

1 2 9 0



UNIVERSIDADE D
COIMBRA

Afonso Cheung Viana de Lemos Nunes

**A CHATBOT FOR THE NEXT GENERATION
LEARNING SYSTEMS**

**Master's Dissertation in Electrical and Computer Engineering, supervised by Prof.
Dr. Jorge Miguel Sá Silva and Prof. Dr. André Miguel de Almeida Marrão
Rodrigues, presented to the Faculty of Science and Technology of the University of
Coimbra.**

July 2023



UNIVERSIDADE D
COIMBRA

A Chatbot for the next generation learning systems

Afonso Cheung Viana de Lemos Nunes

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientadores:

Prof. Doutor Jorge Miguel Sá Silva

Prof. Doutor André Miguel de Almeida Marrão Rodrigues

Júri

Presidente: Prof. Doutor Jorge Manuel Miranda Dias

Vogais: Prof. Doutor Paulo José Monteiro Peixoto

Prof. Doutor Jorge Miguel Sá Silva

Julho de 2023

”Remember, as you step into the world of software development, there are only two types of programming languages: those people always complain about, and those nobody uses.

Enjoy the journey!”

GPT-4

Agradecimentos

Gostaria de agradecer em primeiro lugar à minha família pelo apoio incondicional.

Aos professores Jorge Sá Silva e André Rodrigues pela orientação e pelo apoio constante nos vários projectos que integrei neste grupo de investigação.

Aos amigos.

Ao Marcelo Fernandes, que tem a capacidade de resolver todos os problemas e esclarecer todas as dúvidas, por toda a paciência e camaradagem ao longo dos vários projectos em que tive o prazer de trabalhar.

Ao Rui Silva pela irmandade sempre presente no curso e fora dele.

Ao Moisés Dias por todos os projectos, cadeiras, trabalhos, noitadas de estudo, F3Es, BotOlympics, NLS e tanto mais.

A todos os que contribuíram para o meu percurso de formação académica e pessoal.

Abstract

Integrating the Instituto de Engenharia de Sistemas e Computadores de Coimbra (INESCC) research group from the Departamento de Engenharia Electrotécnica e de Computadores (DEEC), this research project explores the symbiotic relationship between humans and technology within the domain of Human-in-the-Loop Cyber-Physical Systems (HITLCPS). By developing and integrating a ChatBot into the group's primary platforms, IoT Student Advisor and BEst Lifestyle Analyzer (ISABELA) and Best Assisting Tutor and INteractive Advisor (BATINA), we focused on the actuation step in HITLCPS, which emphasizes the interaction between the system and the human, closing the loop.

Working in a long-standing partnership with the University of Coimbra (UC) and integrating ISABELA and BATINA's features, this project takes a step into innovative research with the ON-BOARD initiative, that was recently funded by the Plano de Recuperação e Resiliência (PRR), which will incorporate our developed ChatBot as a digital tutor into the UC official platforms.

Our research explores the potential of Internet of Things (IoT), Artificial Intelligence (AI) and Natural Language Processing (NLP) for enhancing the INESCC-DEEC Human-in-the-Loop (HITL) platforms and establishing the groundwork for future human interactions with Cyber-Physical-Systems (CPS). This dissertation contributes to this expanding field by detailing implementations and proposing a solid architecture for future development.

Keywords

ChatBots, Human-in-the-Loop Cyber-Physical Systems, Internet of Things, Artificial Intelligence, Natural Language Processing

Resumo

Este projecto foi desenvolvido integrando o grupo do Instituto de Engenharia de Sistemas e Computadores de Coimbra (INESCC) do Departamento de Engenharia Electrotécnica e de Computadores (DEEC) e explora a simbiose entre o ser humano e a tecnologia através o conceito de Sistemas Ciber-Físicos Human-in-the-Loop. Através do desenvolvimento de um ChatBot para as plataformas principais deste grupo, IoT Student Advisor and BEst Lifestyle Analyzer (ISABELA) e Best Assisting Tutor and INteractive Advisor (BATINA), o nosso trabalho centrou-se na etapa de actuação dos sistemas HITL, que gere a interacção entre o sistema e o ser humano, fechando o loop.

Através de uma parceria com a Universidade de Coimbra (UC) e integrando as funcionalidades das plataformas ISABELA e BATINA, este projecto inova através da iniciativa ONBOARD que, recentemente financiada pelo Plano de Recuperação e Resiliência (PRR), irá incorporar o ChatBot desenvolvido como um tutor digital nas plataformas oficiais da UC.

Este trabalho explora o potencial dos conceitos Internet das Coisas (IoT), Inteligência Artificial (IA) e Processamento de Linguagem Natural para fortalecer as plataformas HITL do grupo INESCC-DEEC e estabelece um trabalho de base para futuras interacções humanas com Sistemas Ciber-Físicos. Esta dissertação contribui para este tema em crescimento descrevendo em detalhe as funcionalidades implementadas e propondo uma arquitectura sólida para trabalhos futuros.

Palavras-Chave

ChatBots, Human-in-the-Loop Cyber-Physical Systems, Internet das Coisas, Inteligência Artificial, Processamento de Linguagem Natural

List of Figures

2.1	Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030 (in billions)	6
2.2	Mobile data traffic in Western Europe 2011-2027 (in exabytes per month) .	7
2.3	Control Loop for a HITLCPS	11
2.4	Xamarin Architecture	12
2.5	FIWARE Modules	13
2.6	Dialogflow Architecture	14
3.1	ISABELA Architecture	18
3.2	ISABELA's Feedback Pages	19
3.3	ISABELA's ChatBot	20
3.4	ISABELA's Smartwatch Activity	21
3.5	Professor's Course and Questionnaire Lists	22
3.6	Professor's Support Material and Doubts Lists	22
3.7	Creating a questionnaire from pool	23
3.8	Course Report	23
3.9	Student Doubt Answered	24
3.10	BATINA Mobile Courses Pages	25
3.11	BATINA Mobile Questionnaire Pages	25
3.12	BATINA Mobile Questionnaire Pages (cont)	26

List of Figures

3.13	BATINA Mobile Reports	27
3.14	UC Teacher	28
3.15	UC Student	28
4.1	ChatBot Architecture	42
4.2	UI Architecture	43
4.3	Pollux Architecture	44
4.4	Dialogflow's Architecture	45
4.5	ChatBot Alarm Interaction	46
4.6	Canteen Menu Interaction	47
4.7	Historic Fact Interaction	48
4.8	Professor's Info Interaction	49
4.9	Exam Reminder Interaction	50
4.10	Exam Feedback Interaction	50
4.11	Assignment Submission Reminder Interaction	51
4.12	Poor Questionnaire Result Interaction	52
4.13	GPT-3 Toggled On and Off Interactions	53
4.14	Dialogflow Intent List	55

List of Tables

2.1	ChatBot overview over time	10
4.1	Functional Requirements	36
4.2	Requirements' Parameters	36
4.3	Intent recognition and subsequent behavior in Pollux	57
4.4	Background services' messages and matching Dialogflow's Intents	58
4.5	Alarm Test	59
4.6	Canteen Menu with Meals Time Test	59
4.7	Canteen Menu Test	60
4.8	Canteen Menu Lowest Kcal Meal Test	60
4.9	Professor Contact Info Test	61
4.10	History Facts Test	61
4.11	Calculations Test	62
4.12	Random Characters Test	62
4.13	Random Question Test	62
4.14	Random Question Test 2	62

Acronyms

AI Artificial Intelligence.

AIML Artificial Intelligence Markup Language.

ALEKS Assessment and LEarning in Knowledge Spaces.

ALICE Artificial Linguistic Internet Computer Entity.

API Application Programming Interface.

BATINA Best Assisting Tutor and INteractive Advisor.

CNN Convolutional Neural Networks.

CPS Cyber-Physical-Systems.

DEEC Departamento de Engenharia Electrotécnica e de Computadores.

DL Deep Learning.

FCTUC Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

FMUC Faculdade de Medicina da Universidade de Coimbra.

FR Functional Requirement.

GNN Generative Neural Networks.

GPA Grade Point Average.

GPS Global Positioning System.

Acronyms

GPT-3 Generative Pre-Trained Transformer 3.

HITL Human-in-the-Loop.

HITLCPS Human-in-the-Loop Cyber-Physical Systems.

HTTP HyperText Transfer Protocol.

IBM International Business Machines Corporation.

IDE Integrated Development Environment.

INESCC Instituto de Engenharia de Sistemas e Computadores de Coimbra.

IoT Internet of Things.

ISABELA IoT Student Advisor and BEst Lifestyle Analyzer.

JSON JavaScript Object Notation.

KST Knowledge Space Theory.

MAUI Multi-Platform App User Interface.

MIT Massachusetts Institute of Technology.

ML Machine Learning.

NLP Natural Language Processing.

OCB Orion Context Broker.

OS Operating System.

POMS Profile of Mood States.

PRR Plano de Recuperação e Resiliência.

REST Representational State Transfer.

RNN Recurrent Neural Networks.

SASUC Serviços de Ação Social da Universidade de Coimbra.

UC University of Coimbra.

UI User Interface.

URL Uniform Resource Locator.

UWP Universal Windows Platform.

XAML Extensible Application Markup Language.

Contents

- 1 Introduction 1**
 - 1.1 Context and Motivation 1
 - 1.2 Objectives 2
 - 1.3 Methodology 2
 - 1.4 Document Structure 3

- 2 Background Knowledge 5**
 - 2.1 Base Concepts 5
 - 2.1.1 Internet of Things (IoT) 5
 - 2.1.2 Smartphones 6
 - 2.1.3 Artificial Intelligence (AI) 7
 - 2.1.4 ChatBots 8
 - 2.1.5 Human-in-the-Loop Cyber-Physical Systems (HITLCPS) 10
 - 2.2 Framework Technologies 11
 - 2.2.1 Xamarin 11
 - 2.2.1.A Xamarin.Essentials 12
 - 2.2.1.B Xamarin.Forms 13
 - 2.2.2 FIWARE 13
 - 2.2.3 Flask 14

2.2.4	Dialogflow	14
2.2.5	GPT-3	15
2.2.6	Lottie	15
3	Group Platforms	17
3.1	IoT Student Advisor and BEst Lifestyle Analyzer (ISABELA)	17
3.2	Best Assisting Tutor and INteractive Advisor (BATINA)	21
3.2.1	BATINA Dashboard	21
3.2.2	BATINA Mobile	23
3.3	UC Teacher / UC Student	27
3.4	ON-BOARD Project	29
3.5	Similar Academic Platforms	30
3.5.1	StudentLife	30
3.5.2	BigBlueButton	30
3.5.3	ALEKS	31
3.5.4	Socrative	31
3.5.5	Comparative Analysis	32
3.6	GreenBear	33
4	ChatBot Development	35
4.1	Context	35
4.2	Requirements	36
4.2.1	Functional	37
4.2.2	Non-Functional	41
4.3	Architecture	41
4.3.1	Overview	41
4.3.2	UI	42

4.3.3	Pollux	43
4.3.4	Dialogflow	44
4.4	Implemented Features	45
4.4.1	Define an alarm	46
4.4.2	Canteen Menu	47
4.4.3	Historic Facts	48
4.4.4	Professor Contact Info	49
4.4.5	Exam Reminder	49
4.4.6	Exam Feedback	49
4.4.7	Assignment Submission Reminder	51
4.4.8	Poor Questionnaire Result Feedback	51
4.4.9	GPT-3 Interaction	52
4.5	Implementation	53
4.5.1	Dialogflow configuration	53
4.5.2	GPT-3 configuration	55
4.5.3	Pollux configuration	56
4.5.4	Smartphone application	57
4.6	Tests	59
5	Conclusion and Future Work	63
5.1	Conclusion	63
5.2	Future Work	64
A	Introduction Tutorial to ChatBot Technologies	70
B	Green Bear – A LoRaWAN-based Human-in-the-Loop Case-Study for Sustainable Cities	94

Chapter 1

Introduction

1.1 Context and Motivation

This project is the result of a comprehensive research conducted alongside the Instituto de Engenharia de Sistemas e Computadores de Coimbra (INESCC) research group. This group is dedicated to the development of Human-in-the-Loop Cyber-Physical Systems (HITLCPS), a field that emphasizes the symbiotic relationship between humans and technology by creating human-centric systems. These systems infer present and future states based on data acquisition and provide user feedback therefore integrating the human as part of the system. The development of HITLCPS is important in order to adapt technology to human behavior as well as human behavior to technology. It is essential to comprehend this relationship from both a physical and an emotional standpoint, as we move towards an increasingly interconnected future.

Our group's primary development and testing platforms, IoT Student Advisor and BEst Lifestyle Analyzer (ISABELA) and Best Assisting Tutor and INteractive Advisor (BATINA) (discussed in greater detail in Chapter 3), allowed us to investigate this relationship by focusing on the actuation of HITLCPS: the interaction between the system and the human, which closes the loop.

After an initial phase devoted to familiarising ourselves with the pertinent frameworks and technologies (detailed in section 2), we developed and implemented a ChatBot for the group's platforms on both ISABELA and BATINA. We also created an in-depth introduction to Chat-

1. Introduction

Bot technologies to provide the INESCC-DEEC research team a solid foundation for comprehending and implementing the developed architecture on future platforms.

Furthermore, in a long-standing partnership with our group, the University of Coimbra (UC) has demonstrated its dedication to innovative research by integrating ISABELA and BATINA features into its official platforms. Our most recent collaboration with the UC is the ON-BOARD project (described in section 3.4), wherein our developed ChatBot will be used as a digital tutor.

1.2 Objectives

As mentioned previously, our project's primary objective was to build a foundation for ChatBot operations that can manage system-user interactions, to close the loop on the existing HITLCPS platforms, therefore providing feedback and actuation from the system to the user as well as data acquisition from the user to the system. To achieve this, the first objective was to design and implement the ChatBot on the two main developing platforms mentioned previously: ISABELA and BATINA.

As a secondary objective, we aimed at implementing the ChatBot focused on academic life, furthering our partnership with the UC, behaving as a digital tutor for students, providing feedback on their academic results and lifestyle and help them improve their performance, preventing dropout.

1.3 Methodology

This project was developed using the Scrum methodology, which is an agile framework widely adopted in software development [1]. It structures work into iterations known as "sprints", which are typically a few weeks in duration. At the start of each sprint, tasks are defined and planned out. Progress is monitored through weekly meetings where the team discusses their achievements, problems and the next steps. The conclusion of each sprint involves an evaluation of the work accomplished and, if needed, an adjustment to the planning or strategy for subsequent sprints. To keep track of iterations, GitLab was used to maintain a record of features and version control, as well as issues and bugs on the

smartphone applications [2].

As means of communication, the team used Slack and the weekly meetings were hosted in Skype [3, 4].

For software development, JetBrains' Rider and IntelliJ were the chosen Integrated Development Environment (IDE)s due to their great programming tools for different languages and projects, but also GitLab integration [5–7].

1.4 Document Structure

The chapters that provide structure to this document are ordered as follows:

- **Chapter 1** presents the context, motivations and methodology for this study;
- **Chapter 2** describes the base concepts and technologies used;
- **Chapter 3** characterises the platforms on which this project stands on, along with the group's initiatives developed in partnership with the UC, the Faculdade de Medicina da Universidade de Coimbra (FMUC) and the Coimbra City Hall;
- **Chapter 4** explains the proposed system and its integrations in ISABELA and BAT-INA projects;
- **Chapter 5** discusses the results obtained, drawing conclusions from the implemented system and presents future work suggestions.

Chapter 2

Background Knowledge

2.1 Base Concepts

This section summarises the background knowledge acquired from the research conducted, which substantiates the work developed in the next chapters. Although the foundational concepts supporting this project are extensive and intricate, a concise explanation of each core concept applied in this study is provided.

2.1.1 Internet of Things (IoT)

Internet of Things is a network of diverse physical devices equipped with sensors, software and communication protocols whose purpose is to collect, exchange and distribute data over the Internet. Kevin Ashton coined the term "Internet of Things" during a Procter & Gamble presentation in 1999 [8]. These devices may capture the physical world and could involve industrial tools, academic utilities, household items and commonplace consumer objects. Consequently, the physical world can be incorporated into digital platforms and systems in order to better comprehend or monitor certain activities. Some recent examples may include a refrigerator that monitors its fill level or a lightbulb that powers on automatically at dusk. IoT devices may also be able to communicate with each other autonomously, allowing for the automation of these systems.

In 2022, there were an estimated 13 billion IoT devices connected to the Internet [9]. By 2030, it is expected that the number of IoT devices will approach 30 billion [10] as it is shown in

2. Background Knowledge

Figure 2.1. However, having access to data provided by these devices is not enough, we must also know how to utilize it.

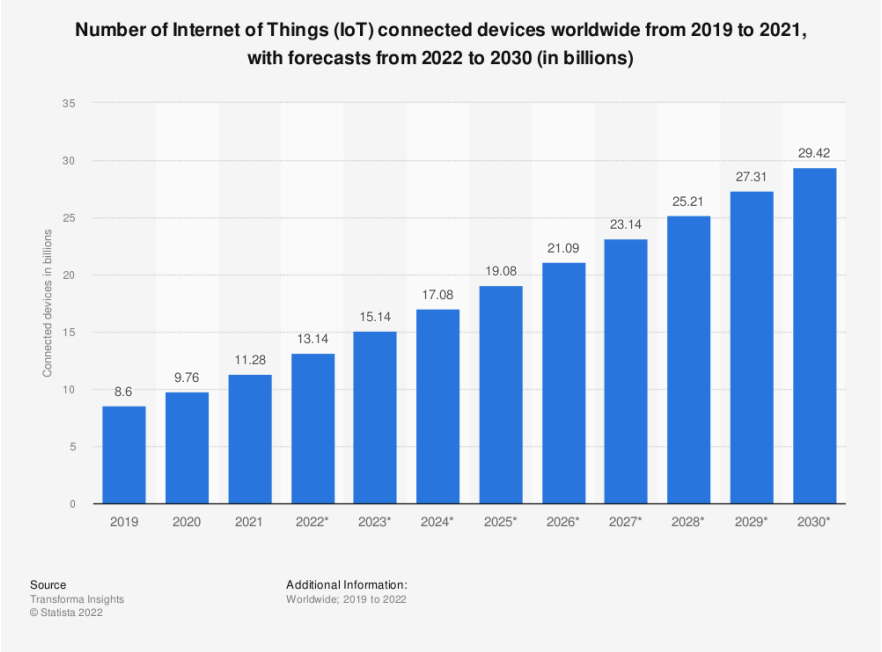


Figure 2.1: Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030 (in billions) [10]

Designing highly sophisticated, reliable, and user-friendly systems will become increasingly more challenging as the number of these devices in our daily lives and the complexity of the systems increase in the coming years.

2.1.2 Smartphones

International Business Machines Corporation (IBM) introduced the IBM Simon, the world’s first smartphone, in 1994. This unveiling represents the born of a new era of mobile phones, as it included numerous applications, such as an address book, calendar, appointment scheduler, calculator, world time clock, electronic notepad, handwritten annotations and standard and predictive stylus input screen keyboards. Since then, smartphones have been an ever-growing success. More than three and a half billion smartphone network subscriptions existed in 2016. Subsequently, there were nearly twice as many subscriptions in 2022 [11]. For reference, in 2022, the population of the Earth was 7.96 billion [12]. Smartphones manage a vast number of aspects of our existence in the twenty-first century, gradually becoming an essential item. The operations of a smartphone today range from commerce to entertainment

to communications to photography and so much more. It is estimated that the average individual, in 2023, uses their smartphone for more than three hours per day. This corresponds to a 480% increase compared to 2011, when it was only approximately thirty-one minutes [13]. Recent reports show that, in Western Europe alone, in 2020, the average mobile traffic per month was about 4.45 exabytes, as shown in Figure 2.2 [14]. There has been an increase in daily smartphone usage as well as data traffic, which indicates that more people are connected to the Internet and spend more time on their smartphones than ever before.

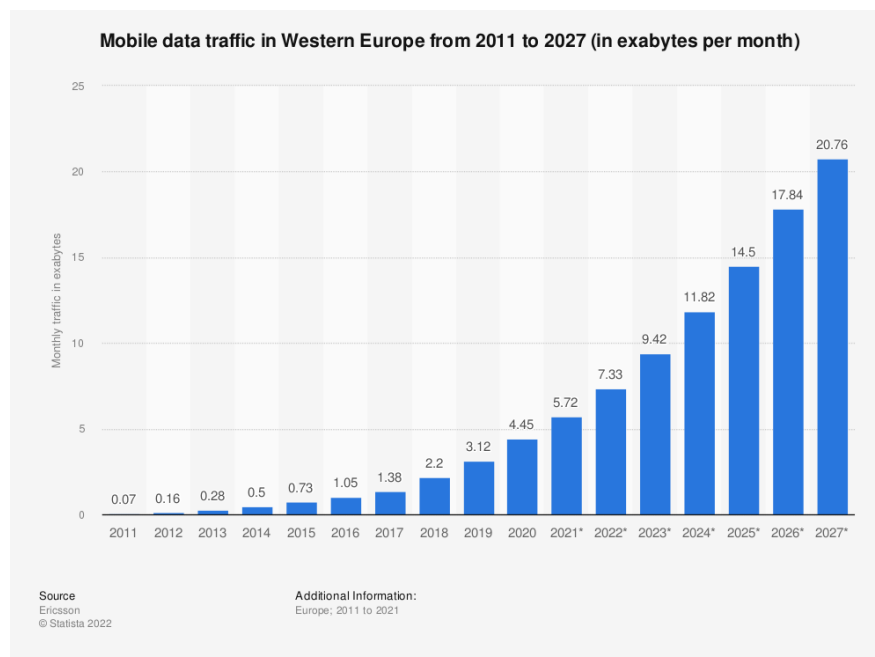


Figure 2.2: Mobile data traffic in Western Europe 2011-2027 (in exabytes per month) [14]

2.1.3 Artificial Intelligence (AI)

AI is the science and engineering behind the creation of devices that are intelligent, especially smart computer programs. AI was born to perform the task of using computers to comprehend human intelligence, but it is not limited to methods that are biologically observable [15]. According to John McCarthy [16], AI comprises numerous subfields and applications. AI is present in master chess game engines [17], speech recognition applications, computer vision software and natural language processing software. AI's versatility in applications comes from the fact that it can be implemented through various branches, including pattern recognition, learning from experience, ontology and inference [18].

AI evolved from the application of knowledge, algorithms and computing ability, commonly

2. Background Knowledge

referred as Machine Learning (ML), to the combination of layers of data, algorithms, and processing power, also known as Deep Learning (DL) [19]. ML is a subset of AI that describes one of AI's fundamental principles. Methods based on ML automatically improve system outcomes through training. DL falls under AI and ML. It instructs a system or machine to process information in layers, classify, interpret and infer the result. Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Generative Neural Networks (GNN) are prevalent DL techniques, which learn from experiences or datasets rather than just instructions [20]. Currently, AI applications are so widespread that they permeate virtually every aspect of society: computer vision, autonomous driving, voice recognition, neural networks and natural language processing, to mention a few, are among the most popular.

2.1.4 ChatBots

In 1966, Joseph Weizenbaum introduced the first ChatBot at the Massachusetts Institute of Technology (MIT). In order to simulate a conversation with a person-centered (Rogerian) psychotherapist, the Eliza program generated responses based on a set of established rules, making natural language conversation with a Computer possible [21]. The program would determine a response based on the user's input keywords. The implementation of the psychotherapist model was ingenious, as the interactions would appear to formulate a natural conversation during a psychotherapy session. This was the primary objective, to create a seamless experience that gave the user the impression that Eliza understood the meaning of every sentence in their inputs, despite the fact that Eliza did not store data from inputs [21].

In 1975, Parry ChatBot was created and designed to behave like a paranoid individual. It was the first ChatBot to successfully pass the Turing test [22]. Like Eliza, Parry interactions were designed based on rules. In addition, it contained an affective model for simulating the ChatBot's emotions.

It is evident that these programs created a significant number of new challenges:

- How to give a ChatBot knowledge about the world?
- Would it be advantageous for the ChatBot to construct a knowledge model of the individual interacting with it?

- Can a ChatBot be designed to be able to handle all interactions, not just a psychotherapy session?

The need arose to supply ChatBot programs with knowledge, not only about the world but also about the context of a conversation or the person speaking to it. The capabilities and usefulness of ChatBots have been vastly enhanced by AI. As previously observed, the earliest ChatBots were rule-based and could only deliver pre-programmed responses in response to specific inputs - they had limited conversational capabilities and followed a simple prescribed logic.

Some years later, around 1995, a new initiative emerged that incorporated AI. It was the first ChatBot with customizable scripts and pattern-matching recursion [23]. Artificial Linguistic Internet Computer Entity (ALICE) was powered by supervised learning, a technique in which the ChatBot's programmer plays a crucial role. By supervising bot interactions, it is the programmer's responsibility to develop new Artificial Intelligence Markup Language (AIML) scripts to enhance the content of the conversation [24].

Alongside a previously-mentioned technology on the rise, a follow-up significant innovation was introduced. The first smartphone to integrate an intelligent personal assistant was Apple's iPhone with Siri; Google Assistant soon followed in Android devices. These ChatBots provided the user with various kinds of assistance and were incorporated into the device's operating system. These were the first ChatBots broadly available that could interpret text, voice and images.

With Apple and Google developing these technologies, ChatBots gained popularity, and Microsoft quickly followed suit with XiaoIce, a social ChatBot with empathy. XiaoIce was introduced in 2014 and promptly went viral. 1.5 million messaging groups added XiaoIce within 72 hours [25].

Table 2.1 displays some of the most important ChatBots in history over time along with some metrics comparisons: time of release, scalability, key features, accomplishment, modality, modeling, domain, key technical breakthroughs and key technical limitations.

2. Background Knowledge

Table 2.1: ChatBot overview over time [26]

Metric	ELIZA	ALICE	Siri	XiaoIce	Chat GPT
Time	1966	1995	2011	2018	2022
Scalability	None	Scripts can be customized	Scalable	Scalable	Scalable
Key Features	Mimicking human behavior in conversation	Easy customisation of scripts (via AIML)	Providing personal digital assistance	Building emotional attachment to users; scalable skill set for user assistance	Ability to generate responses like humans in real-time, based on the user’s input; Multilingual support
Accomplishment	First chitchat bot	Won the Loebner Prize three times	The first widely deployed intelligent personal assistant (IPA)	The first widely deployed social ChatBot; 100MM users; published poem book; host TV programs	Generative Pre-training Transformer architecture and is trained on a massive dataset of internet text;
Modality	Text only	Text only	Text, image, voice	Text, image, voice	Text only
Modeling	Rule-based	Rule-based	Learning-based	Learning-based	Learning-based
Domain	Constrained domain	Constrained domain	Open domain	Open domain	Open domain
Key technical breakthrough	Use of scripts; Keyword-based pattern matching; rule-based response	Using AIML and recursion for pattern matching; multiple patterns can be mapped into same response	Provide both reactive assistance covering a wide range of domains	Emotional intelligence models for establishing emotional attachments with users	Deep learning techniques to generate human-like text; ability to generate text that is coherent and fluent, contextually relevant and can be fine-tuned for specific tasks
Key technical limitation	Limited domain of knowledge	Size of scrip can be huge	Lack of emotional engagement with users	Inconsistent personality and responses in long dialogue	Cannot access Internet in real-time; cannot handle multitasking

2.1.5 Human-in-the-Loop Cyber-Physical Systems (HITLCPS)

The majority of IoT applications today do not include the Human as an integral component of the system and in order to develop technology that better services humans, we must develop systems capable of comprehending their actions, intentions, and emotions [27, 28]. Four phases are depicted in Figure 2.3: data acquisition, present state inference, future state inference and actuation.

HITLCPS applications are meant to react to natural human behavior and do not necessarily require direct human interaction [29]. The acquisition of data through which the human’s state may be inferred is a complex process with a multitude of possible sources of informa-

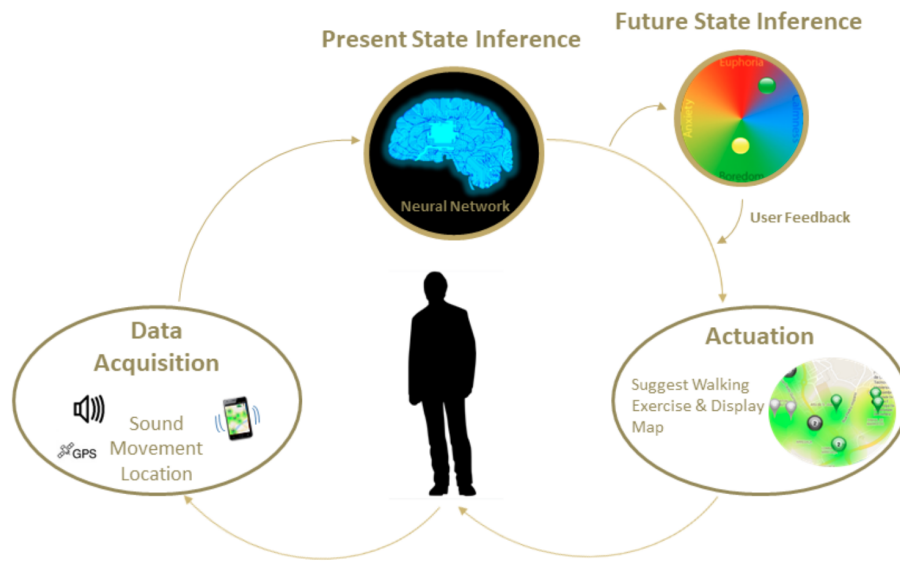


Figure 2.3: Control Loop for a HITLCPS [28]

tion [29]. Many physical data, such as a person’s Global Positioning System (GPS) location and heart rate, can be used to infer their state, however, there are also valuable non-physical data, such as communication and socialization practices [29].

2.2 Framework Technologies

This section emphasises the software and frameworks used to develop this project. Xamarin is the base framework that is used by the INESCC-DEEC group to develop multi-platform applications and FIWARE framework is used to deal with data management and context information. Flask, Dialogflow, GPT-3 and Lottie are frameworks used to develop this project.

