuel platform providing a functional chatbot in which beta users could interact. Implementing the conversation flows in this prototype was complex due to platform limitations, although I managed to do so with over 75 blocks for each language.

The main purpose of the prototype was to detect possible flaws before the actual development, and see how the NLU was performing. Implementing an automated user testing functionality was really difficult, leading to 45 additional blocks only for English users, and in the end it was not completely functional. I found it was too much functionality for the platform capabilities, and I had to find an alternative to evaluate it.

To convince people to use this prototype it needed to provide live schedules. Using user attributes to save entities, logic operators and API calls to a backend server in Glitch, I ended up with a dynamic prototype that provided real time departure times and schedule requests. Also, I started redirecting every user message, apart from button clicks, to Glitch server which was responsible to call Dialogflow, and based on the detected intent would generate appropriate responses.

With all of this, I asked friends and family to test it and every time a message intent was not recognised by Dialogflow it was being saved in a spreadsheet for further training. Additionally, I had to see every conversation, manually, looking for errors and mishandled intents. In total, 53 users used originated more than 1500 unhandled messages accumulated, and the Figure 3.2 represents the total of messages, per week, sent and received by this prototype.

This systems should be easy to use, both by beginners and advanced users. Those who are not used to interact with chatbots, can choose the most common functionalities from a menu, and then respond accordingly to the scripted conversation. To achieve their goals faster, users directly send a detailed request and skip some steps of the conversation. In some replies the bot should provide buttons to guide the user into possible conversation paths. These buttons represent payloads and can improve bot efficiency, by replacing written messages with a single tap. Figure 3.3 shows how a guided conversation compares to handling complex messages.
3.5 Development

To implement this proposal, the requirements were prioritised, and then split into five sprints. The spiral present in Figure 3.4 represent the expected development iterations, with the yellow path and green path for both real stages of the development in both prototype and Google Cloud Platform, respectively.

While the prototype was gathering user interactions, all my effort went to build the final architecture of the bot using Google Cloud Functions. Each requirement, or functionality, consisted on multiple iterations of training possible user intents in NLU, building respective code which supports and followed by tests to ensure it works as expected.
Figure 3.5 shows the total of messages, per week, sent and received by this final version. Messenger platform resend user messages every 20 seconds until it receives a acknowledge response. The huge number of messages sent in March was due to a bug that increased times obtaining a schedule which resulted in hundreds of messages being spammed, thankfully nobody else interacted with the bot until it got fixed. Outgoing messages were always higher than incoming ones, mostly because every user message generates at least one reply. Also, I programmed the bot to update me every time a cloud function would awake, and this way I know when it scales up instances and if response times represent a cold or a warm start.
3.5.1 Dialogflow Training

Intents usually rely on entities to correctly understand the messages, for example, it would be irrelevant to know user wants a schedule if it doesn’t recognise the stations. With this in mind I started the following entities:

- Alerts: With a row for each alert topics and respective synonyms
- Stations: Over 450 rows of various CP stations
- FAQ-categories: Supported FAQ categories and some synonyms

Intents were then trained to look for these entities, although soon I realised that if the user started the phrase with something like 'Can you give me' or 'I would like' it would lead to the fallback intent. To solve these I created some helping entities which allow a single example intent phrase to cover multiple combinations of requests. Figure 3.6 demonstrates 'beforeRequest' entity which resulted from this procedure, and on the right, how it can be used in an intent.

Similar, I also had to create a new entities with possible variations of articles, mainly to be used before station names. Dialogflow provides built in date entities which are very useful although I didn’t want to multiply every training phrase by the different types of dates, and ended with a single 'DateVariation' intent with all of them combined which I can handle differently in the code. In Portuguese agent this allowed to identify abbreviations of dates based on an additional date entity. Both these 'ExtraDates' and 'DateVariation' are represented in Figure 3.7
To support some basic small talk, I added the following intents as well:
• Cancel - The bot will forget the conversation context
• Emojis - It will recognise over 1500 emojis and avoid a fallback response
• Feedback - Asks user feedback and then saves it in the database for analysis
• Help - Presents the menu and asks if the user needs help in any of the supported topics
• Human Handoff - Gives CP contact alternatives and alerts the admin, me, of which user is asking for a live agent
• No - replies with 'Sure, feel free to ask anything else related with CP services'
• Thanks - responds with 'Always here to help!'

