Master's Thesis

DOUBLE MASTER'S DEGREE IN INDUSTRIAL ENGINNEERING AND MANAGEMENT ENGINEERING

InduBot

Final Report

Author:	Marco Facchini Amondarain
Director:	Marta Gatius Vila
Rapporteur:	Lluis Solano Albajes
Edition:	04/18

ETSEIB

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

UPC

ETSEIB

Abstract

Chatbots are disrupting the market to substitute traditional channels of interaction between businesses and customers. People nowadays are using messaging apps more than never before and when booking a flight, ordering a pizza or whatever the transaction, in many cases companies has a chatbot there, 24/7, ready to chat and give a service.

This project consists on the development of InduBot, a chatbot to respond to student's demands when searching academic information about the university. In this final report it is explained the followed phases to design, develop and implement the functional prototype. InduBot is intended to be a first step and a base for developing other chatbots to connect the university with the students.

The resulting chatbot for this project can be found on the messaging app Telegram by searching *TheInduBot*. It has been configured to be always running on a server as a web application and through a webhook and the Telegram Bot API answer at any time to user requests.

The university should be ready for chatbots and contemplate this new channel of communication because it may be the key to transform the relationship with the students in the coming years.

INDEX

1. Glossary

Server: is both a running instance of some software capable of accepting requests from clients and the computer such a server runs on. A **web server** is an information technology that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web. The term can refer either to the entire computer system, an appliance or specifically to the software that accepts and supervises the HTTP requests.

Integrated development environment (**IDE**): is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.

Open source: as a development model promotes a universal access via a free license to a product's design or blueprint, and universal redistribution of that design or blueprint, including subsequent improvements to it by anyone.

Chatbot: a computer program designed to have a conversation with a human being via auditory or textual methods. It is also known as a Conversational Agent, Talkbot, Chatterbot, Conversational Bot, Artificial Conversational Entity or on many occasions simply as Bot.

Turing Test: in artificial intelligence, a test proposed (1950) by the English mathematician Alan M. Turing to determine whether a computer can "think."

Loebner Prize: an annual competition in artificial intelligence that awards prizes to the computer programs considered by the judges to be the most human-like. The format of the competition is that of a standard Turing test.

Linguistic deflection: a way for the chatbot to avoid responding to an input it does not understand by giving a canned response.

Machine Learning: is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed.

Natural Language Processing (NLP): is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding

2. Preface

2.1. Origin of the project

During the history of computer science, there have been many approaches to create conversational systems. Several of these systems try to simulate a human being while others are focused on helping the user in a specific task (i.e., booking a flight, accessing a database, etc.). One of the first well-known examples of the systems from the first group, was Eliza; which emulates a Rogerian psychotherapist and when it appeared in the 60's, some people actually mistook her for a human. But it has been in recent years that many companies have started to create chatbots to drive value for their business. Followed by the rise in this industry many tools and facilities have appeared in the market to easily develop a chatbot and test it with users.

2.2. Motivation

During my student life at university many times I have needed information about courses and other academic issues but never had a quick procedure to obtain it. Sometimes looking at the school's websites took me several minutes only to know simple information such as: who the teacher for Electronics course was or when I was going to have the next exam of the Algebra course. With the boom of chatbots, I found a solution to make students life easier and avoid them to waste time navigating through the web. Besides this project contemplates the implementation of a chatbot in a real case of use with real benefits.

2.3. Previous requirements

In order to conduct this project, some previous knowledge is required. Firstly it is essential to understand the state of art and take advantage of the tools and technologies already developed in the field of chatbots to avoid creating what is already created. Another prerequisite is the familiarity with a programming language to complete the project with a prototype. In this case, the code will be performed in Python which is the programming language taught at ETSEIB[1] and the one more extended in the chatbots industry.

There are also other important resources needed for the implementation and prototyping such as a server or website for running the system.

1 **ETSEIB**: Escola Tècnica Superior d'Enginyeria Industrial de Barcelona.

3. Introduction

3.1. Objectives of the project

The main objective to achieve in this project is the creation of a chatbot to find information about the ETSEIB. Other minor goals are the study of the user experience and the impact this chatbot could have for the University and the students. That is, in a schematic form:

Main goals:

- Design and implement a chatbot to support students with academic information from the ETSEIB website.

Secondary goals:

- Provide a seamless user experience, such as the one of talking with a human by chat.
- Transform the relation university has with the student through the use of new channels of communication.
- Give accurate answers when information comes from a database.
- Set a first chatbot base which can be continued in future developments.

3.2. Scope of the project

Creating a chatbot to improve the student's accessibility to ETSEIB's information is an ambitious proposal. For this project, the scope will be limited to the planning, design, and implementation of a prototype of the chatbot. The field of knowledge of the chatbot will be restricted to the Degree of Industrial Engineering and to the following subset of the information provided by the website of ETSEIB:

- Exams' dates and hours
- Teachers information
- Courses information

The chatbot will be configured to talk in the Catalan language since is the most frequently used one at ETSEIB among students and also the main one for the university website.

The prototype will be a functional beta version but after the end of the project no further developing or maintenance is planned to be done. However, the project will be open in case someone in the future would like to continue it. The complete code of the prototype and the project's memory will be of public access for everyone interested in improving the chatbot.

3.3. Current situation at ETSEIB

The current generation of ETSEIB students is digital native who have used the Internet since a young age and are generally comfortable with technology. They really appreciate the immediacy of response and do not want to spend time on procedures. During the day, they constantly use WhatsApp or other messaging platforms to chat with other people and find it a very useful way of communication.

At ETSEIB, as in most of the universities, when a student wants to know the timetables of a course or the date of an exam the first channel is the website. In the case of ETSEIB, the website presents a responsive layout that can be used from any device without problems. However, the number of clicks, 4 in case a student wants to find the date of an exam, the disposal of the information that may be not so intuitive or the experience of navigating through a static website, maybe not so exciting.

With InduBot the experience of searching information can be transformed into a conversation through a chat as the one of asking information to a friend. The project would like to create a new channel of interaction with the student to consult information online.

4. Context

4.1. Definition and types of chatbots

A chatbot is a service (computer program), powered by rules and sometimes artificial intelligence, and often designed to convincingly simulate the conversation of a human being via text or voice interactions. It can automate certain tasks, by chatting with a user through a conversational interface. The most advanced bots are powered by artificial intelligence, helping it to understand complex requests, personalize responses, and improve interactions over time. The service could be any number of things, ranging from functional to fun, and it could live in any major chat product (Facebook Messenger, Slack, Telegram, Text Messages, etc.). [1]

Since there is not an established classification for chatbots 3 approaches have been chosen, each of them based on different criteria.

4.1.1. Classification 1: Chatbot's purpose

There are two major types of chatbots depending on their final goal: chatbots for entertainment and chatbots for business.

4.1.1.1. Entertainment chatbots

Engineers have been developing chatbots for entertainment for decades, since the famous chatbot, the psychotherapist ELIZA, was introduced in 1966. Creators of these chatbots usually try to make a bot which can look like a human, in other words passing the Turing test. Perhaps all of the bots which participate in Loebner's prize and similar competitions can be included in this group. For example, the Microsoft's bots Xiaoice and Tay meet this definition.

Chatbot responses to user messages should be smart enough for the user to continue the conversation. The chatbot does not need to understand what user is saying and does not have to remember all the details of the dialogue.

One way to assess an entertainment bot is to compare the bot with a human (Turing test). Another option, the quantitative one, are metrics, such as calculating the average length of conversation between the bot and end users or the average time spent by a user per week. If conversations are short then the bot is not entertaining enough.

ELIZA

Back in the 1960s, MIT professor Joseph Weizenbaum developed ELIZA, the first ever chatterbot, that could interact with people like a psychotherapist would. It was designed to imitate human conversations by using pattern matching and substitution methodology. ELIZA would ask open-ended questions and even respond with follow-ups. It uses a script that simulated a Rogerian psychotherapist. [2]

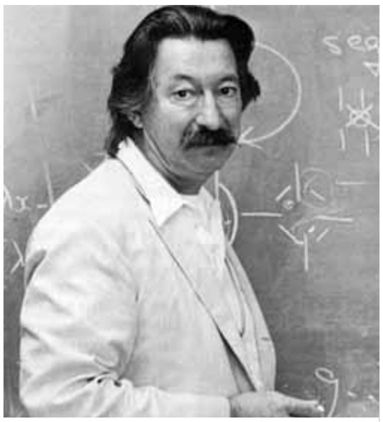
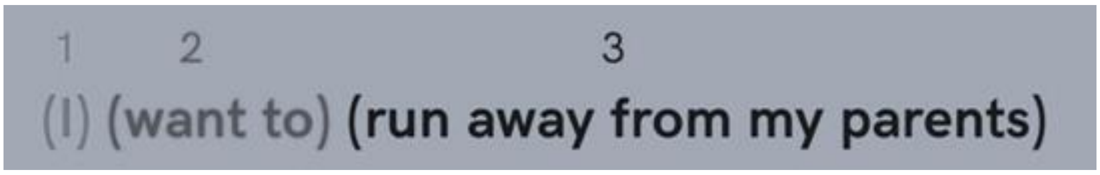


Figure 4.1 Joseph Weizenbaum

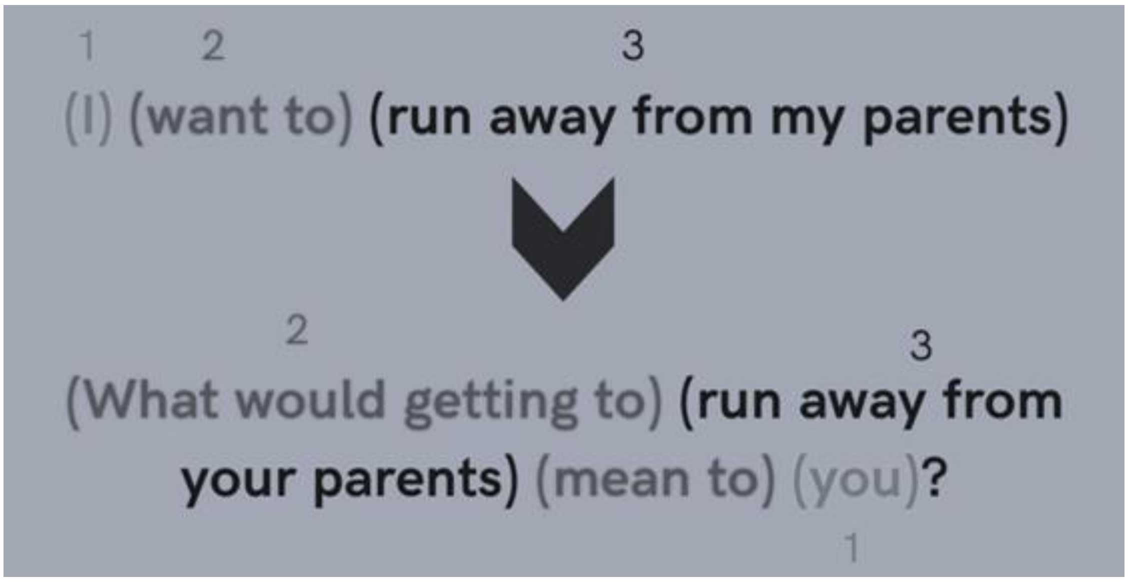
ELIZA works by recognizing keywords or expressions from the input to replicate a response pairing them to a list of possible scripted responses – customized reactions.

Taking the sentence “I want to run away from my parents” it is possible to understand exactly how it works.

ELIZA attributes a weighted value to each word of that sentence:



ELIZA attributes low values to pronouns (I), slightly higher values to action verbs (want to), and the highest value to the actual action (run away from my parents). This allows the programme to know exactly how to flip the sentence around to ask a digging question. How? Simply turn the values into a question, flip the pronoun, and switch the verb to convey meaning. The answer, then, becomes "What would getting to run away from your parents mean to you?".


1 2 3
(I) (want to) (run away from my parents)
2 3
(What would getting to) (run away from your parents) (mean to) (you)?
1

The biggest learning point to get from ELIZA is about complexity. It amazes how simple ELIZA's script actually is, yet plenty of humans got easily tricked.

4.1.1.2. Business chatbots

Chatbots for business are often transactional, and they have a specific purpose. Conversation is typically focused on user’s needs. Travel chatbot, for example, is providing information about flights, hotels, and tours and helps to find the best package according to user’s criteria. Google Assistant readily provides information requested by the user. Uber bot can take a ride request.

Conversations with this kind of chatbots are typically short, less than 15 minutes. Each conversation has a goal, and quality of the bot can be assessed by how many users get to the goal. Has the user found the information he/she was looking for? Has the user successfully booked a flight and a hotel? Has the user bought products which help to solve the problem at hand? Usually, these are the used metrics to measure success and they are easy to track.

Tommy Hilfiger chatbot

Tommy Hilfiger’s chatbot lets consumers globally explore pieces from the brand’s new collection by asking questions that help identify the customer’s individual tastes and required sizes. To purchase products suggested by the chatbot, customers are transferred to tommy.com, where the items will have already been placed in their basket.

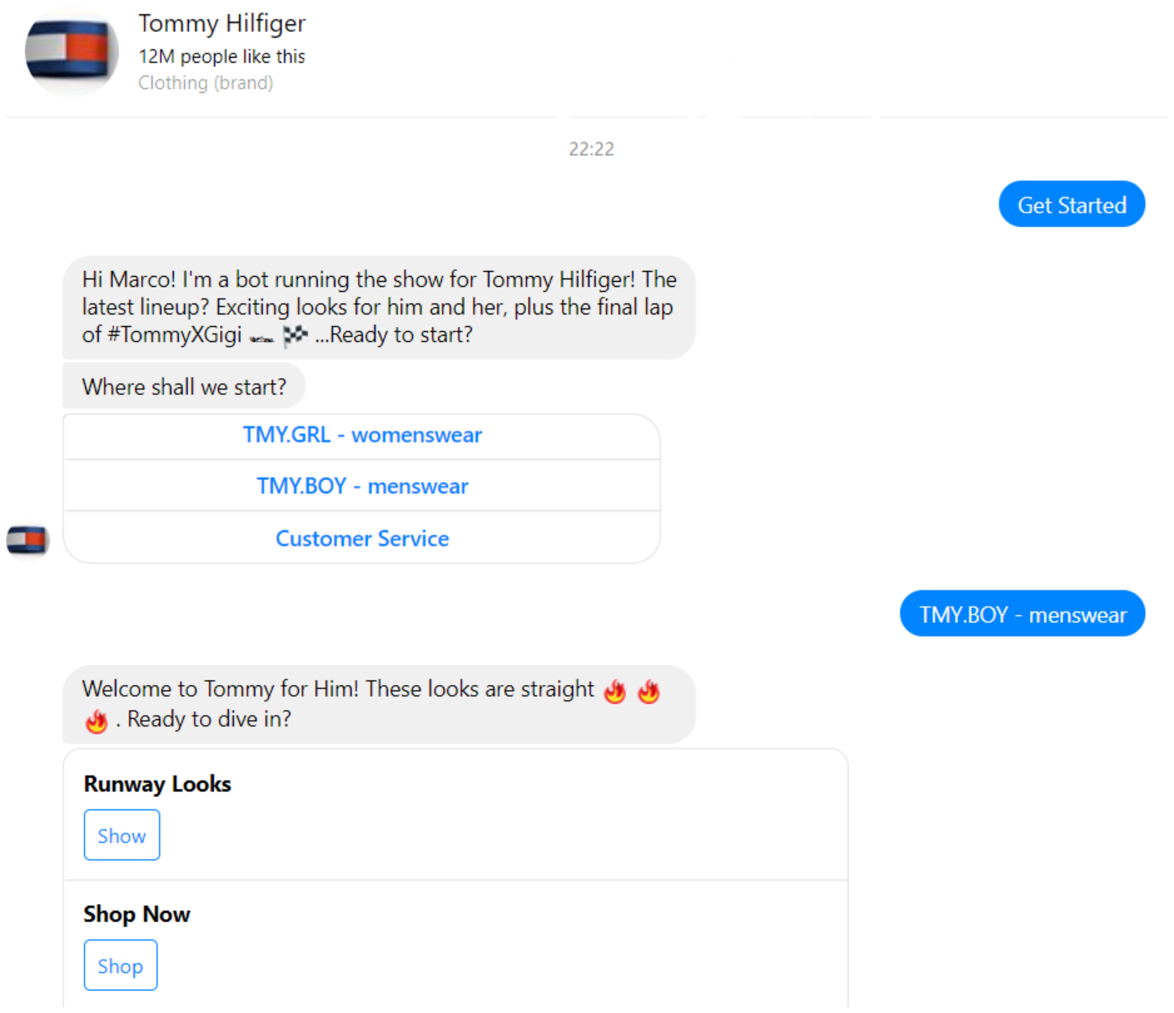

Tommy Hilfiger
12M people like this
Clothing (brand)
22:22
Get Started
Hi Marco! I'm a bot running the show for Tommy Hilfiger! The latest lineup? Exciting looks for him and her, plus the final lap of #TommyXGigi 🏎 🏁 ...Ready to start?
Where shall we start?
TMY.GRL - womenswear
TMY.BOY - menswear
Customer Service
TMY.BOY - menswear
Welcome to Tommy for Him! These looks are straight 🔥 🔥 🔥 . Ready to dive in?
Runway Looks
Show
Shop Now
Shop

Figure 4.2 Tommy Hilfiger's Facebook Messenger Chatbot

4.1.2. Classification 2: Chatbot's logic

When interacting with different chatbots the user will probably find that some react in a completely different way than others. Some have the ability to reply to anything the user throw them, and others will simply keep on apologizing, while some simply seem to have a single path that the user can follow. The reason for that is not every bot is the same and their intelligence level as well as their capabilities are defined by the back-end, which powers them. So a reasonable classification is to divide chatbots depending on their logic. [3]

4.1.2.1. Flow-oriented chatbots

These bots are made to follow a certain path that is defined by a logic tree created for a specific purpose. So the user goes through a set of questions, options based on those inputs, and then the chatbot goes down the pre-defined path.