2.2.1 Xamarin

Xamarin is an open-source platform for developing modern and performant iOS, Android, and Universal Windows Platform (UWP) applications using the .NET framework. As an abstraction layer, Xamarin bridges the gap between shared code across multiple platforms and platform-specific code, as seen in Figure 2.4. Xamarin allows developers to share an average of 90 percent of their application code across multiple platforms. This pattern en-

2. Background Knowledge

ables developers to compose or reuse existing code while still achieving native performance, appearance, and behavior on each platform [30].

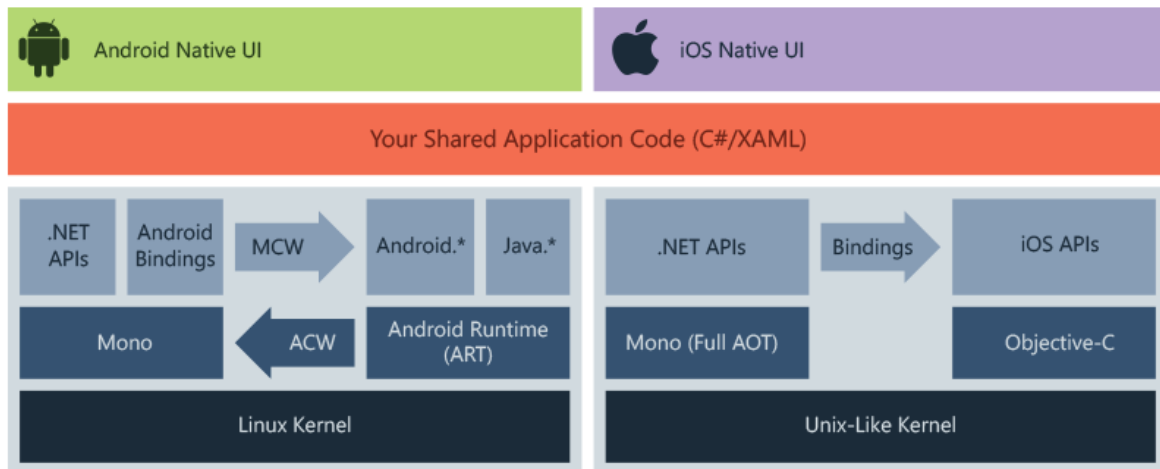


Figure 2.4: Xamarin Architecture[30]

The most recent version of Xamarin is Multi-Platform App User Interface (MAUI), which, in addition to Xamarin's features, offers improved performance with fewer resources and enables the development of a native User Interface (UI) within a single project. Our team is currently upgrading all Xamarin platforms to MAUI.

2.2.1.A Xamarin.Essentials

Xamarin.Essentials is a library that provides developers with a single cross-platform Application Programming Interface (API) that is compatible with all Xamarin.Forms, as well as Xamarin.Android, Xamarin.iOS or UWP application using shared code. It is a tool that considerably accelerates the mobile app development process by providing cross-platform support for the common features that mobile apps typically require [31].

The greatest advantage of Xamarin.Essentials is the wide range of out-of-the-box functionality it provides, such as network connectivity detection, device geolocation, persistent preferences, secure data storage, device information, file system helpers, accelerometer data, phone dialer access and text-to-speech capabilities, among others. All of these features suppress the need for platform-specific code, allowing developers to write their applications in C# [31].

2.2.1.B Xamarin.Forms

Xamarin.Forms is an open-source User Interface (UI) framework. Xamarin.Forms allows developers to build multiplatform applications from a single shared codebase and create user interfaces in Extensible Application Markup Language (XAML) with code-behind in C#. These interfaces are rendered as performant native controls on each platform [32].

2.2.2 FIWARE

FIWARE is an open-source framework of key open standards that enable the development of portable and interoperable smart solutions [33]. The primary and only required component of any platform or FIWARE solution is a FIWARE Context Broker Generic Enabler, which provides an essential function required by any smart solution: the need to manage context information, enabling updates and providing access to context [34]. A very powerful Context Broker currently provided by FIWARE is the Orion Context Broker (OCB) which is a core component of the FIWARE platform, illustrated in Figure 2.5. It is a server that enables the management of the lifecycle of context information, including updates, queries, registrations, and subscriptions [35]. The OCB enables the creation of virtual entities that represent real-world objects or human beings [27, 35]. It is also possible to create and manage context elements, as well as query and update them as required [35].

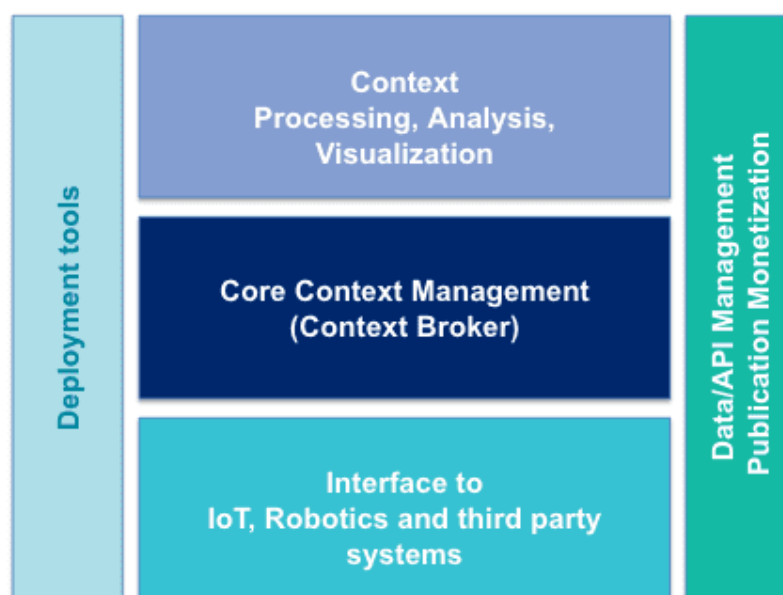


Figure 2.5: FIWARE Modules [34]

2.2.3 Flask

Flask is a lightweight, micro web framework written in Python. It was created to assist Python developers in rapidly building basic web applications without requiring a complex framework. Its "micro" label indicates that it seeks to keep the application's core simple and extensible. Uniform Resource Locator (URL) routing (what code to execute on which page) and request handling are essential and straightforward features provided by Flask [36].

2.2.4 Dialogflow

Dialogflow is a natural language understanding platform that makes it simple to design and integrate a conversational user interface into mobile apps, web applications, etc. It is capable of analyzing a selection of inputs, including text and audio. Additionally, it can respond through text or synthetic speech [37].

It is a platform that provides the ability to create a fully customizable ChatBot that produces individualized responses to the user based on a previously configured intent match. To create a competent ChatBot agent, we must provide it with knowledge inputs. If configured to do so, the agent is capable of maintaining context during interactions. It is also feasible to provide training phrases to improve its user input interpretation and intent matching. It is possible as well to create a webhook service to access external databases and APIs to fulfill the agent responses [37]. This architecture is illustrated in Figure 2.6

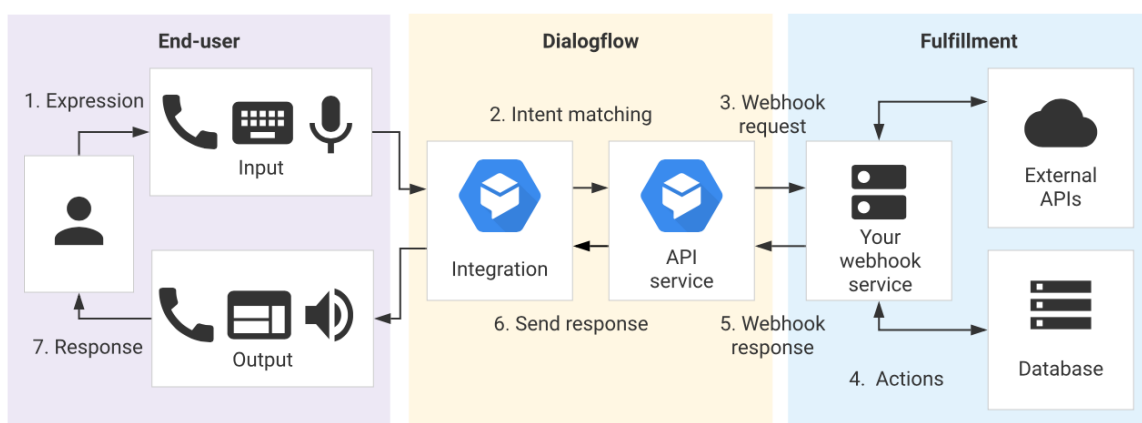


Figure 2.6: Dialogflow Architecture [37]

2.2.5 GPT-3

OpenAI's GPT-3, which stands for Generative Pre-Trained Transformer 3, is a highly advanced language model. It uses deep learning to generate human-like text by anticipating the following word based on the context. With 175 billion machine learning parameters, GPT-3 can comprehend context and generate coherent, pertinent sentences. It is capable of learning from a few examples and accurately performing duties. GPT-3 has numerous applications, including email writing, scripting, tutoring, language translation, and even creative writing, however, it has some limitations and can occasionally generate incorrect or nonsensical output, despite its strengths [38]. OpenAI released an iteration of this language model, GPT-4, that improves all previous features and has access to external plugins, making it the best version available. GPT-4 is currently being integrated in our project.

2.2.6 Lottie

Lottie is a library for Android, iOS and Web (officially) that parses animations exported as JavaScript Object Notation (JSON) files from an open-source Adobe After Effects plugin called Bodymovin [39, 40]. It enables native applications to render animations for approximately the same cost as rendering static assets and the file size of lottie animations is significantly smaller than any animation in video format. In addition to Android, iOS and Web, Lottie provides the following stable versions, developed by its community: React Native, Windows, Qt, Skia, Python and Xamarin [39, 41].

Chapter 3

Group Platforms

3.1 IoT Student Advisor and BEst Lifestyle Analyzer (ISABELA)

ISABELA is a mobile application developed to help students achieve better results in their academic journey. It is based on the four phases of HITLCPS previously mentioned: data acquisition, inference, future inference and actuation. Using IoT devices, it collects sound and light data from the student's University facilities and residence. Using the smartphone's sensors, it also records the student's location, physical activity and social interactions. It implements a FIWARE architecture, displayed in Figure 3.1, which provides a solid ground to store data and communicate with different devices [27].

ISABELA project's objective is to get insights of the student's routines in order to prevent bad habits that could lead to poor academic results or even dropout. The HITLCPS architecture of ISABELA generates a system in which students are the primary agents in modifying their own behavior.

From a research perspective, one of ISABELA's goals is to develop and test modern, robust privacy techniques. As the collection of data continues to increase, so does the need for enhanced privacy safeguards to reduce potential hazards and abuse.

This platform was initially designed to monitor students twenty-four hours a day using smartphone sensors and IoT devices to gain a deeper understanding of their environment and daily behaviors. It monitors the student's activity, sleep, social interactions and location in order to

3. Group Platforms

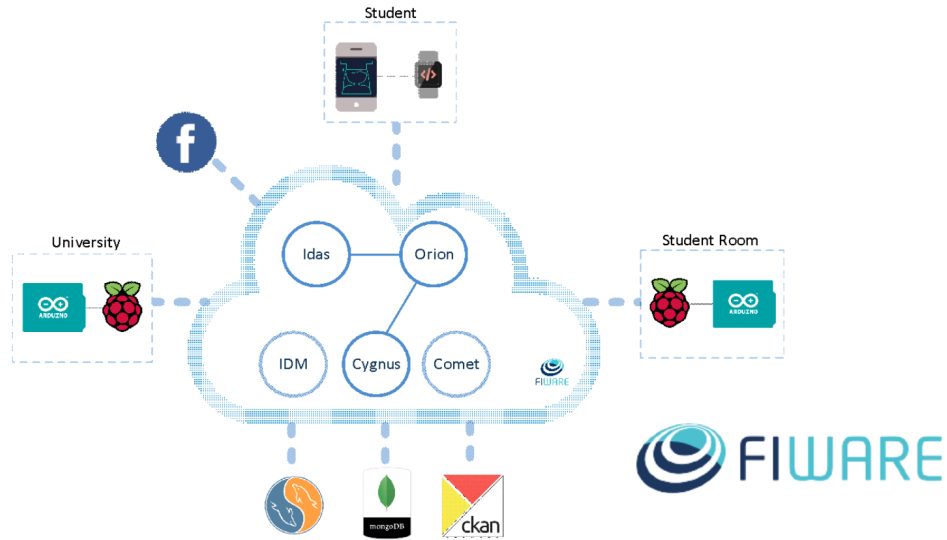


Figure 3.1: ISABELA Architecture [27]

infer their behavior. Figure 3.2 gives an example of the data that is displayed to the student in real-time.

The student is notified via smartphone push notifications and a ChatBot whenever a negative or positive behavior is identified. The function of the ChatBot is to interact with the student, either in response to these inferred bad or good behaviors detected by the system, or in response to textual input from the user. In the case of bad behavior, if the student alters it, the system will recognize the change and cease sending them notifications about their previous misbehavior, it will continue to alert the student otherwise [27].

As ISABELA implements a complete HITLCPS, it has become a core developing and testing platform for new functionalities in our group's research and development projects. In addition, in the context of a partnership with the FMUC's Clinical Psychology group, we are currently adapting the HEXACO Personality Inventory test and the Profile of Mood States (POMS) test into surveys within ISABELA [42, 43].

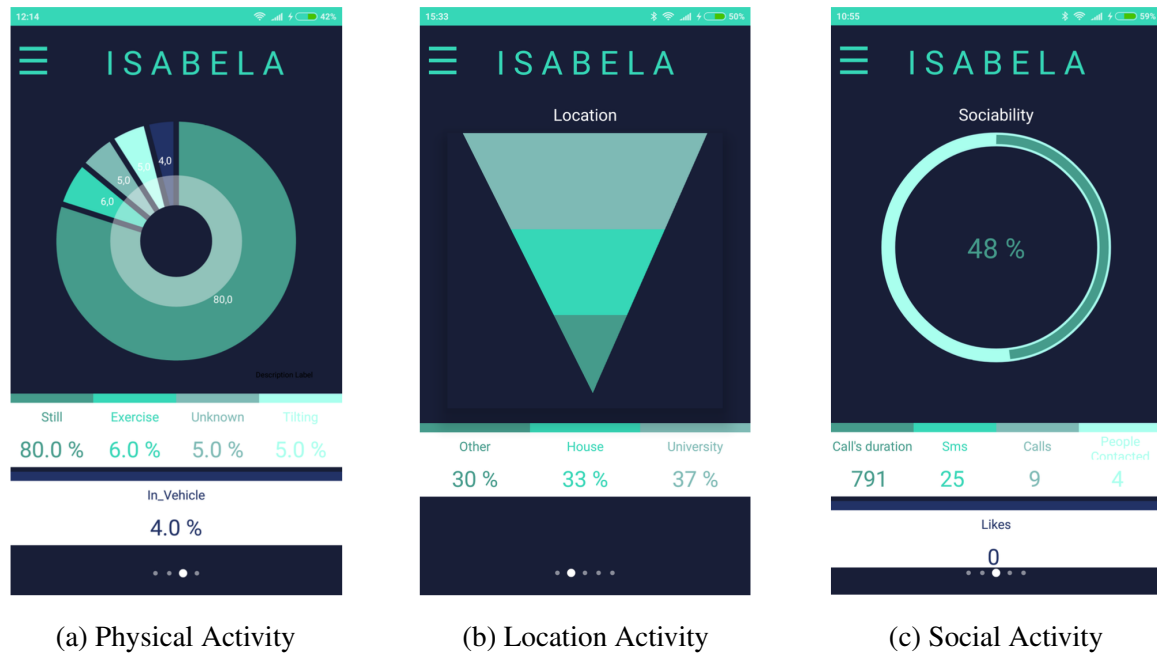


Figure 3.2: ISABELA's Feedback Pages [27]

One of the features in ISABELA's smartphone application is the ChatBot. This ChatBot was developed integrating Google's Dialogflow API (formerly API.AI) directly on the application. As mentioned previously, Dialogflow is a platform that provides the tools required to construct a fully customisable ChatBot and functions as an external API that developers can instruct in real time using Google's technology. Given that Google's machine learning algorithms are among the most advanced, this is a major benefit in NLP for any project [27, 37]. This ChatBot was a basic prototype centered entirely on the ISABELA's platform, which served as the foundation for the work presented in this dissertation.

Based on the student's behavior, ISABELA's ChatBot is equipped with 5 intents:

- Too much time still (not moving)
- Insufficient exercise
- Not enough time at the university
- Excessive time at home
- Insufficient sleep

In addition, it has some intents that take advantage of the deployed IoT devices and smartphone sensors, as well as the ability to deal with user's interactions. Figure 3.3 illustrates

3. Group Platforms

two distinct ChatBot interactions: in Figure 3.3a, the student requests assistance defining an alarm and finding the local room temperature, in Figure 3.3b, the system is triggered to provide the student with feedback regarding their lack of physical activity (triggered "Insufficient exercise" intent) and it suggests the student to go for a run. This alarm interaction was adapted during this project's development to the new designed architecture.



Figure 3.3: ISABELA's ChatBot [27]

ISABELA also includes a smartwatch application that communicates with the smartphone application in order to better track physical activity and movement. Figure 3.4 depicts the foreground activity that displays physical metrics which include steps taken and heart rate using the smartwatch's internal sensors, while the background activity broadcasts these metrics to the smartphone.

ISABELA is the foundation for several subsequent initiatives, including BATINA (section 3.2), ON-BOARD (section 3.4) and GreenBear (section 3.6). As mentioned previously, ISABELA is also a development and testing platform for our ongoing group's projects which result from partnerships with the UC, the FMUC and the Coimbra City Hall.

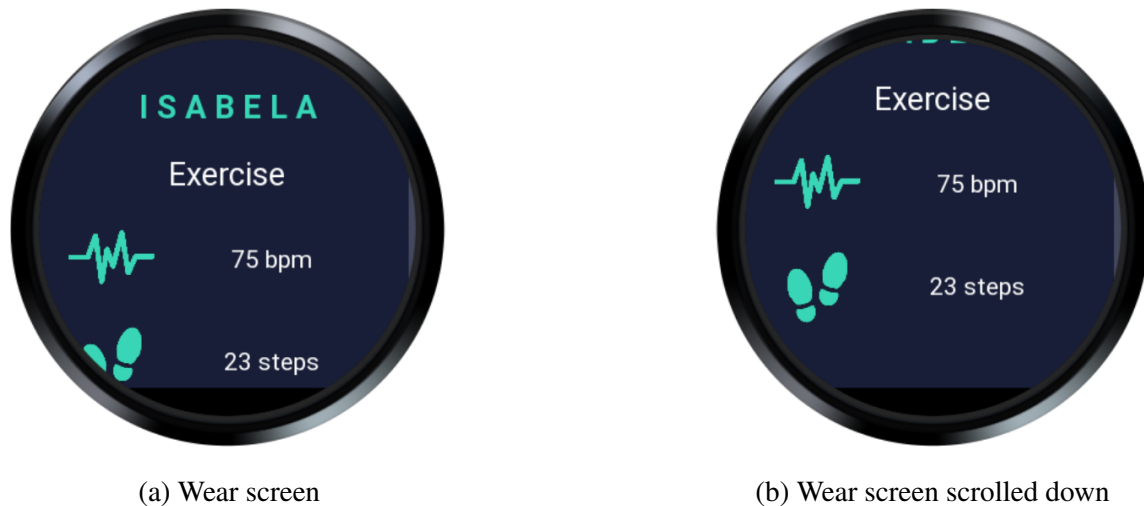


Figure 3.4: ISABELA's Smartwatch Activity [27]

3.2 Best Assisting Tutor and INteractive Advisor (BATINA)

BATINA is a tutoring system comprised of a dashboard web application and a smartphone application based on ISABELA. The primary objective of this system is to be implemented in the classroom and provide resources for professors (who utilize the dashboard app) and students (who utilize the smartphone app). Furthermore, BATINA, alongside ISABELA, is used for the development of new features, in the context of partnership with the Rectory of the UC, to be integrated in the official UC applications: UC Teacher and UC Student (detailed in section 3.3) as well as the ON-BOARD project (detailed in section 3.4).

3.2.1 BATINA Dashboard

The BATINA Dashboard is a Dash-based [44] platform which enables professors to manage their courses, displayed in Figure 3.5a, and classroom-related duties, such as creating and managing questionnaires, illustrated in Figure 3.5b, or providing study materials to students, as seen in Figure 3.6a, as well as a system to answer questions, displayed in Figure 3.6b.

Here are the main features of the BATINA Dashboard:

- **Questionnaires** can be created and monitored throughout its duration. It is possible to construct a questionnaire by writing each question individually or by letting the system automatically pick questions from a question database (Question Pool), as seen in Figure 3.7.

3. Group Platforms

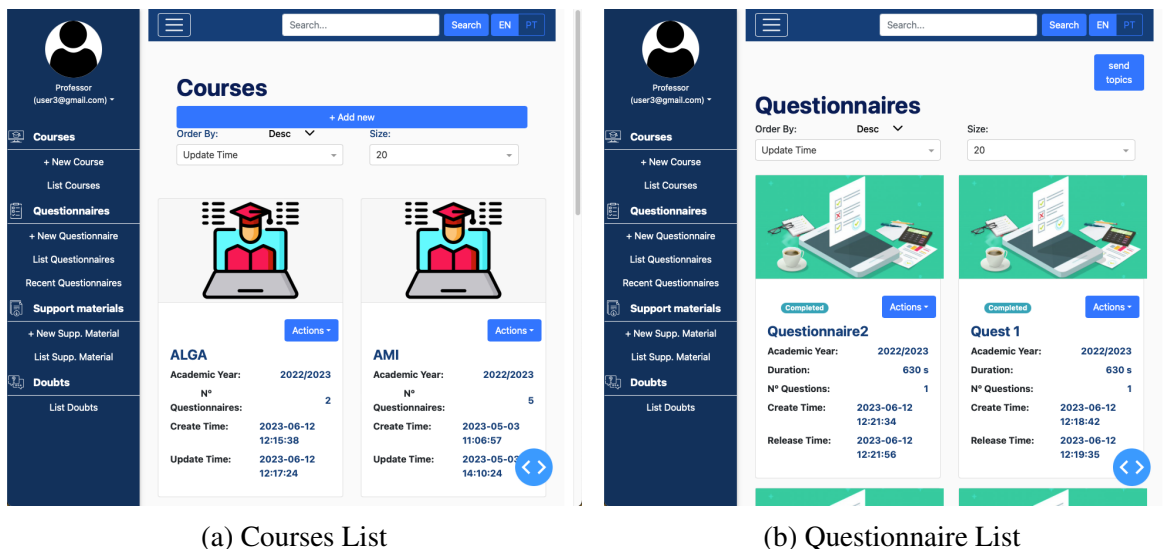


Figure 3.5: Professor's Course and Questionnaire Lists

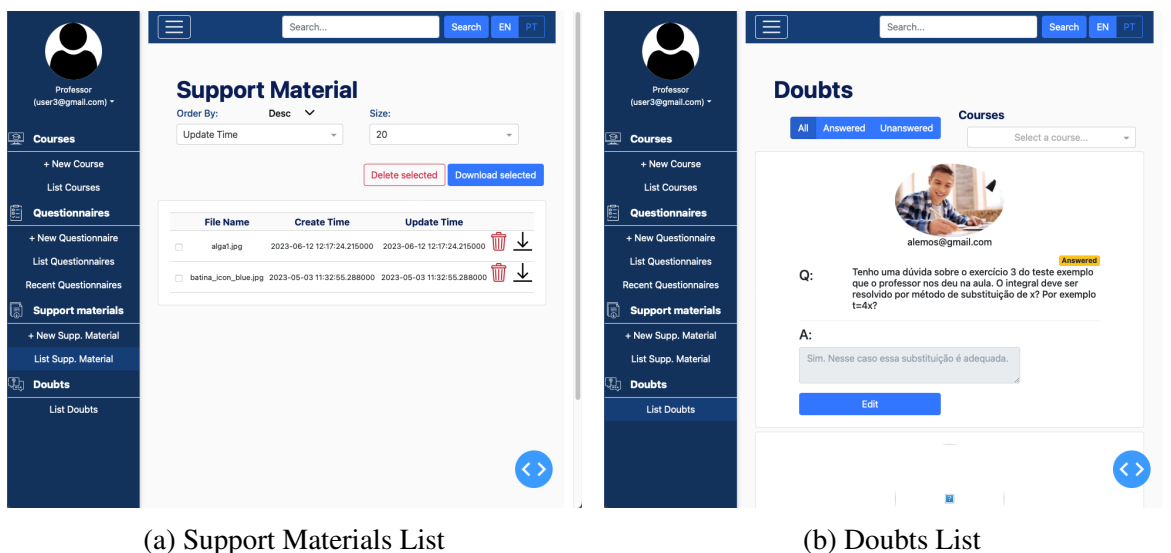


Figure 3.6: Professor's Support Material and Doubts Lists

- **Courses** display the roster of students enrolled in a particular course taught by the professor. It provides charts indicating success and failure rates based on questionnaire results, both for the course as a whole and for each individual student. This allows one to predict, for instance, dropout rates. This example can be seen in Figure 3.8.
- The professor may provide **Support Materials** about the lectures which can be accessed by the students via the smartphone application, illustrated in Figure 3.6a.
- Any student may submit **Doubts** and the professor has a page, depicted in Figure 3.6b, dedicated to answering them. Each time a response is provided, the student who posed the question is notified in real-time as seen in Figure 3.9.

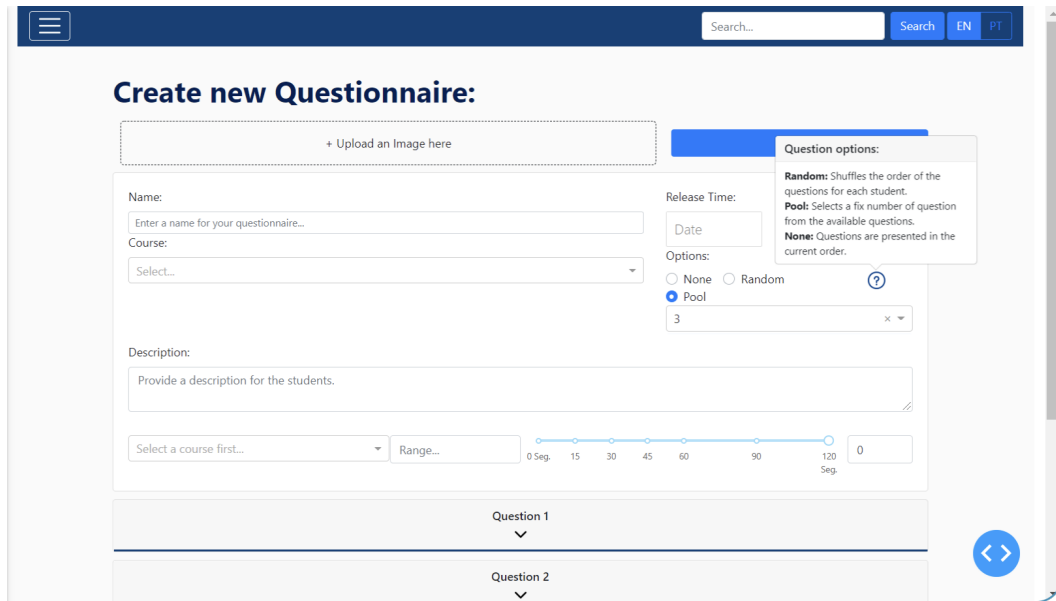


Figure 3.7: Creating a questionnaire from pool

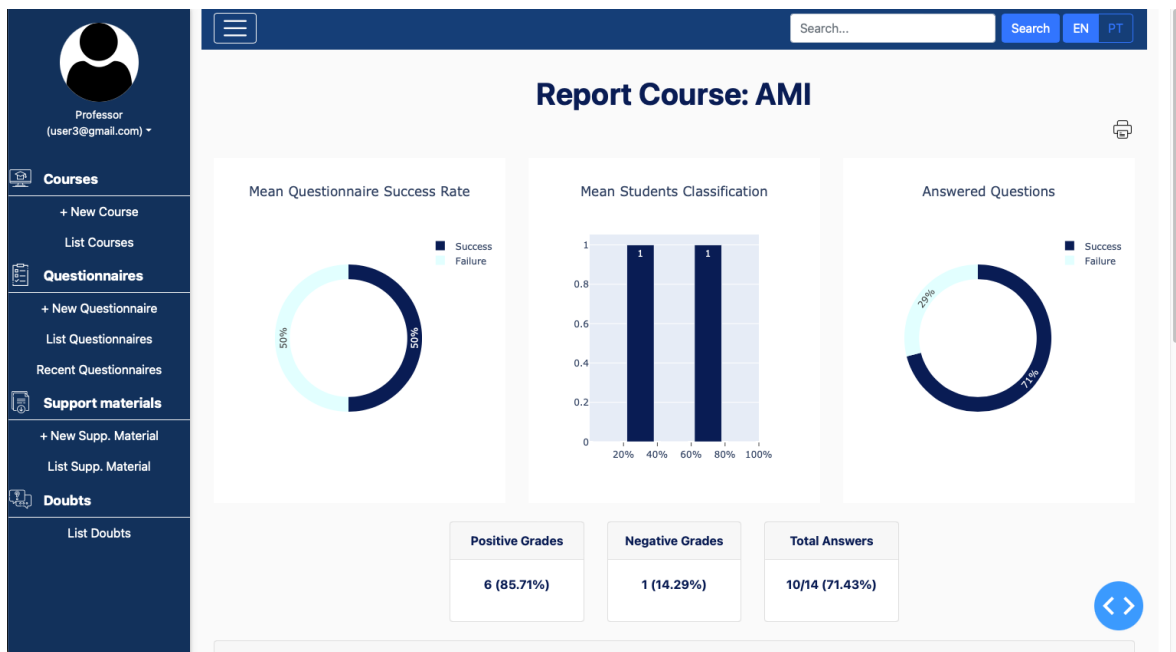


Figure 3.8: Course Report

3.2.2 BATINA Mobile

BATINA Mobile is a smartphone application designed for students. Its primary objective is to provide insight into academic achievement and results through an analytical summary. In addition, it encourages positive habits that contribute to improved academic performance via feedback pages with data and a ChatBot, similar to ISABELA, although BATINA Mobile focuses more on classroom performance and does not use IoT devices to monitor students’

3. Group Platforms

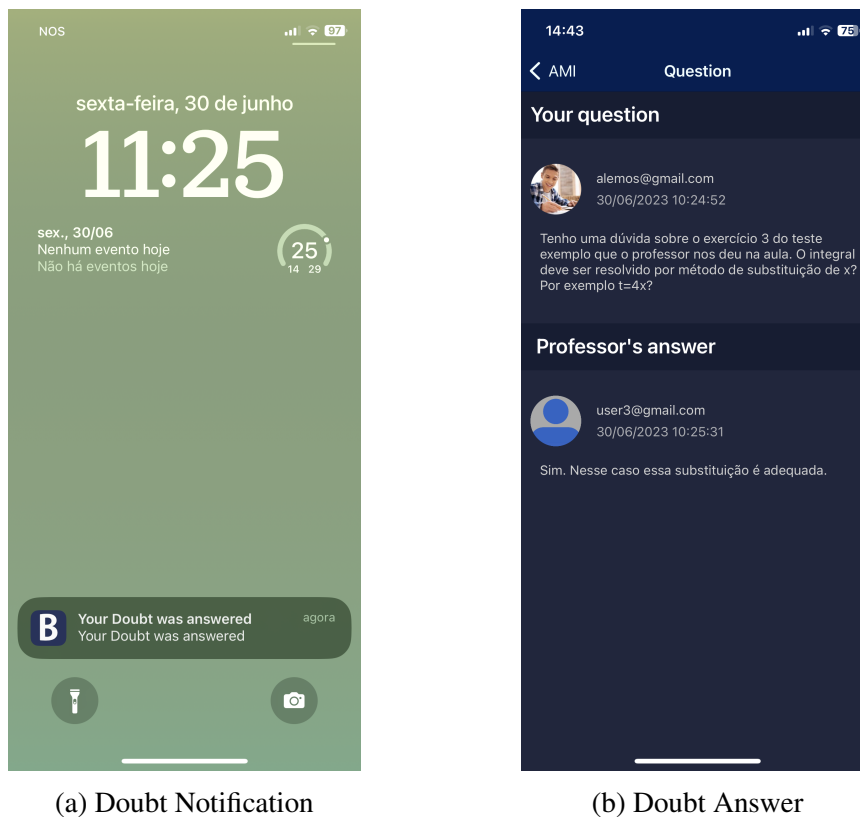


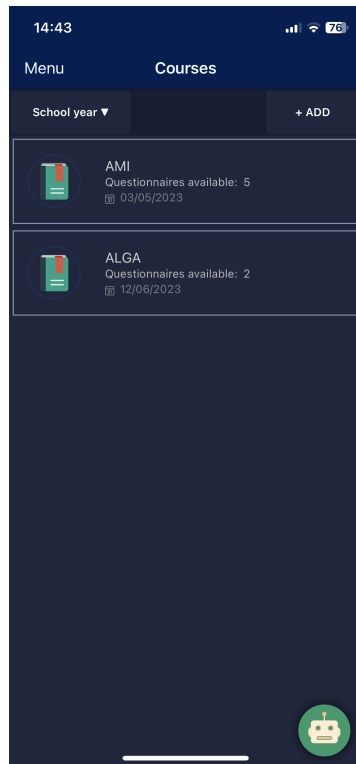
Figure 3.9: Student Doubt Answered

daily habits.