In one of the meeting with CP they provided some FAQ which I used to train the bot 3.8.

Dialogflow has a timeout for user’s context after several minutes, and when users interact using buttons dialog is not called. There are complex context scenarios, for example, when the bot asks what is user’s destination, and then he clicks to add a favorite station. At this moment Dialogflow must only recognise his station while dialog manager handles the context. in which the bot must ask the user
which station he is referring, then add as favorite and save as destination. To accomplish this and other context-related handlers four new intents were added in DialogFlow: 'Inputs-AlertType', 'Inputs-Station', 'Inputs-Date' and 'Inputs-Number'.

3.5.2 Build and Test

To implement the CP Bot architecture the first step was connecting the backend to a channel. This required creating a Facebook page and creating an App in Facebook Developers dashboard, in order to request permission for pages.messaging. They usually review permissions in less than a week and after being accepted it is possible to receive webhook notifications about messages users send to the page. Also, they provide all the security measures to make sure the messages are authentic. With all this setup the backend started by receiving only text messages and button payloads, but in the end it was able to handle images, voice recordings, and user locations as well.

Integrating with Datastore was straightforward because it also belongs to Google Cloud Platform. Then, it got integrated with Dialogflow and I started developing one functionality at the time, testing it manually before moving to the next one. It was not possible to simulate function’s execution locally, which required deploying, and waiting 2 minutes to see it in action. At the moment of this writing, I found that major drawback was fixed when Google, in April 2019, released a new opened sourced Functions Framework for Node.js [40].

A new permission had to be requested in order to implement Notification Manager because, by default, chatbots in Messenger can only send proactive messages to users 24 hours after their last interaction. Apart from the expected functionalities it ended up giving Google Maps directions to near stations, allowing user to save favorite stations, show what personal information is being stored, why it is being kept, and the possibility to delete it.

In May, the agent was robust handling most conversations but I started noticing that most of my friends and family would repeat past requests to obtain information. With the help of my friend David Sardinha, we developed a A4 sized poster, Figure B.1, which was published in my University, in Link Consulting where I was developing the thesis, and in ‘Clube Recreativo Cultural e Desportivo Luzense’. In addition, in June I decided to test a ‘click-to-Messenger’ campaign as a mean to gather more users while I was developing the required user testing and questionnaire functionality. This ad was a screen recording inside CP Bot requesting a schedule and receiving it in a few seconds. With €11 spent over 9 days, it reached more than 5 thousand people, 1255 of them engaged with the post and 69 started conversations. Overall, it resulted in a €0.17 per messaging conversation started and helped finding bugs both in Dialog Manager and in Dialogflow training model.

When CP Bot was several hours without activity, the following message would take around 7 seconds to respond, and even more if it was a schedule request. To reduce these cold start times, I found
that the main factor was the number of dependencies, and ensuring multiple functions use the same version of dependencies also help because they might be retrieved from a cache. An additional measure was to use lazy-variables, allowing parts of the code to only be loaded for specific requests. Also, after 859 deployments in ‘us-central1’ region, I found that ‘europe-west2’ was the one which provided better executions times. Table 3.1 compares the times of executions, for warm starts, before these improvement with the latest version, 340th deployment in ‘europe-west2’. The execution time reduced, in average, 44.71% and the real was calculated manually with a stopwatch, from the time i pressed the send button until the last expected response appeared on messenger.

