1 2 9 0

UNIVERSIDADE Đ COIMBRA

João Pedro Novo Antunes

Intermediary Server-Based Privacy Models for Human in the Loop Environments

Dissertation in the scope of the Master's Degree in Electrical and Computer Engineering in the field of Computers supervised by Professor Jorge Sá Silva and presented to the Department of Electrical and Computer Engineering of Faculty of Science and Technology of the University of Coimbra.

February 2024

Faculty of Science and Technology of the
University of Coimbra

Intermediary Server-Based Privacy Models for Human in the Loop Environments

João Pedro Novo Antunes

Dissertation in the scope of the Master's Degree in Electrical and Computer Engineering in the field of Computers supervised by Professor Jorge Sá Silva and presented to the Department of Electrical and Computer Engineering of Faculty of Science and Technology of the University of Coimbra.

Jury:
Luís Alberto da Silva Cruz
Cristiano Premebida
Jorge Miguel Sá Silva

Coimbra, February 2024

Acknowledgments

I would like to express my gratitude to two exceptional women who have been unwavering pillars of support throughout this journey. To my mother, whose boundless love, encouragement and sacrifices have shaped me into who I am today, I am forever grateful. And to my wonderful girlfriend, whose understanding and encouragement have been my rock during the challenging times. Your unwavering belief in my abilities has kept me moving forward and I cannot thank you enough.

I also want to extend my heartfelt appreciation to all my friends who have stood by my side, offering their invaluable insights, encouragement and sometimes much-needed distractions.

A special thanks goes to my dissertation advisor, Professor Jorge Sá Silva. Your guidance, expertise and dedication to my academic growth have been truly remarkable. Your insights have helped guide through this journey and your patience has been crucial.

In addition, I am grateful to the entire faculty of Universidade de Coimbra for fostering an environment of learning and inquiry that has been crucial to my intellectual development.

Lastly, I would like to acknowledge the support of my family and colleagues who have contributed in their own ways to this accomplishment.

Thank you all for being part of this incredible journey.

Abstract

The importance of privacy is ever growing in today's interconnected world due to increased threats of illicit access and exploitation of personal information. Protecting individual privacy rights has become crucial in the developing digital landscape since numerous organizations are gathering, exchanging and keeping a lot of personal information.The goal of this work is to thoroughly investigate various privacy and security issues that people face in contemporary times. Furthermore, it intends to explore and identify alternative privacy and security approaches, while analyzing the potential function and importance of a Personal Privacy Assistant in increasing user knowledge about security and privacy. The implemented work attempts to improve the comprehension of the different issues and risks that users face while seeking to safeguard their privacy, it also proposes practical solutions that individuals can adopt to feel more in control over their data and their privacy. With an ever increasing presence of advanced technologies like Internet of Things (IoT) and Machine Learning (ML) in areas such as wearable devices and smart homes, a necessity for an evaluation and identification of potential privacy and security concerns and solutions becomes even more relevant and important.

Keywords: Privacy, Security, Internet of Things, Personal Privacy Assistants

Sumário

O crescimento da importância da privacidade no mundo conectado de hoje é exponencial devido ao aumento das ameaças de acesso ilícito e recolha de informações pessoais. A proteção dos direitos de privacidade dos indivíduos tornou-se num ponto incontornável num cenário digital em desenvolvimento, devido ao aumento no número organizações e aplicações que recolhem, trocam e armazenam quantidades enormes de informação pessoal. O objetivo deste trabalho é investigar diversas questões de privacidade e segurança que as pessoas enfrentam nos tempos atuais. Além disso, pretende-se explorar e identificar abordagens alternativas sobre como melhorar a proteção de privacidade e segurança, e, ao mesmo tempo, analisar a função potencial, assim como a importância que um assistente de privacidade pessoal pode ter de forma a aumentar o conhecimento do utilizador. O trabalho implementado tenta melhorar a compreensão das diferentes questões e riscos que os utilizadores enfrentam quando procuram salvaguardar a sua privacidade, propondo também soluções práticas e concretas que os indivíduos podem adotar de forma a se sentirem em maior controlo sobre os seus dados e privacidade. Com a presença cada vez maior de tecnologias como a Internet das Coisas (IoT) e a Aprendizagem Computacional (ML) em diferentes áreas e dispositivos como smartwatches e casas inteligentes, torna-se relevante a necessidade de uma avaliação e identificação de potenciais riscos e soluções relativamente à privacidade e segurança dos dados e dos utilizadores.