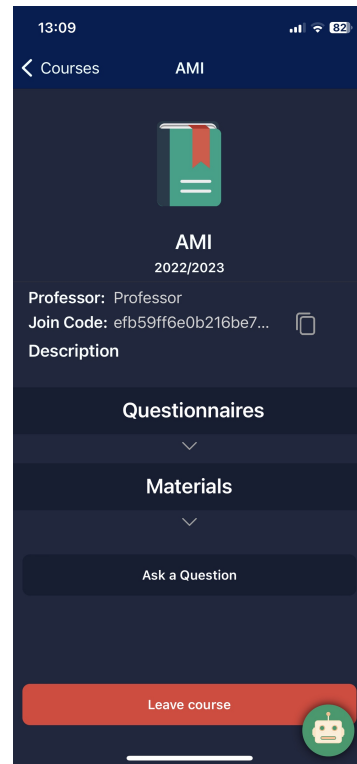
BATINA Mobile has five main pages: Courses Page, Courses Reports, Global Report, Settings Page and ChatBot Page. The Courses Page gives a list of courses the student is currently enrolled in, as well as a history of past courses. Each Course page has the correspondent questionnaires, support materials and the ability to ask questions to the professor, as seen in Figure 3.10.

The questionnaires menu displays uncompleted questionnaires and maintains a record of completed ones, as displayed in Figure 3.11. When a student starts a questionnaire, there is an initial page, displayed in Figure 3.12a, which warns the student to maintain a stable connection and to not leave the questionnaire during its duration, as it will end automatically otherwise. In Figure 3.12b, there is an example of a multi-choice question. The support material menu provides access to any course-specific materials supplied by the professor.

3.2 Best Assisting Tutor and INteractive Advisor (BATINA)

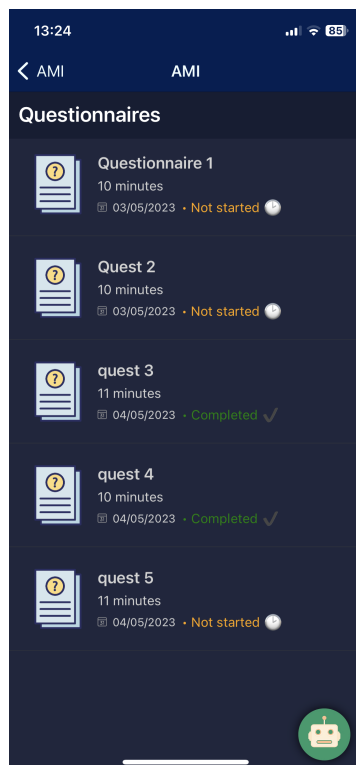


(a) Courses List

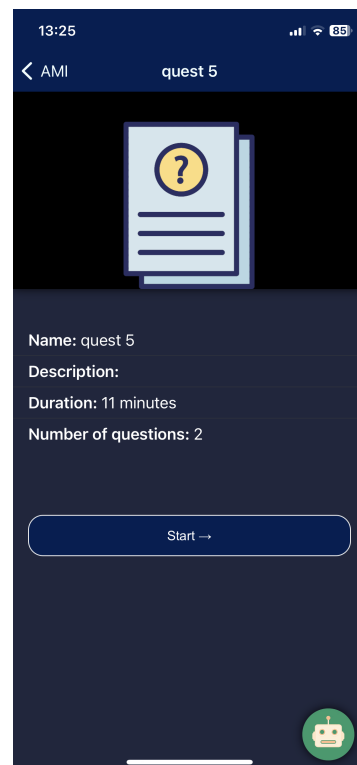


(b) Course Detail Page

Figure 3.10: BATINA Mobile Courses Pages



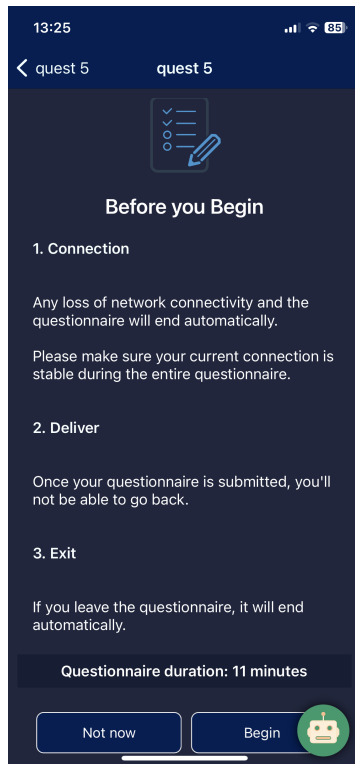
(a) Questionnaire List



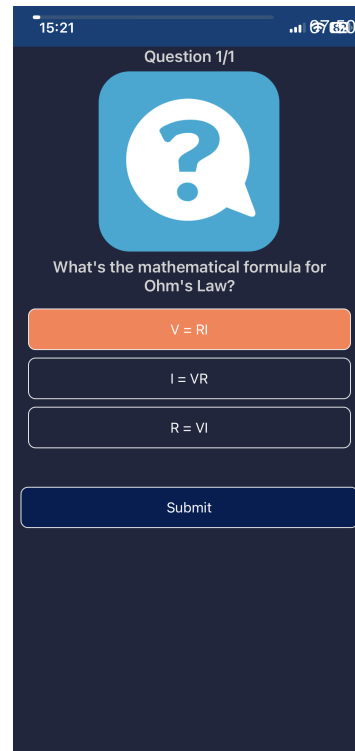
(b) Questionnaire Detail Page

Figure 3.11: BATINA Mobile Questionnaire Pages

3. Group Platforms



(a) Questionnaire Initial Page



(b) Question Example

Figure 3.12: BATINA Mobile Questionnaire Pages (cont)

The Course Report page gives the student an analytical insight about their academic performance based on questionnaire results, illustrated in Figure 3.13a. The student can have an overview of their grades in each questionnaire and which questions got wrong or right. Figure 3.13b depicts the global academic performance through the various courses the student is currently enrolled in, as well as the possibility to check past global reports from previous academic years. It is possible to consult each course individually from this page, by clicking on the correspondent course.

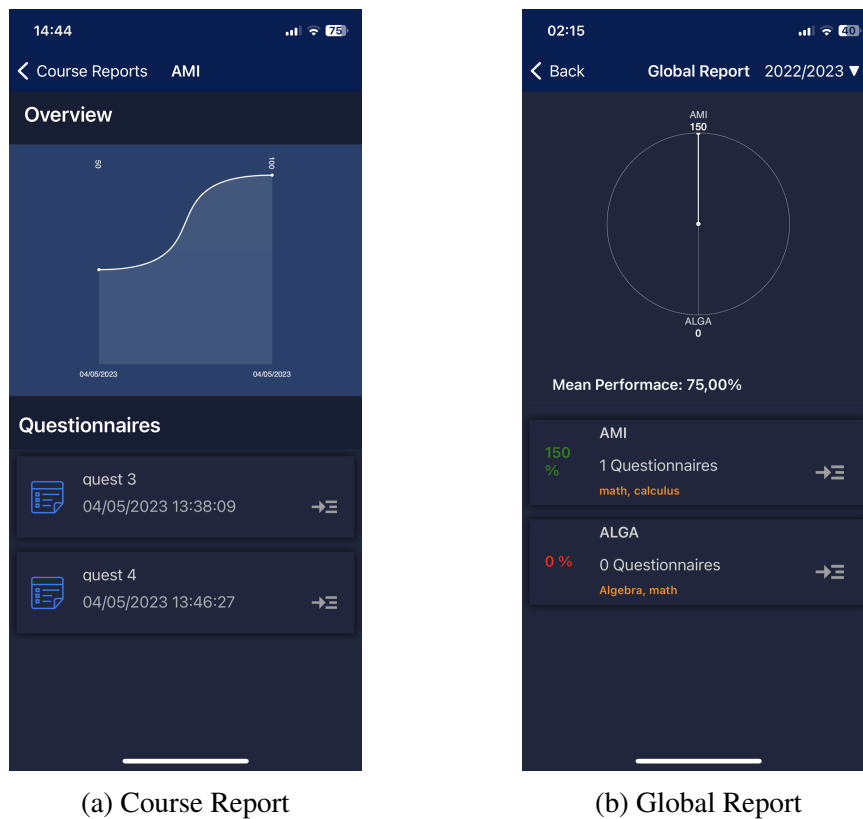


Figure 3.13: BATINA Mobile Reports

3.3 UC Teacher / UC Student

UC Teacher is an academic platform to support classes with the primary goal of providing all University of Coimbra (UC) professors with an instrument to manage their entire teaching activity. This platform exists for the sole purpose of serving the entire UC community. The platform is the subject of significant development and is becoming increasingly vital to the daily lives of all UC instructors, providing features like conducting remote exams, record class sessions or meetings, create groups to share content with specific users, multilingual translation and creation of repositories. It should also be noted that the UC Student platform shares a close connection with UC Teacher [45]. Figure 3.14 displays the UC Teacher's platform. On the left side there are the main menus, where the teacher is able to manage courses, groups, exams, support materials and files, chatting channels and recordings from meetings, classes or exams. On the center there is some statistical data about the professor's activity and a quick action menu.

UC Student is an academic platform available to all students at the University of Coimbra for class support [46]. It shares some primary features with UC Teacher and also includes

3. Group Platforms

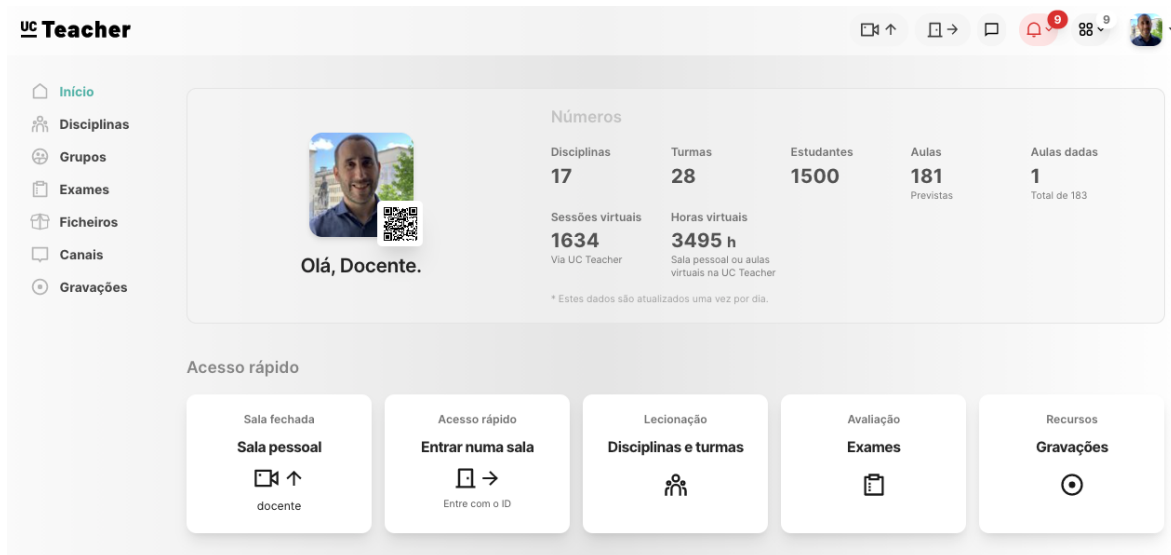


Figure 3.14: UC Teacher [45]

the ability to join or create virtual rooms and inspect enrolled courses, classes and support materials [46]. The UC Student's platform is illustrated in Figure 3.15, and it gives the students the same features as UC Teacher's except the exams, recordings options and the management of courses and support materials.

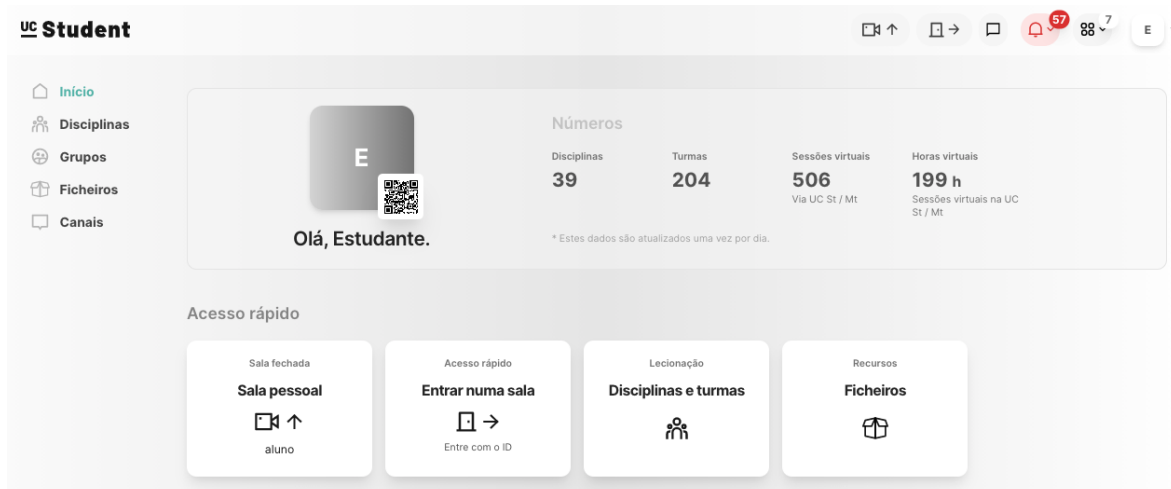


Figure 3.15: UC Student [46]

Our group is working with UC Frameworks, the entity responsible for the official UC platforms, to integrate the most valuable features developed in the ISABELA and BATINA applications into the UC Teacher and UC Student platforms. Some features currently being integrated are the BATINA's questionnaires and the smartphone application.

3.4 ON-BOARD Project

The ON-BOARD project is the product of a successful application of the UC to the Plano de Recuperação e Resiliência (PRR) funding for education and it focuses on personal and digital tutoring procedures for students and teachers, as well as the transformation of organisational conditions and performance in academia. The student dropout rate, especially during the first year of study, is one of the primary concerns of this project. It is based on the following key components:

- **Data Analytics Solution:** This involves the systematic analysis of data to ascertain patterns and predictive models related to academic failure and premature discontinuation of studies.
- **Adaptive Learning:** This element is focused on the enhancement and modernization of student learning experiences.
- **Happy Campus UC:** This encompasses workshops aimed at fostering personal and social connections, leadership and overall well-being.
- **Peer Tutoring:** This component involves collaborative dynamics where upper-year students provide guidance and support to first-year students.
- **Tutoring Preparation:** This entails the exchange of experiences among teaching staff from the UC and other domestic or international institutions, with the objective of sharing exemplary pedagogical practices.

In addition to the collection and analysis of data to identify dropout prediction patterns, the initiative integrates technological tools for adaptive teaching and personalised monitoring of students. Wellness and mental health are also considered by promoting stress and anxiety management techniques among students and instructors [47, 48].

This project is one of the next steps in developing the ChatBot, as an integrated part of our group's partnership with the UC, to contribute as a digital tutor for students. Thanks to the PRR funding, this work will be developed over the course of the upcoming months.

3.5 Similar Academic Platforms

3.5.1 StudentLife

Researchers at Dartmouth College conducted the StudentLife study to objectively measure and evaluate the behavioral trends and mental health of college students using sensor data collected from smartphones, the research was longitudinal in nature and spanned the entire 10 week academic term. The primary objective was to determine how factors such as sleep, physical activity, social interaction and location affect academic performance and mental health. In addition to sensor data (GPS, accelerometer, etc.), variables like phone usage and responses to periodic self-reported surveys from students were integrated. Consequently, the collected data was quite comprehensive [49].

A significant portion of the study was devoted to predicting the student's Grade Point Average (GPA). The researchers were able to predict the GPAs by correlating these variables with academic performance. In an identical manner, class attendance was evaluated using GPS and activity data. It was observed that students with a higher GPA had more regular sleep patterns, spent more time studying, were reserved on their phones and were more likely to be in locations classified as 'academic.' Conversely, students with lower GPAs had irregular sleep patterns, less study time, more time at parties and longer evening conversations. The researchers concluded that this behavior is associated with self-reported stress and loneliness and has a negative effect on academic performance. Students who partied more frequently had lower GPAs, were less likely to attend classes and reported higher levels of stress. The StudentLife study provides a comprehensive digital phenotype of student life and helps to identify and address the factors that influence academic performance and mental health among students. Continuous monitoring using devices could offer an unobtrusive method for detecting behavioral trends among college students, allowing for the possibility of early intervention [49, 50].

3.5.2 BigBlueButton

BigBlueButton's features are geared towards the educational setting, replicating the classroom environment in a virtual space. It enables teachers to, among other things, exchange docu-

ments, play videos, use a virtual whiteboard, share their screen and carry out polls. To keep track of student development, it provides the teacher with an Analytics Dashboard, on which the teacher can monitor who attends classes, who actively participates and how students are evolving (based on poll results), making it easy to identify potential students who are struggling [51, 52].

This software is entirely open-source and was developed by a community of committed developers with a passion for enhancing online learning. It has been deeply embedded into major learning management systems used world-wide. It has been adopted by Canvas, Moodle Cloud, Sakai, Jenzabar, D2L and Schoology as their default virtual classroom [51].

3.5.3 ALEKS

Assessment and LEarning in Knowledge Spaces (ALEKS) is a web-based, artificially intelligent assessment and learning system that uses adaptive questioning to accurately determine exactly what a student knows in a field of study. Its AI is based on a theoretical framework called Knowledge Space Theory (KST) [53, 54]. KST allows the representation in the computer's memory of an enormously large number of possible knowledge states that organize a scholarly subject. Rather than giving a score or series of scores that describe a student's overall mastery of the subject, KST allows for a precise description of what the student knows, does not know and is ready to learn next [53, 55].

The adaptive nature of ALEKS is based on the principles of formative assessment, which involves ongoing evaluation of student understanding to tailor instruction to individual needs. As students interact with the system, ALEKS continually reassesses their knowledge, adjusting the learning path as necessary. This dynamic approach allows for differentiated instruction, accommodating a wide range of learning styles and paces. It also provides immediate feedback, which research has shown to be crucial for effective learning [53, 55].

3.5.4 Socrative

Socrative is a web-based educational platform with two major features: interactive classroom engagement tools and customisable assessment capabilities. The first feature, interactive classroom engagement tools, is designed to facilitate real-time communication and instant

3. Group Platforms

feedback during lessons. Educators can launch quick polls or questions in the midst of a lecture and students can respond immediately, stimulating their participation. This instantaneous feedback enables educators to modify their approach on the spot based on student comprehension and engagement [56].

Alternatively, Socrative's customisable assessment capabilities are designed to enable educators to create, distribute and evaluate quizzes and tests. The types of questions vary from multiple choice to true or false and short answer questions. These assessments can be utilized as assignments, spontaneous quizzes, or unit examinations. Post-assessment analytics provide educators with a detailed understanding of each student's performance, allowing for personalized and effective feedback and subsequent lesson planning [56].

Despite the fact that Socrative's features facilitate real-time feedback and individualized learning, it is crucial to note that the platform's efficacy is largely dependent on the instructor's pedagogical approach. Without an effective teaching strategy, it may not be possible to attain the desired educational outcomes using the platform. It is the responsibility of the educator to use Socrative's features to enhance the learning experience.

3.5.5 Comparative Analysis

Using IoT and smartphone sensors, the ISABELA and BATINA projects seek to improve academic performance and prevent bad habits. While both projects are designed to enhance academic performance, ISABELA focuses on student lifestyle, whereas BATINA focuses on developing performance in the classroom for both professors and students. In the context of a partnership with FMUC, these platforms are also being used to develop new clinical psychology support features. ISABELA also has an integrated ChatBot that provides direct feedback to students and handles their inquiries, which is crucial for promoting an effective interaction between the platform and the student. Similar to ISABELA but without real-time feedback, the StudentLife Study utilises smartphone data to gather insights about student behaviour and academic performance in order to predict student success. This data-driven strategy in HITLCPS is comparable to BigBlueButton's use of analytical data to monitor student engagement and progress. ALEKS also relies on data, but its AI-driven approach goes one step further by adapting the learning topics dynamically based on the student's individual knowledge level. Socrative, meanwhile, integrates real-time feedback into classrooms and

provides immediate insights into classroom methodology.

All of these platforms and studies demonstrate how technology is transforming education to become more data-driven, personalised and able to provide real-time feedback. Even though they provide real-time insights about academic performance and knowledge levels, there is no mutual interaction between the student and the system, with the exception of ISABELA. Consequently, it is essential to cover this void in human-system interaction, therefore the proposal to develop a ChatBot.

3.6 GreenBear

The GreenBear initiative began as a case study, evolving into a collaborative project with the Coimbra City Hall. This evolution underscores its growing relevance by promoting sustainable practices and enhancing the quality of urban life. The initiative employs technology to encourage healthier habits and sustainable behaviors, aligning with the broader concept of smart cities. Developed using Xamarin Forms, the platform is designed to support Android and iOS devices and offers a variety of features to stimulate and monitor user engagement in sustainable actions.

One of the platform's key features is a user-friendly interface that includes a map showcasing local recycling bins, green spaces and public transportation options. In addition, it includes a ChatBot that responds to user questions and provides feedback. The GreenBear initiative is characterized by its rewards-based system, which is designed to encourage users to engage in more sustainable behaviors. This system not only encourages participation, but also permits the collection of data on user behavior.

The collected data is a valuable resource for assessing the efficacy of the platform's incentives and identifying areas for enhancement. The GreenBear initiative is committed to ongoing testing and development to ensure that the platform remains responsive to user requirements and continues to foster long-term sustainability.

The INESCC group has published an article that has been peer-reviewed and published in the *Pervasive and Mobile Computing Journal*, which is classified in the top quartile (Q1) of journals in its field. This article features a ChatBot created using the architecture proposed in this dissertation. The article is provided in Appendix B.

Chapter 4

ChatBot Development

4.1 Context

This chapter overviews the ChatBot development and implementation in the ISABELA and BATINA projects.

As stated before, the core development of this project is a ChatBot for HITLCPS learning platforms. The development integrated the INESCC-DEEC research group's projects, taking advantage of the existing platforms, architectures and features, focusing on the last stage of HITLCPS: user feedback and actuation. To achieve this, we took advantage of the existing smartphone applications, with a special focus on ISABELA and BATINA, developed a Flask server which we named Pollux to handle users' requests and a Dialogflow agent as a NLP platform to understand users' interactions. Later we added another NLP platform: GPT-3 to expand the capabilities of understanding natural language of our implemented system.

ISABELA and BATINA serve as platforms for the development and testing of new concepts and features. Given their design, transferring the features developed on these platforms to other projects within our group, such as GreenBear and ON-BOARD, is a straightforward procedure.

4.2 Requirements

In this section, both the Functional and Non-Functional Requirements for the ChatBot are described. Table 4.1 displays each Functional Requirement (FR) and its corresponding title, which substantiate the designed architecture and the implemented features.

Table 4.1: Functional Requirements

FR	Title	FR	Title
FR0	Dialogflow	FR6	Exam Feedback
FR1	ChatBot Page	FR7	Poor Questionnaire Result Feedback
FR2	ChatBot Interface	FR8	Assignment Submission Reminder
FR3	Alarm Clock	FR9	GPT-3 Interactions
FR4	Canteen Menu Access	FR10	GPT-3 Setting
FR5	Exam Reminder		

To characterise each FR, we utilize the following parameters, described in Table 4.2: priority, description, actors, pre-conditions, event flow and expected result.

Table 4.2: Requirements' Parameters

Parameter	Description
Priority	Must: is essential to the project. Should: is important to the project but the system should work without it. Could: not implemented which don't affect the implementation of important requirements. Will not: not implemented and considered future work.
Description	Overall explanation.
Actors	Intervening systems.
Pre-Conditions	Requisites necessary for functionality.
Event Flow	Description of necessary actions to achieve the desired outcome.
Expected Outcome	The desired result.

4.2.1 Functional

Dialogflow

- **Priority:** Must.
- **Description:** the ChatBot must have Dialogflow has a NLP.
- **Actors:** the user.
- **Pre-Conditions:** none.
- **Event Flow:** the user sends a message.
- **Expected Outcome:** Dialogflow responds to the user message.

ChatBot Page

- **Priority:** Must.
- **Description:** the smartphone applications must have a ChatBot page which displays text interactions.
- **Actors:** the user.
- **Pre-Conditions:** Internet connection.
- **Event Flow:** the user presses de ChatBot page button from any navigation page and the application navigates to the ChatBot page.
- **Expected Outcome:** the ChatBot page loads up with the history of past messages, an entry text box where the user may write messages and a send message button.

ChatBot Interface

- **Priority:** Must.
- **Description:** the ChatBot interface must be able to send and retrieve messages.
- **Actors:** the user.
- **Pre-Conditions:** Internet connection.

4. ChatBot Development

- **Event Flow:** on the ChatBot page, the user writes a message and presses the send button.
- **Expected Outcome:** the message is sent and a few seconds later a response is received.

Alarm Clock

- **Priority:** Should.
- **Description:** the ChatBot should be able to handle an alarm clock request given a date, time and description.
- **Actors:** the user.
- **Pre-Conditions:** Internet connection.
- **Event Flow:** the user asks the ChatBot to set an alarm.
- **Expected Outcome:** the ChatBot sets an alarm for the given date, time and the description.

Canteen Menu Access

- **Priority:** Should.
- **Description:** the ChatBot should be able to provide the SASUC's canteen menu for any given day.
- **Actors:** the user.
- **Pre-Conditions:** Internet connection.
- **Event Flow:** the user asks the ChatBot for the menu of a given day.
- **Expected Outcome:** the ChatBot provides the corresponding menu.

Exam Reminder

- **Priority:** Should.
- **Description:** the ChatBot interface should remind a user to go to bed early on the evening before the exam.
- **Actors:** none.
- **Pre-Conditions:** Internet connection.
- **Event Flow:** when information is gathered from the user's calendar, if the user has an exam on the day after, the ChatBot sends a message to the user.
- **Expected Outcome:** the user receives a message with a reminder to go to bed early on the evening before an exam.

Exam Feedback

- **Priority:** Should.
- **Description:** the ChatBot interface should ask the user about their perception on their performance of a taken exam.
- **Actors:** the user.
- **Pre-Conditions:** Internet connection.
- **Event Flow:** when information is gathered from the user's calendar, if the user has had an exam on that day, the ChatBot sends a message to the user with a pop up question.
- **Expected Outcome:** the user receives a message with a pop up question about their exam performance and answers it in a scale of 1(very badly) to 5(very well).