<table>
<thead>
<tr>
<th>Intent / Payload</th>
<th>Entities</th>
<th>us-central1</th>
<th>europe-west2</th>
<th>Real duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetStarted</td>
<td></td>
<td>1.16s</td>
<td>0.52s</td>
<td>3.04s</td>
</tr>
<tr>
<td>Menu</td>
<td></td>
<td>2.05s</td>
<td>0.58s</td>
<td>3.82s</td>
</tr>
<tr>
<td>Departures</td>
<td></td>
<td>1.82s</td>
<td>0.60s</td>
<td>2.25s</td>
</tr>
<tr>
<td>Inputs Station</td>
<td>station</td>
<td>2.90s</td>
<td>1.60s</td>
<td>3.92s</td>
</tr>
<tr>
<td>Departures</td>
<td>station</td>
<td>2.92s</td>
<td>1.81s</td>
<td>4.38s</td>
</tr>
<tr>
<td>Near Stations</td>
<td></td>
<td>1.58s</td>
<td>2.74s</td>
<td>0.61s</td>
</tr>
<tr>
<td>Send Location</td>
<td>coordinates</td>
<td>2.02s</td>
<td>0.83s</td>
<td>3.40s</td>
</tr>
<tr>
<td>Schedule</td>
<td>origin,destination,date</td>
<td>4.34s</td>
<td>3.34s</td>
<td>4.45s</td>
</tr>
<tr>
<td>View Stops</td>
<td></td>
<td>2.11s</td>
<td>0.92s</td>
<td>2.98s</td>
</tr>
<tr>
<td>Change Schedule</td>
<td>date</td>
<td>4.58s</td>
<td>2.26s</td>
<td>3.69s</td>
</tr>
<tr>
<td>Thanks</td>
<td></td>
<td>1.69s</td>
<td>0.59s</td>
<td>2.48s</td>
</tr>
<tr>
<td>Simulate Purchase</td>
<td></td>
<td>0.85s</td>
<td>0.22s</td>
<td>2.32s</td>
</tr>
<tr>
<td>Select Document</td>
<td></td>
<td>0.74s</td>
<td>0.22s</td>
<td>2.31s</td>
</tr>
<tr>
<td>Inputs Number</td>
<td>document_number</td>
<td>2.77s</td>
<td>0.59s</td>
<td>2.39s</td>
</tr>
<tr>
<td>Apply Discount</td>
<td></td>
<td>0.73s</td>
<td>0.21s</td>
<td>2.24s</td>
</tr>
<tr>
<td>Inputs Discount</td>
<td>discount_category</td>
<td>0.83s</td>
<td>0.24s</td>
<td>2.35s</td>
</tr>
<tr>
<td>Pay with Paypal</td>
<td></td>
<td>2.39s</td>
<td>2.23s</td>
<td>4.88s</td>
</tr>
<tr>
<td>See Alerts</td>
<td></td>
<td>1.84s</td>
<td>0.83s</td>
<td>1.99s</td>
</tr>
<tr>
<td>Inputs Alert Type</td>
<td>alert_type</td>
<td>3.53s</td>
<td>2.36s</td>
<td>4.62s</td>
</tr>
<tr>
<td>See Alerts</td>
<td>alert_type</td>
<td>3.46s</td>
<td>2.73s</td>
<td>5.39s</td>
</tr>
<tr>
<td>Subscribe</td>
<td></td>
<td>1.57s</td>
<td>0.65s</td>
<td>2.88s</td>
</tr>
<tr>
<td>Inputs Alert Type</td>
<td>alert_type</td>
<td>0.91s</td>
<td>0.25s</td>
<td>2.79s</td>
</tr>
<tr>
<td>Subscribe</td>
<td>alert_type</td>
<td>1.64s</td>
<td>0.59s</td>
<td>2.91s</td>
</tr>
<tr>
<td>FAQs</td>
<td>faq_category</td>
<td>1.42s</td>
<td>0.63s</td>
<td>3.15s</td>
</tr>
<tr>
<td><strong>Average times</strong></td>
<td></td>
<td><strong>2.08s</strong></td>
<td><strong>1.15s</strong></td>
<td><strong>3.14s</strong></td>
</tr>
</tbody>
</table>

Table 3.1: Chatbot execution times
Contents

4.1 Automatic Metrics .............................................. 53
4.2 User Testing ....................................................... 57
4.3 Good Practices ................................................... 60
Ever since the first chatbot was developed, creators aim at measuring how well their chatbot is performing in order to improve it. As presented in section 2.8, there are different perspectives on how to evaluate the quality of chatbots. I will evaluate this proposal in line with Jongerious research, with metrics that can be gathered either automatically or through questionnaire-based methods [26]. He distinguishes questionnaire-based methods into:

- **Expert Review**, performed by chatbot experts, to capture bot's performance based on Effectiveness, Efficiency and Satisfaction with which specified users achieve specified goals.