Index

List of Acronyms

API Application Program Interface.

CMU Carnegie Mellon University.

CPS Cyber Physical Systems.

DPO Data Protection Officer.

EU European Union.

GDPR General Data Protection Regulation.

GE Generic Enablers.

HITL Human-in-the-Loop.

HiTLCPS Human-in-the-loop Cyber-Physical Systems.

IoT Internet of Things.

IP Internet Protocol.

IRR IoT Resource Registry.

ML Machine Learning.

NIST National Institute of Standards and Technology.

NLP Natural Language Processing.

OSN Online Social Networks.

PEP Policy Enforcement Point.

PII Personally Identifiable Information.

PPA Personal Privacy Assistant.

List of Figures

List of Tables

1 Introduction

1.1 Context

Privacy and security concerns have taken the center stage as the world progresses and technology and the Internet infiltrate our lives. This new data-driven world collects, processes and stores enormous quantities of data, which proves to be invaluable. Recent events that target data and attempt to violate our privacy rights such as spyware like Pegasus have been wake-up calls, that remind of the urgency to protect our personal data from unauthorized access, to empower the data subject to be more in control and for organizations to be more transparent with their use of personal information. The proposal of this dissertation is to evaluate new solutions that are based on the idea of Personal Privacy Assistants (PPAs). This idea has developed as a possible response to these urgent worries aswell as regulations. PPAs are technologies that enable people to properly manage their online privacy. These assistants are made to provide users personalized recommendations and direction as they traverse the complexities of the digital world and choose the best method for protecting their personal information.

1.2 Objectives

The primary objective of this dissertation is to explore and propose a new view based on a PPA. By examining the potential of a PPA created with the aid of Xamarin, a popular cross-platform mobile app development framework, the chatbot-based PPA seeks to provide users with helpful guidance and pointers on how to secure their personal information, stay safe online and improve their overall privacy posture. The potential of PPAs to empower users has been highlighted by cutting-edge research carried out by Carnegie Mellon University (CMU).[1] It intends to provide new insights into the efficacy and usefulness of these assistants in the context of mobile applications by analyzing and extending this existing research. The aim of the study is to explore the usability of such tools in promoting online safety, safeguarding user privacy through a user-centric approach and evaluation of the created PPA for this modern world reliant on technology with an ever increasing utilization of IoT. It is also an objective to contribute to ongoing efforts to develop more privacy-aware and empowered digital citizens by comprehending the user perspective.

The study developed together with the new proposal for a different paradigm in privacy, goes hand-in-hand with the analysis and investigation of two study platforms, ISABELA and BATINA. These platforms function as practical use cases for the proposal and by evidencing the main security concerns affecting the two platforms, one can fit the devloped proposed as a mean to demonstrate the usability in a real scenario. In order improve user trust in the platforms, the proposed PPA will act as a reliable companion, providing insightful advice to improve users' overall security posture. By showing nearby IoT devices, the PPA can make the user aware of potential privacy and security risks close by.

The intention of the developed work is to serve as a foundational research and development that aims at being utilized and integrated in the two platforms by two colleagues, who are responsible for the continuous development of the platforms. The prototype implementation along with the didactic sheets also play a crucial role in creating a better understanding of the new proposal, which allows the integration of the proposed solution to the platforms. Users' confidence and trust in platforms such as BATINA and ISABELA can be considerably increased by integrating a PPA with a chatbot, that provides answers to the questions asked by the user and gives valuable information.

1.3 Methodology

The methodology used in this work involved a comprehensive approach aimed at understanding foundational concepts, analyzing key literature, evaluating existing platforms and implementing practical solutions. This methodology section outlines the specific steps that were taken to accomplish the objectives layed out.