Poor Questionnaire Result Feedback

- **Priority:** Should.
- **Description:** the ChatBot interface should warn the user about their poor result in a questionnaire.

4. ChatBot Development

- **Actors:** the user.
- **Pre-Conditions:** Internet connection.
- **Event Flow:** when information is gathered from the user's completed questionnaires, if the user has had a poor result, the ChatBot sends a message warning the student to study the recommended materials.
- **Expected Outcome:** the user receives a message with a warning about their poor questionnaire performance and a recommendation to study the support materials.

Assignment Submission Reminder

- **Priority:** Should.
- **Description:** every week, the ChatBot interface should remind a user about assignment submissions coming up in the following week.
- **Actors:** the user.
- **Pre-Conditions:** Internet connection.
- **Event Flow:** when information is gathered from the user's calendar, if the user has an assignment coming up during the following week, the ChatBot sends a message for each submission, reminding them about it.
- **Expected Outcome:** the user receives a message for each assignment submission, with a reminder about the deadline along with all the details about it.

GPT-3 Interactions

- **Priority:** Should.
- **Description:** the ChatBot should have GPT-3 integrated.
- **Actors:** none.
- **Pre-Conditions:** none.
- **Event Flow:** the user sends a message.

- **Expected Outcome:** GPT-3 responds to the user message.

GPT-3 Setting

- **Priority:** Should.
- **Description:** the user should be able to toggle ChatBot interactions with GPT-3.
- **Actors:** the user.
- **Pre-Conditions:** none.
- **Event Flow:** on the settings page, the user presses the GPT-3 interaction toggle.
- **Expected Outcome:** the GPT-3 interaction will be toggled on or off.

4.2.2 Non-Functional

- **Usability:** the ChatBot interface must have a pleasing and appealing UI to provide ease of use.
- **Support:** the ChatBot interface should be supported by the majority of iOS and Android devices.
- **Accessibility:** the system should be simple and straightforward.
- **Scalability:** the system should facilitate the development of new features and ChatBot interactions.

4.3 Architecture

4.3.1 Overview

This section provides an overview of the architecture, while the following sections describe each module portrayed in the diagram. Figure 4.1 depicts the system architecture for the ChatBot's development.

4. ChatBot Development

The UI refers to the smartphone application, which in this case are the ISABELA and BAT-INA applications, as mentioned previously.

The Pollux module is the Flask server developed to handle all communications between the UI and the remaining modules of the system. It forwards the users' messages to Dialogflow in order to be interpreted and handles the response accordingly. If Dialogflow's agent does not interpret the user input it interacts with GPT-3 as an alternative. Finally, it also manages the access to external APIs if needed.

Dialogflow's module has a programmable agent configured to interpret specific user interactions based on intent matching, maintains context during a conversation and provides appropriate responses.

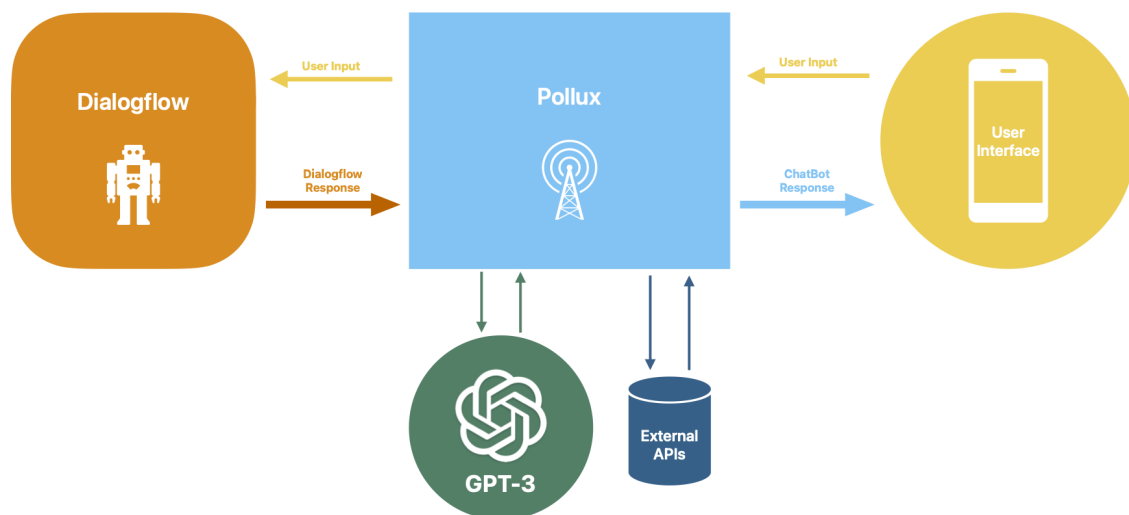


Figure 4.1: ChatBot Architecture

There are two methods to initiate a ChatBot interaction: a user message or a previously programmed message. The last type is used when a student's behaviour triggers an interaction that provides recommendations to improve academic performance.

4.3.2 UI

In the smartphone applications, we developed a ChatBot page to be interacted with, which is ran by the Operating System (OS) on the UI Thread, responsible for handling all user interactions. Subsequently, the OS also runs background services which manage data transfers and external communications. Figure 4.2 displays the relevant background services for

the developed ChatBot. The Calendar Service checks regularly for academic activities and deadlines, the Questionnaire Service checks the student's performance in completed questionnaires via the existing FIWARE database and the ChatBot Service handles all interactions with Pollux. Since the first two are invisible to the student, these services are the ones responsible for triggering a ChatBot interaction without text input from the student.

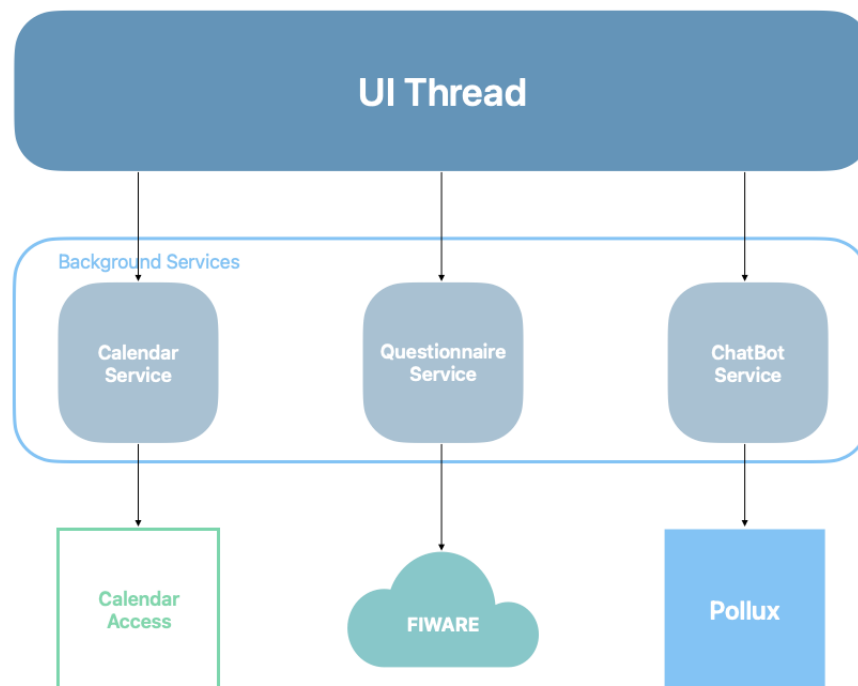


Figure 4.2: UI Architecture

4.3.3 Pollux

The Pollux server has a main thread that handles all the ChatBot's interactions. It is comprised of three main modules which control the flow of each interaction, depicted in Figure 4.3:

- **Dialogflow** module is responsible to forward the received message to the Dialogflow agent and await a response.
- **Intent Matching and Conversion** manages the Dialogflow's response, converts it into simpler types of data to work with and matches the intent with a specific function, subsequently, it also manages the interaction with the **GPT-3** and the **External APIs** modules.

4. ChatBot Development

- The **Response** module handles the response back to the source of the whole interaction.

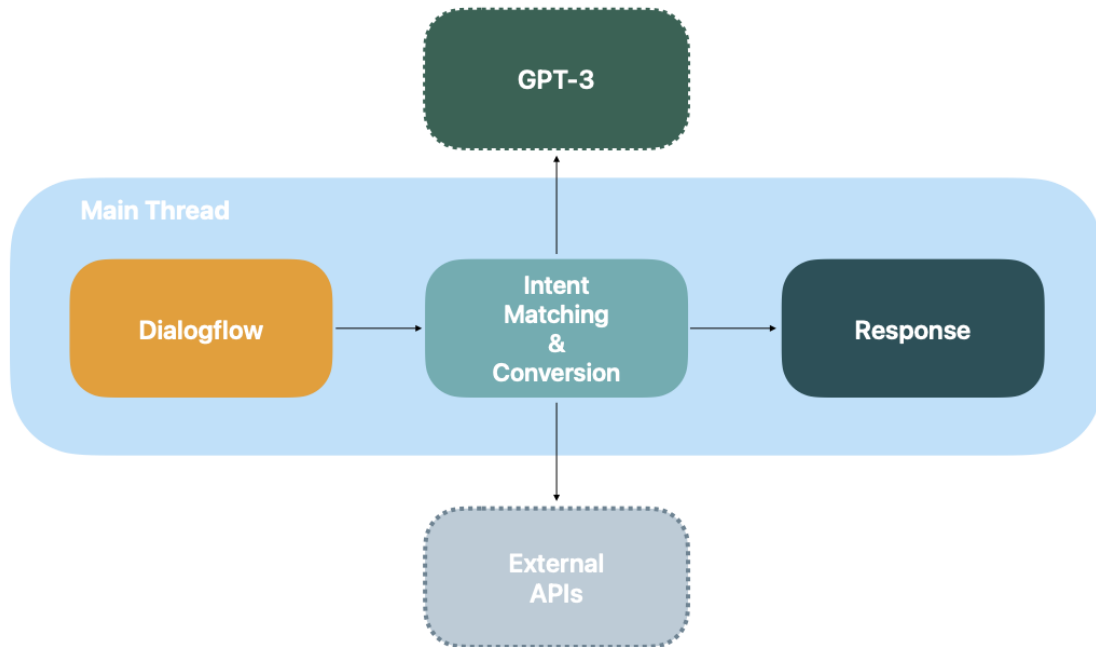


Figure 4.3: Pollux Architecture

The GPT-3 module handles the interaction with OpenAI's agent and the External APIs module manages interactions with external APIs. These modules operate only when the Dialogflow agent is unable to understand the request or when external data is required.

4.3.4 Dialogflow

Dialogflow's agent has an intent matching feature which matches the user interaction to a predefined intent. The intent matching is processed and a response is given back. Every intent has 3 modules:

- **Training Phrases** to know which type of inquiries might lead to which intent.
- **Action and Parameters** that allow for identification of types of entities like time, names, dates and locations.
- **Responses** so Dialogflow can choose one randomly.

The user's utterance, or input, is matched to an intent which has a previously programmed response. This behavior is illustrated in Figure 4.4

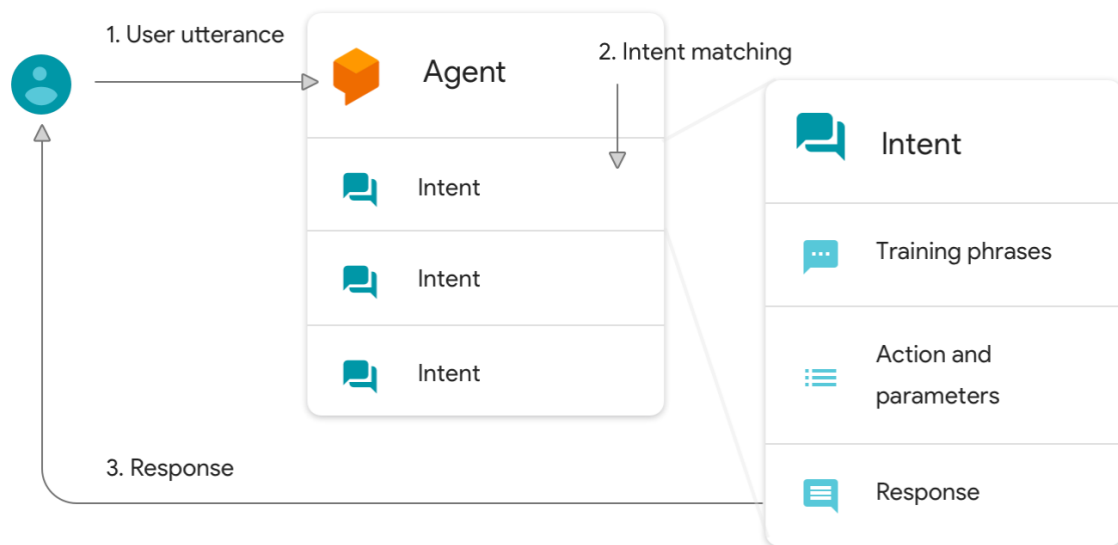


Figure 4.4: Dialogflow's Architecture [57]

4.4 Implemented Features

The ChatBot was developed having the student's in mind, therefore, the following features were designed to improve student's academic success and academic lifestyle. All presented features were implemented on ISABELA and BATINA and may later be adapted to every INESCC-DEEC's group platforms.

4.4.1 Define an alarm

The first step in the development was to program a simple task: making the ChatBot able to define a smartphone alarm given a time, a date and a description by the student. The alarm interaction was adapted from the ISABELA's ChatBot prototype to integrate all of the group's platforms. This is triggered by the student asking for the setting of an alarm, which the ChatBot fulfills if the all the parameters mentioned before are met, otherwise, it will ask for them until the student provides all of them. When successful, the ChatBot creates a smartphone alarm, this is illustrated in Figure 4.5.

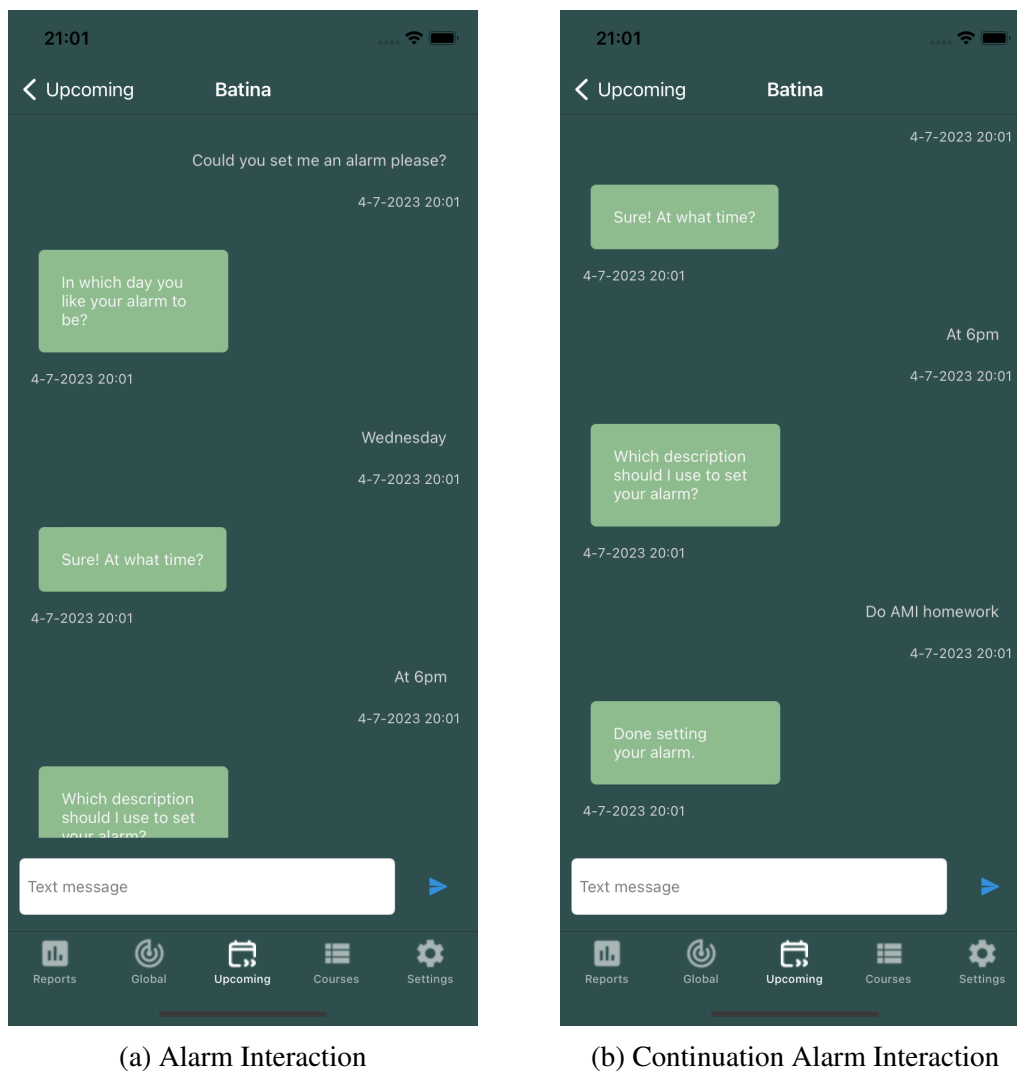


Figure 4.5: ChatBot Alarm Interaction

4.4.2 Canteen Menu

The canteen menu feature is triggered by the student and there are three distinct possibilities: to ask for the complete menu for the day, which includes lunch and dinner, to ask for the menu for one of the meals and to ask for the lowest calorie meal. Here, Pollux makes a HyperText Transfer Protocol (HTTP) request to an external API: the Serviços de Ação Social da Universidade de Coimbra (SASUC) menu API, to retrieve the specific data the student asked for. Figure 4.6 displays this interaction.

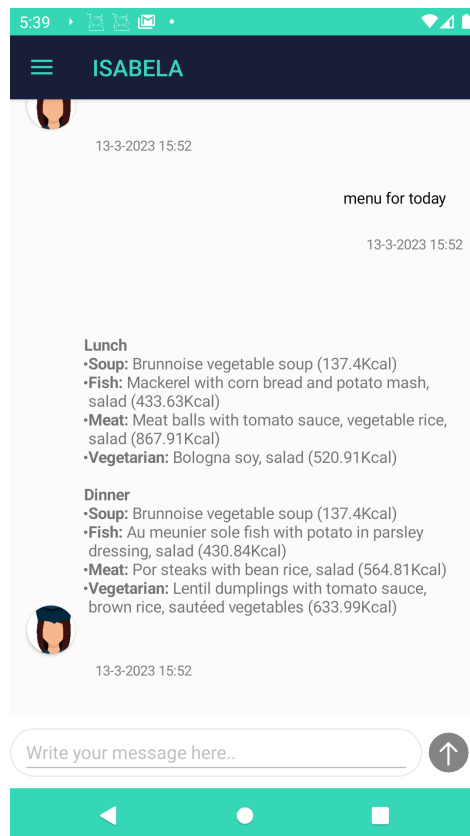


Figure 4.6: Canteen Menu Interaction

4.4.3 Historic Facts

This feature is also triggered by the student, who can ask for some historic facts. The implemented ones are about the UC, the Departamento de Engenharia Electrotécnica e de Computadores (DEEC) and the Faculdade de Ciências e Tecnologia da Universidade de Coimbra (FCTUC) and an example is given in Figure 4.7.

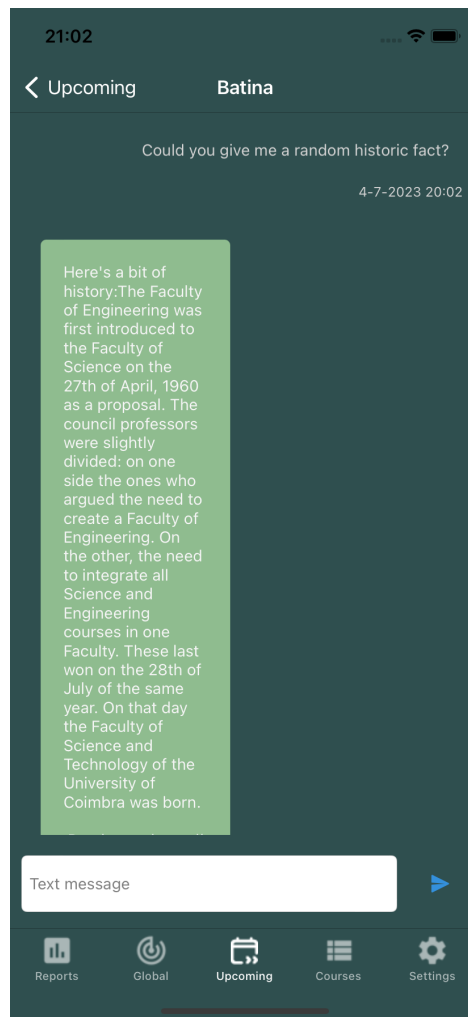


Figure 4.7: Historic Fact Interaction

4.4.4 Professor Contact Info

If the student would like to contact a professor, they may ask the ChatBot about the professor's contact info. The ChatBot then retrieves the asked data and provides it to the student.

Figure 4.8 illustrates this interaction.

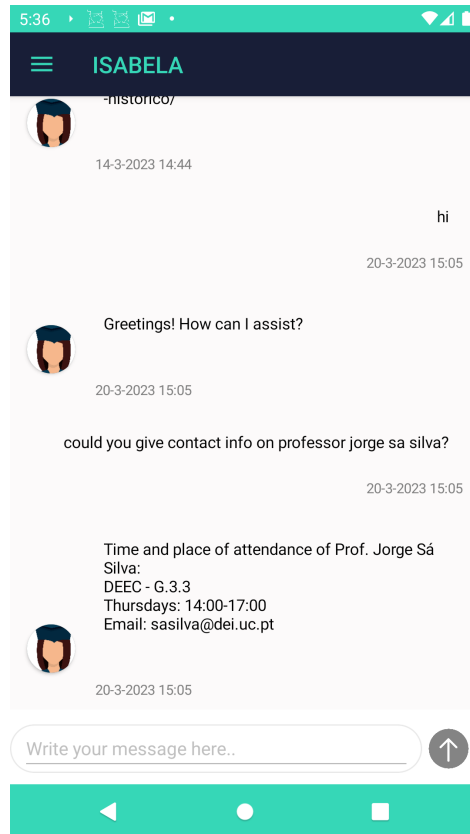


Figure 4.8: Professor's Info Interaction

4.4.5 Exam Reminder

This feature is triggered by a Calendar Service and does not require student interactions. The Calendar Service checks if the student has an exam in the following day and, if so, reminds the student to go to bed early and get a nightful of sleep, as shown in Figure 4.9.

4.4.6 Exam Feedback

This feature is also triggered by the Calendar Service that checks if the student had an exam during the day and if so, sends a message asking how well did it go. The message contains

4. ChatBot Development

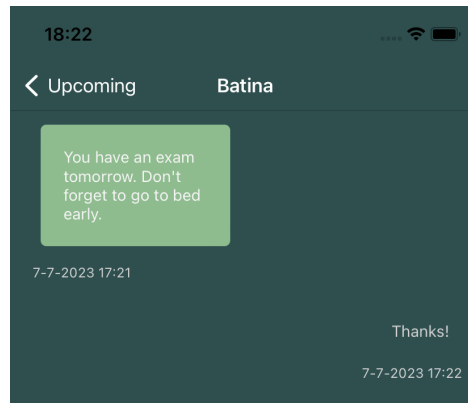
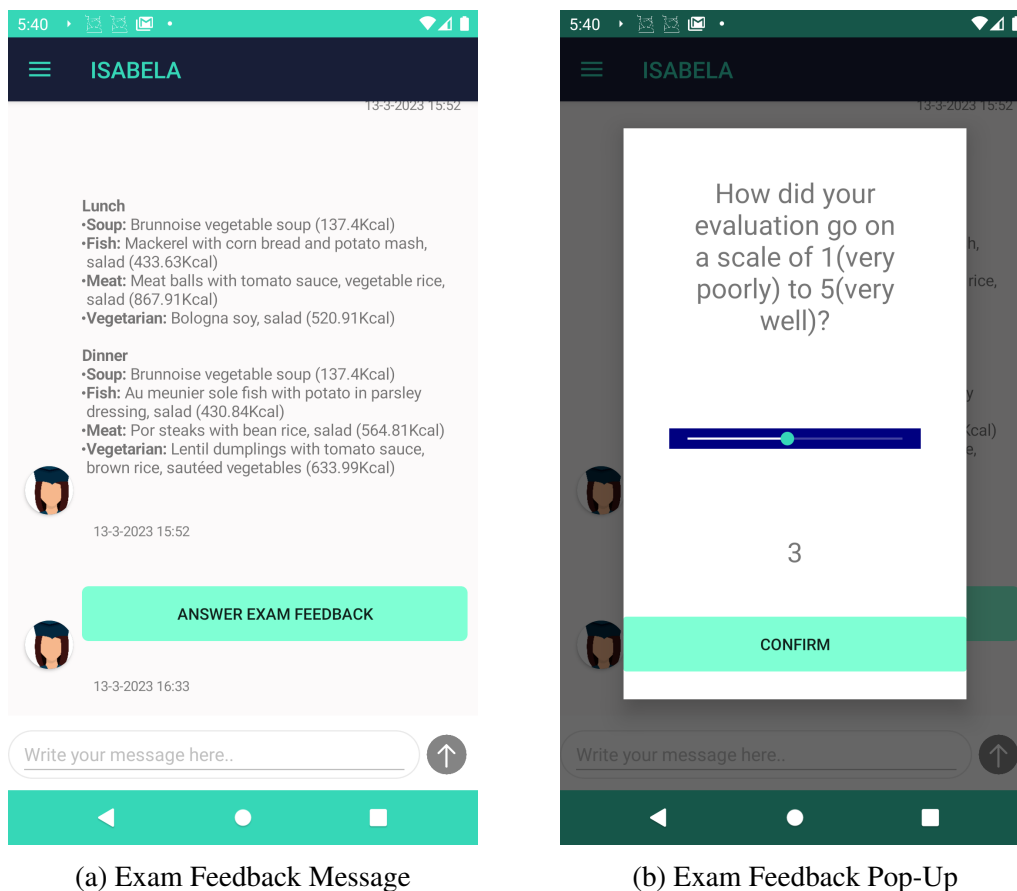


Figure 4.9: Exam Reminder Interaction

a button that shows a pop-up page in which the student can answer on a scale from 1(very bad) to 5(very good). This data is then stored in the database and it can be important to infer the student's perception, later on, comparing the student's expectation with the score of the exam. This interaction is depicted in Figure 4.10.



(a) Exam Feedback Message

(b) Exam Feedback Pop-Up

Figure 4.10: Exam Feedback Interaction

4.4.7 Assignment Submission Reminder

This is another feature developed as the previous one, only this time the Assignments Submission Reminder checks once per week if the student has any assignment's deadlines coming up and if so it triggers the message displayed in Figure 4.11

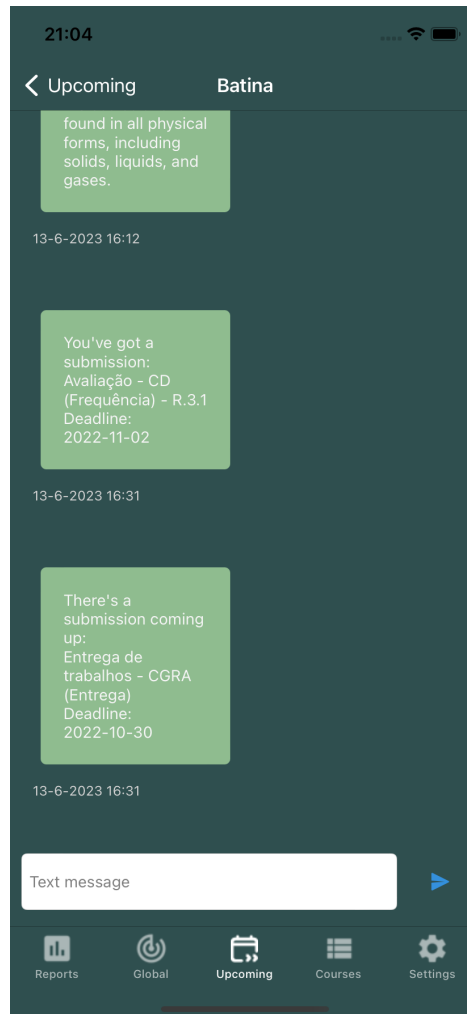


Figure 4.11: Assignment Submission Reminder Interaction

4.4.8 Poor Questionnaire Result Feedback

This feature is triggered by a background service, which runs once a week, that checks if the student had any poor questionnaire results. If so, the ChatBot sends a message warning the student to study the provided support materials, as seen in Figure 4.12.



Figure 4.12: Poor Questionnaire Result Interaction

4.4.9 GPT-3 Interaction

This was the latest feature to be developed, as the technology that powers it is very recent. The GPT-3 interaction is triggered by the student and it has the full capabilities of the GPT-3 language model. It is an optional feature that may be toggled on or off in the settings page. Figure 4.13a refers to the case when the interaction setting is on and the ChatBot answers correctly to the question "What's the capital of Canada?". In Figure 4.13b, however, the interaction setting is off and the ChatBot does not understand the question.

At the time of this implementation, the only concern with this interaction is that it cannot maintain context. Due to time constraints, it was not possible to develop context awareness in this interaction, as doing so would require a substantial amount of time and effort.