- **User's Opinion**, a well-known approach to get insights about bot's perceived performance, gathered by asking users for their opinion. The results of a user evaluation might not reflect the actual chatbot's performance but instead reflect the perceived performance of the chatbot.

I tried to evaluate CP Bot in an automated and scalable way, based on Automated Metrics to measure its efficiency, and User Tests to obtain by perceived performance metrics, related to QoE. Taking into account CP Bot is a closed-domain chatbot, Table 4.1 presents the most relevant quality attributes which were selected to evaluate with users, divided by four categories: *Functionality, Humanity, Affect* and *Accessibility*. *Ethics* should also be included to see how user feel about protecting and respecting privacy when data protection is fully implemented. At the moment, General Data Protection Regulation (GDPR) functionalities allow users to delete their name, favorite stations or the last searched schedule, although does not allow the deletion of past conversations. An additional *Ethics* should be included in future user test when GDPR functionality is fully implemented.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Category</th>
<th>Quality Attribute</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>Functionality</td>
<td>Can detect user intent*</td>
<td>1-100%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Task achievement</td>
<td>1-100%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time elapsed per task</td>
<td># of seconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User turns per task</td>
<td># of turns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maintain conversation context</td>
<td>1-5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ease of use</td>
<td>1-5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept</th>
<th>Category</th>
<th>Quality Attribute</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfaction</td>
<td>Humanity</td>
<td>Represented a friendly personality</td>
<td>1-5</td>
</tr>
<tr>
<td></td>
<td>Affect</td>
<td>Turn tasks more fun or interesting</td>
<td>1-5</td>
</tr>
<tr>
<td></td>
<td>Accessibility</td>
<td>Robust understanding messages</td>
<td>1-5</td>
</tr>
<tr>
<td></td>
<td>Ethics</td>
<td>Protect and respect user privacy</td>
<td>1-5 (not evaluated)</td>
</tr>
</tbody>
</table>

Table 4.1: Evaluation Metrics

### 4.1 Automatic Metrics

In this project I was able to gather metrics from three different sources, which I will describe below. Dialogflow has a smaller number of messages because clicks on buttons don’t require NLU intervention.
4.1.1 Functionality Tests

During all the development I had to manually test the most possible path of conversation I could, created new ones based on previous errors. Testing versions before deploying to production was not reliable. Although this feature was not finished, I started populating example conversations in database, by functionalities, which will be used as tests to verify every past implemented functionality is working as expected. The goal is to automatically update Table 3.1, with more and more rows over time, and a new columns for production versions.

4.1.2 Dialogflow Metrics

From 26th September I started keeping information about how frequently Dialogflow wouldn’t recognise user messages. In almost a month it detected 32 out of 829, and 3 with less the 50% confidence rate. Dialogflow’s dashboard includes some analytics, where it shows two types of data related to the agent and the conversations it has been a part of. The following date ranges are provided: 1 day, 7 days, or 30 days. For each, an additional dotted line, light blue, indicates the data from the previous day or time period. Regarding CP Bot, I was able to obtain the following metrics:

- **NLU data**: All intents are ordered by the number of times it was used. The table included a total of 40 intents detected with the number of sessions it was used and the percentage of those who exited the conversation in the specified intent. The conversational paths users have taken when interacting with the agents is summarised into a dynamic chart to analyse user conversation in a user friendly graphical representation. They are represented in Figure 4.1 and Figure 4.2.

- **Usage data**: Number of sessions and queries per session. Figure 4.3 represents the metrics regarding the past 30 days, which Portuguese and English agents together had a total of 113 sessions, and an average of 5.66 queries per session.