So although the user still gets to make the decision, he/she will need to go down one of the pre-set paths. These bots have quite a lot of buttons, options, and quick replies to restrict the domain of the conversation and will usually keep on apologizing if the user writes anything in they do not understand.

These chatbots are usually created to realize specialized processes that replace the need to talk with a person or use more complicated UIs such as mobile apps or websites. Since they are trained on top of structured data they just can do a set of limited operations. Think of what a bank operator can do for you over the phone: verify your identity, block your stolen credit card, give you the working hours of nearby branches and confirm an outgoing transfer.

Weather bot

A chatbot created to know the current weather and forecast for a determined location. It has a predefined path with buttons to guide the user through the conversation.

The following figure shows the chart-flow for this chatbot:

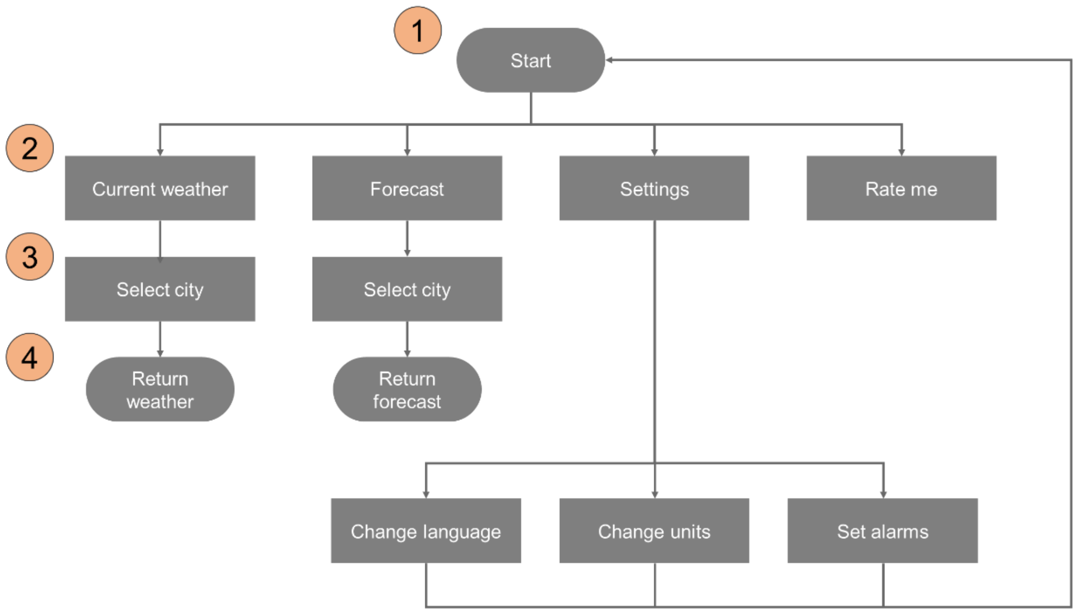
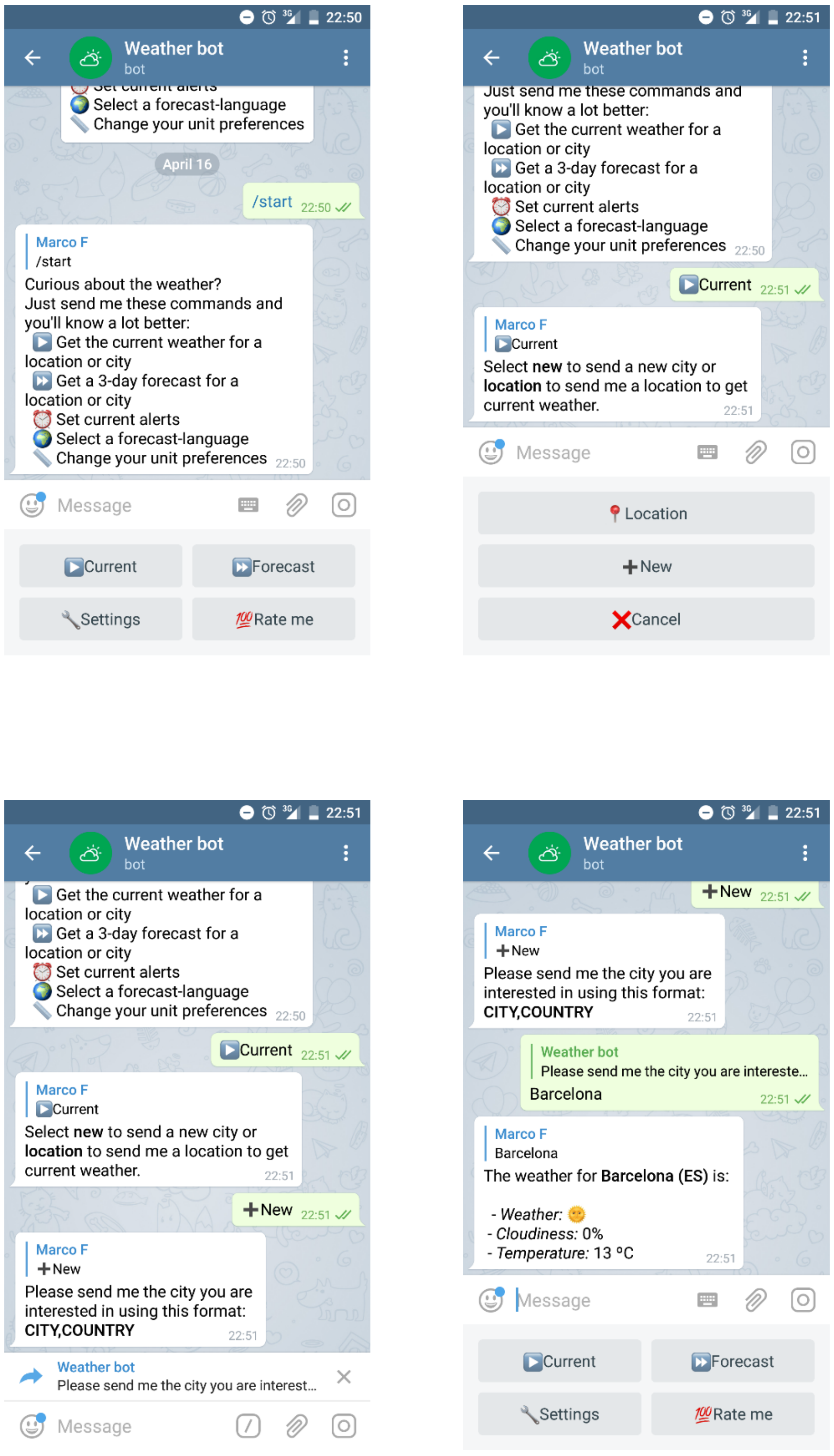


Figure 4.3 Flowchart Weather Bot

To illustrate the highlighted states 1,2,3 and 4 in orange behind there are the corresponding conversations.


22:50

Weather bot
bot

Set current alerts
Select a forecast-language
Change your unit preferences

April 16

/start 22:50

Marco F
/start
Curious about the weather?
Just send me these commands and you'll know a lot better:
Get the current weather for a location or city
Get a 3-day forecast for a location or city
Set current alerts
Select a forecast-language
Change your unit preferences 22:50

Message

Current
Forecast
Settings
Rate me

22:51

Weather bot
bot

Just send me these commands and you'll know a lot better:
Get the current weather for a location or city
Get a 3-day forecast for a location or city
Set current alerts
Select a forecast-language
Change your unit preferences 22:50

Current 22:51

Marco F
Current
Select **new** to send a new city or **location** to send me a location to get current weather. 22:51

Message

Location
New
Cancel

22:51

Weather bot
bot

Get the current weather for a location or city
Get a 3-day forecast for a location or city
Set current alerts
Select a forecast-language
Change your unit preferences 22:50

Current 22:51

Marco F
Current
Select **new** to send a new city or **location** to send me a location to get current weather. 22:51

New 22:51

Marco F
New
Please send me the city you are interested in using this format:
CITY,COUNTRY 22:51

Weather bot
Please send me the city you are interest...

Message

22:51

Weather bot
bot

New 22:51

Marco F
New
Please send me the city you are interested in using this format:
CITY,COUNTRY 22:51

Weather bot
Please send me the city you are intereste...
Barcelona 22:51

Marco F
Barcelona
The weather for **Barcelona (ES)** is:

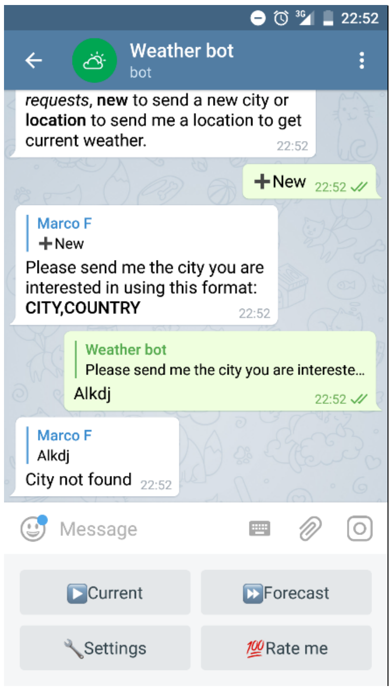
- *Weather:*
- *Cloudiness:* 0%
- *Temperature:* 13 °C 22:51

Message

Current
Forecast
Settings
Rate me

Figure 4.4 Weather Bot Telegram

The chatbot has also mechanisms to handle with inputs that are out of scope. Here there is an example of how it has answered when it does not know a city.



requests, **new** to send a new city or **location** to send me a location to get current weather. 22:52

+New 22:52

Marco F
+New
Please send me the city you are interested in using this format:
CITY,COUNTRY 22:52

Weather bot
Please send me the city you are intereste...
Alkdj 22:52

Marco F
Alkdj
City not found 22:52

Figure 4.5 Weather Bot Telegram

4.1.2.2. Artificially Intelligent chatbots

So the main difference between Artificial Intelligence and flow-chatbots is that they can handle free-text. So the user can simply enter any sentence and the main difference here is that the chatbot will be able to analyse that into a set of parameters and understand what the user's intent was and thus react accordingly.

Of course, this is powered by Natural Language Processing and Machine Learning. To function it accurately, the bot needs a complex back-end that can truly handle anything that is asked by the user. Currently, there are few NLP engines in the market, which are open-source and are owned by the big tech companies. However, it is possible to create an AI engine to train your chatbot and mimic human conversations.

Following there is an example of how NLP works within a chatbot:

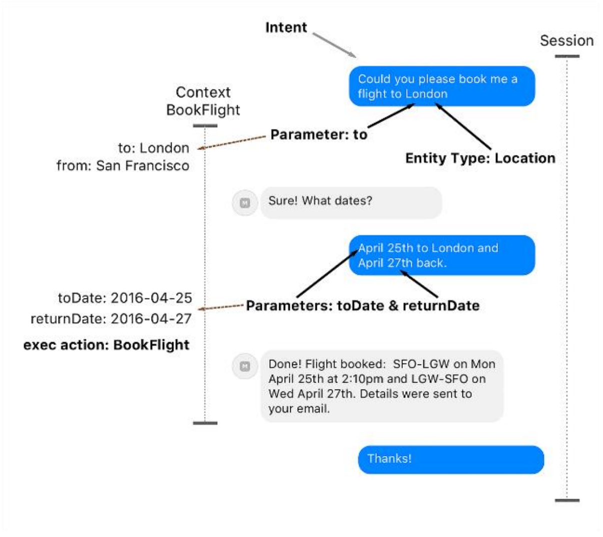

Intent
Session
Could you please book me a flight to London
Context BookFlight
Parameter: to
to: London
from: San Francisco
Entity Type: Location
Sure! What dates?
April 25th to London and April 27th back.
toDate: 2016-04-25
returnDate: 2016-04-27
Parameters: toDate & returnDate
exec action: BookFlight
Done! Flight booked: SFO-LGW on Mon April 25th at 2:10pm and LGW-SFO on Wed April 27th. Details were sent to your email.
Thanks!

Figure 4.6 Flight Bot Facebook Messenger

4.1.2.3. Hybrid chatbots

Hybrid chatbots combine the two types of bots and create a semi-intelligent system that is configured of some NLP capabilities and a pre-defined flow. In fact, most of the bots in the market today, are hybrid bots. This is due to the fact that NLP still has quite a long way to go to completely understand any sentence with different spellings, and sometimes even accents in applicable languages. These misunderstandings can lead to user frustration and abandon of the conversation, so that is the reason why by now hybrid chatbot approaches are the most used ones.

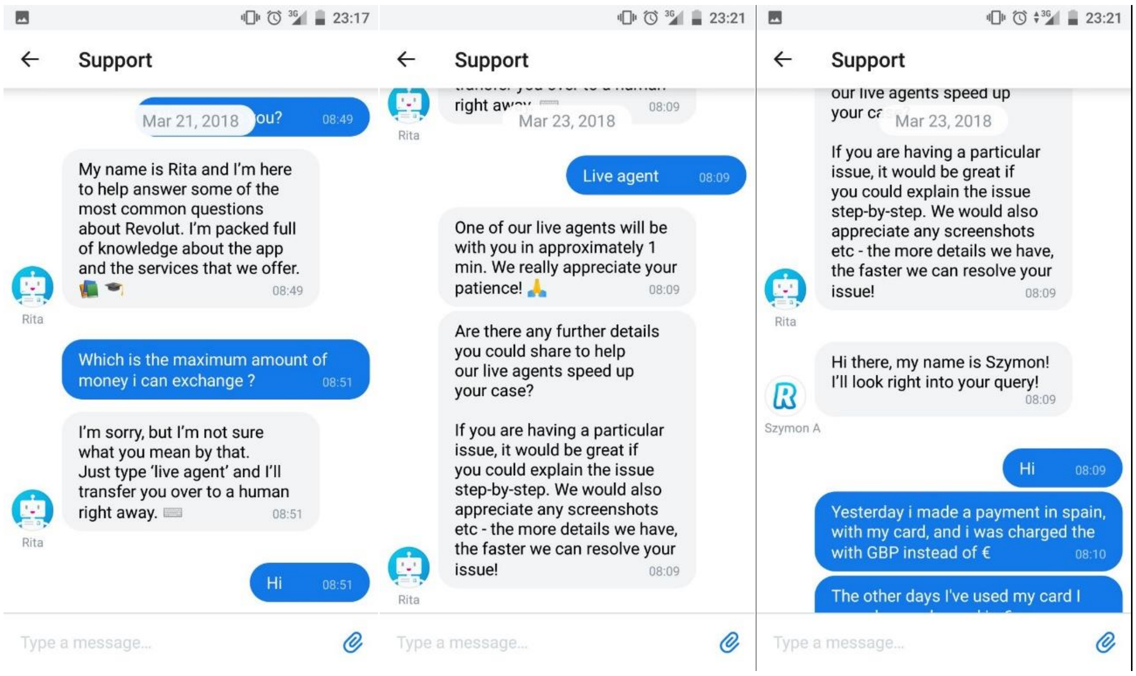
This chatbots guide the user through questions/answers with a certain logic tree, but the bot is also able to handle free text and reply accordingly in a way, to get the user back into one of the set paths to achieve the intended functionality.

4.1.2.4. Human supported bots

These bots go down the path of an artificially intelligent bot, allowing the user to interact using free text, however, there is also always a human operator that is observing and takes over if the bot is no longer handling the requests in an acceptable manner. Of course, these bots can also be trained, by their operators, who can add-in those missing parameters as they occur and teach the bot how to react/reply for future reference.

Revolut Chatbot

The FinTech startup Revolut, dedicated to digital banking that includes pre-paid debit cards, currency exchange, cryptocurrency exchange and peer-to-peer payments, has an app to manage all the finances. The support service has a chat that starts with an AI-powered chatbot that changes to a human when it does not understand the user's requests.


23:17

Support

Mar 21, 2018 ou? 08:49

My name is Rita and I'm here to help answer some of the most common questions about Revolut. I'm packed full of knowledge about the app and the services that we offer. 📚🎓 08:49

Rita

Which is the maximum amount of money i can exchange ? 08:51

I'm sorry, but I'm not sure what you mean by that. Just type 'live agent' and I'll transfer you over to a human right away. ⌨ 08:51

Rita

Hi 08:51

Type a message...

23:21

Support

right aw 08:09

Mar 23, 2018

Rita

Live agent 08:09

One of our live agents will be with you in approximately 1 min. We really appreciate your patience! 🙏 08:09

Are there any further details you could share to help our live agents speed up your case?

If you are having a particular issue, it would be great if you could explain the issue step-by-step. We would also appreciate any screenshots etc - the more details we have, the faster we can resolve your issue! 08:09

Rita

Type a message...

23:21

Support

our live agents speed up your ca Mar 23, 2018

If you are having a particular issue, it would be great if you could explain the issue step-by-step. We would also appreciate any screenshots etc - the more details we have, the faster we can resolve your issue! 08:09

Rita

Hi there, my name is Szymon! I'll look right into your query! 08:09

Szymon A

Hi 08:09

Yesterday i made a payment in spain, with my card, and i was charged the with GBP instead of € 08:10

The other days I've used my card I

Type a message...

Figure 4.7 Revolut Support Chatbot

4.1.3. Classification 3: Answering model

The chatbot can either generate responses from scratch, based on machine learning models or use some heuristic to select a response from a library of predefined responses. [4]

4.1.3.1. Retrieval-based models

Retrieval-based models are easy to build. They also provide predictable results. The user probably will not get 100% accuracy of responses, but at least user knows all possible responses and can make sure that there are no inappropriate or grammatically incorrect responses.