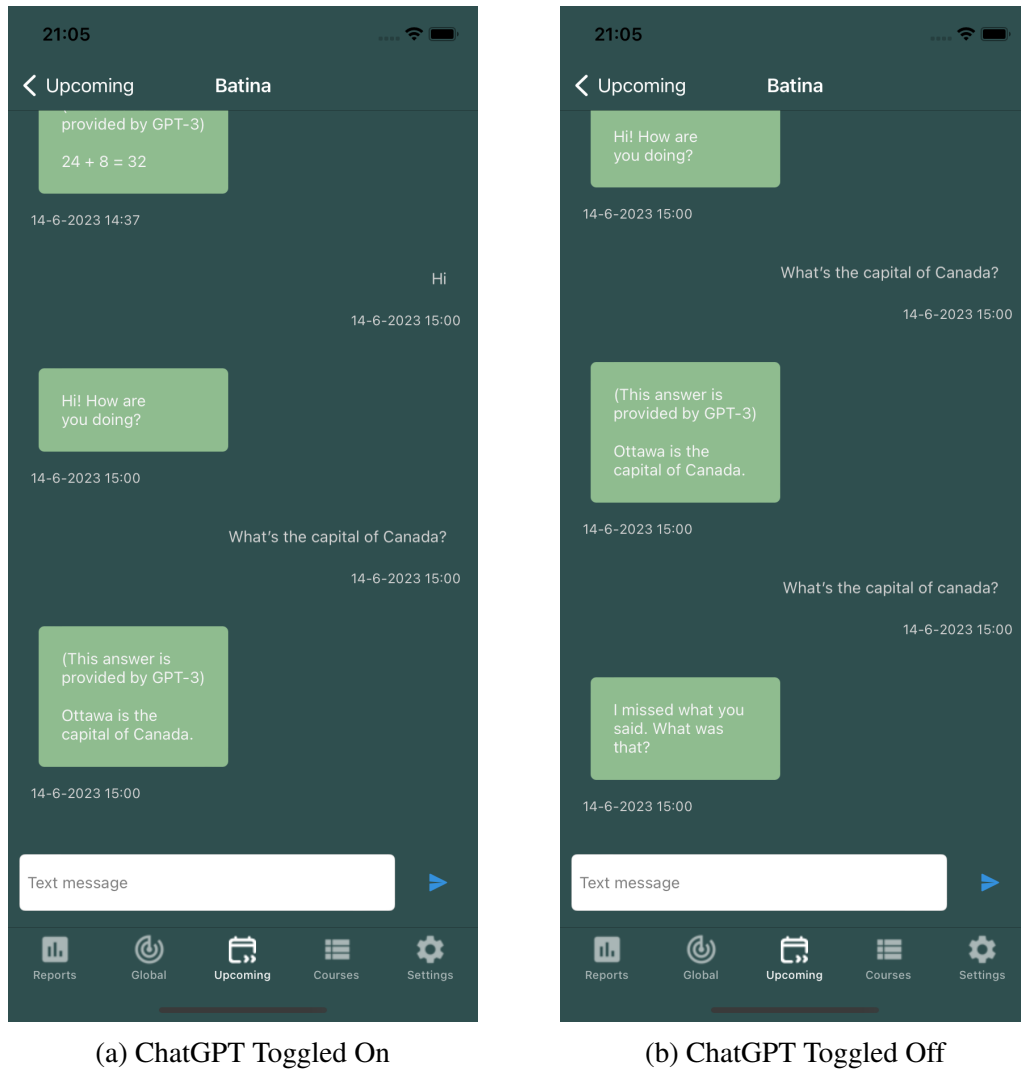


Figure 4.13: GPT-3 Toggled On and Off Interactions

4.5 Implementation

4.5.1 Dialogflow configuration

To implement Dialogflow, we had to create a Dialogflow agent. This was achieved through Dialogflow's Cloud Console.

In order to use the agent, there was a need to configure a remote access account for remote interactions. We created a service account with a specific access key, which can be used to authorize the access to our created agent. The Google Cloud Console provides this generated key on a JSON format file to later be used by the Pollux server.

We then configured a list of intents, as illustrated in Figure 4.14, to formulate responses

4. ChatBot Development

to specific student interactions. As mentioned previously, these intents enable the agent to interpret student queries and provide an appropriate response. We created two types of intents:

- **Visible intents** are intents that are tailored to respond directly to student queries each time a student interacts with the agent. To facilitate intent matching, we provide illustrative phrases that simulate student messages for each intent. We called these "visible" because these intents are triggered by a message which is visible to the student.
- **Invisible intents** are intents that are not triggered by a student interaction, but function identically. In contrast to the prior ones, these are configured with a predefined message. Each of these intents is only activated by a unique message. We called these "invisible" because the trigger is a message invisible to the student.

Here is the full list of intents and the description for each one:

- **Bad Answer Report** is the intent matched when the Poor Questionnaire Result feedback is triggered. This is an Invisible intent.
- **Calendar Exam Feedback** is matched when the student has an exam. This is an Invisible intent.
- **Calendar Exam Reminder Tomorrow** is matched to remind the student to go to bed early on the day before an exam. This is an Invisible intent
- **Calendar Infor Submission** is the intent related to remind the student about assignments submissions. This is also an Invisible intent.
- **Canteen Menu** is the intent matched when the student asks for informations about the menu on the canteens on any given day. This is a Visible intent.
- **Canteen Menu Lowest Calories Meal** is matched when the student asks for the lowest calories dish on any meal's menu. This is a Visible intent.
- **Define An Alarm** is matched when the student asks for an alarm setting. This is a Visible intent.

- **Professor Contact Info** is matched when the student asks for the contact details of a professor. This is a Visible intent.
- **UC History Facts** is matched when the student asks for any random fact or historic facts. This is a Visible intent.

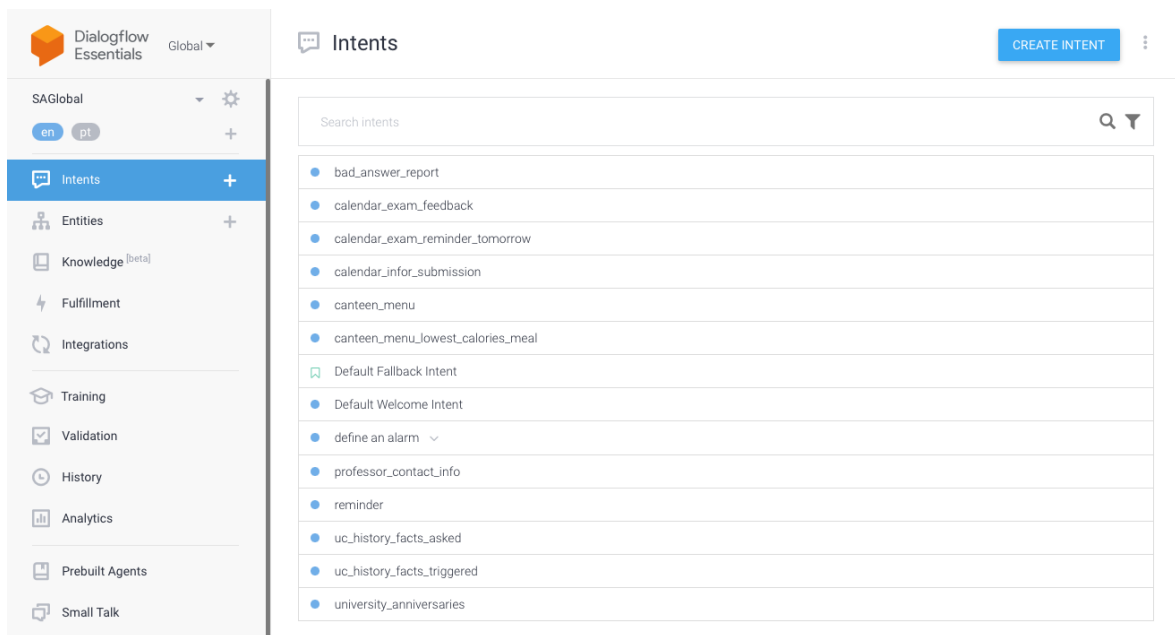


Figure 4.14: Dialogflow Intent List

We also created an entity to deal with the canteen menu interaction: `meals_time`. This is a custom entity used by Dialogflow's agent to identify if the student referred a meal time (lunch or dinner) when asking for the menu. This way we can provide the specific meal's menu if the student asks for it.

4.5.2 GPT-3 configuration

For interactions with GPT-3, using an OpenAI account, we created an API key to give Pollux access to the pre-defined OpenAI's language models. The chosen one was GPT-3 as it was the latest model by the time this implementation took place and it had more features than any of the previous ones.

4.5.3 Pollux configuration

We configured the Pollux server using Flask, as mentioned before. We configured it with only one route which can be accessed through a simple HTTP Post request. We also had to use the *google.cloud* library to manage interactions with Dialogflow and the *openai* library to interact with GPT-3. We included the access key files from Google Cloud and OpenAI in our project.

Every time a request is received, we convert its body to a JSON object in order to better manage its data. The newly converted body is then submitted to a method that deals with Dialogflow's agent. This method firstly interacts with Dialogflow to process the student query and then receives a response with the following fields:

queryResult

- **queryText:** it is the student's query message
- **action:** contains the action to take, if necessary
- **parameters:** may contain pre-defined parameters like date and time
- **fulfillmentText:** is the response given by the Dialogflow's agent
- **intent:** contains the name of the matched intent
- **intentDetectionConfidence:** is a value between 0 and 1 which represents the intent detect confidence
- **languageCode:** is the language in which the interaction is written
- **sentimentAnalysis:** contains the score and magnitude of the student message. The score ranges from -1 to 1 and magnitude from 0 to 1.

The next step is to understand which intent was the matching one, so for each intent there are different tasks and specific code to be run. The different categories are described in Table 4.3. The rest of the intents do not need additional interactions, therefore, are forwarded to the student with Dialogflow's response without Pollux intervention.

Table 4.3: Intent recognition and subsequent behavior in Pollux

Name	Description
canteen menu	Makes a HTTP request to the SASUC's menu API. Scan the response for the appropriate day. If a meal time was specified, selects the corresponding one. If not, picks both. Returns the menu.
canteen menu lowest calories meal	Same the previous one but chooses the lowest calories dish on the menu. Returns the specific dish.
calendar infor sub-mission	Gets the submission title, date and deadline from the body of the request. Returns a response with submission data.
history facts	Picks one historic fact from a static structure within the program. Returns a random fact.
professor contact info	Chooses the matching professor from a static structure within the program. If the name provided exists, returns the appropriate informations. Returns a warning message indicating that it did not recognise the professor's name otherwise.
Unknown	If the intent was not matched, then it interacts with GPT-3 language model. Makes a request to the OpenAI's API with the student query and waits for a response. Returns GPT-3's response with a note informing that response was generated using GPT-3.

4.5.4 Smartphone application

In this section we explore the implementation on the smartphone applications. Since both ISABELA and BATINA were the focus of this project and since the implementations were the same in both applications, the descriptions in this section will only reference BATINA, for simplification and to avoid the likelihood of repetition.

The BATINA application is already functional and has a specific architecture, activities and background services. The ChatBot features were implemented over the existing platform, taking advantage of some of its behaviors.

An essential component of the ChatBot is the ChatBot page, designed to interact with the student and to be interacted with. This page was developed using XAML for the UI elements and C# to program their behavior, which are dynamically changing its content, due to message exchanging between the ChatBot and the student. We projected it to be like a text messaging service, so we designed the text bubbles with XAML elements that show the intended text and the date and time of the message. Since this structure is always the same,

4. ChatBot Development

we created a C# class with objects to represent each component of each message. This way, layout is always the same but the content changes dynamically with each interaction. There is an important parameter in this class which allows visibility of the message, as there are messages that we don't want the student to see. These types of messages are sent by background services when a certain behavior is detected. Table 4.4 displays which message is sent, the service that sends it and the corresponding intent in Dialogflow.

Table 4.4: Background services' messages and matching Dialogflow's Intents

Background Service	Pre-defined Message	Matching Dialogflow Intent
	calendar_exam_feedback	calendar exam feedback
Calendar Service	calendar_inforestudante_submission	calendar inforestudante submission
	calendar_exam_reminder_tomorrow	calendar exam reminder tomorrow
Questionnaire Service	bad_answer_report_result	bad answer report

Every message sent by the student is encapsulated in a JSON payload which has the following structure:

- **texts:** this field contains a text message written by the student or pre-defined messages written by the background services. In case of the latter, this field can have more arguments with data for Pollux.
- **session_id:** this field is normally the internal studentID of the application. This is very important to maintain context during a conversation.
- **language_code:** this refers to the language being used by the student. It is set to the language of the application, which might be Portuguese or English.
- **gpt_interaction:** this field contains the boolean value of the GPT-3 interaction setting. It is *true* if the setting is on and *false* otherwise.

This payload is attached to the body of a REST POST request to the configured Pollux address and port. After performing the request, the application waits for the Pollux response with the previously mentioned structure, in section 4.5.3, and displays the contents of the *fulfillmentText* field to a new message in the ChatBot page.

4.6 Tests

In this section we provide some functional tests and tables 4.5 to 4.14 describe examples of the performed tests to evaluate student interactions. If Dialogflow has a matching intent, then it provides the answer. If not, and the intent is classified as *unknown*, GPT-3 will provide it, as shown in tables 4.11 to 4.14.

Table 4.5: Alarm Test

Matched Intent: define an alarm	
Content of Message	Response
Can you define me an alarm?	In which day would you like your alarm to be?
Next Friday	Sure! At what time?
5pm	Which description should I use to set your alarm?
Read a book	Done setting your alarm.

Table 4.6: Canteen Menu with Meals Time Test

Matched Intent: canteen menu + meals time	
Content of Message	Response
	Here's the menu for 2023-07-08 lunch:
	Soup: Vegetable soup (127.95Kcal)
What's for lunch tomorrow?	Fish: Spanish style hake with rice, salad (678.44Kcal)
	Meat: Thigh roasted chicken w/rice vegetables, salad (605.57Kcal)
	Vegetarian: Bologna soy, salad (520.91Kcal)

Table 4.7: Canteen Menu Test

Matched Intent: canteen menu	
Content of Message	Response
What's the menu for today?	Here's the menu for 2023-07-07:
	<p>Lunch</p> <p>Soup: Potato and onion broth with kale strips (134.14Kcal) Fish: Baked sole with lemon, roast potato, salad (380.11Kcal) Meat: Spaghetti Bolognese (pork) (744.29Kcal) Vegetarian: Seitan stew with rice (940.31Kcal)</p> <p>Dinner</p> <p>Soup: Potato and onion broth with kale strips (134.14Kcal) Fish: Hors d'oeuvres with russian salad, salad (411.92Kcal) Meat: Chichen curry (977.33Kcal) Vegetarian: Tofu burger with crisp, lettuce and tomato salad (419.85Kcal)</p>

Table 4.8: Canteen Menu Lowest Kcal Meal Test

Matched Intent: canteen menu lowest calories meal	
Content of Message	Response
What's the lowest calories meal for today's lunch?	Here's the lowest Kcal meal on the menu: Fish: Baked sole with lemon, roast potato, salad (380.11 Kcal)

Table 4.9: Professor Contact Info Test

Matched Intent: professor contact info	
Content of Message	Response
Could you give me the contact info of professor Jorge Sá Silva?	Time and place of attendance of Prof. Jorge Sá Silva: DEEC - G.3.3 Thursdays: 14:00-17:00 Email: sasilva@deec.uc.pt

Table 4.10: History Facts Test

Matched Intent: uc history facts	
Content of Message	Response
Could you tell me a random fact?	Here's a bit of history: By signing "Scientiae Thesaurus Mirabilis", King Dinis created the oldest university of the country and one of the oldest in the world. Dated from 1290, the document originated the General Study, recognised in the same year by Pope Nicholas IV. A century after the birth of the nation, the University of Coimbra germinated. It starts operating in Lisbon but, in 1308, it is transferred to Coimbra, alternating between the two cities until 1537, when it is definitely settled in the city of river Mondego.

Table 4.11: Calculations Test

Matched Intent: Unknown	
Content of Message	Response
How much is 2+2?	(This answer is provided by GPT-3) 2+2 is 4.

Table 4.12: Random Characters Test

Matched Intent: Unknown	
Content of Message	Response
jaksdfkldjsafgkdjslagfhads	(This answer is provided by GPT-3) This is not a valid sentence.

Table 4.13: Random Question Test

Matched Intent: Unknown	
Content of Message	Response
What's the capital of Brasil?	(This answer is provided by GPT-3) Brasilia.

Table 4.14: Random Question Test 2

Matched Intent: Unknown	
Content of Message	Response
	(This answer is provided by GPT-3)
	That depends on the purpose of the ChatBot.
What's the best question to ask a ChatBot?	If the ChatBot is designed to provide information, a good question to ask might be \ "What can you tell me about [topic]?" If the ChatBot is designed to provide customer service, a good question to ask might be \ "How can I [task]?"

Chapter 5

Conclusion and Future Work

5.1 Conclusion

With this project, we aimed at building a ChatBot to integrate the ISABELA and BATINA learning platforms, based on the HITLCPS concept, as well as lay a groundwork for human interactions with CPS. We explored the concepts of IoT, AI and NLP and evaluated how they could be implemented to enhance these systems and produce more human-centric applications and interactions.

We developed a ChatBot architecture using Dialogflow as a NLP tool to interpret users' queries, a Flask server to manage ChatBot interactions with the system and the INESCC-DEEC's HITL platforms to interact with users. All of the developed features were successfully implemented in the ISABELA and BATINA platforms.

The development of HITLCPS in IoT architectures is a recent concept and not many works have been done in this area. This project aimed to contribute to further research and development in these areas by providing a solution with detailed implementations and a solid architecture. Although we believe this project's objectives were fulfilled, we also believe there is more work that can be done to improve this solution.

Integrating OpenAI's GPT-3 contributed to our project's potential and processing power. However, there was a desire for further exploration, which is detailed in the next section.

We concluded that, in order to improve the ChatBot's intent matching and NLP, there are some field tests that need to be done to provide insights on how impactful the ChatBot is in

everyday academic life, as well as to fine-tune Dialogflow and GPT-3 operations and rules.

5.2 Future Work

In this section, we mention some of the features that could be improved in future works in this field.

As future work we need to expand the intents in Dialogflow, adding new functionalities and better context about academic issues and lifestyle.

As stated previously, we felt the need to further explore GPT-3 and GPT-4 functionalities. This could be improved by training these language models with tailored datasets, in order to develop a better ChatBot adapted to students' academic lifestyle and interactions. Even though creating a dataset like this requires a large study of students' interactions with technology and CPS, this would hugely improve the familiarity between students and the ChatBot.

For the coming months, we expect to improve this project with the ON-BOARD initiative, which aims to integrate this system into official academic tools from the UC and implement the ChatBot as digital tutor. We believe this integration will provide new insights about the next functionalities to implement in this system.

In conclusion, these challenges offer a great opportunity for future research work and contribute to the existing partnerships' projects with the UC, the FMUC and the Coimbra City Hall.

Bibliography

- [1] Scrum Alliance. Scrum. <https://www.scrumalliance.org/about-scrum>. Accessed: 2023-07-08.
- [2] GitLab. Gitlab. <https://about.gitlab.com>. Accessed: 2023-07-08.
- [3] Slack. Slack. <https://slack.com>. Accessed: 2023-07-08.
- [4] Skype. Skype. <https://www.skype.com/en/>. Accessed: 2023-07-08.
- [5] JetBrains. JetBrains. <https://www.jetbrains.com>, . Accessed: 2023-07-08.
- [6] JetBrains. Rider. <https://www.jetbrains.com/rider/>, . Accessed: 2023-07-08.
- [7] JetBrains. Jintellij idea. <https://www.jetbrains.com/idea/>, . Accessed: 2023-07-08.
- [8] Kevin Ashton. In the real world, things matter more than ideas. 1999.
- [9] Ericsson. Ericsson mobility report november 2022, 2022.
- [10] Lionel Sujay Vailshery (Statista). Number of internet of things (iot) connected devices worldwide from 2019 to 2021. <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>, 2021. Accessed: 2023-06-16.
- [11] Ericsson. Number of smartphone mobile network subscriptions worldwide from 2016 to 2022. <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>, 2023. Accessed: 2023-06-19.
- [12] Population Reference Bureau. Population mid-2022. <https://www.prb.org/international/indicator/population/table>, 2022. Accessed: 2023-06-19.

Bibliography

- [13] Josh Howarth. Time spent using smartphones. <https://explodingtopics.com/blog/smartphone-usage-stats>, 2023. Accessed: 2023-06-19.
- [14] Ericsson. Mobile data traffic in western europe from 2011 to 2027 (in exabytes per month). <https://www.statista.com/statistics/1133848/western-europe-monthly-data-traffic/>, 2022. Accessed: 2023-06-19.
- [15] John Mccarthy. What is artificial intelligence?, 2007. URL <http://www-formal.stanford.edu/jmc/>.
- [16] Stanford University. Professor john mccarthy. <http://jmc.stanford.edu>. Accessed: 2023-06-27.
- [17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
- [18] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.
- [19] Bo Zhang, Jun Zhu, and Hang Su. Toward the third generation artificial intelligence. *Science China Information Sciences*, 66(2):1–19, 2023.
- [20] Imran Ahmed, Gwanggil Jeon, and Francesco Piccialli. From artificial intelligence to explainable artificial intelligence in industry 4.0: a survey on what, how, and where. *IEEE Transactions on Industrial Informatics*, 18(8):5031–5042, 2022.
- [21] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, jan 1966. ISSN 0001-0782. doi: 10.1145/365153.365168.
- [22] Chatbots.org. Parry. <https://www.chatbots.org/chatbot/parry/>. Accessed: 2023-06-21.
- [23] Bayan Abu Shawar and Eric Atwell. Alice chatbot: Trials and outputs. *Computacion y Sistemas*, 19:625–632, 2015. ISSN 14055546. doi: 10.13053/CyS-19-4-2326.

- [24] Richard S. Wallace, Robert Epstein, Gary Roberts, and Grace Beber. *The Anatomy of A.L.I.C.E.*, pages 181–210. Springer Science+Business Media, London, 2009. ISBN 978-1-4020-6710-5. doi: 10.1007/978-1-4020-6710-5_13.
- [25] Li Zhou, Jianfeng Gao, Di Li, and Heung Yeung Shum. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46:53–93, 2020. ISSN 15309312. doi: 10.1162/COLI_a.00368.
- [26] Heung yeung Shum, Xiao dong He, and Di Li. From eliza to xiaoice: challenges and opportunities with social chatbots, 1 2018. ISSN 20959230.
- [27] José Marcelo da Silva Lopes Fernandes, Jorge Sá Silva, and Fernando Boavida. Isabela iot student advisor and best lifestyle analyzer.
- [28] David Nunes, Jorge Sa Silva, and Fernando Boavida. *A Practical Introduction to Human-in-the-Loop Cyber-Physical Systems*. IEEE Press, 2018.
- [29] David Sousa Sousa Nunes, Pei Zhang, and Jorge Sa Silva. A survey on human-in-the-loop applications towards an internet of all. *IEEE Communications Surveys and Tutorials*, 17:944–965, 4 2015. ISSN 1553877X. doi: 10.1109/COMST.2015.2398816.
- [30] Microsoft Corporation. Xamarin. <https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin/>, . Accessed: 2023-06-27.
- [31] Microsoft Corporation. Xamarin.essentials. <https://learn.microsoft.com/en-us/xamarin/essentials/>, . Accessed: 2023-06-27.
- [32] Microsoft Corporation. Xamarin.forms. <https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms/>, . Accessed: 2023-06-27.
- [33] FIWARE Foundation. Fiware. <https://www.fiware.org/about-us/>, . Accessed: 2023-06-28.
- [34] FIWARE Foundation. Fiware. <https://www.fiware.org/catalogue/#core>, . Accessed: 2023-06-28.
- [35] FIWARE Foundation. Fiware-orion. <https://fiware-orion.readthedocs.io/en/master/>, . Accessed: 2023-06-27.

Bibliography

- [36] Pallets. Flask. <https://flask.palletsprojects.com/en/2.3.x/>. Accessed: 2023-06-27.
- [37] Google. Dialogflow. <https://cloud.google.com/dialogflow/docs>. Accessed: 2023-06-27.
- [38] OpenAI. Gpt - openai api. <https://platform.openai.com/docs/guides/gpt>. Accessed: 2023-06-27.
- [39] Airbnb. Lottie docs. <https://github.com/airbnb/lottie>, . Accessed: 2023-07-05.
- [40] Hernan Torrisi. Bodymovin. <https://exchange.adobe.com/apps/cc/12557/bodymovin>. Accessed: 2023-07-05.
- [41] Airbnb. Lottie github. <https://github.com/airbnb/lottie>, . Accessed: 2023-07-05.
- [42] Michael C. Ashton Kibeom Lee. The hexaco personality inventory. <https://hexaco.org>. Accessed: 2023-07-08.
- [43] Oxford Reference. Profile of mood states. <https://www.oxfordreference.com/display/10.1093/oi/authority.20110803100348404>. Accessed: 2023-07-08.
- [44] Plotly. Dash. <https://dash.plotly.com>. Accessed: 2023-06-29.
- [45] UC Frameworks. Uc teacher. <https://www.uc.pt/ucframework/apps/ucteacher/>, . Accessed: 2023-06-30.
- [46] UC Frameworks. Uc student. <https://www.uc.pt/ucframework/apps/ucstudent/>, . Accessed: 2023-06-30.
- [47] Ana Bartolomeu and Karine Paniza. On-board: na luta contra o abandono escolar. *Notícias UC*. URL <https://noticias.uc.pt/artigos/on-board-na-luta-contr-o-abandono-escolar/>.
- [48] João Pedro Campos. Meio milhão de euros para combater abandono escolar na universidade de coimbra. *Jornal de Notícias*. URL <https://www.jn.pt/local/noticias/coimbra/coimbra/>

meio-milhao-de-euros-para-combater-abandono-escolar-na-universidade-de-coimbra.html/.

- [49] Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T. Campbell. Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. pages 3–14. Association for Computing Machinery, Inc, 2014. ISBN 9781450329682. doi: 10.1145/2632048.2632054.
- [50] University of Dartmouth. Studentlife study. <http://studentlife.cs.dartmouth.edu>. Accessed: 2023-06-29.
- [51] Big Blue Button. Big blue button. <https://bigbluebutton.org>, . Accessed: 2023-06-29.
- [52] Big Blue Button. Teacher features. <https://bigbluebutton.org/teachers/features/>, . Accessed: 2023-06-29.
- [53] ALEKS Corporation. Aleks. <https://www.aleks.com>, . Accessed: 2023-06-29.
- [54] ALEKS Corporation. Knowledge space theory. https://www.aleks.com/about_aleks/knowledge_space_theory, . Accessed: 2023-06-29.
- [55] Scotty D Craig, Xiangen Hu, Arthur C Graesser, Anna E Bargagliotti, Allan Sterbinsky, Kyle R Cheney, and Theresa Okwumabua. The impact of a technology-based mathematics after-school program using aleks on student’s knowledge and behaviors. *Computers & Education*, 68:495–504, 2013.
- [56] Socrative. Socrative. <https://www.socrative.com>. Accessed: 2023-06-29.
- [57] Priyanka Vergadia. Deconstructing chatbots: Getting started with dialogflow. <https://medium.com/google-cloud/deconstructing-chatbots-getting-started-with-dialogflow-4f91deb32135>. Accessed: 2023-07-05.

Appendix A

Introduction Tutorial to ChatBot Technologies

This document introduces ChatBot technologies and lays the groundwork for our implementation. It serves as an instructional introduction to the three distinctive frameworks of this project: Dialogflow, Flask, and Xamarin. This document is extremely important for the INESCC-DEEC research group, as it explains in great detail how to implement and utilize ChatBots on the group's platforms.



Dialogflow and Xamarin Introduction

2023/2024

1. Introduction

The objective of this assignment is to introduce you to the world of conversational AI frameworks and the tools required to develop a chatbot. You will be introduced to the concept of Natural Language Processing (NLP) and its importance in building intelligent chatbots that can communicate with users in a natural and intuitive way. In addition, you will gain an understanding of the role played by a managing server in managing and maintaining the chatbot.

This assignment will also explore the importance of a user interface (UI) in designing an engaging and user-friendly chatbot. You will learn how to create a UI that enables users to interact with the chatbot seamlessly, and how to incorporate design elements that promote user engagement and satisfaction.

By the end of the assignment, you will have a solid understanding of the key components required to build a chatbot, including NLP, managing servers, and user interfaces. You will also have the opportunity to put their skills into practice by developing your own chatbot using a platform such as Dialogflow, which offers a range of tools and features for building conversational AI applications.

Dialogflow is a cloud-based conversational AI platform that enables developers to design and integrate intelligent chatbots and virtual assistants into their applications. With Dialogflow, developers can create natural language processing (NLP) models that can understand user inputs and generate appropriate responses. (Read more at <https://cloud.google.com/dialogflow/es/docs>)

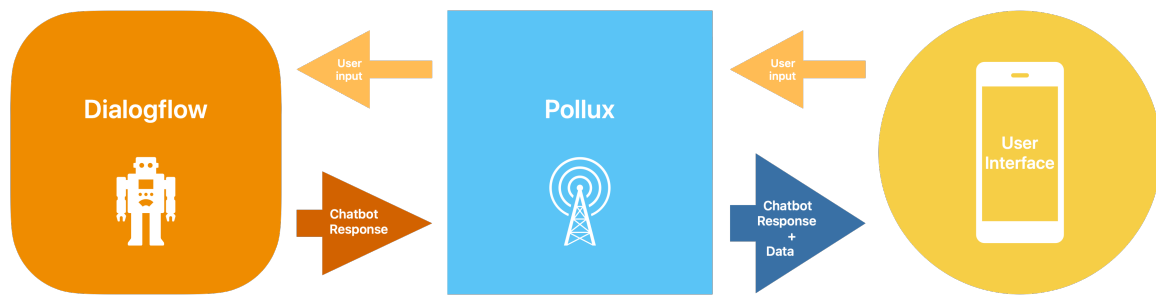


Fig. 1 - Simple architecture diagram

Fig. 1 displays the architecture of the system you are going to develop. On one side (yellow) you have the User Interface (UI) which is going to be a smartphone app developed using Xamarin framework. On the other side (orange) you have Dialogflow - Google's framework to develop chatbots. To establish a connection between the two you are going to develop a server using the Flask framework, and run it locally (on your machine) to manage all the interactions between the user and Dialogflow.

2. Requirements

2.1. Visual Studio

In order to develop a Xamarin smartphone app you need the Visual Studio IDE. You can download it here: <https://visualstudio.microsoft.com/>

Note: Make sure you install Visual Studio and not Visual Studio Code.