<table>
<thead>
<tr>
<th>Intent</th>
<th>Sessions</th>
<th>Count</th>
<th>Exit %</th>
</tr>
</thead>
<tbody>
<tr>
<td>plan_journey</td>
<td>31</td>
<td>145</td>
<td>28.97%</td>
</tr>
<tr>
<td>inputsStation</td>
<td>22</td>
<td>65</td>
<td>13.85%</td>
</tr>
<tr>
<td>menu</td>
<td>7</td>
<td>53</td>
<td>50.94%</td>
</tr>
<tr>
<td>plan_journey.inputs_date</td>
<td>11</td>
<td>51</td>
<td>19.61%</td>
</tr>
<tr>
<td>next_trains</td>
<td>17</td>
<td>42</td>
<td>26.19%</td>
</tr>
</tbody>
</table>

**Figure 4.1**: Top five of the table with all intents handled
4.1.3 Dashbot Metrics

Their features are divided into three main groups:
• **Understand**: They collect more than 30 metrics to understand interacting users. They are constantly adding new, varying from Demographics, to sessions tracking and funnels of messages.

• **Real time**: Three useful functionalities, allow pausing the bot, responding manually to users, and see recent messages in an easy friendly platform, demonstrated in Figure 4.4.

• **Tool For Action**: The audience builder which allows targeting users based on multiple parameters, as demonstrated in Figure 4.5. This powerful feature is not available in the free version, although, they allow broadcasting messages to all engaged users in the last 6 months, and creating Email, Webhook, and Slack alerts for specific types of messages.

![Figure 4.4: Live transcripts functionality in Dashbot](image1)

![Figure 4.5: Audience Builder functionality in Dashbot](image2)
4.2 User Testing

Instead of performing all user testing personally, which would have been very time-consuming, I thought about performing them directly through the chatbot in a scalable way. This was possible by improving the Dialog Manager and Dialogflow modules to handle task-related intents, and tracking the functionality metrics automatically. Users that completed all the tasks were then asked to perform a questionnaire about their experience in a Likert scale, which 1 represents ‘Strongly disagree’, 3 for ‘Neither agree or disagree’ and 5 for ‘Strongly agree’.

The user could have been redirected to a questionnaire in an online form, but asking the questions directly in the bot, seemed easier and more natural. I also integrated the chatbot with QuickCharts API to produce charts with live results from the database. Figure 4.6 shows how these statistics look in the middle of the conversation, and bigger versions of the charts are available in Figures B.3 and B.4. To support this, the bot started keeping track of user’s current task, and the time between each tasks is given and when they get completed or skipped. Testing every functionality would lead to long user tests, instead I decided to evaluate only five most relevant intents. Figure 4.7 is a collection of conversations taken from users performing the tasks:

1. Asking the schedule from Lisbon Oriente to Lagos
2. See the times of departures of the next trains in Carcavelos
3. Find more information about CP services for customers with special needs
4. Simulate a ticket purchase from Cais do Sodre station to Cascais
5. Subscribe service alerts

From a total of 201 users, 61 testers, mostly family and friends, were gathered through private messages on Facebook and LinkedIn. There were bugs obtaining number of retries in this evaluation, reason for not tracking number of messages to achieve first task, and some zeros in other tasks. Also, for some users it only counted written messages, ignoring the number of button taps. The charts are created with data from the users, 24, that completed all the tasks at 17th October. Overall, I was able to obtain the following metrics:

- On the first task those who directly asking the schedule were able to obtain it in around 20 seconds, although the average was 51.4s. The 62% competition rate was low mainly because 23 users didn’t understand or wanted to perform the tasks.
- Requesting the times of departures in a station took an average of 28.4s and couldn’t be achieved by 4 users.
• Finding more information about services for reduced mobility users wasn’t achieved by 3 users and one user took more than 7 minutes which increased the average to 48s.

• Another 3 users weren’t able to simulate the complete purchase of a specific ticket and those who could took on average 72s.

• On average 53.2s were necessary to subscribe an alert topic, and 4 of them didn’t accomplish it.

All these times will be shorter for real user intents, reducing the latency from task description message, time spent reading the task, and thinking about how to accomplish it. Based on the 21 users who responded to the questionnaire, every quality attribute measured obtained great satisfaction rates, but this result would be more valuable if those who got stuck also responded to it and analyse the reasons for task failures.