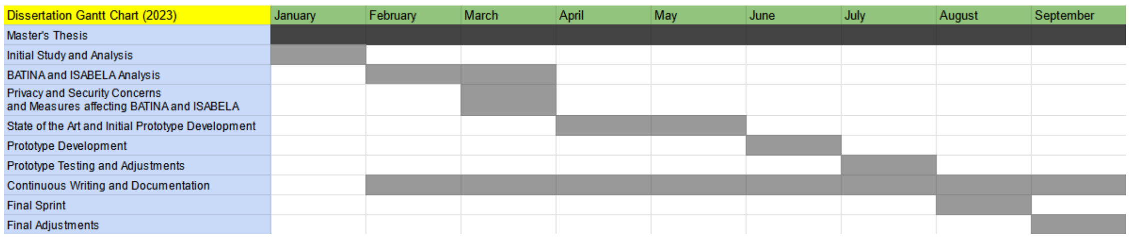
One of the crucial aspects of this work involves an in-depth review of the literature that serves as the foundation for this work. This evaluation included a study of the research topic's underlying ideas, with a special emphasis on comprehending the principles of data privacy and user rights. Giving thorough thought and examination of the General Data Protection Regulation (GDPR) provided a solid understanding of the legal obligations as well as rights and best practices for protecting user privacy. It also included a careful assessment of a study on PPAs done by Carnegie Mellon University. This research was essential in understanding the possible benefits, problems and practical elements of incorporating privacy-enhancing technology into applications.

To gain a better perspective on data security and user privacy concerns, two unique study platforms, ISABELA and BATINA, were chosen for a deeper analysis. Each platform was examined to identify potential risks and areas for enhancement in terms of data protection and user privacy. This process involved a detailed exploration of the platforms' architecture, data handling mechanisms and interaction.

The theoretical insights gained were translated into a practical solution through implementation. A practical approach was adopted using the Xamarin frameworks, along with Python and Arduino programming languages to develop the application components necessary for the study. Xamarin was used for the development of a mobile application, while Python was utilized for the implementation of the intermediary servers that integrated the various components. Not excluding the work developed using Dialogflow for the creation of a functional and easy to use chatbot.

This work was developed using an iterative development approach, allowing for constant refinement and enhancement of the overall research. At the same time, a thorough analysis of relevant literature was conducted, gathering valuable insights from testing phases. Regular meetings and recommendations by Professor Jorge Sá Silva were crucial to a linear and constant evolution of the work. These iterative cycles, in union with the developmental phases, contributed significantly to both the features of the practical work and the development of a text that is both comprehensible and professionally presented.

This process was established right in the beginning of the work, with the creation of a Gantt chart that was adjusted throughout the development of the dissertation. The final Gantt chart can be observed below.



Dissertation Gantt Chart (2023)	January	February	March	April	May	June	July	August	September
Master's Thesis	■	■	■	■	■	■	■	■	■
Initial Study and Analysis	▨								
BATINA and ISABELA Analysis		▨	▨						
Privacy and Security Concerns and Measures affecting BATINA and ISABELA			▨						
State of the Art and Initial Prototype Development				▨	▨				
Prototype Development						▨			
Prototype Testing and Adjustments							▨		
Continuous Writing and Documentation		▨	▨	▨	▨	▨	▨	▨	▨
Final Sprint								▨	
Final Adjustments									▨

Figure 1: Gantt Chart depicting the dissertation development

1.4 Document Structure

The dissertation is structured in a logical and progressive way. The different sections and their logic are described below:

- Chapter 1 serves as an introduction to the dissertation, a context regarding the work that was developed throughout is provided, aswell as defining clear objectives and indicating a methodology.
- Chapter 2 introduces key concepts, technologies and related research. It covers all the foundational concepts, the tools and frameworks used in practical work, aswell as a a review of state-of-the-art research to support the project.
- Chapter 3 studies two different platforms, ISABELA and BATINA, it analyzes their architecture, what they are used for and how they work.
- Chapter 4 presents the final culmination of the analysis and study. It introduces a new proposal involving a PPA and intermediary servers, enabling user interactions with chatbots and IoT devices while enhancing anonymity and security. The chapter also discusses how this proposal addresses issues related to BATINA and ISABELA.
- Chapter 5 provides a conclusion to this dissertation by presenting a retrospective about the research findings and their practical implementation. It serves as a reflective analysis of the work developed while at the same time laying the groundwork for future possibilities and potential enhancements.