Follow the instructions for installation here:

- Windows: <https://learn.microsoft.com/en-us/xamarin/get-started/installation/?pivots=windows-vs2022>

- MacOS: <https://learn.microsoft.com/en-us/xamarin/get-started/installation/?pivots=macos-vs2022>

The installation process may take a while. While it is running, head over to the next step.

2.2. PyCharm

To be able to develop and run your server (Pollux) you need to install python and flask. To make it simple you should use JetBrains IDE for python - PyCharm. You can download it here: <https://www.jetbrains.com/pycharm/>.

Note: if you don't have python installed on your machine, download it and install it from <https://www.python.org/downloads/>. Note 2: Make sure you download the **Community Edition** if you don't have a JetBrains license.

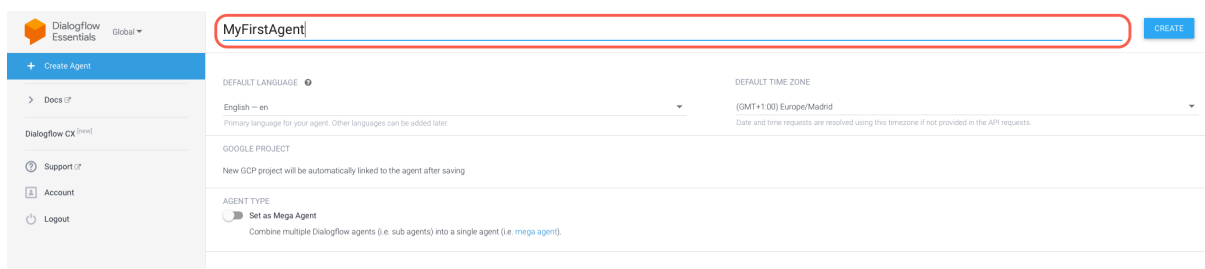
Install PyCharm on your machine. Open it up and create a new Python project.

2.3. Dialogflow

In order to use Dialogflow you need a Google account. If you don't have one, you should create one now. When you are done, head over to <https://dialogflow.cloud.google.com/#/login>. Once you are logged in you need to create an agent.

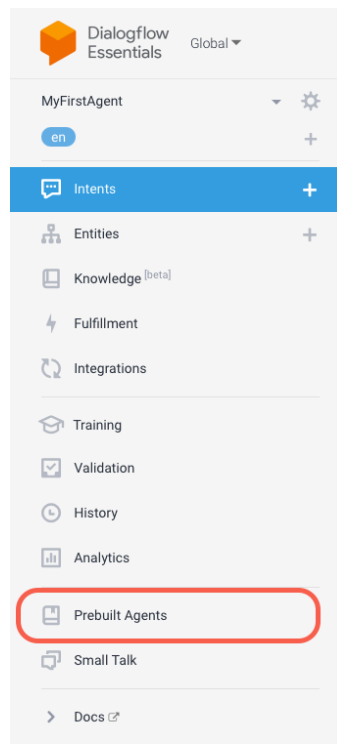


Click the button "Create Agent"

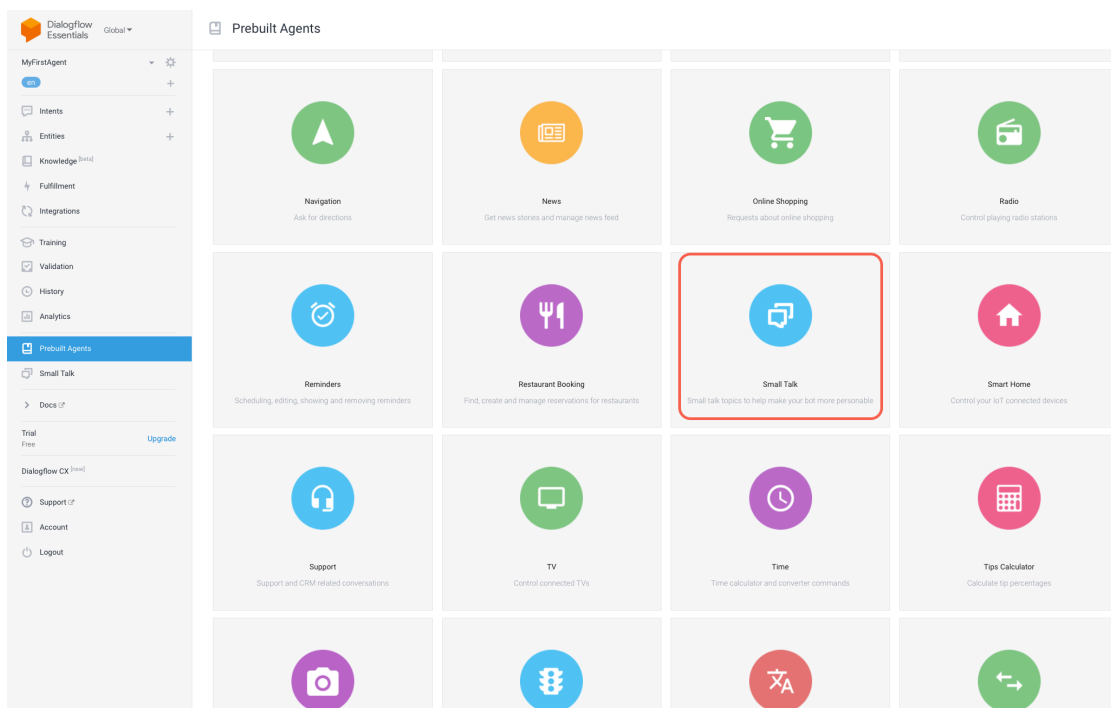


Give your agent a name and click "Create"

3. Exercise: Dialogflow Agent



On the sidebar, click the “Prebuilt Agents”



Scroll down and find the “Small Talk” prebuilt agent. Import it.



Create an agent from a template

New agent name

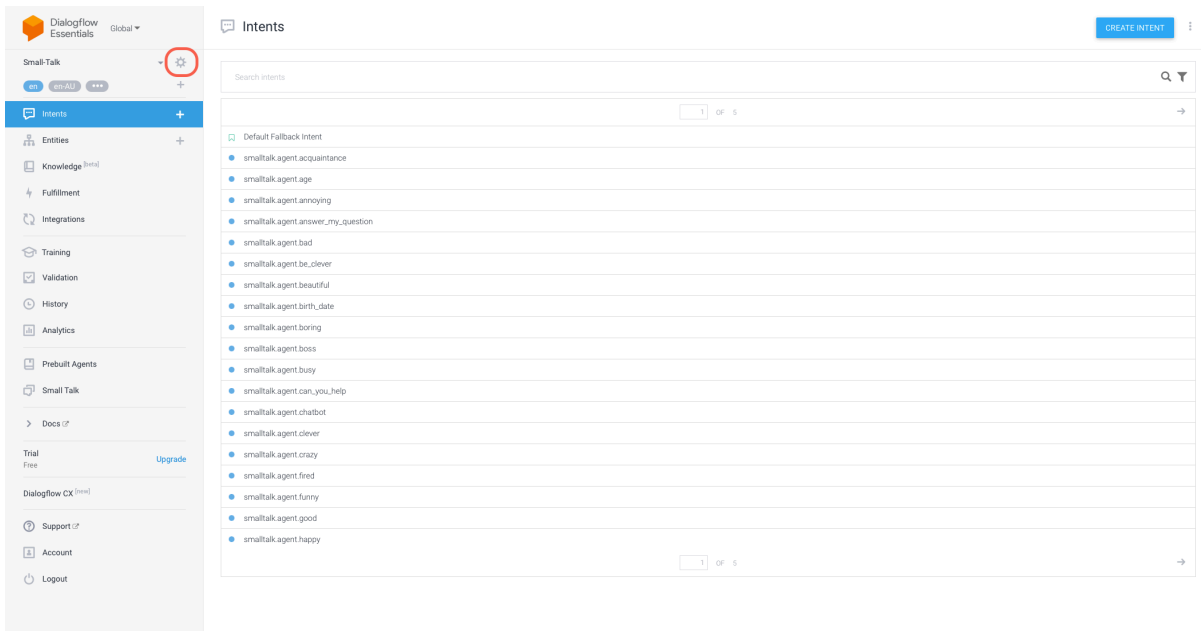
Small-Talk

Google Cloud project

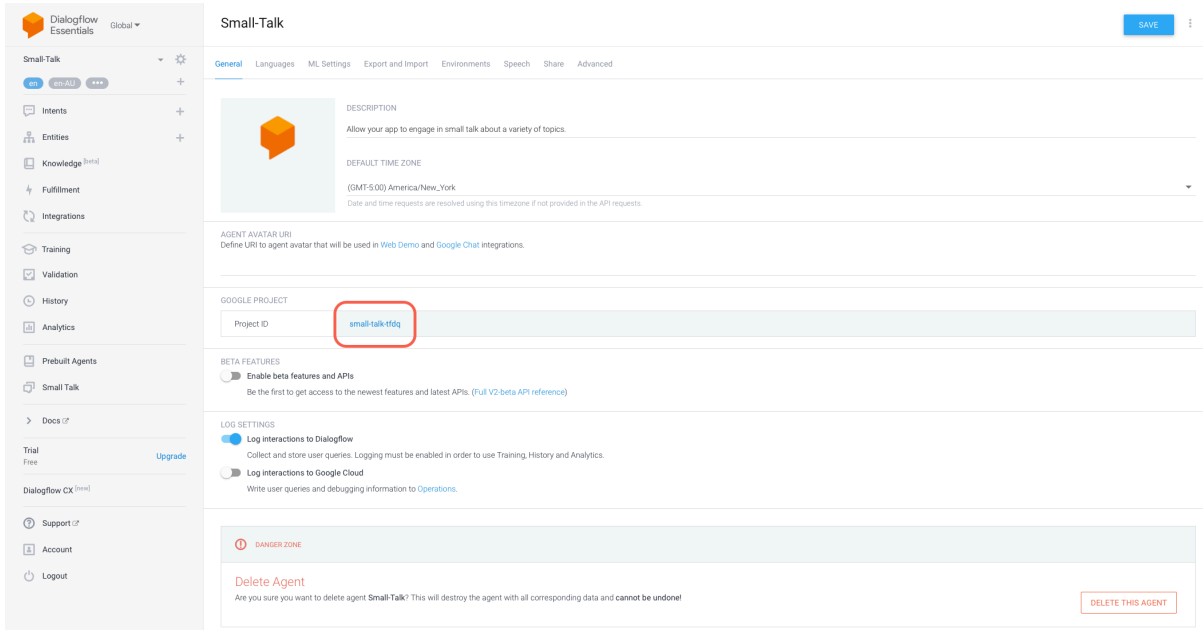
New GCP project will be automatically linked to the agent after saving

CREATE AGENT FROM TEMPLATE

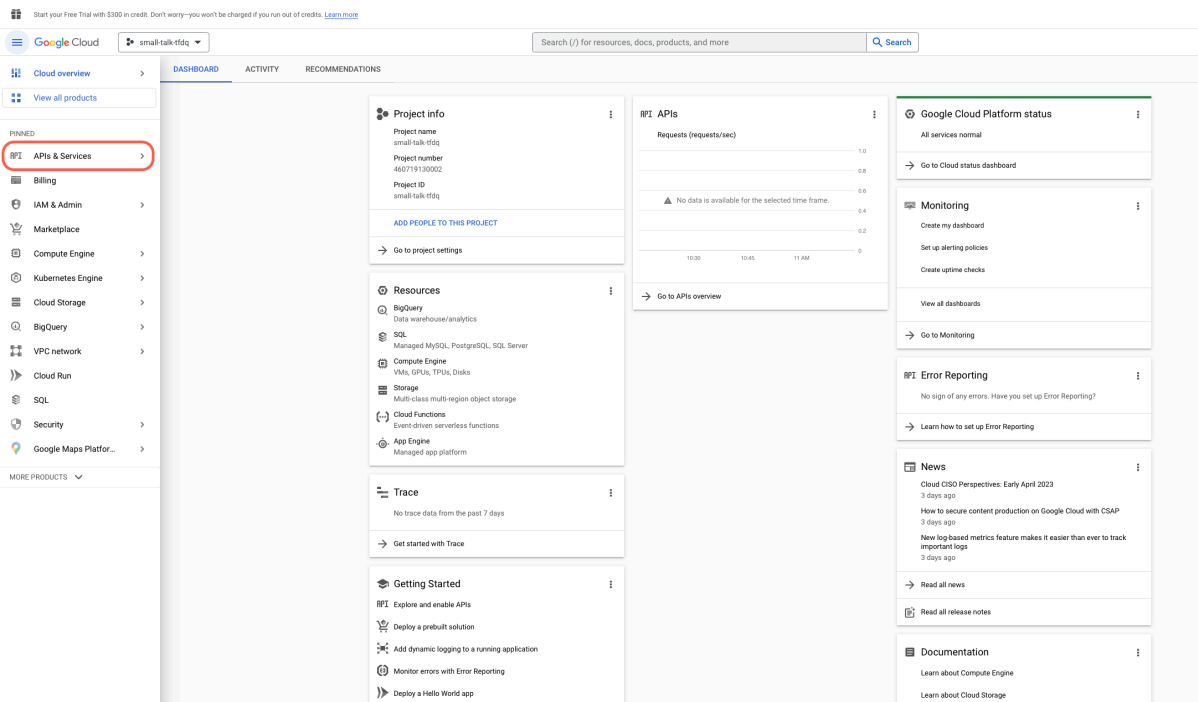
Leave the name on default and create your new agent from template



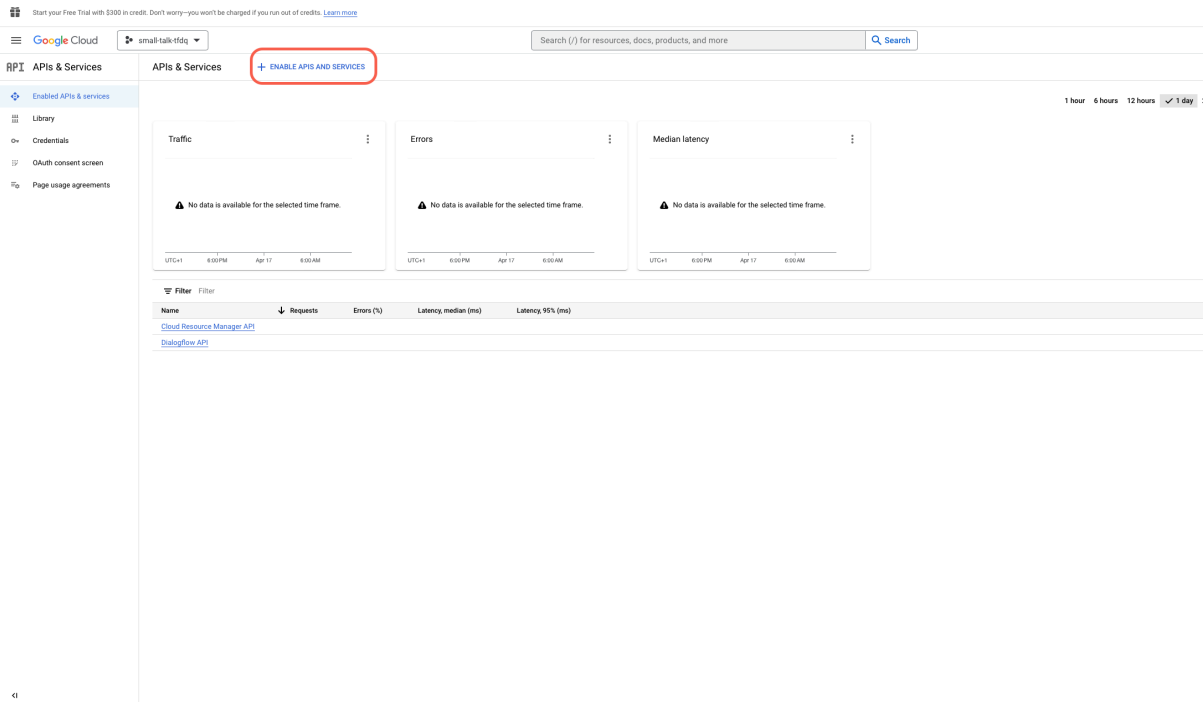
Next open your agent settings



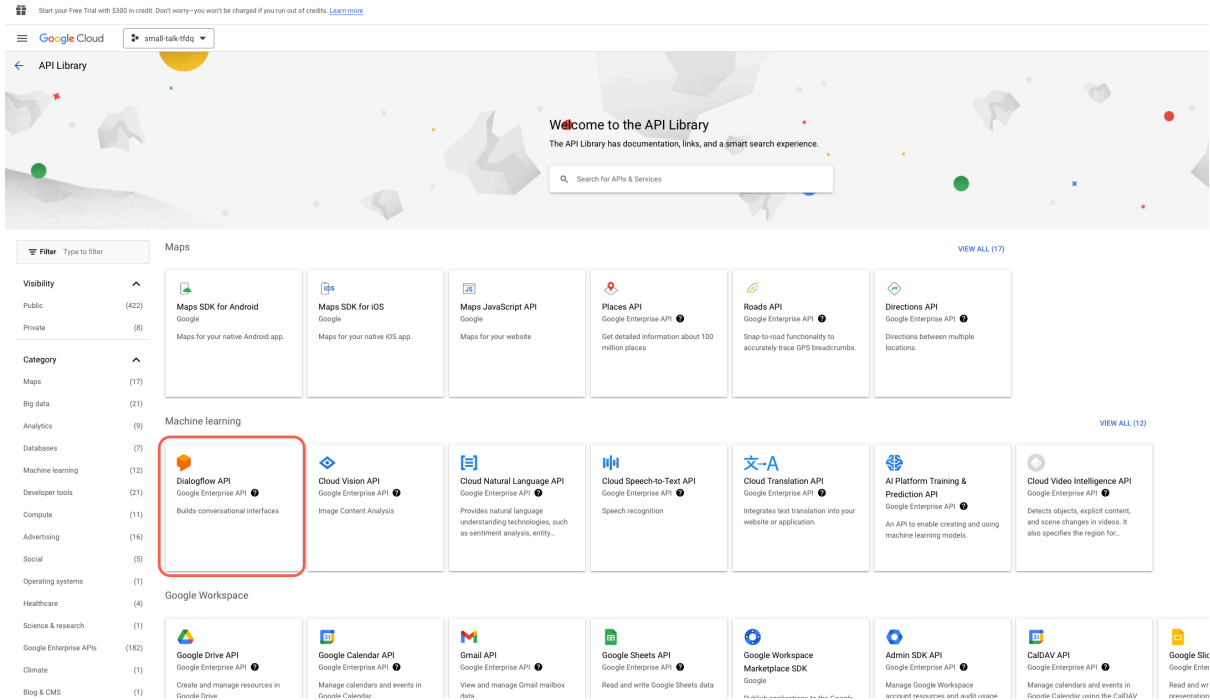
Click on your Project ID



A new tab is going to open up and you are now in the Google Cloud Console. Go to API & Services



Go to “Enable API and Services”



Find the Dialogflow API

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud small-talk-tfdq

Product details

Dialogflow API

Google Enterprise API

Builds conversational interfaces

[MANAGE](#) [TRY THIS API](#) **API Enabled**

[OVERVIEW](#) [PRICING](#) [DOCUMENTATION](#) [SUPPORT](#) [RELATED PRODUCTS](#)

Overview

Builds conversational interfaces (for example, chatbots, and voice-powered apps and devices).

Additional details

Type: [Soft & APIs](#)
Last updated: 4/20/22
Category: [Machine learning](#), [Google Enterprise APIs](#)
Service name: dialogflow.googleapis.com

Pricing

Intent Detection Audio Query Length for Enter	USD 0.0065 /75 second
Audio session for interacting with Dialogflow v	
Text session for interacting with Dialogflow C	
Agent Assist for Chat	

Note: Additional charges may also apply from infrastructure you use to call the API. If you pay in a currency other than USD, the prices listed in your currency on [Google Cloud Billing](#) apply. Please refer to [Google Cloud Price List](#) for the latest pricing.

We're currently using USD to calculate costs, which can be changed in the billing setup. Final prices in your bill will be set in accordance with your billing setup, and might be subject to exchange rates.

Tutorials and documentation

[Learn more](#)

Make sure it is enabled

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud small-talk-tfdq

Search (/) for resources, docs, products, and more [Search](#)

Navigation menu: Cloud Overview, View all products

PINNED: APIs & Services, Billing, **IAM & Admin**, Marketplace, Compute Engine, Kubernetes Engine, Cloud Storage, BigQuery, VPC network, Cloud Run, SQL, Security, Google Maps Plat...

IAM & Admin

+ ENABLE APIS AND SERVICES

Traffic Errors Median latency

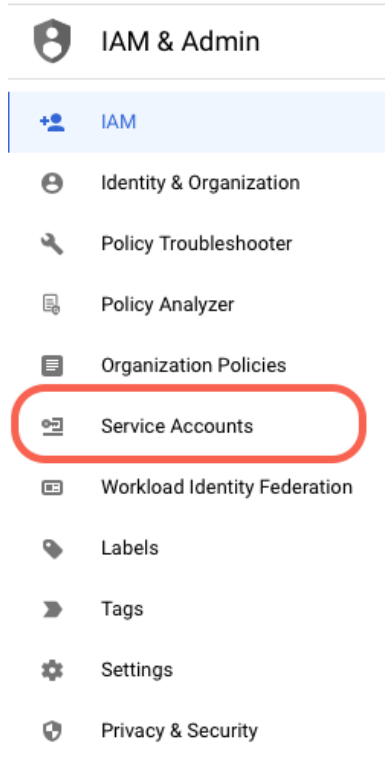
No data is available for the selected time frame.

UTC+1 6:00PM Apr 17 6:00AM UTC+1 6:00PM Apr 17 6:00AM UTC+1 6:00PM Apr 17 6:00AM

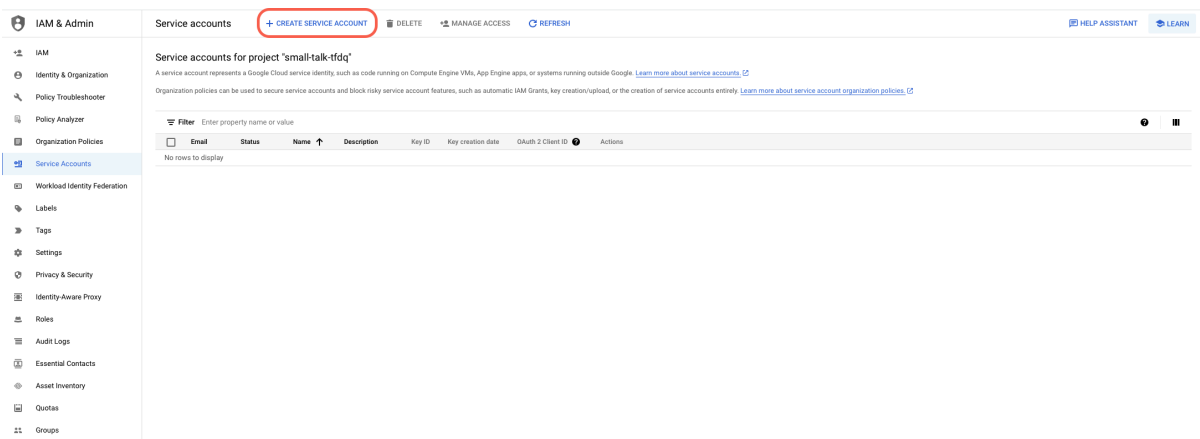
Filter Filter

Name	Requests	Errors (N)	Latency median (ms)	Latency 95% (ms)
Cloud Resource Manager API				
Dialogflow API				

Go back and open the side menu. Go to IAM & Admin



On the side menu go to Service Accounts



Create a service account

1 Service account details

Service account name
Display name for this service account

Service account ID * X ↺

Email address: test-566@small-talk-tfdq.iam.gserviceaccount.com 📧

Service account description
Describe what this service account will do

[CREATE AND CONTINUE](#)

2 Grant this service account access to project (optional)

3 Grant users access to this service account (optional)

[DONE](#) [CANCEL](#)

Give your account a name. Click on “Create and Continue”

✓ Service account details

2 Grant this service account access to project (optional)

Grant this service account access to small-talk-tfdq so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Select a role IAM condition (optional) ?

Filter Type to filter

Quick access	Roles
Currently used	Owner
Basic	
By product or service	
Access Approval	
Access Context Manager	
Actions	

[MANAGE ROLES](#)

3 Grant users access to this service account (optional)

[DONE](#)

Next select the “Owner” role and click "Done”

Service accounts [+ CREATE SERVICE ACCOUNT](#) [DELETE](#) [MANAGE ACCESS](#) [REFRESH](#) [HELP ASSISTANT](#) [LEARN](#)

Service accounts for project 'small-talk-tfdq'

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more about service accounts](#).

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. [Learn more about service account organization policies](#).

Filter Enter property name or value

<input type="checkbox"/>	Email	Status	Name ↑	Description	Key ID	Key creation date	OAuth 2 Client ID	Actions
<input type="checkbox"/>	test-566@small-talk-tfdq.iam.gserviceaccount.com	✓	test		No keys		115140659964381825160	⋮

Click on your new account

← test

[DETAILS](#) [PERMISSIONS](#) [KEYS](#) [METRICS](#) [LOGS](#)

Service account details

Name [SAVE](#)

Description [SAVE](#)

Email
test-566@small-talk-tfdq.iam.gserviceaccount.com

Unique ID
115140659964381825160

Service account status

Disabling your account allows you to preserve your policies without having to delete it.

✓ Account currently active

[DISABLE SERVICE ACCOUNT](#)


Advanced settings ∨

Go to “Keys”

← test

DETAILS PERMISSIONS **KEYS** METRICS LOGS

Keys

 Service account keys could pose a security risk if compromised. We recommend you a

Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using [organization policies](#).

[Learn more about setting organization policies for service accounts](#)

ADD KEY ▾

Type	Status	Key	Key creation date	Key expiration date
No rows to display				

Add a new key

Create private key for "test"

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

JSON
Recommended

P12
For backward compatibility with code using the P12 format

CANCEL **CREATE**

Make sure the JSON option is selected and click "Create"

Private key saved to your computer



small-talk-tfdq-da4af70c18b1.json allows access to your cloud resources, so store it securely. [Learn more best practices](#)

CLOSE

A file should be downloaded and a pop up like this should show up. Move this file to your Python project directory.

Exercise 2 - Pollux Server

Coding the server

Now you are going to create a middleman between Dialogflow and the user: the server (we named it Pollux for easier referencing).

Head over to your PyCharm and import the following libraries:

```
from flask import Flask, request
from google.cloud import dialogflow
import os
import json
```

If any of these imports is not recognised, make sure you have all these libraries installed. You may need to install them for these project specifically. You should find on the left icons, a **Python Packages** menu. Make sure you have the **google-cloud-dialogflow** package installed.

Next you are going to configure the access to your Dialogflow agent. Firstly you need to move the JSON file you downloaded from Google Cloud Console to the directory of your PyCharm project. Next you declare these variables:

```
project_id = 'smalltalk-tfdq'
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = 'smalltalk-
tfdq-2ee70168ff37.json'
app = Flask(__name__)
```

The `project_id` variable is the name of your Dialogflow Project ID, which should be “smalltalk-XXXX”. The `os.environ[“GOOGLE_APPLICATION_CREDENTIALS”]` variable is created to store the JSON file from the Google Cloud Console referenced before.

Now, from Dialogflow’s and Flask’s reference guide:

```
@app.route("/chatbot", methods=['POST'])
def chat_bot():
    body = request.json
    responseBody =
    detect_intent_texts(body['session_id'], body['query'], body['language_code'])
```

The `detect_intent_texts` method is the main focus of the interaction with the Dialogflow agent. This last snippet of code is just to format Dialogflow’s response to JSON.

```
responseHelper = {"responses": responseBody}
return app.response_class(
    response=json.dumps(responseHelper),
    status=200,
    mimetype='application/json'
)
```

Now let's develop the `detect_intent_texts` method:

```
def detect_intent_texts(session_id, texts, language_code):

    session_client = dialogflow.SessionsClient()
    session = session_client.session_path(project_id, session_id)
    responseObject = {}
    responses = []
    text_input = dialogflow.TextInput(text=texts, language_code=language_code)

    query_input = dialogflow.QueryInput(text=text_input)
    response = session_client.detect_intent(request={"session": session, "query_input":
query_input})
    responseObject = parse_object(response._pb.query_result)
    responses.append(responseObject)
    return responses
```

You should add the **parse_object** and **helping methods** to the code:

```
def parse_object(obj):
    j = {}
    for f in obj.ListFields():
        name = f[0].name
        field = f[1]
        if 'Struct' in str(field.__class__):
            field = struct_to_json(field)
        elif 'google.protobuf.internal.containers' in str(field.__class__):
            field = parse_array(field)
        elif str(field.__class__.__name__) in ['Intent', 'SentimentAnalysisResult', 'Sentiment', 'Text']:
            field=parse_object(field)
        elif 'google._upb._message' in str(field.__class__):
            field = parse_array(field)
        elif 'google.protobuf.struct_pb2.Struct' in str(field.__class__):
            field = struct_to_json(field)
        elif 'google.protobuf.pyext._message.RepeatedCompositeContainer' in str(field.__class__):
            field = parse_array(field)
        elif 'google.protobuf.pyext._message.RepeatedScalarContainer' in str(field.__class__):
            field = parse_array(field)
        print(field.__class__)
        j[name] = field
```

```
print(j)
return j
```

```
def parse_array(array):
    j=[]
    for i in array:
        if isinstance(i, (int, float, str, bool)):
            j.append(i)
        else:
            j.append(parse_object(i))
    return j
```

```
def struct_to_json(struct):
    j = {}
    for k in struct.keys():
        j[k] = struct[k]
    return j
```

And lastly you need to route the server like this:

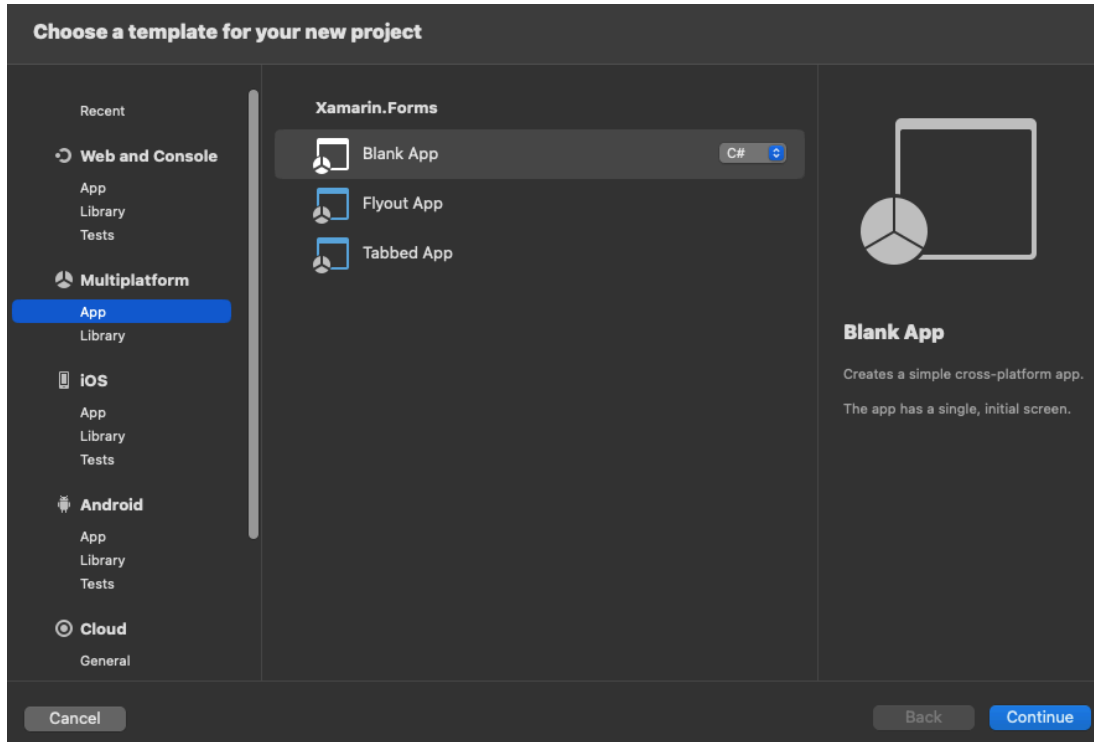
```
if __name__ == '__main__':
    app.run(port=9001, host="0.0.0.0")
```

You should choose a port (9001 or 9002 recommended) and the host should be as given.