![User test results on 17th October 2019](image_url)

**Figure 4.6:** User test results on 17th October 2019
Figure 4.7: Examples of user test conversations
4.3 Good Practices

Similar to any other software development process, a chatbot also has a life cycle. After several months developing the chatbot for CP, I suggest following the model present in Figure 4.8, and being agile allows handle change, reducing overall project risks and maximising customer value. It starts with an initial conception phase to define the requirements and make sure a bot is a suitable solution, followed by a development stage in which the product is build and the Natural Language model is trained. All supported use cases must be tested and when the bot meets the expectations it goes to production stage becoming accessible to general users.

Chatbots projects should grow gradually, starting to support the most important user intents and only support new ones making sure they are robust handling them. Both the development and the production phases are composed of iterative stages which allow for incremental refinements as well as support for new intents in each iteration. This agile architecture allows risk reduction by having short sprints of well-defined tasks. In the end of each sprint requirements should be reassessed with the client, and according to business need, user feedback or changes in the budget they might need to be changed or re-prioritised.

Below, I will describe some good practices for each stage of this proposed chatbot life cycle.

![Figure 4.8: Proposed chatbot life cycle](image)

4.3.1 Conception

4.3.1.A Analyse

Before initiating any project, it’s essential to define goals, along with ways to quantify success. This is a crucial phase, where requirements must be defined, analysed and then decide whether or not the chatbot is a recommended solution. Most successful applications and websites ensure a great User
Experience (UX), and bots are no different in that regard. Bot's UX is fully based on messages and it is crucial to write the requirements from the user perspective forcing the project team to think about to whom the features are for, and why they are important.

During this first stage, three main aspects must be defined:

1. Chatbot goals
   
The goals should be feasible and measurable, allowing to evaluate its success over time. These should be the project motivation and can vary from multiple types, such as, providing 24/7 customer support, personalising user experience or even improving task’s efficiency.

2. Chatbot requirements
   
These requirements can be captured as user stories, which also forces us to think from the end user's perspective. A user story is a short description of something that customers will do, focused on the value or result they get from the UX and always follow the same format:

   As a {user role}, I want {requirement}, so that {reason}

For example, designing a chatbot for a railway company the following user story might be identified:

As a regular train customer, I want the bot to remember the services I use, so that I can receive respective alerts

If possible target user
A great way to gather and prioritise the requirements is by directly asking the target user. Also, humans can pretend to be the bot for a while and learn about users’ expectations.

3. Target users
   
Selecting the channels where bot will live should be driven by the target audience. Messaging platforms might have advantages to increase bot discovery and engagement, although custom solutions in website or mobile apps can provide additional data privacy. Additionally, bots should be developed with the logic independent from used channel, favouring future channel integrations.

4.3.1.B Design and Prototype

After establishing the requirements, it is important to create conversation design flows that represent the conversation at a high level. These flows describe the paths users might follow in the conversation, and should optimise the number of messages exchanged to handle user intents, except in open-domain chatbots where the goal is to entertain the user. Humans do not communicate in sequential dialogues and tend to change their minds, as result new paths are expected to be added over time. To design the flows I suggest using the design elements present in Figure 4.9.
Using these elements we are able to build a graphical representation of supported conversations. Also, they describe the logic decisions used to select the most appropriate answer. Having these conversation flows is a huge help building both prototype and the final dialog manager. The goal here it to
always steer the conversation forward with contextual prompts based on understood user intent.

Conversation designers should provide a greeting and walkthrough for 1st time users, and allow user to adjust notifications or subscriptions. It is also critical to understand when users ask for help. Receiving user feedback is also a recommended practice that can bring additional user insights.

Users should know they are talking to a robot, but implementing a pleasurable personality humanises the experience and more closely aligns the chatbot with your brand. Writing chatbot replies doesn’t simply mean paying attention to correct grammar, appropriate tone and accurate lexical choices, but also playing with words, with cultural standards and with shared knowledge. Alexa, for example, is playful and has its own opinions, moreover, her opinions and preferences vary from country to country.

In the end, the team should decide whether or not to proceed with the development phase. I recommend listening first to feedback from target users, and go ahead if the bot can make use of its knowledge to achieve the end goals while providing better customer experience.