2 Concepts and Technologies

2.1 Foundational Concepts

Internet-of-Things

Internet-of-Things, better know as IoT, is defined by as a system of interrelated computing devices, objects or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. A *thing* in IoT can be anything ranging from a heart monitor to a smartwatch that can be assigned an IP address. [2]

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware to collect, send and act on data they acquire from their environments. [2] They then share that sensor data by connecting to an IoT gateway or other edge device where data is either sent to the cloud or used locally.

IoT devices and the data that they collect provide great convenience and benefits to consumers, but that comes at the expense of large quantities of personal information collected about users and their activities. The types and quantities of data collected varies greatly depending on the situation, but it ranges from information about a user's heart rate, location, fatigue level, concentration and much more. This large amount of data could benefit everyone through better research, with more information being available to be studied and ending up improving the common person's life, but it can also cause harm if used inappropriately, such as by an insurer raising the premiums of users with high blood pressure.[3]

Multiple sensors in close proximity, can combine their data in a process known as sensor fusion, which allows for more accurate and specific inferences that would not be possible with the data from a single sensor. Inferences can be extremely useful for a range of purposes, such as knowing the attention state of an individual and warn the user to return focus, but these inferences can also be highly personal and unexpected. Individuals are generally uncomfortable with organizations using IoT data to infer information about them, with user preferring anonymous data collection. [4]

As stated above, users have different attitudes to different circunstances of data that is being collected and processed, the following figure can bring some insights into what are some of those attitudes regarding different factors such as Data Type, Device Type, Location, Retention Time and Inferred Data[4].

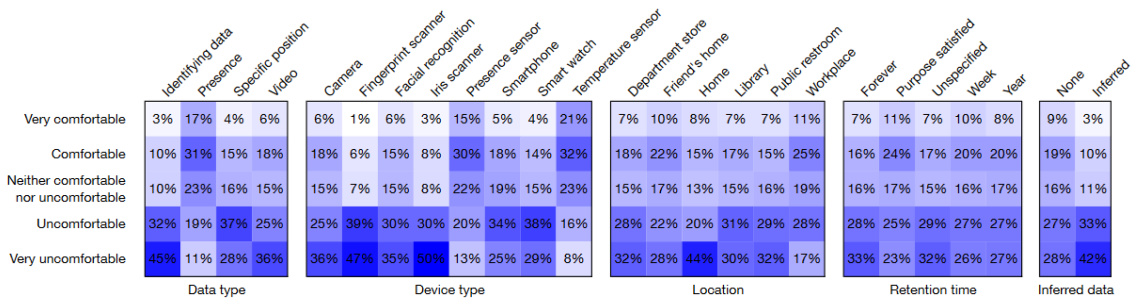


Figure 2: Statistics showing the relation between various factors and participants' comfort level.[4]

The platforms that were studied, also take advantage of IoT devices and the data that they gather and process to assist the user and provide insights and tips that aim at improving the user's life and well-being. But as discussed above, this has be done in a clear and transparent way, to give the user confidence about how one's data is being collect and used.

Privacy

Privacy is a fundamental human right that underpins freedom of association, thought and expression, as well as freedom from discrimination.[5] It is defined by the NIST as an assurance that the confidentiality of and access to, certain information about an entity is protected. It is also the freedom from intrusion into the private life or affairs of an individual, when that intrusion results from illegal access. Additionally it is the right of a party to maintain control over their data.[6] Privacy includes the following rights[5]:

- Right to be free from interference and intrusion
- Right to associate freely with whom you want
- Right to be able to control who can see or use information about you

And there are different ways to look at privacy, them being:

- Physical privacy
- Surveillance
- Information Privacy (how personal information is handled)

With the world becoming increasingly digitized, the old risks become even more relevant and new concerns arise. The unethical practices of Big Tech companies such as Amazon, Google, Apple, Meta and Microsoft have faced scrutiny from the public. As an example, users from 40 different states in the United Sates sued Google accusing the company of misleading users. They believed that the company would stop collecting their information after a location tracking was turned off in their settings. However, that did not happen and Google continued to collect this sensitive information. [7]