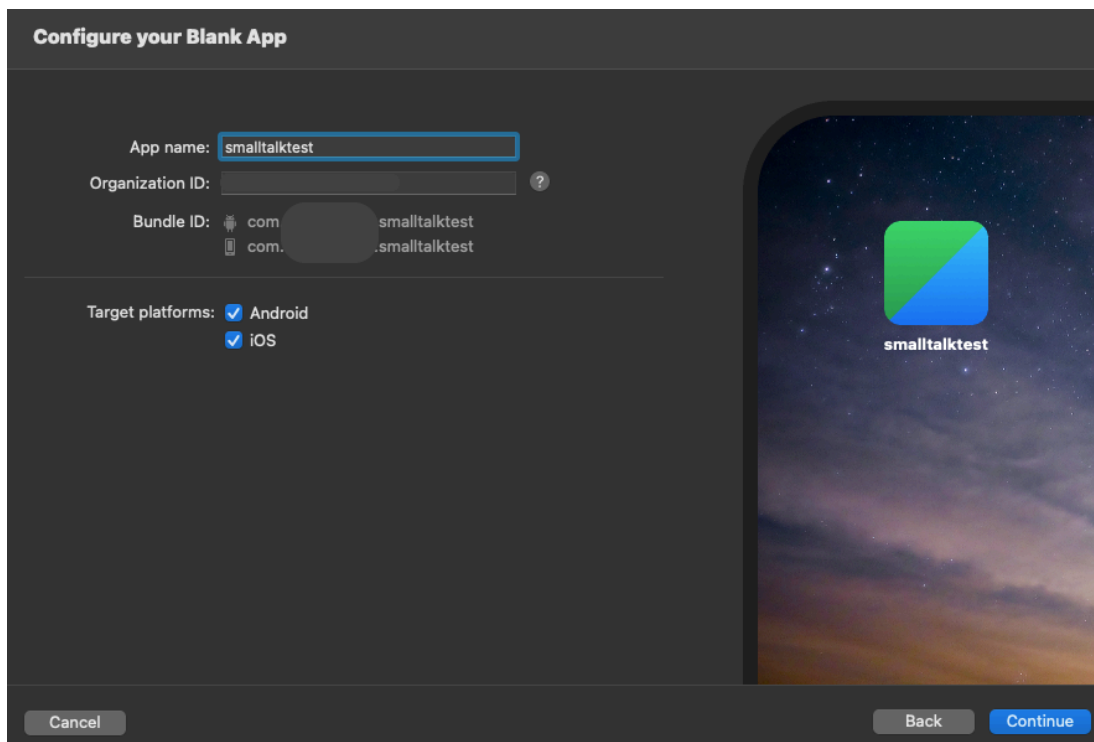
Exercise 3 - Xamarin App

Now you are going to create the Xamarin App to interact with the user.

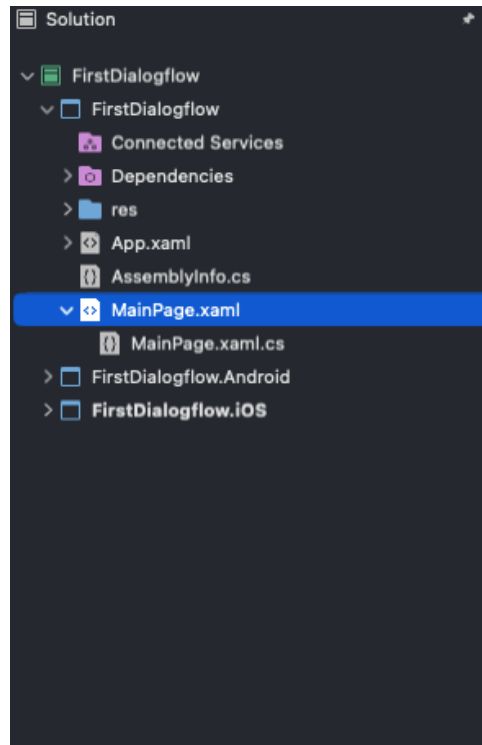
Open Visual Studio and create a new project.



Select a Multiplatform Blank App in C#

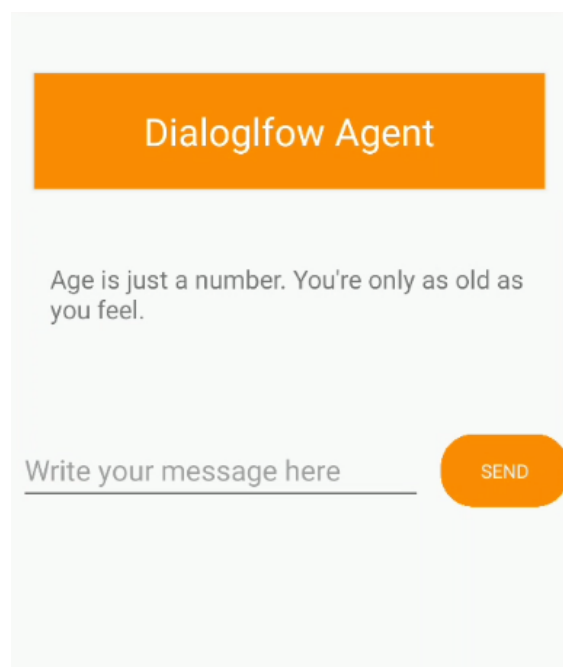


Give your app a name of your choice



Your project directory should look like this

Your main focus here are these two files: **MainPage.xaml** and **MainPage.xaml.cs**. The **MainPage.xaml** is the visual layout of the app, where you can place buttons, labels, pictures, etc. The **MainPage.xaml.cs** is the behaviour part of the app (i.e. what actions buttons perform). On the xaml file you are going to create this layout:



Simple Xamarin app layout: A title, a text box to receive Dialogflow's responses, an input to send messages and a Send button.

Before starting to code, there is some package management to be done. Head over to the toolbar and choose the “**Project**” tab. Select “**Manage NuGet Packages...**”. Search for “newtonsoft.json” and install the **Newtonsoft.Json** package. Next search for “restsharp” and install the **RestSharp** package. After installing both packages, close the NuGet Packages Manager.

Note: **If there are any Dependencies to update, update them.**

The following code generates the previous layout. On the **MainPage.xaml**, switch the **StackLayout** from your default app for the following one:

```
<StackLayout>
  <Frame BackgroundColor="DarkOrange"
    Padding="20" CornerRadius="0" Margin="20, 40">
    <Label Text="Dialogflow Agent"
      HorizontalTextAlignment="Center" TextColor="White"
      FontSize="24"/>
  </Frame>
  <Label Text="Answers here"
    x:Name="AnswerLabel"
    FontSize="16"
    Padding="30,0,30,0"/>
  <Label FontSize="16"
    Padding="30,24,30,0">
  </Label>
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="*" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="Auto" />
    </Grid.ColumnDefinitions>
    <Entry
      x:Name="InputQuery"
      Grid.Row="0"
      Placeholder="Write your message here"
      Margin="10, 10, 5, 10"
      HorizontalOptions="FillAndExpand"
    >
  </Entry>
```

```

<Button
    Grid.Row="0"
    Grid.Column="1"
    Text="Send"
    BackgroundColor="DarkOrange"
    TextColor="WhiteSmoke"
    ScaleX="1"
    ScaleY="1"
    x:Name="SendButton"
    Margin="0, 10, 5, 10"
    CornerRadius="20"
    FontSize="12"
    HeightRequest="10"
    WidthRequest="80"
    Clicked="SendButton_Clicked"
    >
</Button>
</Grid>
</StackLayout>

```

Next, on the top of the **MainPage.xaml.cs** make sure you add the following lines:

```

using RestSharp;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

```

After adding those packages to the code, take a look at the code for the MainPage class. Paste the relevant code into the proper areas:

```

public partial class MainPage : ContentPage
{
    private static string serverIP = "10.101.230.206";

    private static readonly string ChatBotProxyAddress = $"http://{serverIP}:9001"; //
chatBotProxy"
    private static string responseDialogflow;

    public MainPage()
    {

```

```

        InitializeComponent();
    }

    async void SendButton_Clicked(System.Object sender, System.EventArgs e)

    {
        var query = InputQuery.Text; // Save text from Entry

        InputQuery.Text = ""; // Reset the Entry to empty text

        var respContent = await waitClient(query);

        AnswerLabel.Text = responseDialogflow;
    }

    public static async Task<string> waitClient(string query)

    {
        var options = new RestClientOptions($"{ChatBotProxyAddress}")
        {
            RemoteCertificateValidationCallback = delegate { return true; }
        };

        var clientRest = new RestClient(options);
        var request = new RestRequest("chatbot", Method.Post);

        request.AddHeader("Content-Type", "application/json");
        request.AddHeader("Accept", "application/json");

        var json = JsonConvert.SerializeObject(new
        {
            query,
            session_id = "test",
            language_code = "en"
        });

        request.AddJsonBody(json);
        var response = await clientRest.ExecuteAsync(request);
    }

```

```

if (string.Equals(response.StatusCode.ToString(), "OK"))
{
    var chatResponse = JObject.Parse(response.Content);

    var resp = chatResponse["responses"][0];
    var chatText = resp?["fulfillment_text"].ToString();

    responseDialogflow = chatText;
    return chatText;
}
return null;
}
}
}

```

Exercise 4 - Test your implementation

4.1 - Run your Pollux server locally on debug mode. Check your **local IP** (normally displayed in PyCharm console. If not run ipconfig/ifconfig. Copy your IP to the Xamarin app in Visual Studio and check if the port is the same.

4.2 - Configure a new Android device in Visual Studio. To run smoothly, Pixel (API 30) is recommended.

4.3 - Run the app in debug mode. This should open a Pixel emulator and open your app on it. If the app does not open, please run the app again without closing the emulator.

4.4 - Send the following message: “How old are you?” and register the response.

4.5 - Try three other examples of messages and register the corresponding answers.

4.6 - On Dialogflow create a new intent with the following parameters:

- Intent name: create_alarm
- Training phrases:
 - “could you set me an alarm?”
 - “remind me of something”
 - “set alarm please”
- Text responses:
 - “Glad to help!”
 - “It’s done”

4.7 - Send one of the three training phrases from 4.6. Did you get a response?

4.8 - The alarm intent could be better. Notice that if the user does not provide a date and time, how are you going to set an alarm. Also if the user does not provide a description, what’s the alarm about? Let’s fix this. In Dialogflow go to the create_alarm intent. In action and parameters insert 3 new parameters: date, time and description. Set them to required. Define a prompt for each parameter.

These prompts are going to be activated if the user does not meet one or more of the parameters. Test

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
<input checked="" type="checkbox"/>	date	@sys.date	\$date	<input type="checkbox"/>	In which day yo...
<input checked="" type="checkbox"/>	timestamp	@sys.time	\$timestamp	<input type="checkbox"/>	Sure! At what t...
<input checked="" type="checkbox"/>	description	@sys.any	\$description	<input type="checkbox"/>	Which descripti...

New intent parameters

the intent again.

Appendix B

Green Bear – A LoRaWAN-based Human-in-the-Loop Case-Study for Sustainable Cities

Green Bear – A LoRaWAN-based Human-in-the-Loop Case-Study for Sustainable Cities

Oscar Torres Sanchez*, José Marcelo Fernandes*, André Rodrigues*[†], Jorge Sá Silva[‡], Fernando Boavida*, Jorge Eduardo Rivadeneira*, Afonso Viana de Lemos[§], and Duarte Raposo[¶]

*Centre of Informatics and Systems of the University of Coimbra, CISUC, Coimbra, Portugal

Email: {otorres, jmfernandes, arod, boavida, jrivadeneira}@dei.uc.pt

[†]Polytechnic of Coimbra, Coimbra Business School Research Centre | ISCAC, Coimbra, Portugal

Email: andre@iscac.pt

[‡]University of Coimbra, Institute for Systems Engineering and Computers, INESC, Coimbra, Portugal

Email: sasilva@deec.uc.pt

[§]Department of Electrical Engineering, University of Coimbra, Coimbra, Portugal

Email: aln@student.uc.pt

[¶]Instituto de Telecomunicações, University of Aveiro, Aveiro, Portugal

Email: dmraposo@av.it.pt

Abstract—The success of any smart city project or initiative lies in two key factors: on one side, a clear drive to contribute to sustainability and, on the other side, the involvement and contribution of citizens. These two factors should, thus, be taken into account when designing and implementing both the technical solutions and the functionality of smart city systems. This paper presents the *Green Bear* smart city platform, an IoT system that uses LoRaWAN nodes to gather information on city green spaces, bike lanes, and recycling depots, as well as a user-centered mobile application that allows for active user participation, feedback, and incentivization. This solution is being implemented in the city of Coimbra, Portugal. After an overview of the system, the paper provides a description of each of the main system modules. The Green Bear prototype was subject to functional and technological assessment, and the results are presented and discussed.

Index Terms—Sustainable cities, LoRaWAN, Human-in-the-loop, TheThingsNetwork.

I. INTRODUCTION

Information and Communication Technology (ICT) is driving the transformation of our cities. During the last two decades, widespread access to ICT has impacted and shaped all sectors of society [1]. Innovation in computer science is leading to an unprecedented change in the way that cities function and the way citizens live, with impact on operability, sustainability, quality of life, and social interaction. Clearly, modern urban living is connected with and driven by the ICT's evolution and response, transforming the development of communities [2].

Urban sustainability projects deal with complex environmental, physical, social, economic systems, involving many aspects, of which political vision, investment, environmental care initiatives, smart transportation, eco-friendly builds, renewable energy use, green spaces, and committed citizens are just some examples. In turn, these lead to a multitude of

technological challenges, in order to determine how ICT can actively help addressing and solving the complex requirements of a sustainable city [3].

Internet of Things (IoT) is an essential ICT component for sustainable cities infrastructures. IoT technologies can connect multiple urban domains to the Internet, gathering vast amounts of data that can be processed and analyzed to improve city sustainability [4]. Low Power Wide Area Networks (LPWANs) have gained relevance in IoT deployments in sustainable cities due to fulfilling some requirements such as long-range communication and extended device battery life [5]. Long-Range Wide Area Network (LoRaWAN) technology is one of the most deployed LPWAN technologies [6]. In addition to LPWAN's general advantages, LoRaWAN uses an unlicensed frequency band (868 MHz band for Europe), adaptive data rate, and high interference immunity. Moreover, companies like The Things Network (TTN) provide extensive public gateways coverage, making IoT LoRaWAN-based solutions easy to implement in cities [7]. Essential characteristics of LoRaWAN are the transfer of small data packets, high energy efficiency, the ability to interconnect remote areas, and easy interoperability, making it ideal for supporting the smart, sustainable city concept [8].

IoT solutions for sustainable cities usually focus on devices and sensors that collect people's daily activities, and consider humans as one more data-source, not as agents that are an integral part of the system, both producing data and being the main beneficiary of the whole system, thus closing the loop. One advantage of using the Human-in-the-Loop (HiTL) concept is that the actions, intents, emotional states, and physical aspects of humans are considered and integrated into the solutions. For a smart, sustainable city, human behavior and feedback are vital, because the system can be enhanced, influenced,

or disrupted by the interaction between human players and the system. Once humans are added to the system, they can provide information to be processed and, based on this, the system can infer future states and close the loop providing user feedback. This was, for instance, the approach we took in the system proposed in [9], where we proposed a HiTL cyber-physical system to improve students' performance. With the work presented in the current paper, we build on our previous work to develop a larger scale case study, involving the deployment of a more significant IoT infrastructure.

This paper describes the Green Bear system, developed by the authors with the aim of exploring and demonstrating the use of IoT, LoRaWAN public networks, and HiTL technologies in a sustainable city environment. Green Bear provides user feedback and guidance through a virtual friend chatbot, called the bear. The paper describes the overall solution, technical implementation details, and case-study assessment in the city of Coimbra, Portugal. The main contributions of the paper can be summarized as follows:

- an overall architecture for a Human-in-the-Loop, IoT and LoRaWAN-based solution for smart cities, exploring the concepts of sustainability and of the active participation of citizens;
- a real-world case-study implementation of the proposed system, involving the development of LoRaWAN prototypes that obtain data from users and can be placed in strategic locations throughout the city; the use-case resorted to several types of devices, adapted to a variety of different uses, such as garbage recycling, outdoor activities, or bicycle paths;
- an assessment of the proposed system, with identification of issues, limitations, and resulting research directions.

The remainder of the paper is organized as follows. Section II identifies the related work, their limitations, and distinguishing aspects of the Green Bear proposal. Section III provides an overview of the proposed system, discussing the overall approach and some sample application scenarios. Section IV details the technical aspects of the Green Bear implementation. An assessment of the system is presented in section V, essentially covering scalability issues. The conclusions and guidelines for further research are presented in section VI.

II. RELATED WORK

The development of human-centered systems using LoRaWAN to enhance urban sustainability is not novel. For instance, in [10] a human-centered methodology using LoRaWAN for gas concentration monitoring in public buildings was presented. The paper addresses several topics, such as sensor collection, long-range connectivity, cloud engine, automatic control, and mobile applications. However, the paper does not close the human loop, as the system does not try to modify employees' behavior to improve gas concentration measurements. In [11], a framework for Real-Time Human-In-The Loop Simulation with Mobile Agents, Chat Bots, and Crowd Sensing for Smart Cities was developed. In this work, humans are considered as sensors, based on three

models. The concerned data is used to analyze digital social media usage and its impact on social interaction, mobility, and opinion-making. Nevertheless, contrary to what happens in Green Bear, this work is based on simulation, and not on LoRaWAN prototypes that can easily be deployed in any urban environment. Deployment of a collaborative mesh network based on Bluetooth low energy (BLE) and long-range wide-area network (LoRaWAN) technologies promoting human-machine collaboration is presented in [12]. This paper builds on OperaBLE as the key enabler for human-in-the-loop systems seeking safer working conditions and sustainable digitalization of Industry 4.0. In [13], the authors developed new ways for managing environmental monitoring through a Mobile Crowd Sensing platform. The platform uses the microphone of the citizens' smartphones to build large-scale noise maps. The system can output strategies to reduce noise. In [14], an innovative garbage monitoring system using NB-IoT (Narrow Band IoT) was proposed. This solution used a set of sensors placed in garbage cans to provide information to edge nodes, that can correctly classify this information. NB-IoT is an LPWAN technology that allows the system to transmit the management garbage information with low-power consumption. The authors of [15] evaluated bike sharing and bike rental variations using IoT to enhance bike supplying services per season in China's smart city context. In [16], detailed strategies for building Green IoT solutions that make cities smarter, sustainable, and ecofriendly are presented. The study explained the best techniques to use IoT for improving energy efficiency, reducing pollution, managing waste, and sustainability.

Considering the mentioned pieces of work, and as far as we know, our solution is the only one that takes a Human-in-The-Loop-Cyber-Physical-System (HiTLCPS) approach to enhance urban sustainability and citizens' quality of life, by closing the loop and giving feedback to the users. Moreover, Green Bear prototypes use LoRaWAN for long-range communication and BLE connections for short-range communication, leading to extremely low power consumption in nodes, as required by green city implementations.

By combining a human-in-the-loop approach with chatbots for human assistance, and low-power LoRaWAN and BLE transmission technologies, Green Bear is a technological solution that seeks to improve participatory and sustainable processes in intelligent cities.

III. GREEN BEAR OVERVIEW

A. Overall Approach

Green Bear is a solution that consists of a mobile application interacting with LoRaWAN nodes strategically located in the city, which collect data on the users' activity, under their full control. The solution includes LoRaWAN nodes using a public TTN gateway, a TTN application to integrate external services, a back-end developed to support Internet services, a dashboard that allows system management, and a mobile application that integrates humans in the system. IoT LoRaWAN devices, mobile devices, and the voluntary human participation in the

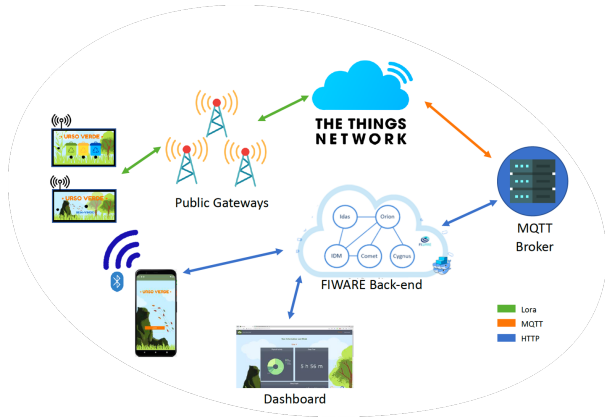


Fig. 1. Green Bear general architecture

system are the main components of an integrated solution that aims to improve the participants' lifestyle and the dynamics of the city. Fig 1 shows the overall architecture of the Green Bear solution.

Green Bear users register through the mobile application. To protect the users' identity, their name is never used nor maintained in the system; instead, randomly generated identifiers, called user ID, are assigned to each user. The system does not collect any personal information from the users, and the identifiers are solely used as primary key in the system database. When performing various activities, the users are informed of the earned points. These points are reported to the system, and the global ranking is updated on a weekly basis. With this setting, a variety of incentives can be created, such as prizes, discounts in stores, or discounts in public services, depending on corporate or governmental agreements.

In addition to providing a way for the users to check the earned points, the mobile application has a chatbot to interact with the users and collect additional data. For instance, the chatbot can provide recommendations on sleep hours and physical activity, depending on the observed activity. Notifications help participants to become more involved in the system. Additionally, the chatbot allows the system to ask questions and set mobile parameters, assisting the user in configuring the system. The overall objectives of the mechanisms implemented in the mobile application are to foster the citizens' interaction with city spaces, e.g., green spaces, outdoor activities, recycling dynamics, and to incentivize the users' physical activity, sleep time, and quality of life.

B. Green Bear activities

Green Bear's architecture was devised in such a way that it does not limit the applications and activities that may be of interest in any smart city environment. In this sub-section, we present the activities implemented in the Coimbra case-study as a way of example, but many more are possible.

Recycling bins: In this activity, users receive one point each time they report a full recycling bin. For instance, if there are three main types of recycling bins at each location (e. g.,

paper/cardboard, glass, and plastic/metal), users may receive up to three points per day per recycling bin spot. LoRaWAN devices located in the bins register that the user has notified the bin status. Then, the saved information is sent by the recycle bin spot on a daily basis, through a LoRaWAN connection, to the central system.

Green Spaces: Users earn one point when performing activities in parks or green spaces. Many European cities have adapted spaces for leisure activities, and the World Health Organization (WHO) makes several recommendations on activities of this type [17]. In the Green Bear case-study, if users spend at least one hour in a green space, they will receive one point. For implementing this, LoRaWAN devices determine the time spent by the user on the grounds and, if it is longer than one hour, save the information in their local memory, and periodically send the user identifiers to the central system.

Bike Lanes: Similarly, the use of bike lanes gives users one point. People can use these spaces on bicycles, skates, skateboards, scooters, or even on foot. LoRaWAN devices located on the bike lanes gather information on the users and assign points that are periodically sent to the central system.

Sleep Hours: Based on an estimated number of hours of rest, the user will be rewarded with one point per day. The number of hours of sleep is collected through the user's phone, and points will be assigned if the user complies with a recommended six sleep hours (average adult).

Physical Activity: Furthermore, following WHO physical activity recommendations, if the user achieves the minimum threshold, he/she also earns one point, which is then reported to the system on a daily basis.

It should be noted that all the data pertaining to the users' activities is anonymized, through the use of the randomly generated identifier assigned to each user at registration time. The system does not save nor maintains any association between the actual identity of the user and his/her identifier.

IV. GREEN BEAR IMPLEMENTATION

Green Bear's architecture consists of LoRaWAN nodes, TTN public infrastructure, MQTT broker connectors, back-end implementation through FIWARE, dashboard, and end-user mobile application. Figure 1 shows the referred components, which are going to be described in detail in the current section.

A. LoRaWAN Devices

LoRaWAN nodes are fixed nodes deployed throughout the city with the objective of interacting with end-user mobile applications, collecting data, and forwarding it to the central system. These nodes were implemented using Pycom devices. A Pycom FiPy includes a microcontroller that can be programmed using Micropython. In addition to LoRaWAN, FiPy devices have four other connectivity options, namely SigFox, Wi-Fi, Bluetooth, and LTE [18], which allows the development of heterogeneous, flexible systems, with a variety of communication capabilities. Pycom devices are based on ESP32

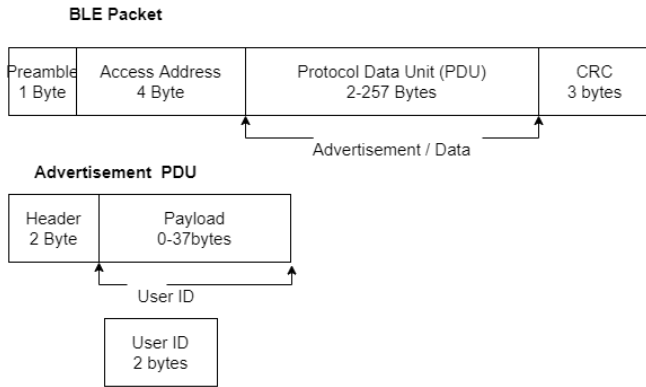


Fig. 2. BLE packet format in Green Bear

SoC, with ultra-low power consumption, 4MB RAM, 8 MB Flash Memory, and several peripherals for IoT applications. Furthermore, the Pycom firmware allows developers to use microcontroller functions such as interrupts, timers, analog-to-digital converters, digital-to-analog converters, and general-purpose input/output pins, among others [19].

Two communication technologies are used in Green Bear fixed-node devices: Bluetooth Low Energy (BLE) for collecting user identifiers, and LoRaWAN for sending user identifiers and collected data to the central system. The BLE packet frame comprises a preamble, Access Address, a variable Protocol Data Unit (PDU) and a Cyclic redundancy check (CRC), as is shown in Figure 2.. The variable size PDU is used for advertisement PDUs or for data PDUs. In Green Bear, an advertisement PDU has a variable payload (0-37 bytes) and is used by mobile phones to notify the respective user ID. The user ID is a 2-byte identification code sent to Green Bear fixed devices in advertisement PDUs. In addition to the identifier, the Pycom device can obtain several other data from BLE packets, namely the communication RSSI, the MAC address, proximity Universally unique identifier (UUID), and measured Tx power [20].

On other hand, LoRa supports long-distance communication with the TTN public gateways. A FiPy can be configured with operating class (class A or class C), spreading factor (SF), data rate (DR), and frequency band [21]. The node to gateway communication range is directly affected by the SF configured at the nodes, as shown in the study presented in [22] and illustrated in Fig 3. However, increasing the SF affects the number of bytes in the message [23], and message airtime allowed for TTN public gateways [24]. Therefore, the ISM band (868 MHz Europe) limitations allow a maximum number of bytes sent in a single message and the number of messages per day.

The LoRaWAN message format is shown in Figure 4. The message contains header and trailer fields that consume 13 bytes for uplink or downlink messages, and a variable frame payload. Hence the maximum size of a message will depend on the data sent in the payload. This frame payload contains application-specific information, such as the Green Bear IDs.