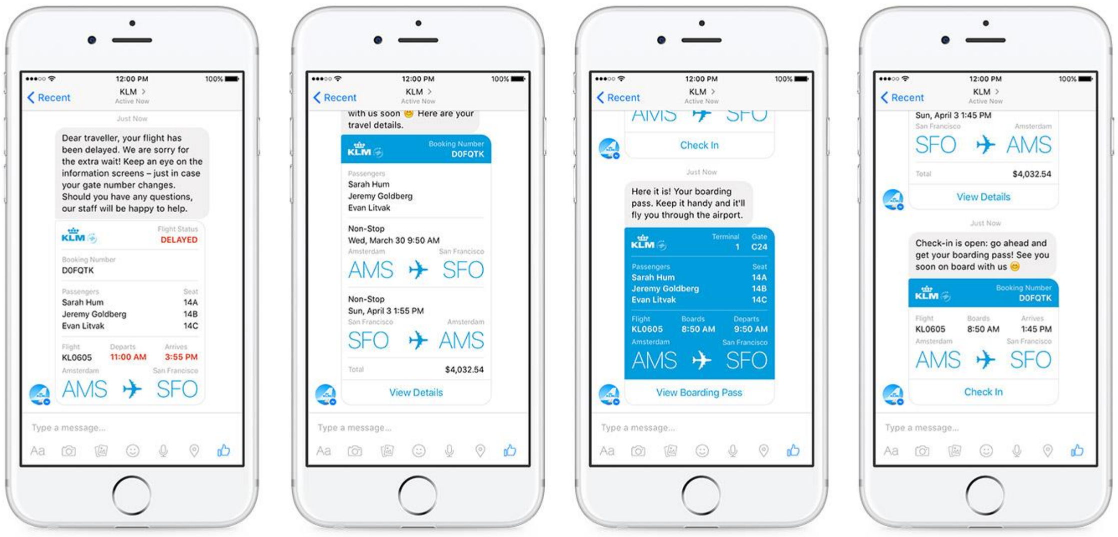
Retrieval-based models are the most practical at the moment for business and procedures since many algorithms and APIs are readily available for developers.

The chatbot uses the message and context of conversation for selecting the best response from a predefined list of chatbot messages. The context can include a current position in the dialog tree, all previous messages in the conversation, previously saved variables (e.g. username).

If the bot does not use context then it is stateless. It will only respond to the latest user message, disregarding all the history of the conversation.

KLM Chatbot

The Dutch Airlines KLM offers the users to receive their flight documentation via Facebook Messenger. After booking their flight on KLM.com the users can choose to receive the booking confirmation, check-in notification, boarding pass and flight status updates via Facebook Messenger. This makes their travel information easy to find in a single place, available at the airport, end route or at home. And if there are any questions the chat is available 24/7.



Dear traveller, your flight has been delayed. We are sorry for the extra wait! Keep an eye on the information screens – just in case your gate number changes. Should you have any questions, our staff will be happy to help.

Flight Status DELAYED
Booking Number D0FQTK
Passengers Sarah Hum 14A, Jeremy Goldberg 14B, Evan Litvak 14C
Flight KL0605 Departs 11:00 AM Arrives 3:55 PM
Amsterdam AMS ✈ SFO San Francisco

with us soon 😊 Here are your travel details.

Booking Number D0FQTK
Passengers Sarah Hum, Jeremy Goldberg, Evan Litvak
Non-Stop Wed, March 30 9:50 AM Amsterdam AMS ✈ SFO San Francisco
Non-Stop Sun, April 3 1:55 PM San Francisco SFO ✈ AMS Amsterdam
Total $4,032.54

Here it is! Your boarding pass. Keep it handy and it'll fly you through the airport.

Terminal 1 Gate C24
Passengers Sarah Hum 14A, Jeremy Goldberg 14B, Evan Litvak 14C
Flight KL0605 Boards 8:50 AM Departs 9:50 AM
Amsterdam AMS ✈ SFO San Francisco

Sun, April 3 1:45 PM San Francisco SFO ✈ AMS Amsterdam
Total $4,032.54

Check-in is open: go ahead and get your boarding pass! See you soon on board with us 😊

Booking Number D0FQTK
Flight KL0605 Boards 8:50 AM Arrives 1:45 PM
Amsterdam AMS ✈ SFO San Francisco

Figure 4.8 Weather Bot Telegram

4.1.3.2. Pattern-based heuristics

Heuristics for selecting a response can be engineered in many different ways, from if-else conditional logic to machine learning classifiers. The simplest technology is using a set of rules with patterns as conditions for the rules. Early chatbots used pattern matching to classify text and produce a response. This is often referred to as "brute force" as the developer of the system needs to describe every pattern for which there is a response. This type of models is very popular for entertainment bots. ELIZA and many other chatbots that compete to pass the Turing test follow this heuristic.

A standard structure for these patterns is AIML (artificial intelligence markup language), a widely used language for writing patterns and response templates.

An example of pattern matching written with AIML:

```
<aiml    version    =    "1.0.1"    encoding    =    "UTF-8"?>
```

```
   <category>
      <pattern>    WHO     IS     ALBERT     EINSTEIN     </pattern>
      <template>Albert Einstein was a German physicist.</template>
   </category>

   <category>
      <pattern>     WHO     IS     Isaac     NEWTON     </pattern>
      <template>Isaac  Newton  was  an  English  physicist  and
mathematician.</template>
   </category>

   <category>
      <pattern>DO     YOU     KNOW     WHO     *     IS</pattern>
      <template>
         <srai>WHO                IS                <star/></srai>
      </template>
   </category>
</aiml>
```

The machine then produces:

```
User:    Do    you    know    who    Albert    Einstein    is?
Chatbot: Albert Einstein was a German physicist.
```

When the chatbot receives a message, it goes through all the patterns until it finds a pattern which matches the user message. If the match is found, the chatbot uses the corresponding template to generate a response.

Despite the most popular language for pattern matching is AIML it can be written in any programming language with a proper use of regular expressions.

4.1.3.3. Machine learning for intent classification

The inherent problem of pattern-based heuristics is that patterns should be programmed manually, and it is not an easy task, especially if the chatbot has to correctly distinguish hundreds of intents. Imagine building a customer service chatbot which has to respond to a refund request. Users can express it in hundreds of different ways: “I want a refund”, “Refund my money”, “I need my money back”. At the same time, the bot should respond differently if the same words are used in another context: “Can I request a refund if I don’t like the service?”, “What is your refund policy?”. Humans are not good at writing patterns and rules for natural language understanding, computers are much better at this task.

Machine learning allows training an intent classification algorithm. It is just needed a training set of a few hundred or thousands of examples, and it will pick up patterns in the data.

Such algorithms can be built using any popular machine learning library like scikit-learn. Another option is to use one of the cloud API: wit.ai, api.ai or Microsoft LUIS. However, it is necessary a great number of examples for the training set to guarantee a reliable accuracy.

4.2. Chatbot industry

Subsequent to reviewing what a chatbot is and which types exist, it is necessary to examine their present and trends to understand the importance of chatbot industry.

4.2.1. Present

It has been from 2016 that thousands of companies started to develop their own chatbots. The future is uncertain, the Artificial Intelligence paradigm and the adoption of new habits of communication will decide the course of the industry.

The global chatbot market was valued at over $190 million in 2016 and is only expected to grow in the coming years. Chatbots are here to stay, and it is not hard to see why. According to a study by Aspect Software Research, 44% of consumers said they would prefer to interact with a chatbot over a human customer service representative. [5]

4.2.2. Trends

Chatbots are a huge trend today. Top brands are seizing the opportunity to meet their customers where they are already spending time in: messaging apps. That the reason there is a massive number of chatbots appearing in various different industries, from e-commerce and fashion to more conservative sectors like banking.

There are three major trends that are prompting the rise of chatbots:

- The ever-present app fatigue
- The explosion of messaging apps
- The increased commoditization of AI algorithms

As app downloads and usage decrease, the use of messaging apps continues to surge. People all over the world use messaging apps more than any other channel and this is why brand-customer communications are increasingly moving to chat.

Companies are building their own chatbots leveraging AI for their domain of knowledge and ability to personalize conversations. These branded chatbots live on existing messaging apps, such as Facebook Messenger, Telegram, Kik, Viber and more.

The chatbots people interact with today may be built to do simple things, but as bot makers start to leverage advanced machine learning technologies, bots will become more intelligent and capable of interacting with users in more contextually relevant ways.

To support the previous paragraphs following there are some statements from chatbot researches:

- "80% of businesses want chatbots by 2020" - Oracle
- "The global chatbot market is set to grow at CAGR of 37.11 during the period of 2017–2021" - Orbis Research
- "Chatbots expected to cut business costs by $8.000 million by 2022" - Juniper Research

Another important point is the popularity acquired in the last 2 years since the announcement of the Facebook and Telegram platforms to develop chatbots. Google trends shows the evolution of people searches in google about the term chatbot.

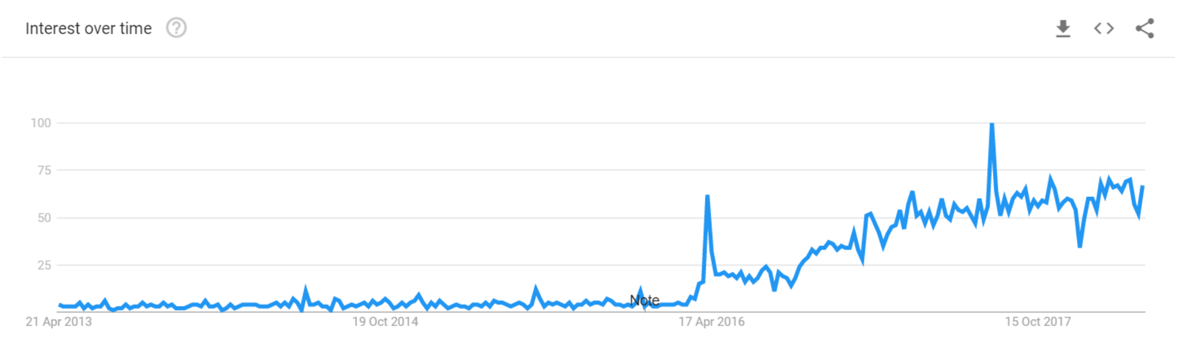


Figure 4.9 Google Trends results for "chatbot"

During the last years, Gartner has included chatbots in their famous hype cycle of emergent technologies. In 2017 Gartner conversational user interfaces were situated in the Innovation Trigger but very close to Peak of Inflated Expectations. That means there has been a number of success stories, often accompanied by scores of failures. Some companies take action; many do not and media interest has triggered significant publicity. [6]

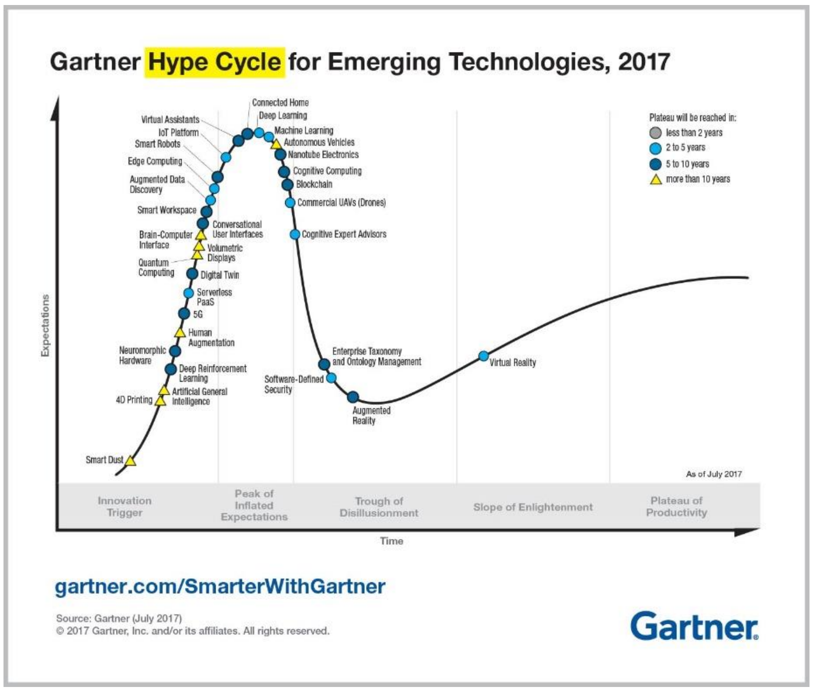


Figure 4.10 Gartner Hype Cycle of Emerging Technologies 2017

5. Methodology

The overall purpose of this project is to create a chatbot to give information about the ETSEIB. On the one hand, this includes understanding the capabilities of chatbots and what can be achieved with this technology. On the other hand, it involves building a software and all the corresponding steps of planning, designing, developing and implementing.

This chapter presents the applied methodology that was decided to be most suitable for addressing the formulated objectives.

The first approach when the project started was to employ an adaptation of the Waterfall model to develop the Chatbot, but after several weeks it was identified the necessity to change the development path. Finally, it was adopted the Agile methodology that incorporates iteration and continuous feedback to successively refine the prototype.

The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is linear and sequential. This brought with it many disadvantages such as the non-possibility to come back after a phase is overcome.

5.1. Agile model

“Agile Development” is an umbrella term for several iterative and incremental software development methodologies. The most popular agile methodologies include Extreme Programming (XP), Scrum, Crystal, Dynamic Systems Development Method (DSDM), Lean Development, and Feature-Driven Development (FDD). [7]

While each of the agile methodologies is unique in its specific approach, they all share a common vision and core values. They all fundamentally incorporate iteration and the continuous feedback that it provides to successively refine and deliver a software system. They all involve continuous planning, continuous testing, continuous integration, and other forms of continuous evolution of both the project and the software. They are all lightweight, especially compared to traditional waterfall-style processes, and inherently adaptable. Another characteristic of agile methods, that does not apply to this project because is being performed individually, is that they all focus on empowering people to collaborate and make decisions together quickly and effectively.

For this project, the adopted methodology is the iterative and incremental development. This method of software development is modeled around a gradual increase in feature additions and a cyclical release and upgrade pattern.

Iterative and incremental software development begins with planning and continues through iterative development cycles involving continuous user feedback and the incremental addition of features concluding with the deployment of completed software at the end of each cycle.

Iterative and incremental development is a discipline for developing systems based on producing deliverables, for this project functional prototypes. In incremental development, different parts of the system are developed at various times or rates and are integrated based on their completion. In iterative development, it is planned to revisit parts of the system in order to revise and improve them. User feedback is consulted to modify the targets for successive deliverables.

Iterative and incremental software development came about in response to flaws in the waterfall model, a sequential design process in which progress flows steadily downwards. It differs from the waterfall model because it is cyclical rather than unidirectional, offering a greater ability to incorporate changes into the application during the development cycle.

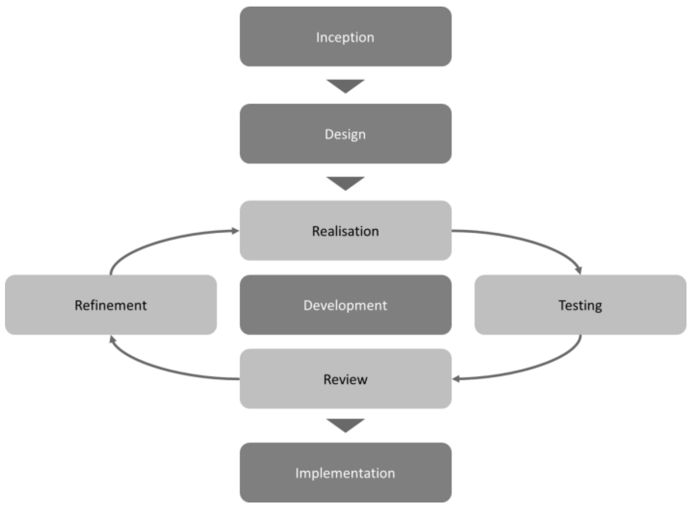


Figure 5.1 Agile Methodology Diagram of Steps

Iterative and incremental development can be grouped into the following phases:

- **Inception**: deals with the scope of the project, requirements, and risks at higher levels.
- **Design**: delivers working architecture that moderates risks identified in the inception phase and satisfies non-functional requirements.
- **Development**: fills in architecture components incrementally with production-ready code, which is produced through the analysis, implementation, design, and testing of functional requirements
- **Implementation**: delivers the system to the production operating environment

In the Annex is it possible to find a comparative example of how waterfall and agile methodologies work.

5.2. Planning

For this project there was an initial planning considering the waterfall methodology but, as explained before, it was dismissed so the final planning corresponds to an iterative approach of many phases in the development.

From the beginning of the project, there has been a willingness to discover and learn about chatbots and discover possible technologies and tools to use for the final prototype. The experimentation coding and the research have accompanied the whole project to guarantee the best results and ensure the success of the project through the incorporation of new ideas and solutions to the different phases.