### 4.3.2 Development

Most of project sprints belong to this phase, which brings the project from conception to reality. At first, the requirements must be prioritised, and then design an architecture to support them. The team divides the proposed requirements into development steps by sprints, and in the end of each sprint the team can adjust the development plan according to its priorities. The team has choose wisely the NLU provider as it is directly related to how smart does it understand user messages.

From the first development iteration, the training set for the NLU module will start its continuous development. Most NLU services provide a real-time dashboard where it is possible to control every aspect of the conversational agents. At the same time the dialog manager is either programmed from scratch or built on top of platform.

#### 4.3.2.A NLU Training

Training the bot is the most important factor in determining its performance. Bad training will inevitably lead to a poor performing chatbot and frustrated users. Based on the flows created during conception, the agent’s training consists of creating intents and filling them with example expressions that represent respecting user intention. If available, FAQ are a good source of information with common user intents and respective replies. These sentences should be varied and preferably come from end users. Besides the proposed functionalities it should handle followup questions and common messages such as:

- Emojis/Stickers
- Thanks
If two intents have similar training utterances, it becomes harder to decide which one the user is referring to. Usually, this means they should be joined into a common intent and the answer is based on present entities and user's conversation context. Providing schedule to users the bot might ask their origin and destination, and in both cases, the expected response will be a station. Instead of training both 'inputs.origin' and 'inputs.destination' intents, a single 'inputs.station' should be created and handled using current context.

Tagging entities has a few rules as well. Entities are keywords that tagged in sentences to extract information. NLU usually provide multiple ways to identify custom entities, they can be aggregated simpler entities, be regexp expression or simple words with synonyms.

The Wizard-of-Oz approach can also be used in chatbots to gather new conversation examples. This approach let users believe they are communicating with a chatbot, while in fact a real human secretly plays the role of the chatbot [41].

4.3.2.B Build

For chatbots that are built using platform the dialog manager is usually implemented defining blocks of operations that are triggered by user messages or through logic operators. Inside blocks it is possible to call webhooks that handle the message and provide dynamic information in the replies. This design choice can bring some limitations in exchange for an easier building phase, which I recommend using for prototyping and only use in production for simple projects without sensitive information.

Alternatively, the dialog manager is programmed as an event based webhook script that before handling them guarantees the messages are authentic using secure signatures. Multi-language bots start by retrieving user information from the database to decide the language of the conversation. After identifying message's intent, entities, and context, the logic that mimics conversation flows redirects the message to respective handler function, which decides what to reply.

4.3.2.C Test

Each development iteration must test bot's performance handling user intents to make sure everything works as expected. If possible, the development team should build a test set with messages and ex-
ected responses that can identify if implemented functionalities are not working properly. This can be particularly useful if training utterances from multiple intents overlap and can misleading intent recognition resulting in incorrect replies.

As previously said, user messages often diverge from what is expected, so it’s really important to test the product with real users throughout the process. It might work perfectly for 90% of the users but there are so many ways of expressing ourselves that the best way to really test the bot’s understanding capabilities is by giving tasks to new users and see how they perform. These evaluations often result in fallback utterances that are aggregated for further training of NLU module.

Testing the user stories can identify bugs and new paths in the conversation flow, which lead to new development iterations. For bots being published into multiple channels all of the interfaces should also be tested. After a successfully tested version, a nice way to demonstrate chatbots functionalities is through a demo. When the product owner decides the implementation is acceptable, its time to publish and monitor how it performs.

4.3.3 Production

Once the bot is publicly accessible, maintaining it is an essential part of its long-lasting success, where learning from the user’s behaviour is crucial. Analytics tools are strictly necessary to monitor the performance and continually improve it. This process of matching received messages to existing intents or highlighting the need for new responses is a continuous cycle that moves the chatbot from a decision tree structure to a natural conversation over several sprints. Messages with low confidence intent detection confidence rates should also be saved in database for further inspection and training. Otherwise they should be counted and trigger a somebody when they reach a certain amount.