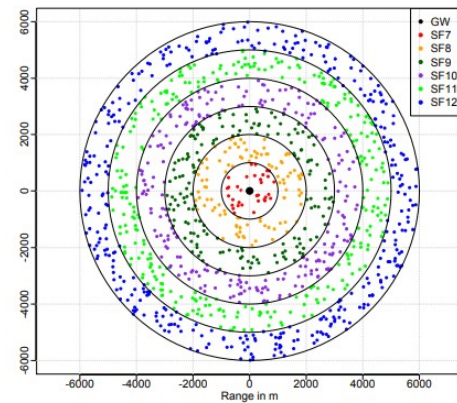


Fig. 3. Distance vs Spreading Factor [22]

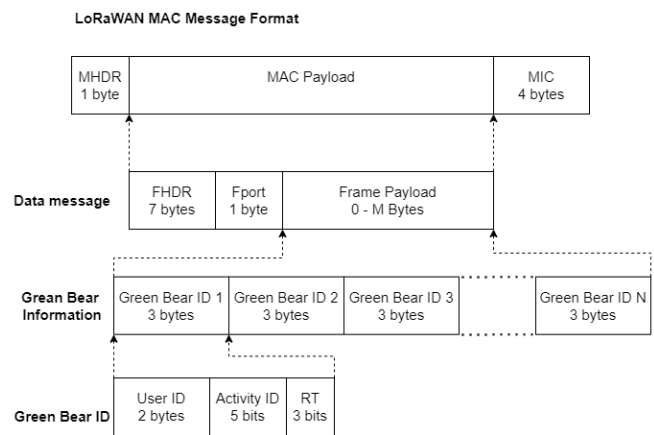


Fig. 4. LoRaWAN message in Green Bear

A Green Bear ID includes the user IDs extracted from BLE packets, activity ID (which contains the device's identifier, based on each activity), and the Recycling Type (RT) (which is reserved for the recycling bins activity).

Green Bear nodes are composed of the Pycom device, LiPo battery, the developing board, LoRaWAN antennas, and control buttons to trigger the operations of the node. Fig. 5 depicts the LoRaWAN nodes' general look. These devices operate in sleep mode until a user presses a control button that triggers an interruption and awakens them. This manual approach to the triggering of Green Bear nodes was adopted as a first step, for simplicity reasons, although more automated ones can be adopted in the future. The approach is used in the various activities where a fixed node is used for collecting data from users: recycling bins, green spaces, and bicycle lanes. Each Green Bear device is programmed for its corresponding activity. Once the user manually activates the device, it will gather the user ID and add the information for the rest of the fields of the Green Bear ID (Fig. 4). For instance, in the case of recycling bins, the user should notify the status of the depot by pressing the corresponding button. The device will store the user's information, the activity, and the type of recycling bin.



Fig. 5. Green Bear nodes

In the case of green spaces, a user should press the button on entering the space and then on exiting it; based on this, the Green Bear nodes calculate the duration of the visit and assign one point if it is longer than 1 hour. Finally, in the case of bike lanes, the user should adopt a similar proceeding. Users can earn one point for each depot for the recycling bins activity, getting a maximum of 3 points per day. In the case of green spaces and bike lanes, the users can earn one point per day, for each of these activities.

B. The Things Network

TTN is a collaborative ecosystem released by the Things Industries to develop and deploy networks, devices, documentation, and solutions over LoRaWAN. TTN, through its community edition, releases a decentralized LoRaWAN network with hundreds of public gateways around the world. TTN provides a platform that allows creating and managing devices, program downlink and uplink payload formatters, manage integrators and keys. Once the nodes are created in the platform, they can establish link credentials and initiate communication.

The nodes send LoRaWAN frames containing identifiers and type of function; the platform should structure the message for external connections. TTN allows for the definition of payload formatters in JavaScript, which decode incoming node messages and reformat them before the platform sends them to subscribing applications. The decoded payload intends to establish the object format for those external connections. Furthermore, TTN supports Message Queuing Telemetry Transport (MQTT) connections, working as a broker, enabling third-party applications to publish and subscribe on established topics. The subscribing upstream traffic topics include joining, data, and ack information, while TTN can push external data or commands to the nodes using the downstream topics, therefore providing end-to-end communication between external applications and LoRaWAN nodes.

C. MQTT broker connectors

These connectors are located inside the TTN platform and support two methods for clients, namely subscribing to

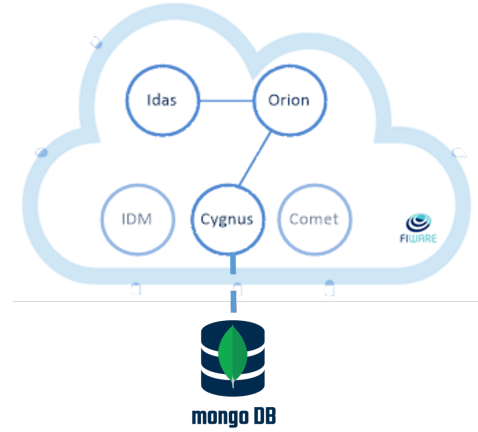


Fig. 6. Green Bear Back-end/Fiware Architecture

upstream traffic, and publishing downlink traffic. Regarding upstream traffic, it is possible to subscribe to the following topics: join, up, down/queued, down/sent, down/ack, down/failed, and service/data. Regarding publishing downlink traffic, the only accepted topics are /down/push and /down/replace, where sending information from an external source to the devices is available.

D. Fiware

FIWARE is an open-source framework with tools and components to integrate smart solutions with third-party components. FIWARE allows interfacing IoT projects to capture updates on information context and translate that on expected actuations [25]. Additionally, it allows publication, management, processing, analyzing, and visualizing applications information.

The Green Bear's back-end is based on our previous work in the ISABELA's project [26], and was developed using FIWARE. The back-end contains the connectors for the interaction of LoRaWAN nodes with the dashboard and mobile application. FIWARE receives updated information notifications from the nodes through the MQTT broker, stores the data in a database, and provides the mobile application and the dashboard with this information using the HTTP protocol.

The modular nature of the FIWARE ecosystem allows it to be the backbone of several applications, enabling the easy deployment of modules, called Generic Enabler (GE), which perform specific functionalities. In figure 6 we can see the modules used in the backend architecture of Green Bear and their interconnections.

The main GE of FIWARE is the ORION module [27]. This module works as a context broker, which receives information, represents that information as virtual entities, and manages subscriptions and registrations to said entities. The FIWARE project adopts the NGS9/10 information model, which is based on attributes and entities, and is represented using JSON format [28]. This information model is used by the ORION context broker to represent real world devices/physical

sensors to virtual entities, which can be accessed through an Application Programming Interface (API) that enables end applications to create/delete/retrieve entities, and to update existing ones.

On one hand, the ORION GE for itself does not allow the system to store historic data. As such the addition of other GEs is required. To connect the ORION with other modules, the FIWARE catalogue has the Cygnus [29] module. This module can work as a connector from the broker to several storage-capable modules or third-party apps. The Cygnus GE is based in the Apache Flume technology [30]. When a new entity accesses the module, it is redirected to a specific channel and forwarded to the selected storage solution (e.g., MySQL, MongoDB and CKAN channel). In the Green Bear project, the information is stored in a MongoDB database.

On the other hand, a database system, by itself, does not provide APIs to retrieve the data from applications. To tackle this issue, the FIWARE catalogue offer a different GE, namely the Short-Term History (STH)-Comet GE [31]. This module provides a RESTful API which allows applications to retrieve information in a historic and aggregated manner. Both the Green Bear mobile application and the dashboard retrieve historic information concerning user activity (e.g., sleep information or 'green' activities) through this API. Additionally, this API can aggregate the information using different time intervals, e.g., per week, per day, or per hour. This functionality allows us to create the end-user application, without the need to develop complex back-end APIs or to deal with database native queries.

As explained before, FIWARE's main objective is to integrate IoT solutions with cloud services and third-party applications. In order to achieve this, an interface layer to communicate with IoT devices is needed. IoT devices have all sorts of hardware and there is great heterogeneity in terms of communication protocols. The IDAS GE implements a set of IoT agents that work as a translation layer between IoT devices and the ORION context broker [32]. These agents are developed for several of the most common IoT protocols such as JSON/MQTT, LwM2M, Ultra Light and OPC UA. In the Green Bear project, we use the JSON/MQTT IoT agent in conjunction with the MQTT Mosquitto broker, to interpret the data coming from the TTN.

Although Green Bear anonymizes data through a randomly assigned user ID, since we are dealing with information pertaining to human activity, security and privacy is one of the most important requirements of this project. The Identity Management GE called IDM or Keyrock [33] allows us to protect communications between GEs, IoT devices, and applications. The IDM allows the addition of authentication and security to devices, users, and applications, as well as the implementation of authorization policies. Additionally, in terms of privacy, and to comply with the EU privacy laws, namely the General Data Protection Regulation of May 2018 [34], all data captured by the mobile application is anonymized in the device before being sent to the server.

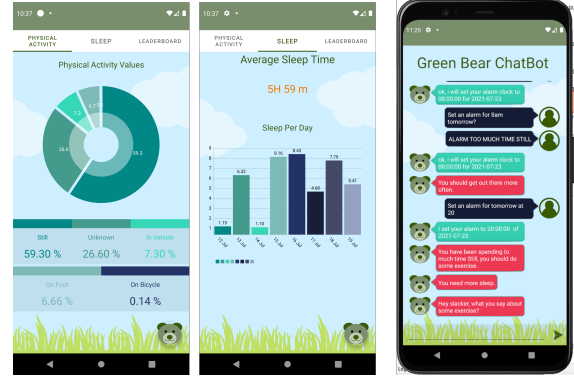


Fig. 7. Mobile Application Activities

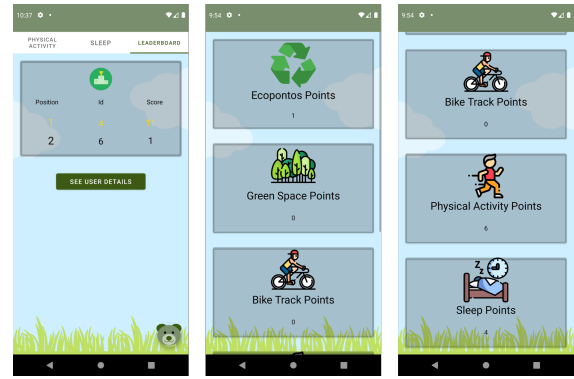


Fig. 8. Mobile Application leaderboard, and user's points display.

E. Mobile Application

Users interact with the Green Bear system through an Android mobile application 7. The main objective of of this application is to allow any registered user to interact with Green Bear LoRaWAN devices over Bluetooth, without the need for additional hardware. Most people, nowadays, have a smartphone, and Bluetooth BLE has become a standard in modern smartphones. As such, this approach means that most people will be able to use Green Bear off-the-shelf. As future work, we plan to move our development to a cross-platform technology (e.g., Xamarin) to be able to cover the iOS user base as well.

When a user first logins, in the application, a unique ID is assigned to the user. This ID is registered in the back-end server and subsequently mapped to an internal sequential number, which is then utilized as the user's identifier sent in BLE beacons. The mobile application periodically sends a BLE beacon advertisement. When a user is near a Green Bear node that has been activated by pressing its interface button, this BLE beacon is captured and the user identifier is stored to register the user specific action. These actions are later sent to the Green Bear server through TTN.

In addition to allowing users to interact with Green Bear nodes, the mobile application also collects user information, namely user activity and sleep patterns. In order to gather

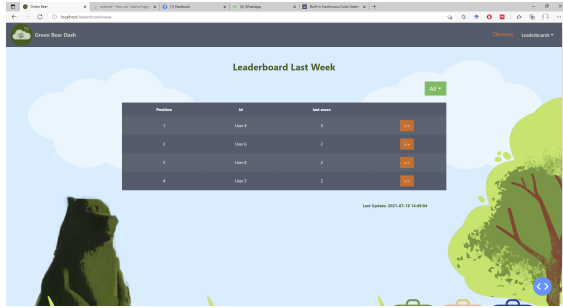


Fig. 9. Leaderboard dashboard's perspective

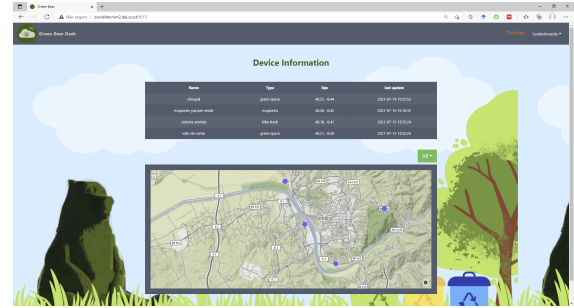


Fig. 10. Device management in the Dashboard.

information on the activity of users, the Activity Recognition API [35] was used. This API is able to automatically detect activities, by periodically reading and processing short bursts of sensor data. The API is able to classify user's activity into one of several possible categories, namely: running, walking, on bicycle, in vehicle, on foot, tilting, and still. To detect user sleep patterns, we used the Sleep API from Google [36]. This API is able to perform two tasks: firstly, it periodically classifies temporal intervals into "sleeping" or "not sleeping"; and secondly, it is able to daily report sleep segments, which include the timestamp of the start and the timestamp of the end of sleeping periods. As explained before, this data is used to assign points to users, using healthy lifestyle criteria. This data is periodically sent to the FIWARE back-end through the available HTTP/Rest API.

The mobile application is also used as a front-end to the user. In the mobile application, as can be seen in Figure 7, the user has access to the collected information, aggregated by week, namely, the percentage of each type of activity he/she performed in the last 7 days, and the total number of sleep hours per day. It is also possible for the user to check the number of points obtained as a result of each performed action, and how they rank against other users (user IDs) in the leaderboard, as can be seen from figure 8.

Finally, in order to enhance the interaction with the user, thus closing the loop, a chatbot was developed, as seen in Figure 7. The chatbot can report problems to the users about their lifestyle, such as having low exercise levels or sleeping less than the recommended daily hours.

F. Dashboard

The Green Bear dashboard was developed using DASH [37]. The DASH framework allows the development of data-intelligence and data-analytics, specific applications, in Python. This allows for a fast tracking of the development of front-end applications, in projects that deal with a large volume of data and require its visualization.

In the Green bear project, the dashboard serves as a means of administrating the system. This platform can be used, for instance, by the officials of a municipality to gather data and act accordingly, in a smart city context. The dashboard can also be used to check the ranking of users, so that incentives are given. The administration team has access to the leaderboards

of the various activities (e.g., physical activity, sleep, recycling, bike tracks, green spaces) through the dashboard, as illustrated in Figure 9. Based on this, it is possible to define prizes based on the type of activity, such as, for instance, discounts in a sportswear shop to top scoring users of bike lanes or other physical activity. The dashboard also allows to visualize the leaderboards for different temporal intervals.

Additionally, the dashboard serves as management tool for the system itself. As can be seen in Figure 10, it is possible to list all the connected LoRaWAN nodes in the dashboard. It is also possible to see their current position and the type of device in each case. In the future we want to extend the dashboard capabilities to show the current energy status of each device. This would allow to manage and plan the preventive maintenance of devices. Additionally, since we know the location of each device and the interaction of the users with them, it would be possible to further extend the functionalities of the system to create a heat map, showing the affluence of people to certain spaces. This would be particularly important in the context of the Covid-19 pandemic or future pandemics, for instance, where restricting the number of people in certain spaces can be important to maintain public safety.

V. ASSESSMENT

The functional assessment of the solution was carried out in the city of Coimbra, Portugal. The project was developed using TTN's public gateways; in the case of Coimbra, there are eight gateways, which cover most of the city. Furthermore, in its urban area Coimbra has parks such as Choupal, Vale de Canas, and Parque Verde do Mondego, and bike lanes in areas such as Portela. The recycling bins scenario was evaluated using the Coimbra garbage depots as part of this solution. Considering the scenarios, this initial phase comprised three LoRaWAN nodes, one for the recycling bins, one for green areas in Choupal Park, and one in the city's bike lanes.

For the assessment of the mobile application, the authors' smartphones were used in the initial phase. The mobiles collect sleep and physical activity data, and generate recommendations through the chatbot. Fig. 7 illustrates the functionality used in the evaluation. In addition, this information is shown in the dashboard, along with the score obtained in the various activities.

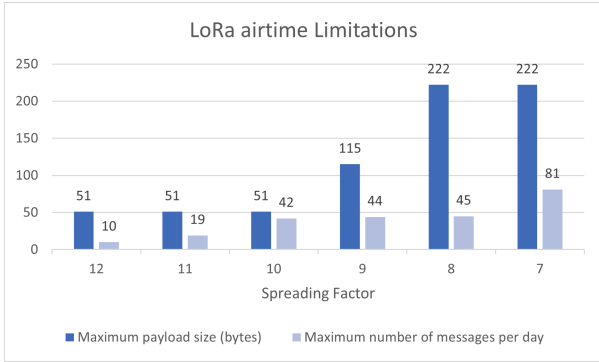


Fig. 11. Limitations on LoRaWAN messages [38]

After several weeks of use during which data pertaining to several identifiers (i.e., users) was collected, the corresponding leaderboard was assembled and could be viewed through the dashboard. By clicking on a specific user identifier, additional information on the points earned by that user in that period can be seen. Fig. 9 shows the leaderboard for a weekly period. Furthermore, the dashboard allows to check the user's score in each of the five case-study categories mentioned in section III.B. Using their mobile application, users can check their score in each category and get chatbot notifications.

Participants' identities are protected both on the dashboard and mobile devices. Thus, the platform administrators have no way of knowing the name of the leading scorer, while mobile users know their own position but not the names of the other participants on the leaderboard. With this personal data protection scheme, claiming incentives is done by simply presenting the user smartphone.

In addition to a user-oriented functional assessment, the system was subject to technical assessment, as well. This is presented and discussed in the following paragraphs. Green Bear nodes can detect multiple gateways, but they communicate with the one that presents the highest RSSI. Upon startup, each node sends its position through LoRaWAN. This enables the system to automatically gather the position of all active nodes in the city. This information is shown in the dashboard.

As described in section IV, the spreading factor (SF) determines the maximum size of the payload of uplink messages, thus limiting the maximum number of Green Bear IDs in each message. On other hand, [38] states that the duty cycle should be 1% in the 868 MHz frequency band to comply with the European Telecommunications Standards Institute (ETSI). As a consequence, TTN establishes a fair access policy that sets a maximum number of messages per day depending on the size of the message. The maximum payload size is as shown in Figure 11 (dark blue bars). Also, Fig. 11 shows the maximum number of messages per day (light blue bars) considering the maximum payload size. Given these limitations, Figure 12 shows the maximum number of user identifiers that can be sent in just one Green Bear message (blue line), as well as the maximum number of user IDs per day (orange bars).

For example, analyzing Figures 11 and 12, the maximum

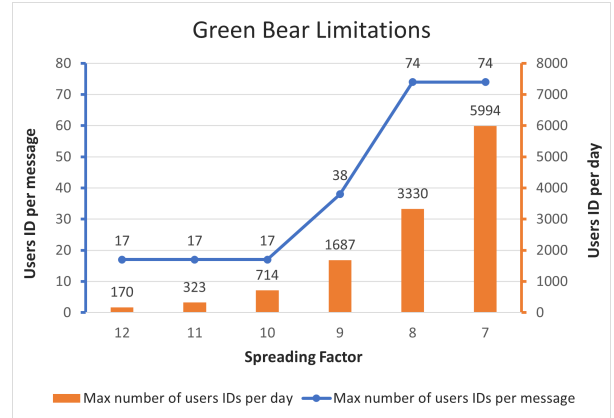


Fig. 12. Green Bear Limitations

number of user identifiers that a node can send, considering one message per day, is 74 for SF 7. Hence if more than 74 users interact with a device, more messages must be used to communicate with the system. Considering the maximum of 81 messages per day given in Figure 11 for SF 7 and a maximum of 74 users per message, a device can report a maximum of 5994 users per day (Figure 12). To extend this solution using TTN, it is necessary to calculate the maximum number of messages per day that can be sent using the different SFs.

TTN implements capacity limitations and a fair access policy. However, it is possible to use private networks for implementations requiring higher number of users. Although private networks use proprietary gateways, they can still use The Things Network configuration and platform through The Things Stack. This would allow the nodes' capacity to get extended, in turn leading to more identifiers in the system per day. Private network implementations will be analyzed in subsequent phases of the Green Bear project.

VI. CONCLUSIONS

Sustainable cities play an essential role in global ecosystems, involving social, environmental, and economic components. This is why the implementation of green technologies that help the operation of cities is becoming increasingly important. IoT projects for sustainable cities, where human behavior is part of the system, improve city dynamics and the quality of life of all those who live and work in them.

Despite its limitations, LoRaWAN has gained space in implementing IoT projects. Furthermore, due to their wide area communication capabilities, LoRaWAN nodes can be used for building generalized participatory systems. Scoring systems, gamification strategies, and social or economic incentives can generate more significant participation and less resistance to technologies. Therefore, technological solutions in sustainable cities must be accompanied by an easy adaptation to the environment, ease of use by people, and incentives for the citizens' involvement in the dynamics of their city. This is the case of the Green Bear case-study presented in this paper.

Further phases of the Green Bear project aim to deploy more nodes throughout the city and increase the number of users. This will require higher involvement or public/private entities and better user incentives. Developments from a technical point of view will explore alternative node wake up solutions (e.g., proximity sensors or capacitive touch), additional system functionality (e.g., more participatory activities), enhancements to the mobile application in order to improve the user experience, and the expansion of the system capacity. Once more nodes are deployed, the data provided by Green Bear can help the evaluation of the global solution from several points of view, which is also an important future work guideline. In addition, the collected data can be helpful in monitoring, controlling and designing city operational processes, with sustainability and quality of life in mind.

ACKNOWLEDGMENT

The research presented in this paper was carried out in the scope of the FikaLabs Contest and of the EEA Grants Portugal 2014-2021 - Bilateral Initiative number 63 "PrivacyCoLab".

José Marcelo Fernandes wishes to acknowledge the Portuguese funding institution FCT - Foundation for Science and Technology for supporting his research under the Ph.D. grants SFRH/BD/147371/2019.

REFERENCES

- [1] S. E. Bibri and J. Krogstie, "On the social shaping dimensions of smart sustainable cities: A study in science, technology, and society," *Sustainable Cities and Society*, vol. 29, pp. 219–246, feb 2017.
- [2] M. E. Cortés-Cediell, I. Cantador, and M. P. R. Bolívar, "Analyzing Citizen Participation and Engagement in European Smart Cities," <https://doi.org/10.1177/0894439319877478>, vol. 39, no. 4, pp. 592–626, oct 2019. [Online]. Available: <https://journals.sagepub.com/doi/full/10.1177/0894439319877478>
- [3] R. Villarroel Walker, M. B. Beck, J. W. Hall, R. J. Dawson, and O. Heidrich, "Identifying key technology and policy strategies for sustainable cities: A case study of London," *Environmental Development*, vol. 21, pp. 1–18, mar 2017.
- [4] B. Shabandri, S. R. Madara, and P. Maheshwari, "IoT-Based Smart Tree Management Solution for Green Cities," pp. 181–199, 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-15-0663-5_9
- [5] A. Lavric and A. I. Petrariu, "LoRaWAN communication protocol: The new era of IoT," *2018 14th International Conference on Development and Application Systems, DAS 2018 - Proceedings*, pp. 74–77, jun 2018.
- [6] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "Overview of Cellular LP-WAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2018*. Institute of Electrical and Electronics Engineers Inc., oct 2018, pp. 197–202.
- [7] P. J. Basford, F. M. J. Bulot, M. Apetroaie-Cristea, S. J. Cox, and S. J. Ossont, "LoRaWAN for Smart City IoT Deployments: A Long Term Evaluation," *Sensors 2020, Vol. 20, Page 648*, vol. 20, no. 3, p. 648, jan 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/3/648>
- [8] A. V. Terleev, A. A. Khalturin, and V. A. Shpenst, "LoRaWAN gateway coverage evaluation for smart city applications," *Proceedings of the 3rd 2021 International Youth Conference on Radio Electronics, Electrical and Power Engineering, REEPE 2021*, mar 2021.
- [9] J. Fernandes, D. Raposo, N. Armando, S. Sinche, J. S. Silva, A. Rodrigues, V. Pereira, H. G. Oliveira, L. Macedo, and F. Boavida, "ISABELA – A Socially-Aware Human-in-the-Loop Advisor System," *Online Social Networks and Media*, vol. 16, p. 100060, mar 2020.
- [10] S. I. Lopes, A. M. Cruz, P. M. Moreira, C. Abreu, J. P. Silva, N. Lopes, J. M. Vieira, and A. Curado, "On the design of a Human-in-The-Loop Cyber-Physical System for online monitoring and active mitigation of indoor Radon gas concentration," *2018 IEEE International Smart Cities Conference, ISC2 2018*, feb 2019.
- [11] S. Bosse and U. Engel, "Real-Time Human-In-The-Loop Simulation with Mobile Agents, Chat Bots, and Crowd Sensing for Smart Cities," *Sensors 2019, Vol. 19, Page 4356*, vol. 19, no. 20, p. 4356, oct 2019. [Online]. Available: <https://www.mdpi.com/1424-8220/19/20/4356/htm> <https://www.mdpi.com/1424-8220/19/20/4356>
- [12] C. Garrido-Hidalgo, D. Hortelano, L. Roda-Sanchez, T. Olivares, M. C. Ruiz, and V. Lopez, "IoT Heterogeneous Mesh Network Deployment for Human-in-the-Loop Challenges Towards a Social and Sustainable Industry 4.0," *IEEE Access*, vol. 6, pp. 28 417–28 437, may 2018.
- [13] M. Zappatore, A. Longo, and M. A. Bochicchio, "Using mobile crowd sensing for noise monitoring in smart cities," *2016 International Multidisciplinary Conference on Computer and Energy Science, SpliTech 2016*, aug 2016.
- [14] Y. Zhu, G. Jia, G. Han, Z. Zhou, and M. Guizani, "An NB-IoT-based smart trash can system for improved health in smart cities," *2019 15th International Wireless Communications and Mobile Computing Conference, IWCMC 2019*, pp. 763–768, jun 2019.
- [15] C. Lyu, J. Bie, and H. Wang, "DISCOVERY OF TEMPORAL VARIATIONS OF PUBLIC BIKE RIDERSHIP USING INTERNET OF THINGS FOR BUILDING SMART CITY," *Microprocessors and Microsystems*, p. 104320, aug 2021.
- [16] F. A. Almalki, S. H. Alsamhi, R. Sahal, J. Hassan, A. Hawbani, N. S. Rajput, A. Saif, J. Morgan, and J. Breslin, "Green IoT for Eco-Friendly and Sustainable Smart Cities: Future Directions and Opportunities," *Mobile Networks and Applications 2021*, pp. 1–25, aug 2021. [Online]. Available: <https://link.springer.com/article/10.1007/s11036-021-01790-w>
- [17] W. Gasparini, "Health-sport", from the European to the local level: the circulation of European norms and categories of thought," 2019. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02417011>
- [18] Pycom, "MicroPython Modules." [Online]. Available: <https://docs.pycom.io/firmwareapi/micropython/>
- [19] —, "FiPy - Pycom -Five Network Development Board for IoT." [Online]. Available: <https://pycom.io/product/fipy/>
- [20] K. E. Jeon, J. She, P. Soonsawad, and P. C. Ng, "BLE Beacons for Internet of Things Applications: Survey, Challenges, and Opportunities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 811–828, apr 2018.
- [21] L. Alliance, "What is LoRaWAN® Specification - LoRa Alliance®," 20. [Online]. Available: <https://loro-alliance.org/about-lorawan/>
- [22] A. Duda, U. Grenoble, and M. Heusse, "Spatial Issues in Modeling Lo-RaWAN Capacity," *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems - MSWIM '19*, 2019. [Online]. Available: <https://doi.org/10.1145/3345768.3355932>
- [23] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.
- [24] "Airtime calculator for LoRaWAN." [Online]. Available: <https://avbentem.github.io/airtime-calculator/ttn/eu868/64>
- [25] FIWARE, "Developers Catalogue - FIWARE." [Online]. Available: <https://www.fiware.org/developers/catalogue/> (accessed 27/09/2021)
- [26] J. Fernandes, D. Raposo, N. Armando, S. Sinche, J. S. Silva, A. Rodrigues, V. Pereira, H. G. Oliveira, L. Macedo, and F. Boavida, "Isabela – a socially-aware human-in-the-loop advisor system," *Online Social Networks and Media*, vol. 16, p. 100060, 2020.
- [27] Telefonica, "Welcome to orion context broker." <https://fiware-orion.readthedocs.io/en/master/>, (Accessed on 27/09/2021).
- [28] S. Krčo, B. Pokrić, and F. Carrez, "Designing iot architecture (s): A european perspective," in *2014 IEEE world forum on internet of things (WF-IoT)*. IEEE, 2014, pp. 79–84.
- [29] Telefonica, "Home - fiware-cygnus," <https://fiware-cygnus.readthedocs.io/en/latest/>, (Accessed on 27/09/2021).
- [30] "Flume 1.9.0 user guide — apache flume," <https://flume.apache.org/FlumeUserGuide.html>, (Accessed on 09/27/2021).
- [31] Telefonica, "Home - fiware-sth-comet," <https://fiware-sth-comet.readthedocs.io/en/latest/>, (Accessed on 09/27/2021).

- [32] F. Academy, "Iot agents - academy," <https://fiware-academy.readthedocs.io/en/latest/iot-agents/idas/index.html>, (Accessed on 09/28/2021).
- [33] G. de Internet de Nueva Generación, "Home - fiware-idm," <https://fiware-idm.readthedocs.io/en/latest/>, (Accessed on 09/28/2021).
- [34] C. Castelluccia, S. Guerses, M. Hansen, J. Hoepman, J. van Hoboken, B. Vieira *et al.*, "Privacy and data protection in mobile applications: A study on the app development ecosystem and the technical implementation of gdpr," 2017.
- [35] Google, "Activity recognition api — google developers," <https://developers.google.com/location-context/activity-recognition>, (Accessed on 09/28/2021).
- [36] —, "Sleep api — google developers," <https://developers.google.com/location-context/sleep>, (Accessed on 09/28/2021).
- [37] Plotly, "Dash Overview," 2021. [Online]. Available: <https://plotly.com/dash/>
- [38] TheThingsNetwork, "Fair Use Policy explained - End Devices (Nodes) - The Things Network." [Online]. Available: <https://www.thethingsnetwork.org/forum/t/fair-use-policy-explained/1300>