The final planning of the project is reflected in the following Gantt diagram:

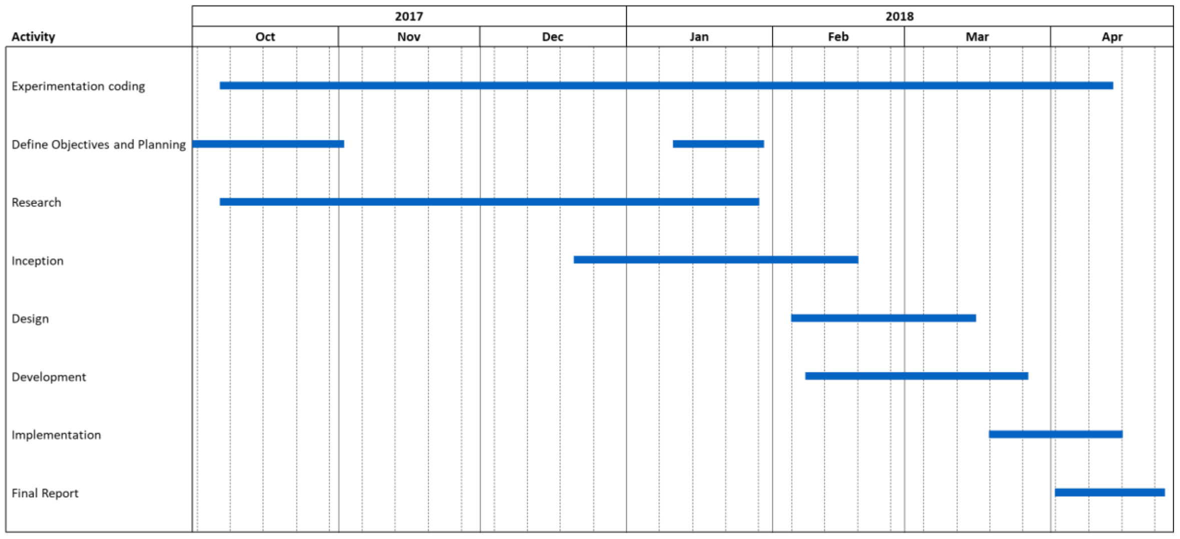


Figure 5.2 Gantt Diagram Planning

In January there was a redefinition of the objectives consequence the new knowledge acquired about chatbots with the research and coding. The planning was also reviewed and adapted to ensure the timings and finalisation of the prototype before the end date.

6. Design & Implementation

The developed chatbot corresponds to a purpose-specific chatbot, designed to answer questions about ETSEIB to students and implemented in the best manner with the available resources of time and knowledge.

6.1. Requirements

Before starting with InduBot's design, the first step has been to determine in detail and exhaustively what capacities should the chatbot have, under which criteria should be taken the different design decisions and which characteristics will differentiate it from other existing systems. The identification of all these requirements is essential to define minimum objectives that must be achieved.

Within the set of requirements, there are two large groups, which together affect decisions in different areas: functional and non-functional.

Below are the differences between each group and a list that includes all the identified requirements, their motivation and their influence on the final product.

6.1.1. Functional requirements

This group includes the requirements that directly affect the capabilities of the bot, both comprehension of the input, generation of an answer in natural language and accuracy in the information delivered.

The following requirements have been identified:

Speed

One of the prime purposes for the existence of a chatbot is to help the users instantly. The speed of the chatbot should be faster than the time a human will take to write the answer. When building the chatbot, it should be integrated with knowledge-based database and programmed to fetch information and respond quickly. For example, if a student wants to know about the date of an exam, the chatbot should provide the information faster than what it takes to search it on the university website.

Purpose oriented

The chatbot has been built to provide students information about the university, so the flow of the conversations should always be oriented towards that goal. If a user just wants to have a conversation with the chatbot it may found it boring since it has been conceived for another purpose and conversation is not its strength. This is why the chatbot must clearly state in the presentation its purpose and do not create false expectations for the user.

The following image reflects the welcome screen of the chatbot when starting a conversation with it. The scope is clearly stated and the different options are presented as buttons so the user knows exactly what to expect.

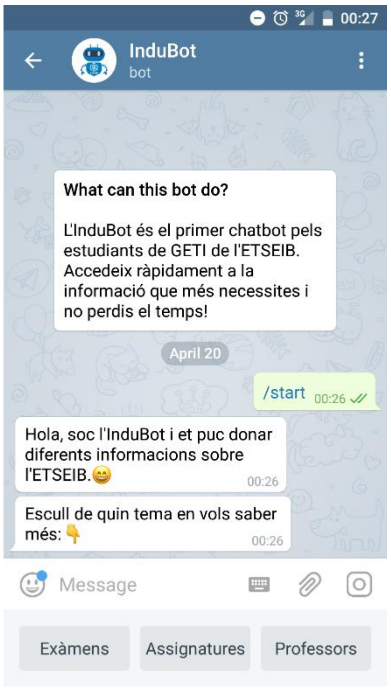

00:27
InduBot
bot
What can this bot do?
L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!
April 20
/start 00:26
Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😄 00:26
Escull de quin tema en vols saber més:👇 00:26

Figure 6.1 InduBot welcome screen

Comprehension capabilities

The chatbot must be able to understand natural language within a certain domain and be able to determine the user input topic independently of the construction of the sentence.

Good comprehension capabilities of a chatbot should ensure the error-free experience for the user. When a user types a spelling mistake or makes an error in a sentence, the program should enable the “auto-correct” feature to avoid that little mistakes can influence the comprehension of the chatbot.

When the user is requested to write the name of a course, InduBot is able to check if it exists within the list of ETSEIB courses of the Degree of Industrial Engineering, and in case it matches with an accuracy of more than 95% the conversation can proceed. Otherwise, the chatbot gives the 3 most similar options for that subject.

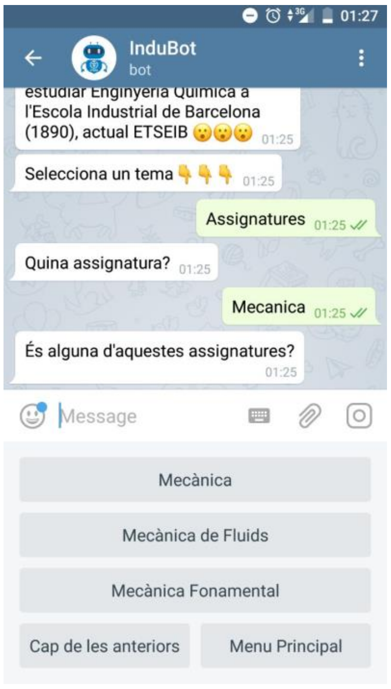

01:27
InduBot
bot
estudiar Enginyeria Química a l'Escola Industrial de Barcelona (1890), actual ETSEIB 😮😮😮 01:25
Selecciona un tema👇👇👇 01:25
Assignatures 01:25
Quina assignatura? 01:25
Mecanica 01:25
És alguna d'aquestes assignatures? 01:25

Figure 6.2 InduBot Suggestions

Answer capabilities

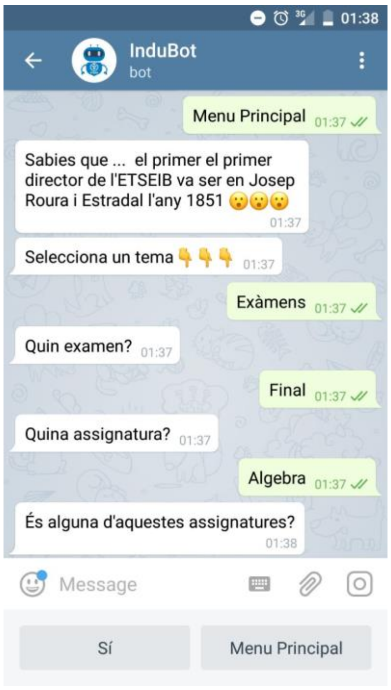
Once the chatbot understands what the user says, it should be able to choose or generate a response, based on the current input and the context of the conversation. The given answers must be coherent with the context derived from the entry.

The only reason to accept an incoherence with the conversation thematic will be the entry of an unknown word. In such a case, the expansion of the vocabulary with the necessary relative information must be sufficient to for correctly interpret and answer in the future. However, the chatbot should be able to guide the conversations to avoid this kind of situations.

User engagement

The chatbot should be capable of initiating conversation with the users and interact with them to share information. Secondly, InduBot should be capable of "balanced text-use". This means that the chatbot must use a combination of both short descriptions and engaging content like emojis to hold the user's attention. In addition, it is important to welcome the user with onboarding steps and present an intuitive layout to improve user's experience.

InduBot makes an extensive use of emoticons and quick answers to invite the user to continue the conversation and be eager to use the chatbot.



Menu Principal 01:37

Sabies que ... el primer el primer director de l'ETSEIB va ser en Josep Roura i Estradal l'any 1851 😮😮😮 01:37

Selecciona un tema 👇👇👇 01:37

Exàmens 01:37

Quin examen? 01:37

Final 01:37

Quina assignatura? 01:37

Algebra 01:37

És alguna d'aquestes assignatures? 01:38

Figure 6.3 InduBot emojis use

Deflection capabilities

In case the chatbot does not understand the input or the user is just trying to crash the chatbot's logic it is important to have the capacity to give a canned response. The user will get the feeling he/she is speaking with a smart chatbot.

Language style and vocabulary

The chatbot should speak the users' language, with words, phrases, and concepts familiar to the user, rather than keywords. Follow real-world conventions, making information appear in a natural and logical order. And the vocabulary should be adapted to the domain of the university.

Consistency

Users should not have to wonder whether different words, situations, or actions mean the same thing. Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

6.1.2. Non-functional requirements

Interoperability

Interoperability simply means the ability of computer systems or software applications to exchange and make use of information. In this case, the chatbot would be implemented in only one channel but the way it is programmed should be able to support multiple channels.

Scalability

The chatbot should be designed to be scalable so that it can support numerous users and additional modules at the same time. Also, the chatbot should be built to accommodate itself in most server environments. So regardless of any server environment, chatbot should be capable of working on either of them.

Languages

The chatbot must understand Catalan for its operation, but it should be considered to extend the available languages in the future.

6.2. Chatbot Design

The chosen design for the chatbot is presented in this section. Firstly, it is explained the selection of the used technologies and the adopted architecture, and then it is detailed the applied design to comply with the requirements and achieve the stated objectives of the project.

6.2.1. Technologies

To develop the chatbot there are two considerations referred to technologies, the programming language used for all the code and business logic and the databases where the knowledge is stored.

Programming languages

Currently, there are many programming languages in the market, each one with its advantages and drawbacks. For this project, the selection has been made according to what other programmers usually do when programming chatbots. In addition, it has been considered the versatile and easy to use of the language to face not only the coding of the chatbot logic but also the need to support the API and webhook to connect with the messaging service platform.

After analysing the valid programming languages used to program chatbots Python was found the most appropriate because:

- It is simply to build an API and connect with the messaging platform. It supports Facebook Messenger, SMS, Slack, Telegram, Twitter, Kik, etc.
- It is the more popular language for Machine Learning and Natural Language Processing.
- It has good support for serving web content: when scaling up the application to allow it to receive many messages per second.
- It is portable: it is easy to run the same code on Linux, MacOS, or Windows.

Additionally, most of the forums on the internet recommend Python to build chatbots. That reinforces the legitimacy of the election.

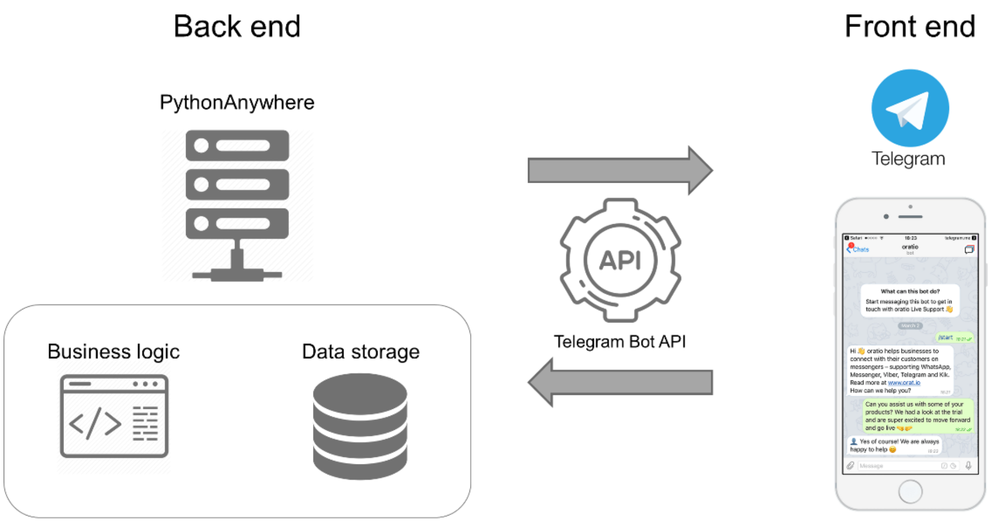
Databases

This project requires storing the knowledge about the university in some kind of source. When implementing the prototype it has been decided using dictionaries was the easiest way to access the information. The information will be stored in CSV files and the dictionaries created when starting the application. An approach using MySql was frustrated by the incompatibilities and limitations of queries with the web hosting provider PythonAnywhere.

6.2.2. Architecture

Before starting to get into details is important to have the big picture of how it is addressed the architecture. The chatbot can be divided into 2 layers: the back end and the front end.

- Back end: the layer where business logic and data storage is managed. This code resides on the server.
- Front end: the channel through the user interacts with the chatbot. It can be a web application or a messaging service.



What can this bot do?
Start messaging this bot to get in touch with oratio Live Support 👋

/start

Hi 👋 oratio helps businesses to connect with their customers on messengers – supporting WhatsApp, Messenger, Viber, Telegram and Kik. Read more at www.orat.io
How can we help you?

Can you assist us with some of your products? We had a look at the trial and are super excited to move forward and go live 🤜🤛

👤 Yes of course! We are always happy to help 😊

Figure 6.4 Schema of InduBot architecture

For this project the back end is being hosted in a web hosting service, the front end is provided through a messaging service and the connection between them is done by an API.

6.2.2.1. Back End

Web hosting service: PythonAnywhere

To host the business logic of the chatbot and all the data files for InduBot's functional operation it has been selected PythonAnywhere. It is not only a web hosting service, but it also provides an online Integrated Development Environment (IDE), which allows to code from any device (even smartphones) just by entering a browser and accessing the web application.

The IDE was created to work specifically with Python, which is the used programming language for this project. This feature grants that there are enough documentation and threads in forums to perform the coding without unexpected problems that can stop the progress of the development.

Web framework: Flask

To support the development of the application and provide a standard way to deploy the chatbot it has been used Flask. The functionalities of this web framework made it easier and reduce the time of coding when setting up the server.

6.2.2.2. Front End

Messaging Service: Telegram

To interact with the user there are several options: create a web application, develop a mobile app or use an existing messaging service. The interface should be user-friendly, allow instant answers and be supported by the maximum devices possible. When considering channels the first option was Whatsapp, but since it does not support chatbots with an API, there were two clear options: Facebook Messenger and Telegram. Facebook Messenger has more tools and capabilities such as Payments or Analytics, which are not necessary for this project, and is more popular among students. But Telegram offers an easier API to use, better documentation and security. So for this project Telegram resulted to be the most suitable path of communication between the user and InduBot.

In addition, Telegram is increasing its user base much faster than the other big messaging services Whatsapp, Facebook Messenger, Viber, Line, etc. And despite being smaller it has a huge potential for growth thanks to its privacy policies and the push on chatbots.

6.2.2.3. APIs

To connect the back end with the front end it has been used the Telegram Bot API, which is basically an HTTP-based interface created for developers to build chatbots for Telegram.

6.2.3. Functional Design

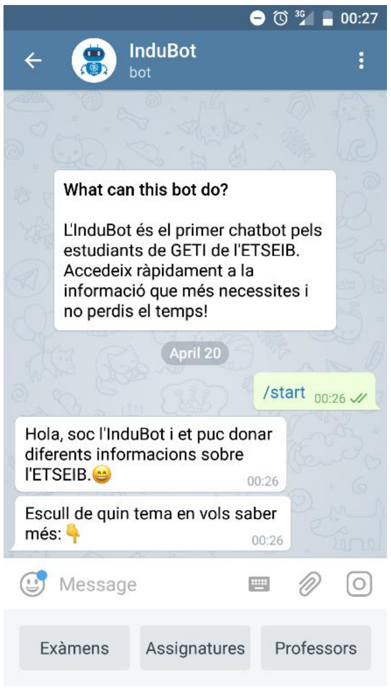
The functional design considers what the chatbot will do and what answers it should provide. It also identifies the scope of the chatbot, the conversational flow with the user and the UX and personality.

6.2.3.1. The chatbot functionalities

The first step is to answer the following questions about InduBot to describe what the bot does:

- What is the main user goal?
    - Obtain information about the ETSEIB.
- What is a successful conversation?
    - Ask for the information and receive the answer faster than it will take to look at the website.
- What are all features the bot has?
    - The bot has information about exams, teachers, and courses.

With the answer to these questions, it has been elaborated a description of InduBot that is the one used to present itself on Telegram.


00:27
InduBot
bot
What can this bot do?

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:26

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😄 00:26

Escull de quin tema en vols saber més:👇 00:26

Figure 6.5 InduBot welcome screen

The scope of InduBot has been limited to this project to answering questions about Exams, Teachers, and Courses of the Degree of Industrial Engineering of the ETSEIB. Information about procedures, Erasmus, internships or any other information that may appear on the ETSEIB website is out of scope.

6.2.3.2. The conversational flow

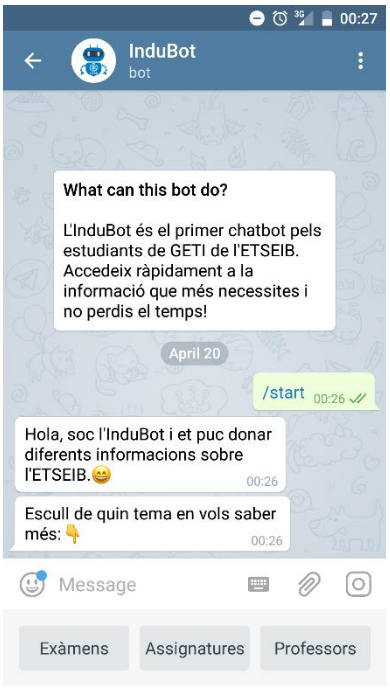
The flow of the conversation determines the necessary steps to go through for the user to reach the goal. Here it is included the questions the chatbot has to say to guide the user to the end and the information the user has to give to get a great reply.

It is very important to conceive each step of the conversation as a state. When certain parameters are fulfilled it is possible to jump from one state to another.