Monitoring user conversations require chatbot specific analytics. They must save received messages and chatbot replies to provide the following useful functionalities:

- Message and User activity over specific time periods
- Measure engagement through time and messages per session
- Show user retention rates
- Produce a message funnel
- Identify messages with sentiments
- Tracing the most common conversation paths
- Live operator takeover
5.1 Limitations ................................................................. 69
5.2 Future Work ............................................................... 70
The output of this research is a fully functional chatbot trained to respond dozens of CP related intents using user’s natural language inside a Messenger conversation. It was developed to understand most common user intentions, and a menu is always available, where users can choose a functionality and CP Bot guides the conversation accordingly. CP Bot resulted from an initial analysis, producing a prototype used by 50 people to gather feedback and training examples. The final version became a stateless solution fully hosted in Google Cloud Platform and with more functionalities than initially planned.

Dashbot was integrated for message analytics since the beginning, and allowed to see useful metrics regarding past conversations and allow pausing the bot when a CP representative is available. The process of matching user intents to existing features or highlighting the need for new responses is a continuous cycle that moves the bot from a decision tree structure to a natural conversation. The most successful chatbots are constantly adapting and revising their conversation flows in response to their users.

I can conclude developing chatbots brings no limitations integrating with existing systems and service, and Dialogflow has a powerful NLP behind conversational agents, which can be controlled in both dashboard and though an API. Messenger ads were a really powerful functionality to target at specific groups of users, and worked perfectly to obtain early testers. In the end it was rewarding to see the projected was really well received by the beta testers and even acquired the following feedback:

- Keep up the good work
- Excellent information and service
- Everything working perfectly
- Application really useful and fast
- Very good and complete

Satya Nadella, the CEO of Microsoft, said “Pretty much everyone today who’s building applications … will build bots as the new interface”. I agree with his statement, as chatbots can bring a much easier and faster way for users to accomplish their goals. Even reduced mobility customers can interact using voice, and instead of teaching the user how to use a program, the chatbot is the responsible to mimic possible conversation paths.

5.1 Limitations

One of the biggest concerns about using chatbots is definitely its data privacy and security. When receiving personal information the user should know why it is being saved and preferably hidden before
forwarding it for analytics or NLU provider. Comparing to chatbots inside website, those integrated with IM have an additional integration and must be compliant with their platform policies. Only in 2019, Messenger was updated 8 times. Some important features are being eliminated to make the mobile app lightweight. These include the end of location quick reply and list templates which will have impact in providing near station information and showing train times. CP Bot interacted with a total of 207 users, but most of them are Portuguese, which resulted in much less training of the English conversational agent. Over time I would translate training phrases just to simulate interactions, although real testing with English speakers is required to take conclusions about English agent performance.

5.2 Future Work

All the proposed functionalities in CP Bot are working correctly, but in order to bring CP Bot into production some additional features should be implemented, which were divided into the following topics:

**NLU Training**

- Automatically generate NLU intent and entities training data in Dialogflow from spreadsheets, specially to update for FAQ
- Implementing sentiment analysis

**Development**

- Integrate with ticket emission service to reserve seat and associate ticket purchases with the user
- Finish GDPR functionalities, which includes forwarding data deletion requests to both Dialogflow and Dashbot
- Support MB Way payments to accommodate Portuguese citizens

**Testing and Analytics**

- Run automated tests based on example conversations to assure chatbot is responding correctly through all trained conversation paths Perform scalability tests to see how it performs on peaks
- Iterate new user testings to obtain more intent training phrases
- Implement A-B testing functionality, giving two groups different responses and comparing the metrics obtained for each
- Implement a dashboard to segment audiences based on tags and send targeted messages
Bibliography


Conversational Design Flows
Figure A.1: Initial Conversation Flow
Figure A.2: Near stations conversation flow
CP Bot Advertising and Evaluation

This poster was made with the help of my friend David Sardinha, Designer Student in IADE. The screenshot on the left shows how the first interaction was handled at the time, asking the user’s preferred language and then provided some examples of what they might ask. And on the right side, it shows an example of how a schedule can be represented in the conversation. The poster was directed to Portuguese CP customers, and the QR code includes a referral parameter so I could see where the user opened it. The three big sentences can be translated to:

- Do you want to see the schedules?
- Look for the next departures and possible delay?
- Receive important alerts?
Figure B.1: CP Bot Poster
Figure B.2: Example of chatbot ad with deep links
Figure B.3: User test results - part 1
Figure B.4: User test results - part 2