To represent the flow of the conversation it has been used a flowchart. This diagram allows structuring the interaction the chatbot has with the user and most importantly it can be used to determine the purpose orientation requirement, guiding the user through a defined path state by state until the reaching the goal.

The chatbot logic tree can be split into 3 categories, each with its own logic subtree. When starting the chat, InduBot presents itself to the user and displays the different options as shown in the figure below.

The generic flowchart is shown below.



What can this bot do?

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:26

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😄 00:26

Escull de quin tema en vols saber més:👇 00:26

Figure 6.6 InduBot welcome with 3 categories

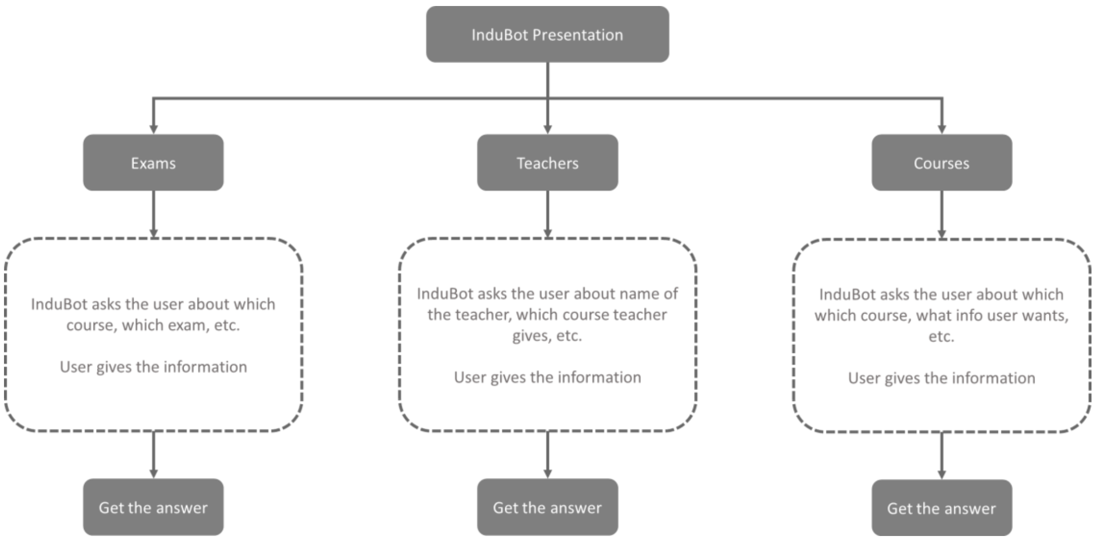

InduBot Presentation
Exams
Teachers
Courses
InduBot asks the user about which course, which exam, etc.
User gives the information
InduBot asks the user about name of the teacher, which course teacher gives, etc.
User gives the information
InduBot asks the user about which which course, what info user wants, etc.
User gives the information
Get the answer
Get the answer
Get the answer

Figure 6.7 InduBot flowchart

Each of the branches of the tree has a more specific flowchart that shows in detail the interaction user-chatbot.

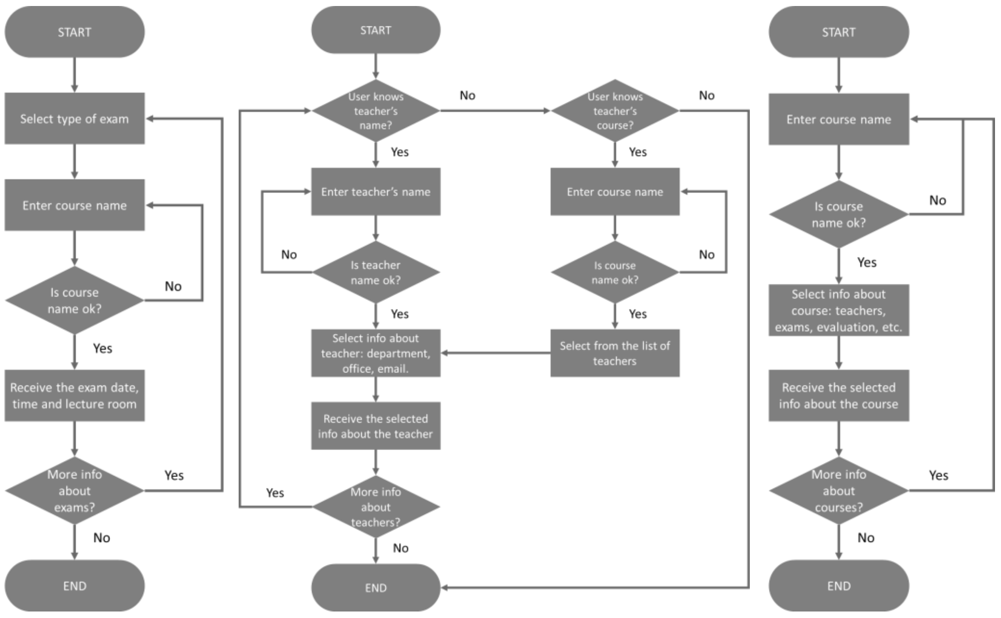
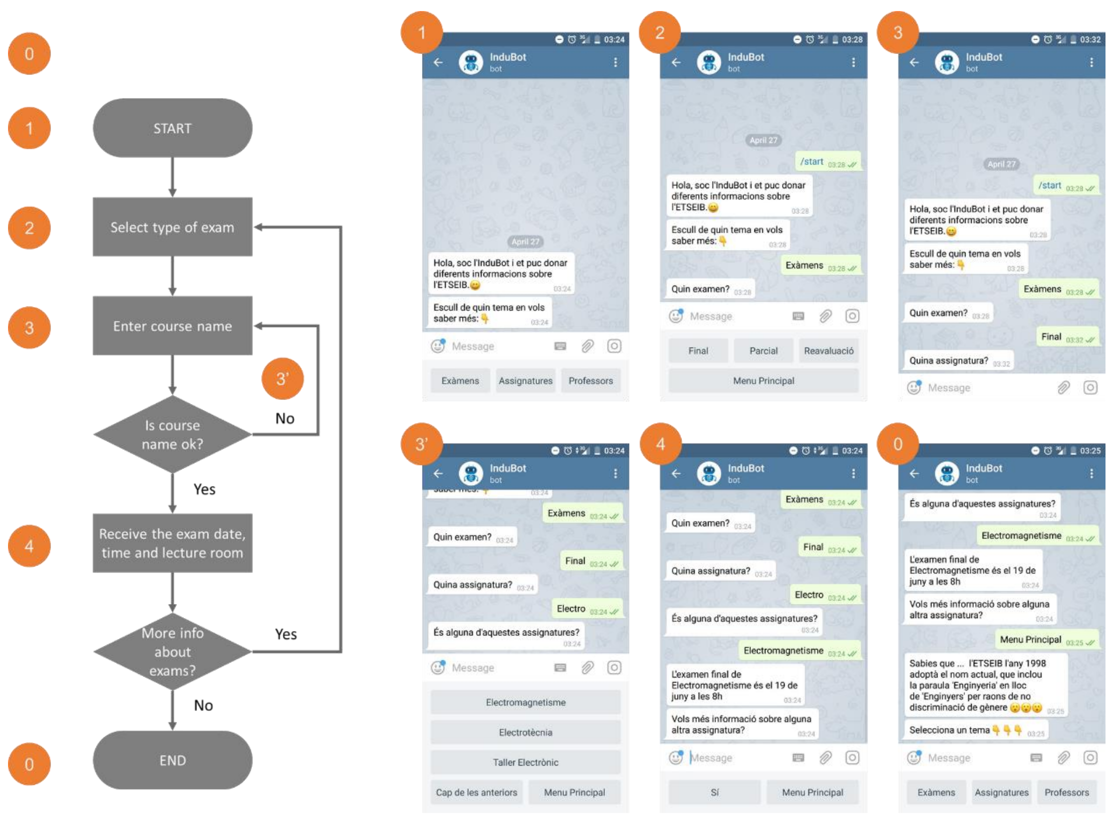

START
Select type of exam
Enter course name
Is course name ok?
No
Yes
Receive the exam date, time and lecture room
More info about exams?
Yes
No
END
START
User knows teacher's name?
No
Yes
Enter teacher's name
Is teacher name ok?
No
Yes
Select info about teacher: department, office, email.
Receive the selected info about the teacher
More info about teachers?
Yes
No
END
User knows teacher's course?
No
Yes
Enter course name
Is course name ok?
No
Yes
Select from the list of teachers
START
Enter course name
Is course name ok?
No
Yes
Select info about course: teachers, exams, evaluation, etc.
Receive the selected info about the course
More info about courses?
Yes
No
END

Figure 6.8 InduBot flowcharts by category

All the steps in the flow consider the possibility the user makes a mistake so there are functions to handle errors and redirect the user to the beginning or the previous step.

To fully understand the flow of a conversation lets illustrate it with an example with the states indicated in every moment. The following example contemplates the request of information for the final exam of the course of *Electromagnetisme*.



1

April 27
Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😀 03:24
Escull de quin tema en vols saber més:👇 03:24

2

April 27
/start 03:28
Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😀 03:28
Escull de quin tema en vols saber més:👇 03:28
Exàmens 03:28
Quin examen? 03:28

3

April 27
/start 03:28
Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😀 03:28
Escull de quin tema en vols saber més:👇 03:28
Exàmens 03:28
Quin examen? 03:28
Final 03:32
Quina assignatura? 03:32

3'

Exàmens 03:24
Quin examen? 03:24
Final 03:24
Quina assignatura? 03:24
Electro 03:24
És alguna d'aquestes assignatures? 03:24

4

Exàmens 03:24
Quin examen? 03:24
Final 03:24
Quina assignatura? 03:24
Electro 03:24
És alguna d'aquestes assignatures? 03:24
Electromagnetisme 03:24
L'examen final de Electromagnetisme és el 19 de juny a les 8h 03:24
Vols més informació sobre alguna altra assignatura? 03:24

0

És alguna d'aquestes assignatures? 03:24
Electromagnetisme 03:24
L'examen final de Electromagnetisme és el 19 de juny a les 8h 03:24
Vols més informació sobre alguna altra assignatura? 03:24
Menu Principal 03:25
Sabies que ... l'ETSEIB l'any 1998 adoptà el nom actual, que inclou la paraula 'Enginyeria' en lloc de 'Enginyers' per raons de no discriminació de gènere 😮😮😮 03:25
Selecciona un tema 👇👇👇 03:25

Figure 6.9 InduBot exam flowchart with states

6.2.3.3. UX and personality

The first aspect before continue is giving a name to the chatbot and creating an avatar for it. The given name comes from the fusion of the words Industrial (from Industrial Engineering) and Chatbot deriving in InduBot. For the avatar it was thought that it should look like a robot with some features to identify it with ETSEIB. The result is the shown in the picture at right. The colours also correspond to the ones of the university.

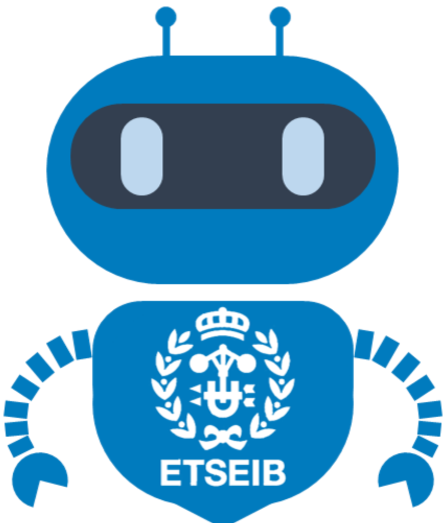


Figure 6.10 InduBot Avatar

The only interactions the users will have with InduBot is through conversation, so building a rich and detailed personality makes the chatbot more believable, relevant and fun to the users.

Personality creates a deeper understanding of the chatbot's end goal, and how it will communicate through the choice of language, mood, tone, and style.

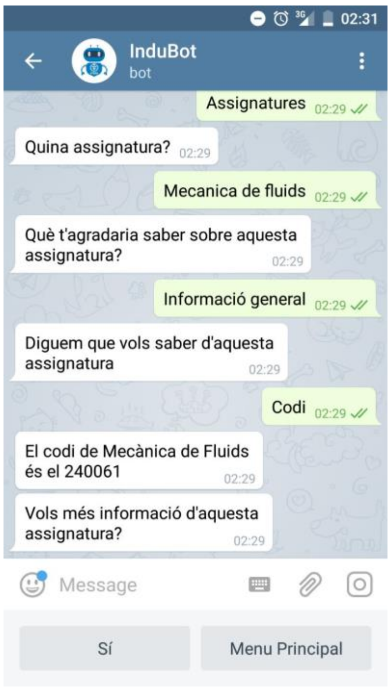

InduBot
bot
Assignatures 02:29
Quina assignatura? 02:29
Mecanica de fluids 02:29
Què t'agradaria saber sobre aquesta assignatura? 02:29
Informació general 02:29
Diguem que vols saber d'aquesta assignatura 02:29
Codi 02:29
El codi de Mecànica de Fluids és el 240061 02:29
Vols més informació d'aquesta assignatura? 02:29

Figure 6.11 InduBot buttons and short answers

InduBot will interact with students of Industrial Engineering who want to get punctual information about the university. So, probably, the interactions will be short and the user will want a straight to the point answer to his/her questions. To craft a personality that copes with this, InduBot's answers should be short and specific, giving the required information. Besides InduBot will also use a friendly tone and in a happy mood. In the figure at left it is possible to observe how InduBot answers exactly what is required, the code of the subject, in a clear way.

Once defined the personality it has been analysed the UX elements offered by Telegram. Actually, it supports text, images, emoji's, gifs, audio, buttons and much more. However for this project it has been used text entries, buttons to guide the conversation and quick replies and emoji's to enrich the conversation. Indeed emojis are essential for building the personality of InduBot, in many cases is easier a happy face than a text to express agreement. The figure at left shows the buttons with quick replies.

6.2.4. Ontology

In computer science and information science, an ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really exist in a particular domain of discourse. An ontology compartmentalizes the variables needed for some set of computations and establishes the relationships between them.

The components of an ontology can be grouped into 4 different categories:

- Classes: Sets, collections, concepts, classes in programming, types of objects, or kinds of things, such as the class *Course*.
- Attributes: Aspects, properties, features, characteristics, or parameters that objects (and classes) can have. For example, the class *Course* has the attribute *Code*.
- Relations: Ways in which classes and individuals can be related to one another.
- Individuals: Instances or objects to a class. For example the *Course* (instance) of *Àlgebra Lineal* (individual).

Furthermore, a domain is necessary to contextualize the ontologies. Is not the same to refer to a card in a poker domain ("playing card") than to do it in a computer hardware domain ("video card"). In this project, the domain is restricted to the field of the university.

The following figures illustrate the ontology of InduBot.

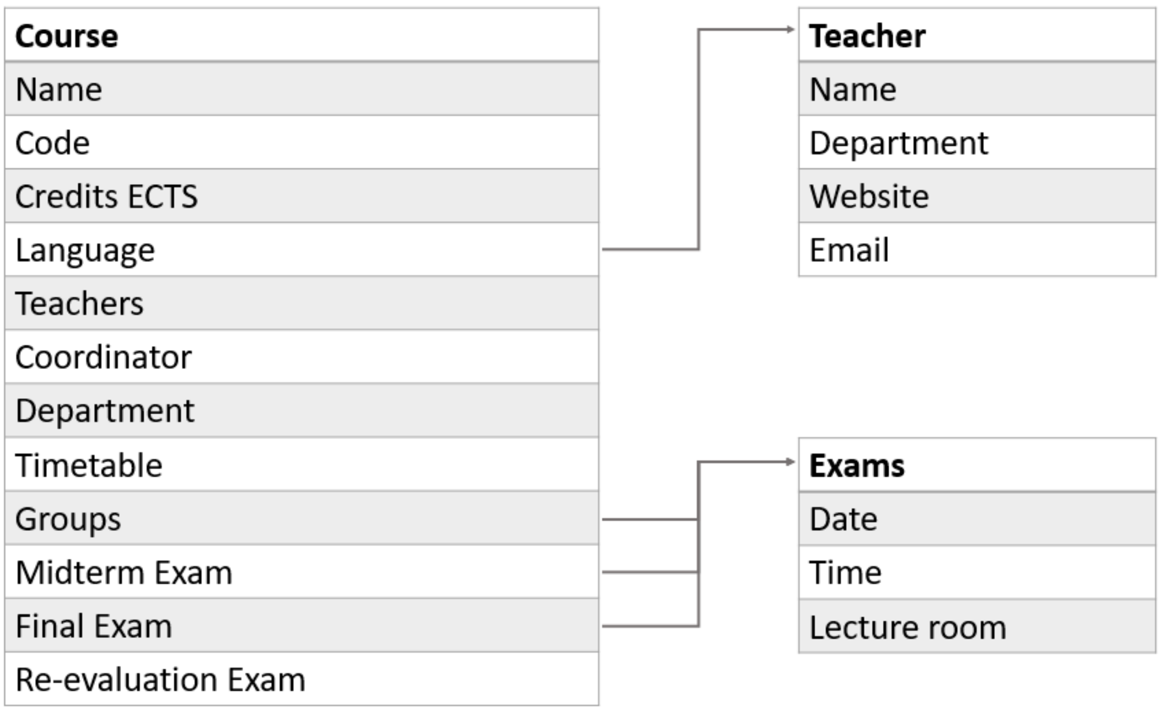


Figure 6.12 InduBot general ontology

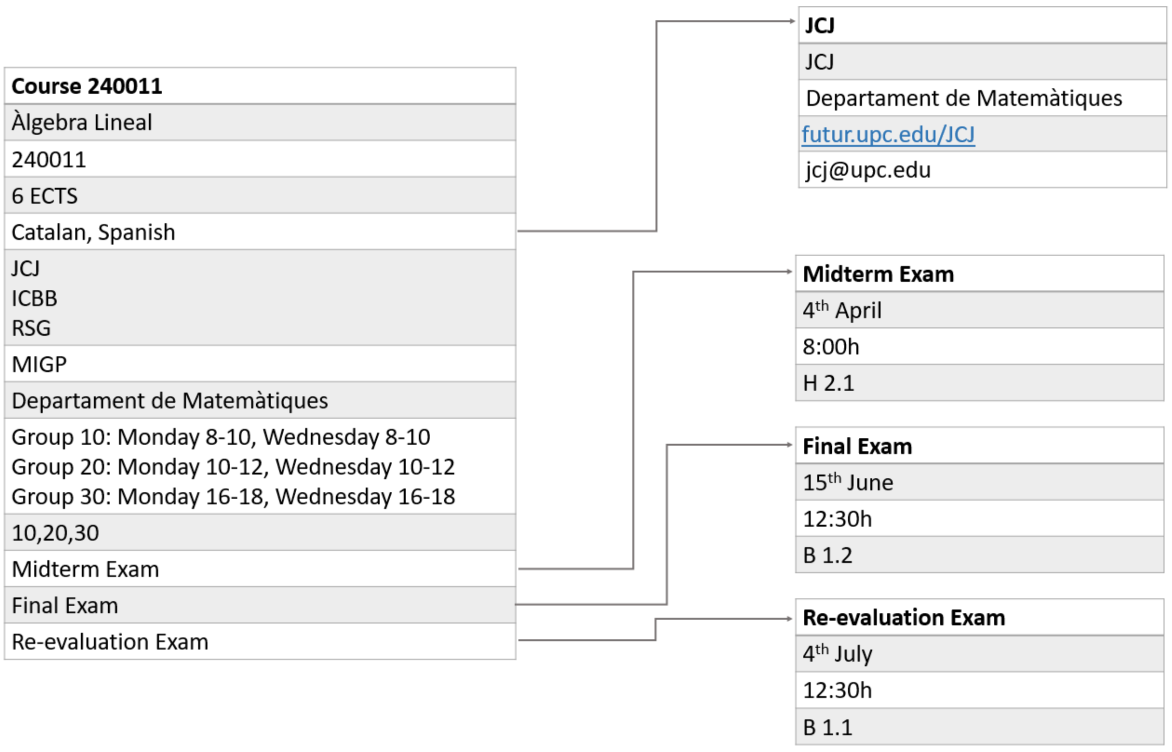
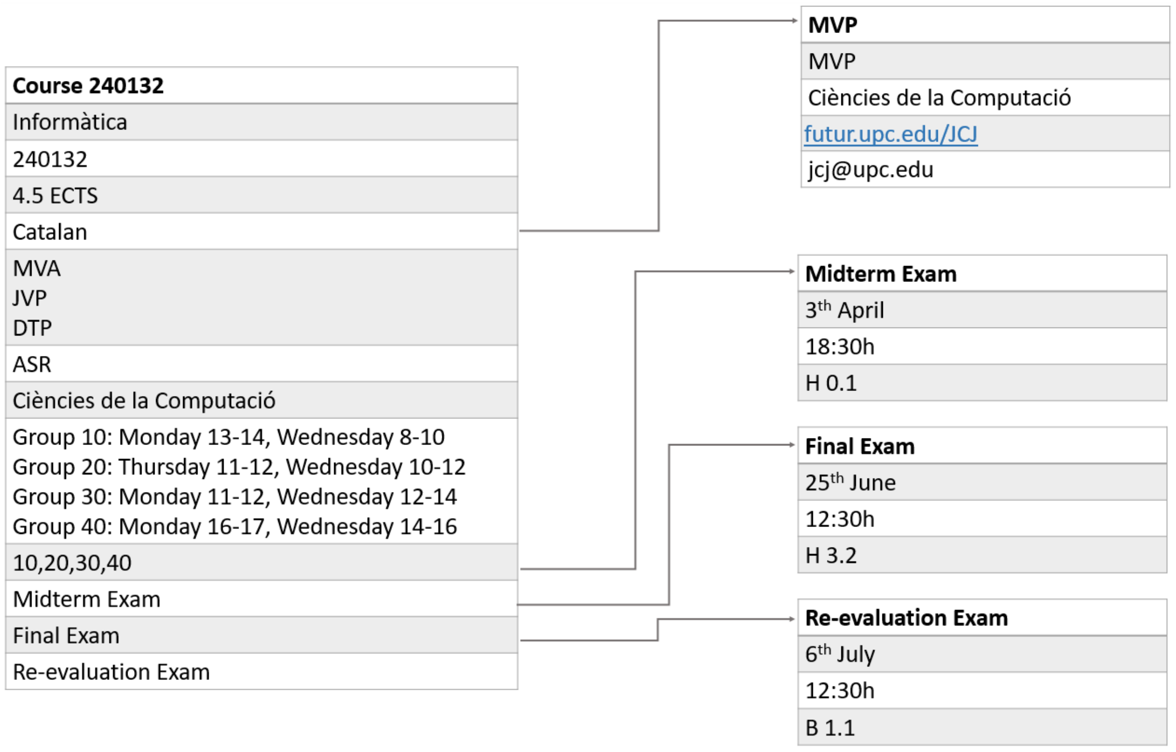


Figure 6.13 InduBot example ontology for Àlgebra Lineal



Course 240132
Informàtica
240132
4.5 ECTS
Catalan
MVA JVP DTP
ASR
Ciències de la Computació
Group 10: Monday 13-14, Wednesday 8-10 Group 20: Thursday 11-12, Wednesday 10-12 Group 30: Monday 11-12, Wednesday 12-14 Group 40: Monday 16-17, Wednesday 14-16
10,20,30,40
Midterm Exam
Final Exam
Re-evaluation Exam

MVP
MVP
Ciències de la Computació
futur.upc.edu/JCJ
jcj@upc.edu

Midterm Exam
3th April
18:30h
H 0.1

Final Exam
25th June
12:30h
H 3.2

Re-evaluation Exam
6th July
12:30h
B 1.1

Figure 6.14 InduBot example ontology for Informàtica

6.2.5. Technical Design

Once the functional design and the ontologies are defined it is necessary to specify the business logic and the knowledge base of InduBot.

As explained in the conversational flow, InduBot design is conceived to guide the user through questions/answers with a certain logic tree. The flow should be understood as a compilation of states, each one accessible by fulfilling specific parameters. For example, if the user wants to know the date of an exam it is necessary to specify the course and the type of exam (final, midterm or recovery).

At this point in the project, came one of the most important decisions for a chatbot: how the input is going to be processed. Here are the main approaches:

- Machine Learning for intent classification: dismissed
- Pattern Matching: tried but finally dismissed
- Retrieval model with keywords and script: used one

From the beginning, it has been dismissed the use of the heuristic of machine learning for intent classification since it does not exist a training set for such model. A training set of conversations from ETSEIB students talking with academic support asking about exams and courses. Of course, there was the option to create one, but it would have required the involvement of academic support and the permission from students, so the idea was totally rejected.

Here there were two possible approaches for the business logic: a first one that uses a hybrid logic with pattern matching techniques. And the second approach with a more rigid flow with structured data and much more focused on performance. Finally, the second option results to fit better for this project and has been implemented as the final one.

First approach: Hybrid logic and pattern based heuristics

During the project objectives definition, it was stated that a goal is to provide the user an experience similar to the one of talking with a human through a chat. To achieve this it is necessary to allow free text input, so the user does not have the sensation of being limited when formulating questions. This first design approach considers a conversation flow, meaning that depending on what the user asks it could be required to facilitate some information about the topic to deliver an accurate answer. However, the intended flow is flexible, and the user could jump from topic to topic.

The chatbot follows a predefined flow and tries to understand what the user wants to know, sometimes questions can be answered within one interaction, other times it would be necessary more iterations. It is also important the idea of how the input is processed, pattern matching techniques compare the input with a pattern and pairs it with the most similar one. Afterward, it answers with the corresponding predefined answer from the matching template. To illustrate how it works it can be taken the example of answering a course exam's date.

The user wants to know when is the exam's date of a certain course, and can ask it in different ways. And the chatbot has a list of templates with patterns, which fortunately one would match with the exam's date questions.

Assuming the patterns / answer for the intent GET FINAL EXAM DATE are:

1) *Quan és l'examen final de* < COURSE > / L'examen final de < COURSE > és el dia < DAY >
2) *Quin dia és l'examen final de* < COURSE > / L'examen final de < COURSE > és el dia < DAY >
3) *L'examen final de* < COURSE > quin dia és / L'examen final de < COURSE > és el dia < DAY >
4) *L'examen final de* < COURSE > quan és / L'examen final de < COURSE > és el dia < DAY >

And the patterns / answer for the intent GET EXAM DATE MISSING EXAM TYPE (when exam is not defined between FINAL or MIDTERM) are:

1) *Quan és l'examen de* < COURSE > / *Final o Parcial?*

2) *Quin dia és l'examen de* < COURSE > / *Final o Parcial?*
3) *L'examen de* < COURSE > quin dia és / *Final o Parcial?*
4) *L'examen de* < COURSE > quan és / *Final o Parcial?*

And the patterns/answer for the intent GET EXAM TYPE are:

1) *Final*
2) *Parcial*

Where COURSE is the entity the chatbot needs to identify.

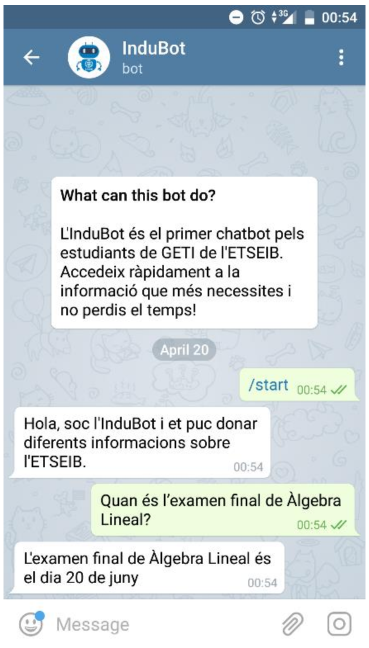
And the scripted answer is:

Where DAY is the entity for the exam's date and is obtained from a database.

Easy Scenario. If the user asks:

Quan és l'examen final de Àlgebra Lineal?

The input matches with the pattern *Quan és l'examen final de* < COURSE > from the intent GET FINAL EXAM DATE. The entity COURSE is easily identified and the program just needs to search in the database the corresponding day to answer the correct date to the user.



What can this bot do?

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:54

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB. 00:54

Quan és l'examen final de Àlgebra Lineal? 00:54

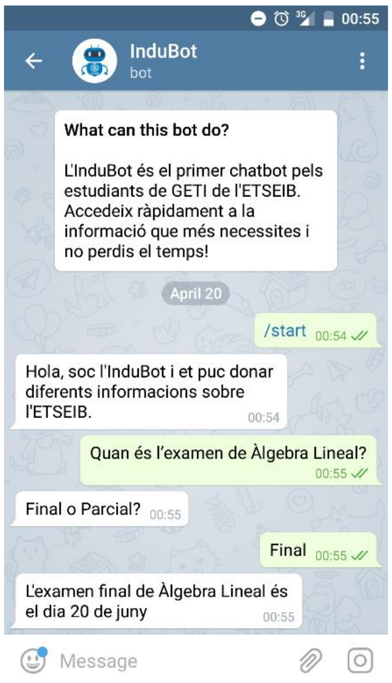
L'examen final de Àlgebra Lineal és el dia 20 de juny 00:54

Figure 6.15 InduBot First Approach: easy scenario

Medium scenario. If the user asks:

`Quan és l’examen de Àlgebra Lineal?`

The input is matched with the `Quan és l’examen de < COURSE >` from the intent GET EXAM DATE MISSING EXAM TYPE. Here the chatbot will ask about which kind of exam is with the following question: `Final o Parcial?`. Then the user will answer `Final` and the information from the corresponding course final exam will be retrieved.



00:55

InduBot
bot

What can this bot do?

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:54

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB. 00:54

Quan és l’examen de Àlgebra Lineal? 00:55

Final o Parcial? 00:55

Final 00:55

L'examen final de Àlgebra Lineal és el dia 20 de juny 00:55

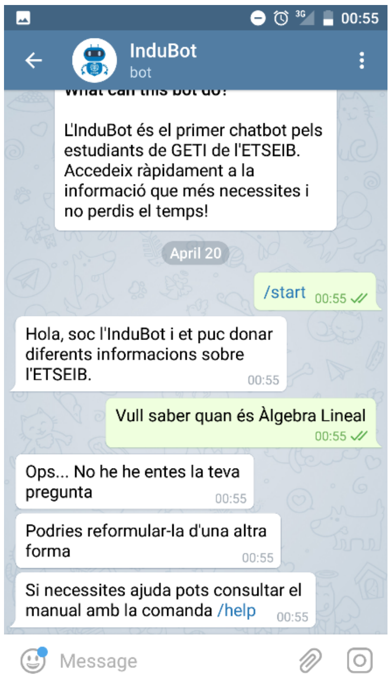
Message

Figure 6.16 InduBot First Approach: medium scenario

Difficult scenario. If the user asks:

`Vull saber quan és Àlgebra Lineal`

There is no pattern to match with, the chatbot will give the predefined answer for situations when it does not understand the user’s input.


InduBot
bot

What can this bot do?

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:55

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB. 00:55

Vull saber quan és Àlgebra Lineal 00:55

Ops... No he he entes la teva pregunta 00:55

Podries reformular-la d'una altra forma 00:55

Si necessites ajuda pots consultar el manual amb la comanda /help 00:55

Figure 6.17 InduBot First Approach: difficult scenario

Here there are many possible initiatives to attempt to educate the user on how to properly post questions in order to receive an answer and improve results. For example:

- Every time the user mistakes entering a question because the words are a little different from the ones in the patterns, try to discover the intent and propose a way to ask the question so the chatbot can understand.
- Create a manual to display before starting the conversation with the chatbot that shows all or part of the accepted questions that the user can formulate and the chatbot will correctly process and answer.

Another feature is the misspellings made by the user. The input has to be corrected before trying the matching with patterns to increase success matching rate. To do so it can be created a spelling corrector that checks input before processing it.

These are some of the applied ideas to guide the conversation towards the desired result and provide better answering.

This first approach is strong in some aspects:

- Once identified the intent is easy to extract the entity and formulate the scripted answer with the appropriate information retrieved from a database, giving an accurate result.
- It is quite easy to implement, it just necessary to write down different patterns for each intent the chatbot has to answer.

However, there are many points that make it a bad approach:

- When delivering an accurate answer to the user it can fail because of the need to precisely formulate the questions in a certain way.
- The learning curve for the user is not fast, it has to memorize the different ways to pose questions. Besides, if there is not a manual at the beginning where the capabilities of the chatbot are clearly stated, the user may not discover all the possibilities the chatbot offers since the layout does not present the different options.
- Being required to repeat the same wording structure every time you want some information can be boring for the user.
- Despite the user may have a good experience when the chatbot understands the input because it is similar to chat with a person, on many other occasions it can be frustrating
- To cope with different ways of asking questions it is necessary to write hundreds of alternative patterns for each intent, trying to include all the variations the user may use.

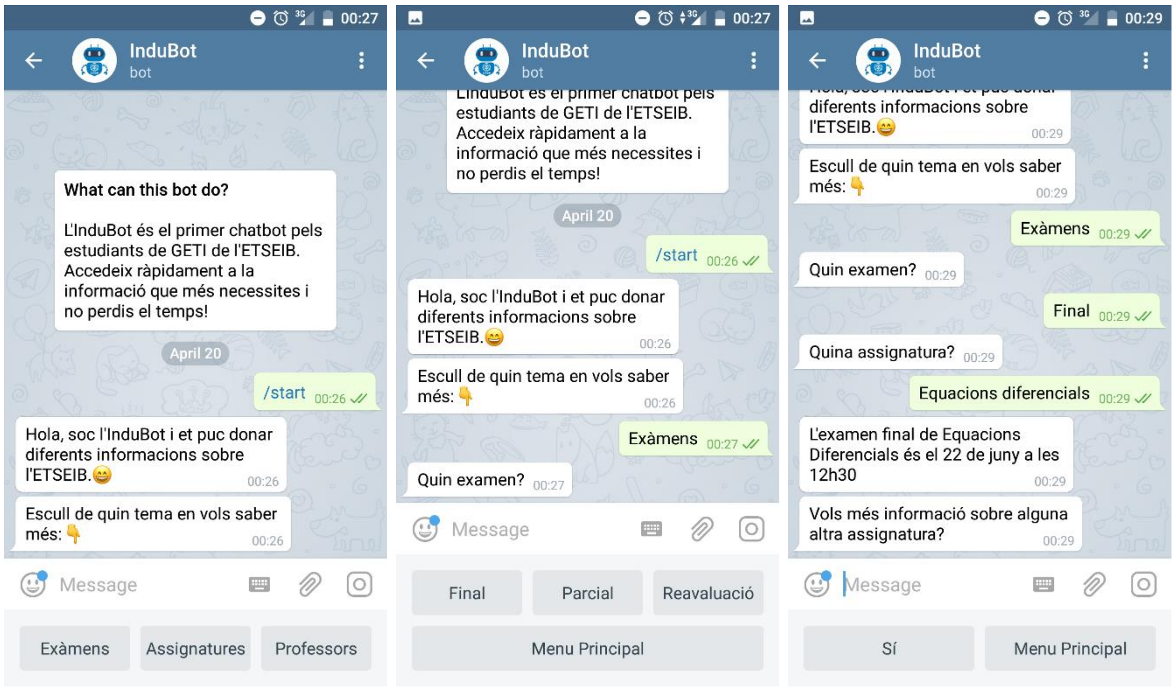
Finally, this first approach was dismissed and a new alternative was formulated.

Second approach: Flow-oriented and retrieval based heuristics

The strategy to elaborate this second design has been based on delivering accurate answers to the user in a fast and intuitive way. To do so the chatbot needs to minimize the intellectual load and increase speed. The design contemplates a layout with quick replies that are self-explanatory, meaning that the quick replies offer the scope and limitations of the chatbot domain.

The chatbot has a predefined logic tree that restricts the conversation to one of the branches. The user is guided from the beginning and has limited freedom to jump from one state to another, once it has chosen to go down a branch the chatbot will try to reach the end. The conversation is like a script, the questions and answers are already written, so the user has to choose usually from quick replies. In some circumstances, it is required to facilitate free text input, but it is restricted to the necessary keywords to retrieve information from the databases.

The following figures show the quick replies with buttons that are used to guide the conversation and inform the user about the possible paths there are.



What can this bot do?

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:26

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😄 00:26

Escull de quin tema en vols saber més:👇 00:26

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:26

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😄 00:26

Escull de quin tema en vols saber més:👇 00:26

Exàmens 00:27

Quin examen? 00:27

diferents informacions sobre l'ETSEIB.😄 00:29

Escull de quin tema en vols saber més:👇 00:29

Exàmens 00:29

Quin examen? 00:29

Final 00:29

Quina assignatura? 00:29

Equacions diferencials 00:29

L'examen final de Equacions Diferencials és el 22 de juny a les 12h30 00:29

Vols més informació sobre alguna altra assignatura? 00:29

Figure 6.18 InduBot Second Approach

6.2.5.1. The business logic

InduBot mechanism is simple, the user sends a message, Telegram Bot API sends that message to the web hosting service PythonAnywhere where the web application with the business rules is running, the message is processed by the web application and an answer is sent to the user through the Telegram Bot API. In this section, the focus will be on the business logic inside the web application that is in charge of processing the message and create the answer.

To understand InduBot's business logic it is necessary to define the following elements:

- State: is the scenario where the conversation currently is. The first scenario 0 corresponds to the welcome messages and the first layout. Afterward, the user chooses a path with his/her answers and decisions to reach a final state that fortunately gives a response to his/her initial doubt. All the possible situations are contemplated within a state, otherwise, it could crashes and the chatbot will restart the conversation from the beginning.
- Input: the message the user sends to the chatbot. This message contains more information a part from the plain text written by the user. But for this chatbot, the needed one is the text and the chat id.
- State variables: these variables are the ones required to elaborate the answer. They give essential information about the user and the conversation. Some of these variables are the current state, the chat id, the course, the teacher, etc.
- Script: the different answers InduBot sends to the user are gathered in a group of scripts. Each script corresponds to a state and has different predefined answers depending on the input and the state variables.

The core logic is built around four different modules:

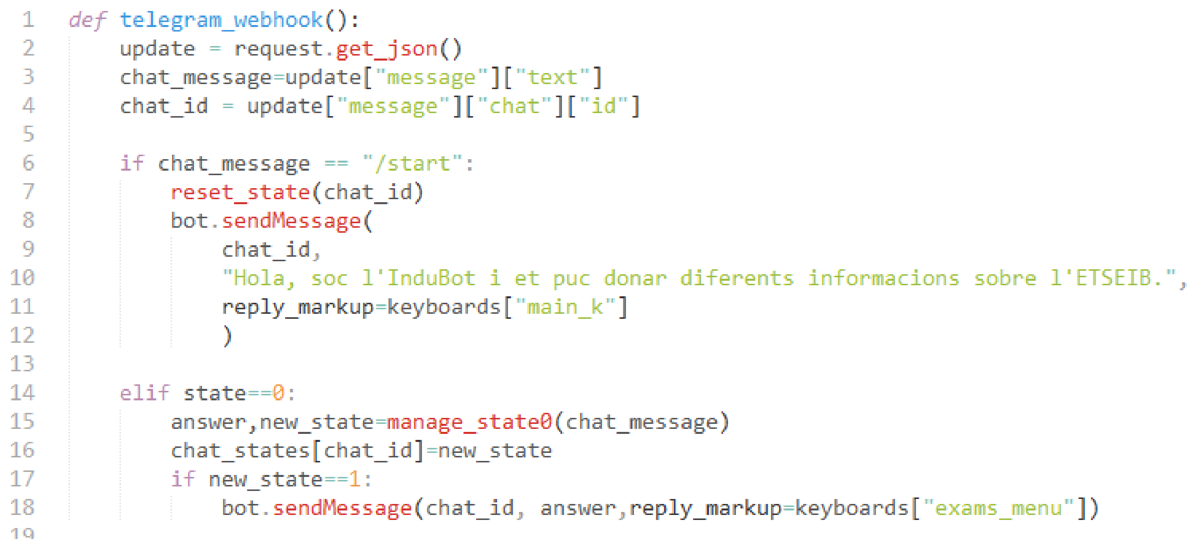
- Administrator module: receive the messages, manages the different modules to create the answer and sends the answer to the user.
- Answers module: contains the script with the answers for each state. It creates the answers using the script and the data from knowledge module.
- Knowledge module: contains the knowledge data and structures it in a way is easy to access.
- Layout module: contains the elements to create a better UX such as emoji or keyboards.

Besides this modules, there are other elements such as the state variables that are used by the various modules and has not been included in any of them.

Administrator module

The administrator module corresponds to the main function, called telegram_webhook that gets the message from the user and depending on the state calls a function from the answers module.

To illustrate following there is a piece of code:



```
1   def telegram_webhook():
2       update = request.get_json()
3       chat_message=update["message"]["text"]
4       chat_id = update["message"]["chat"]["id"]
5
6       if chat_message == "/start":
7           reset_state(chat_id)
8           bot.sendMessage(
9               chat_id,
10              "Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.",
11              reply_markup=keyboards["main_k"]
12              )
13
14      elif state==0:
15          answer,new_state=manage_state0(chat_message)
16          chat_states[chat_id]=new_state
17          if new_state==1:
18              bot.sendMessage(chat_id, answer,reply_markup=keyboards["exams_menu"])
19
```

Figure 6.19 InduBot Webhook function

The code begins with getting information from the user message. In line 3 it is retrieved the plaintext from the input and in the 4th line, it is stored the chat_id, a variable to identify each user and to do not cross conversations.

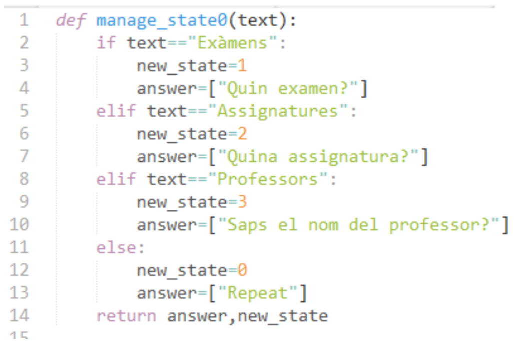
From line 6 below there are the business rules of the conversation flow. If chat message is the command /start, the conversation resets to the beginning. For the other cases, the function telegram_webhook checks the state and calls the corresponding function for each state. This other functions are called manage_stateX and are the responsible to create the answer and set the new state depending on the input. Once an answer is returned to the main function telegram_webhook, it is sent to the user the answer and the corresponding keyboard if required. The keyboard corresponds to the layout of the buttons and how are they set up.

Answers module

The answers module are the collection of functions in charge of producing the answer from the current state and the text input. As seen in the previous module the functions are called manage_stateX, one function for each possible state.

These functions contain a mini script for each state and the business rules to choose the answer, and in some cases call the knowledge module for the required information.

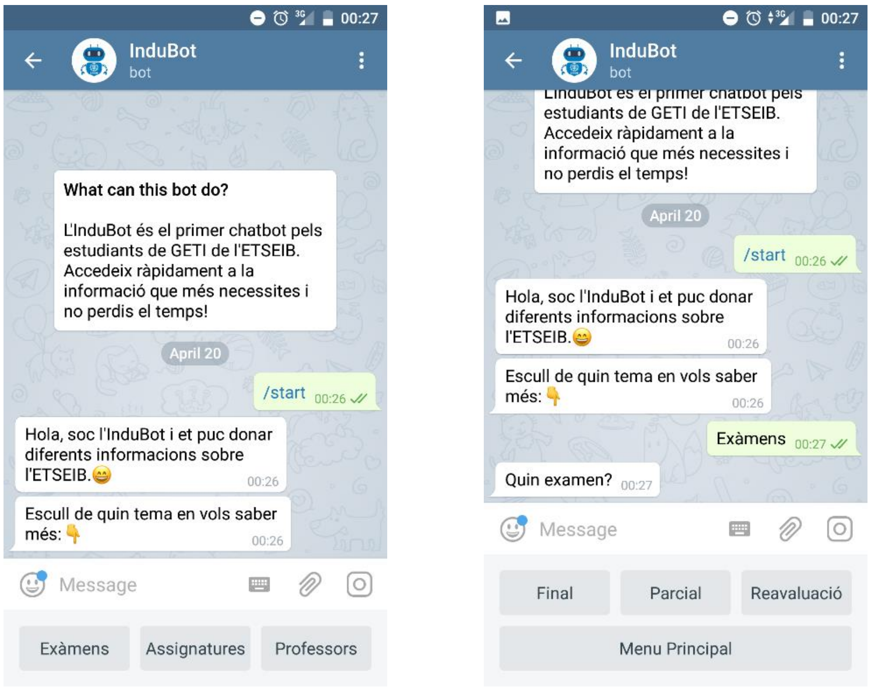
When a user starts a conversation with InduBot the first state is the state 0. Here the user is presented with the presentation layout where it can choose from 3 options: *Exàmens, Assignatures,* and *Professors*. The following function is in charge to create the answer for this state 0:



```
def manage_state0(text):
    if text=="Exàmens":
        new_state=1
        answer=["Quin examen?"]
    elif text=="Assignatures":
        new_state=2
        answer=["Quina assignatura?"]
    elif text=="Professors":
        new_state=3
        answer=["Saps el nom del professor?"]
    else:
        new_state=0
        answer=["Repeat"]
    return answer,new_state
```

Figure 6.20 InduBot Manage_State0 function

As it is possible to see, depending on the input, the new state and the answer varies. If the user chooses *Exàmens*, the conversation will change to state 1, and the answer will be *Quin examen?*. If the user input is not one of the 3 presented options, he/she will be required to repeat the step, and will not move further on remaining in the same state 0. The screens through which the user will pass are:



00:27
InduBot
bot

What can this bot do?

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:26

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😄 00:26

Escull de quin tema en vols saber més: 👇 00:26

Message

Exàmens | Assignatures | Professors

00:27
InduBot
bot

L'InduBot és el primer chatbot pels estudiants de GETI de l'ETSEIB. Accedeix ràpidament a la informació que més necessites i no perdis el temps!

April 20

/start 00:26

Hola, soc l'InduBot i et puc donar diferents informacions sobre l'ETSEIB.😄 00:26

Escull de quin tema en vols saber més: 👇 00:26

Exàmens 00:27

Quin examen? 00:27

Message

Final | Parcial | Reavaluació

Menu Principal

Figure 6.21 InduBot States 1: Exàmens

In the piece of code below, there is an example of function manage_stateX that interact with the knowledge module. In fact, in the 4th and 5th lines, the function is creating an instance from the class Teacher and calling the method get_email() to get the teacher's email. The 6th line is the answer and corresponds to the email of the teacher obtained from the instance t.

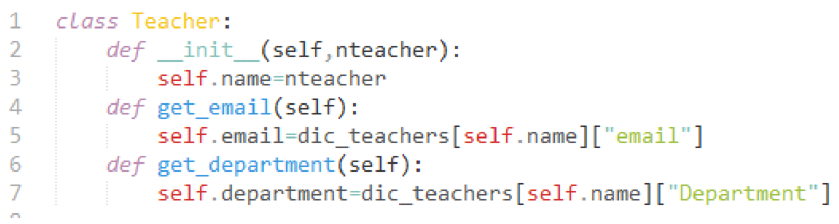
```
1  def manage_state311(text,state):
2      if text=="Email":
3          new_state=3111
4          t=Teacher(options["current_teacher"])
5          t.get_email()
6          answer=[u"El seu email és "+t.email]
```

Figure 6.22 InduBot manage_state311 function

Knowledge module

The knowledge module contains all the data about the university and necessary to construct some answers. The data is stored in dictionaries and is obtained through instances of classes. There are 2 dictionaries. One of the information about courses and the other with information about teachers. To encapsulate and organize the data the program has 3 classes, one for each of the topics: *Exàmens, Professors,* and *Assignatures.* These classes contain attributes and methods that allow to easily access the desired information by doing an instance.

Following there is the class Teacher with some attributes and methods:



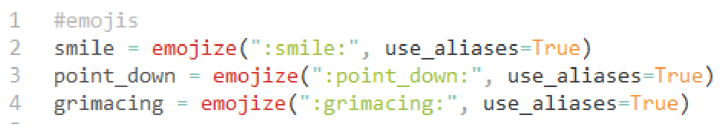
```python
class Teacher:
    def __init__(self,nteacher):
        self.name=nteacher
    def get_email(self):
        self.email=dic_teachers[self.name]["email"]
    def get_department(self):
        self.department=dic_teachers[self.name]["Department"]
```

Figure 6.23 InduBot Teacher class

For example, to get the email of a teacher there is a method, get_email(), that searches in the dictionary dic_teachers and creates an attribute with the found email.

Layout module

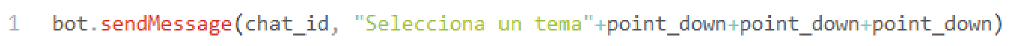
The layout module contains the emoji's and keyboards that are used to improve the UX. The emoji are stored in variables to be able to use them just by typing the variable name.



```python
#emojis
smile = emojize(":smile:", use_aliases=True)
point_down = emojize(":point_down:", use_aliases=True)
grimacing = emojize(":grimacing:", use_aliases=True)
```

Figure 6.24 InduBot Emojis variables

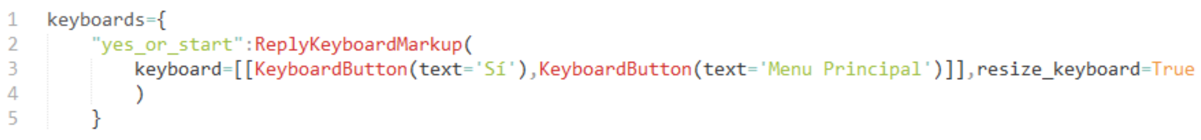
Here the emoji point_down has been used:



```python
bot.sendMessage(chat_id, "Selecciona un tema"+point_down+point_down+point_down)
```

Figure 6.25 InduBot sendMessage function

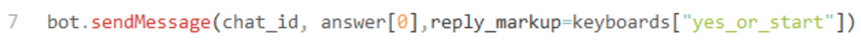
The keyboards are the layout of buttons used for quick replies and which are used to guide the user through the conversation flow.



```python
keyboards={
    "yes_or_start":ReplyKeyboardMarkup(
        keyboard=[[KeyboardButton(text='Sí'),KeyboardButton(text='Menu Principal')]],resize_keyboard=True
        )
    }
```

Figure 6.26 InduBot keyboards for buttons

The figure above shows the code of the keyboard predefined and stored in a dictionary. Then when needed it is retrieved and send it to the user.



```python
bot.sendMessage(chat_id, answer[0],reply_markup=keyboards["yes_or_start"])
```

Figure 6.27 InduBot sendMessage function with buttons

The result looks like the following figure.

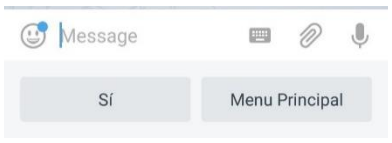


Figure 6.28 InduBot with buttons

6.3. Implementation

This chapter presents the complete implementation of the prototype end to end, from configuring the web hosting to the design of the buttons and emoji's that the user receives on the screen through Telegram.

6.3.1. Configuration

6.3.1.1. Create Telegram Bot

First, is necessary to understand what a Telegram Bot is; at the core, it is a special account that does not require an additional phone number to set up. The messages, commands, and requests sent by users are passed to the software running on the servers (PythonAnywhere). Telegram's intermediary server handles all encryption and communication with the Telegram API. The communication with the server (PythonAnywhere) is done via a simple HTTPS-interface called Telegram Bot API. [9]

Figure 6.29 The Botfather

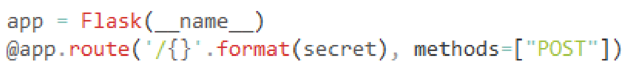
To create the account it is necessary to talk with The Botfather. He indicated all the steps for the creation and correct configuration of InduBot. The following commands were used:

- /newbot: to create the newbot, choose the name (InduBot) and the username (TheInduBot). Here it is obtained the authorization token, required to set the connection with the API and allow the server-bot communication.
- /setdescription: to define the description that is presented to users the first time they open a chat with the chatbot.
- /setuserpic: to select the profile picture for InduBot.

6.3.1.2. PythonAnywhere and Flask

PythonAnywhere is the development and hosting environment where the chatbot (web application) is running. The servers of PythonAnywhere are already set up and ready to use. A remarkable feature is that they offer a free plan, the one used for this project. [12]

To implement the chatbot it has been created a free account on PythonAnywhere and set up a web application using the Flask framework. Basically what the Flask application will do is handling the request from the webhook (explained in the Telegram Bot API section). The following code creates the Flask app and configures it to run a function when it gets a POST request on the secret URL. This URL has to be difficult to guess to secure the communication and avoid inappropriate uses. [12]



```
app = Flask(__name__)
@app.route('/{}'.format(secret), methods=["POST"])
```

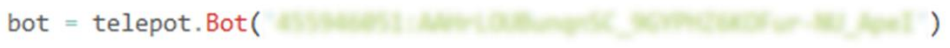
Figure 6.30 Flask App

6.3.1.3. Telegram Bot API

The chatbots in Telegram are controlled using HTTPS requests to the Telegram Bot API. To do so it is possible to use the Telegram Bot API oneself or profit from a framework such as Telepot that is written in Python. In this project it was chosen the second option, taking advantage of the tutorials, documentation, and examples Telepot offers. [10]

To set the API it is necessary to use the authorization token obtained during the chatbot creation with The Botfather. Next, there is the decision to select how to configure the HTTPs requests. For this project, and following the advice from Telepot it has been configured a webhook. A webhook (also called a web callback or HTTP push API) is a way for an app to provide other applications with real-time information. A webhook delivers data to other applications as it happens, meaning the data is obtained immediately. Unlike typical APIs where it is necessary to poll for data very frequently in order to get it real-time. This makes webhooks much more efficient for both provider and consumer. The only drawback to webhooks is the difficulty of initially setting them up. However, since speed in answering and getting real-time data is one of the most important requirements, the webhook has been set. To illustrate how it was made following there are the written lines of code. [12]

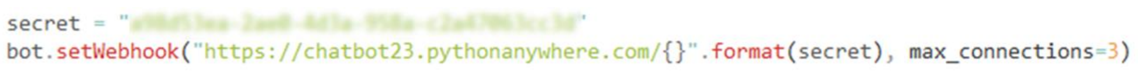
The first step is to set the chatbot with the corresponding authorisation token, obtained during the creation of the chatbot. For security reasons, the token has been blurred in this caption.



```
bot = telepot.Bot('[redacted]')
```

Figure 6.31 Create the Chatbot

Secondly, it has to be settle the webhook. This will be the connection between the server and Telegram. The chatbot is using a publicly-accessible website so it is important to secure the connection by using an unguessable URL to avoid inappropriate uses.



```
secret = "[redacted]"
bot.setWebhook("https://chatbot23.pythonanywhere.com/{}".format(secret), max_connections=3)
```

Figure 6.32 Create the webhook

With the webhook, Telepot notifies Telegram that when InduBot gets a message it has to send it to that unguessable URL. Then the web application running on PythonAnywhere will handle the message and answer to the user. It is also worth noting that the configured webhook uses secure HTTPS (PythonAnywhere has it for default) rather than HTTP, and Telegram (quite sensibly) will only send webhooks over HTTPS.

6.3.2. Business Logic

As explained in the technical design section the business logic is distributed in modules, but basically it consists on a webhook that; receive the user messages, calls other functions that create the answer and sends the message with the answer back to the user. For the implementation, it has been used the integrated development environment (IDE) offered by PythonAnywhere that allows coding from any device just from the browser. This gives incredible freedom when coding because it has been possible to program from different devices, and also permits to has all the configurations and packages on the cloud, without depending on a specific device.

6.3.3. Knowledge Base

For the knowledge base of courses and teachers from ETSEIB, it has been used information from PDFs and the website. In many cases, the PDFs contains the information in such a way it is possible to extract it as a table, adjust it with excel and store it as a CSV file. For some other data such as teachers' names, it has been needed some web scrapping programming that has been done by the local command prompt with Python since the PythonAnywhere IDE does not support web scrapping. By using this method it has been obtained the information necessary to create the knowledge module with the information contained in the ontology about exams, courses, and teachers.

For the implementation, it has not been used a database. The chatbot reads two CSV files, one for teachers and another for courses, where all the data is stored. This files are in the web hosting server PythonAnywhere and are accessed from the knowledge module functions when needed.

6.4. Testing and results

6.4.1. Testing

The testing has been conducted iteratively while developing the chatbot to find software bugs and verify it meets the requirements. In most of the cases, the tests have been done by using the chatbot with the Telegram application in a mobile device and checking how it responds to different inputs and which was the user experience. All the detected necessary changes have been accomplished to build the final operative prototype ready for use.

During the months of the project, more than 15.000 messages have been exchanged with InduBot to test its capabilities and ensure its readiness for the launch of the final version of the prototype.

6.4.2. Results

Before launching InduBot it has been performed a test with real people to check the advantages the chatbot offers to ETSEIB students. As user experience is difficult to check because of the subjective connotation of each individual the test has focused on the speed approach marked as one of InduBot requirements. The test consisted of comparing how it takes to search using ETSEIB website (http://www.etseib.upc.edu/) versus the time it takes with InduBot.

The test has been divided into two searches:

- Find the date of the final exam of *Mecànica de Fluids*
- Find the timetables of the course *Equacions diferencials*

And to guarantee the reliability of the results it has been done with 5 students who had never before interacting with InduBot. To conduct the test it has been used the same device, a smartphone with the application Telegram already installed and good access to the internet.

When searching through the website the time considered: from starting to type ETSEIB in google until the information is found. In the case of using InduBot the time considered: from starting to search the chatbot in Telegram until the information is located.

The results are included in the following table:

	Website		**InduBot**	
	Test 1	Test 2	Test 1	Test 2
Student 1	49''	45''	23''	25''
Student 2	43''	38''	21''	22''
Student 3	52''	51''	18''	24''
Student 4	51''	43''	24''	27''
Student 5	42''	48''	17''	21''

Figure 6.33 Results ETSEIB website vs InduBot

As observed in the results the chatbot can speed up the searching process to half of the time in almost all the cases. If applied to other information searches it may vary, but the results are consistent about the advantage InduBot supposes for students when looking for exams and courses information.

7. Project costs

This chapter considers the associated costs associated with the design, plan and developing of the chatbot.

The resources used for the development of the project are:

- Hardware: physical computer resources
- Software: programs and applications
- Human resources: people that worked on the project

Hardware

The whole project has been developed with only one computer and the tests have been done on a smartphone.

Software

For the developing of the prototype, it has been used PythonAnywhere as IDE and server, Telegram as messaging platform and Microsoft Office Professional 2013 for the intermediate deliverables and the final document.

Human Resources

The design and developing of the chatbot have been done by the author of the project as well as the final document. In some occasions, it has also been required advice from the director of the project.

The power supplies, costs of facilities and other indirect costs have not been considered because of the difficulty to calculate them and the little proportion they supposed compared to the labour expenses.

Product expenses			
Item	Units	€/unit	Total
Laptop +external monitor	1	700	700
Smartphone	1	300	300
Microsoft Office Professional 2013	1	70	70
PythonAnywhere server	1	Free	0
Telegram	1	Free	0
Total			**1.070**

Figure 7.1 Product Expenses

Labour expenses

Task name	Time	€/hour	Total
Business Analyst	80	40	3.200
Developer	180	25	4.500
Back office	100	15	1.500
Director	5	60	300
Total			**9.500**

Figure 7.1 Labour Expenses

The total cost for the project is estimated in 10.570 € despite the extensive use of open source applications for the developing of the chatbot.

8. Environmental impact

Even if a software as the one developed in this project has a very environmental impact it is important to track the power consumption and the effects on the Greenhouse gasses emissions.

To measure the environmental impact of this project first it should be identified the three phases of the product life cycle:

- Developing the product
- Operating the product
- Decommissioning the product

The first stage corresponds to the design, development and implementation of the chatbot. During this phase it has been used different resources:

- Computer + monitor: power consumption when coding and writing the reports
- Smartphone: power consumption when testing
- Web hosting: power consumption of the hardware running the server that hosts the web application

The second phase considers the consumption of the web hosting another time.

And the last stage corresponds to the decommissioning that in this case is just turn off the software without any residual elements.

So to quantify the impact it just necessary to calculate the power consumption of the computer, smartphone and web hosting servers.

The laptop + monitor have been used for the research, the coding and writing the reports. In total they have been used for 360 hours. The power is 12W for the laptop and 30W for the monitor. The resulting consumption has been 15.120Wh. In this case it is considered that the total amount of power consumption corresponds to the project programs running on the computer.

The smartphone has been used during 20 hours for the test and its power is 3W. The resulting consumption is 60Wh. As with the computer, the total amount of energy is considered to be used when testing.

Last one there is the server. In this case the configuration of the application has been done with a webhook, so the time the software is running is minimum. Another aspect is the life of the application, PythonAnywhere closes the websites after 3 months of inactivity. Also, it should be considered that the account used corresponds to the free one which has a limited bandwidth and CPU usage per day. For these reasons and compared to the energy used by the computer during the developing of the prototype it has not been calculated the power usage for the web hosting services.

The total power consume for the project is 15.180Wh which is equivalent to 11,3 kg of CO_2. [8]

Conclusions

InduBot has been created satisfying most of the proposed objectives as well as confirming the deadlines. After the execution of the whole project, there are many aspects to point out which have been of important relevance.

The main objective of designing and implementing a chatbot to facilitate information searches to ETSEIB students has been fulfilled with a fully operative and functional prototype. It performs the defined tasks of information retrieval and consultation faster than it takes to look at ETSEIB's website and with a friendly interface such as it is Telegram. Besides, InduBot is a totally purpose oriented chatbot that with its intuitive design guides the user to get the desired information within few interactions.

The secondary goals have all been achieved except for one: the goal of having a seamless experience, similar to the one of talking with a human. This requisite that was marked at the beginning of the project has not been accomplished since the final approach of InduBot contemplates a restricted use of free text input and an extensive use of quick replies and buttons limiting the options in many cases. This restrictions in the user input augments the accuracy and speed of the interactions but also diminishes the sensation of having a real conversation, similar to the one of speaking with a human being.

InduBot could be considered an attempt to create a new channel of communication between students and the ETSEIB and it is also a base if someone in the future wants to continue with the chatbot development. The way in which it has been designed allows a high scalability in case lots of student would like to use the chatbot. Furthermore, with changes just in the content of the scripts it is possible to change the language of interaction from Catalan to any other. And in case another university likes InduBot and wants one similar chatbot for their students it is very easy to adapt, it just needs with few changes and nurturing the program with other data about courses, exams and teachers.

Improvements

The resulting chatbot performs well in most of the requirements but there are many improvements out of the scope of the project that may contribute to create an outstanding user experience and drive more value to the students.

The first improvement is the NLP capabilities. Including Natural Language Processing to the chatbot will allow the user introducing free text and asking questions whenever they want.

Another upgrade will be augmenting the range of the offered information to more data such as including FAQs about academic procedures or getting access to internship offers.

Other improvements maybe allowing a human supervised logic that allows to change to a live agent in case the student does not find the desired information. Also important is the channels of interaction with InduBot; right now it has been configured to work with Telegram, but it is also possible to set it up to be working on other messaging platforms.

Challenges and limitations

During the different phases of the project there are many challenges and limitations that represented a difficulty and restricted the result of the same. The first one has been the incipient rise of some of the used technologies and tools for chatbots, which obligates to get information from not consolidated sources of information. In many cases the only source of information have been blogs and forums. This also affected the research and elaboration of the state of art of the chatbots in some points.

Another issue has been the information quality when creating the files that contain the data to nurture the chatbot with knowledge. In many cases the data was contained in PDFs or it was necessary to create a script and make some web scrapping. To obtain consistent and homogenous data it takes some time to organize it and check its quality.

Another limitation has been the election of the web hosting service PythonAnywhere; while at the beginning it seems to be all advantages, because it is free, very easy to use, prepared to work with Python and create chatbots with Telegram, etc. At some point it started to present limitations such as the difficulties to configure a database, the non-allowance of doing web scrapping from the web application (this would have been useful to get real time data from the ETSEIB website to nurture InduBot knowledge base), and the low limit to the maximum number of users the chatbot can handle every second.

Bibliography

Websites

[1] https://www.wikipedia.org/

[2] https://chatbotslife.com

[3] https://medium.com

[4] https://chatbotsmagazine.com

[5] https://www.forbes.com

[6] https://www.gartner.com

[7] https://www.versionone.com

[8] https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator

Documentation

[9] https://core.telegram.org/bots

[10] https://core.telegram.org/bots/api

[11] https://www.python.org/

Tutorials

[12] https://blog.pythonanywhere.com/

Forums

[13] http://stackoverflow.com

[14] http://www.python-forum.org/

Data used by InduBot

[15] https://etseib.upc.edu/ca

ETSEIB

[16] https://futur.upc.edu/

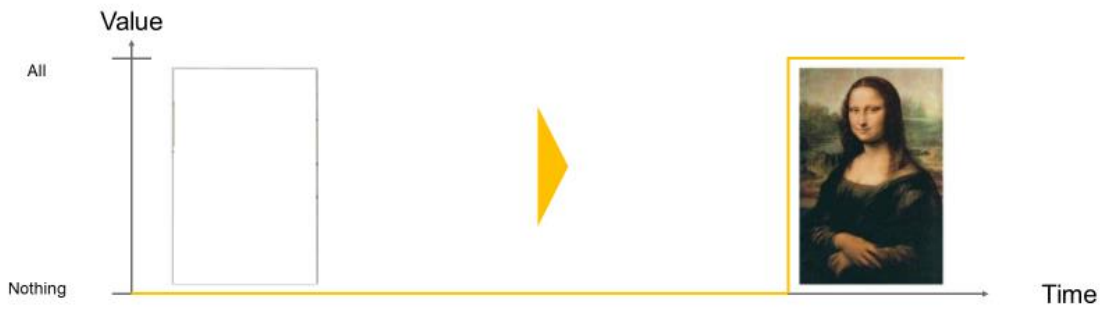
Appendix

Methodology comparative example

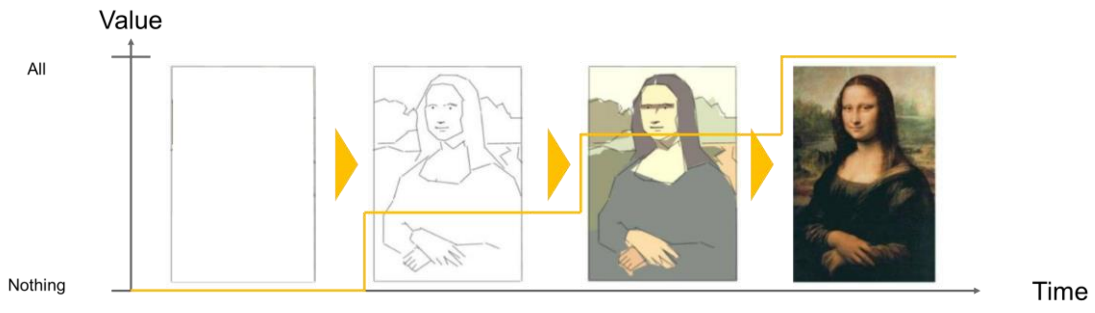
To compare the two models approach imagine Leonardo da Vinci when it was ordered to create the Mona Lisa. Medici probably specifies him the idea of what they want, the requirements. For example, he might have been told to paint a woman in a pastoral setting with a landscape in the background. To understand the reasoning of this comparison it should be assumed Mona Lisa could be delivered in parts.

Figure 4Mona Lisa Requirements

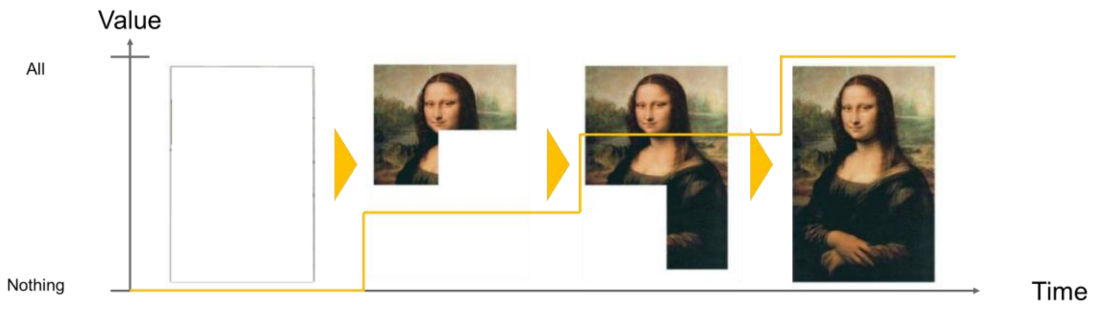
If Leonardo would have opted for the Waterfall model he would have present to the Medici the portrait when finished. If something would have been wrong and not as desired Leonardo would not have time for changes and would have had to start from the beginning.



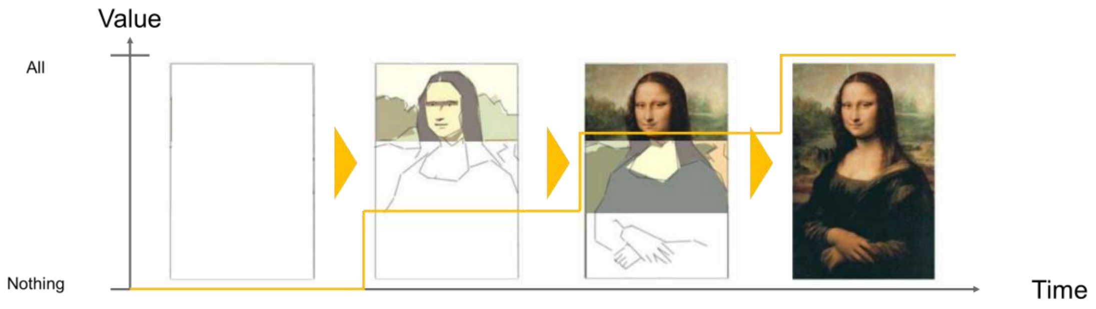
Instead, if he would have opted for just an iterative model, showing the Medici the scratch of the picture and the evolution of it, Leonardo would have had the opportunity to readdress the painting if necessary. Even if after each iteration there would have not been any parts to deliver.



If Leonardo would have chosen the incremental approach he would have divided the painting into parts and painted them one by one. After painting each he would have been able to deliver it to Medici and avoid making them wait for the final result. The problem would have been if Medici would have not liked one part, that Leonardo would have had to repeat it. However, it is better than with the Waterfall approach, one part instead of all the painting.



The last option for Leonardo would have been to adopt the Agile model using the incremental and iterative approaches together, the result would have been something like the following picture.



Leonardo would have delivered first a scratch of what he had thought and in parallel small finished parts for Medici before the final portrait.