As it is possible to observe, the problems affecting the privacy of the users are serious and it is a common belief that people have lost their ability to be confident about their privacy online. The proliferation of IoT devices have increased this problem, these devices can collect and process a multitude of data in large quantities, the types of data include all kinds of personal information, behavioral patterns, biometric data, localization and so on. One of the key issues is that IoT devices are able to collect and process data without implementing proper security and privacy measures, being black boxes for the user. User's may not be fully aware of the scope and nature of data collection, thus leaving them vulnerable and unable to properly safeguard their privacy. This excessive accumulation of personal data is problematic, increasing even further risks of unauthorized access, misuse and overall privacy violations.[8]

This lack of explicit consent and transparent data processing practices only exacerbates these concerns. Users have the right to be fully informed, with companies providing clear, concise and transparent privacy policies that protect the user and his rights. This potential misuse of personal data can adversaly affect individual autonomy, freedom and general well-being, impacting the public trust in technology.[8] This discussion on privacy and IoT, aswell as the concerns regarding the enormous volume of data-collecting devices along with the prevailing lack of transparency and user consent, not only accentuates the risks but also increases the scope and capabilities for malicious/threat actors. The proliferation of such an immense quantity of information and data exacerbates the potential dangers.

Privacy is one of the key discussion points throughout this disseration and it is explored in detail some of they principal aspects regarding the major concerns aswell as providing with possible solutions to attenuate these risks and giving to the user more control and confidence.

Machine Learning

Machine Learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn.[9]

Using the capabilities of ML, researchers are able to make new tools and materials that help people in their lives, such as personal assistants like the ones that will be discussed later in the dissertation. But these new data collection and analytics methods are beginning to see more and more laws and regulations that place restrictions on what reasons data can be collected and used for. A major example is the European Union General Data Protection Regulation, GDPR includes the following guiding principles that apply to ML.[10]

- Purpose limitation - This states that data subjects must be informed about the purpose of data collection and processing.
- Data minimization - Data that is collected should be adequate, limited and relevant.
- Transparency - The focus is on empowering data subjects to decide which of their data is used by third-party data controllers.

The Machine Learning topic is important, because these models often require large amounts of data and sometimes personal data, which is the case for the study platforms that will be analyzed.

Chatbots

Chatbots can be defined as computer programs that leverage AI and Natural Language Processing (NLP) to understand user questions and automate responses. The goal is to simulate human-like conversations.[9]

There are lots of chatbots available, that are user-friendly and simple to use and setup, some of the more well known are:

- IBM Watson
- DialogFlow

Chatbots bring lots of benefits that can be applied in many fields and applications. Firstly, they improve the user's engagement, by managing the interactions 24x7, they eliminate the waits that user's had to face with email or phone support. [9] This is very useful, by providing a seamless and always available help, the use of chatbots, framed inside the work being developed in this dissertation, is capable of addressing the user's concerns and provide valuable information on how he can be more secure and improve the privacy posture, thus increasing a user's overall trust and confidence in the system. This flexibility and availability are two of the main advantages of using chatbots.

Human-in-the-Loop

Human-in-the-Loop (HITL) is a mix of supervised ML and active learning, where humans take part in both the training and the testing stages of the construction of the algorithm. This creates a feedback loop that allows the algorithm to produce increasingly better results. HITL primes itself on being very versatile in its applicability.[11] With HITL there is an acknowledgment that AI systems are not standalone entities but rather tools designed to assist humans, leveraging their expertise, preferences and ethical considerations to enhance the decision-making process. By combining human judgment with AI capabilities, HITL creates more effective and meaningful AI systems.[12]

One of the key benefits of incorporating HITL is transparency. With humans involved at critical decision points, there is a demand for systems to be designed in a way that is understandable to humans. This increased transparency helps to reveal the underlying logic and reasoning of AI systems, making it harder for the process to remain hidden.[13]

In addition to transparency, HITL design strategies bring human judgement and preference into the loop. While AI systems aim to assist humans, the value of such systems lies not only in efficiency or correctness but also in human agency. By involving humans in decision-making, HITL systems ensure that humans have the opportunity to contribute their expertise, preferences and ethical considerations.[13]

In the realm of machine learning, HITL involves humans in the training and deployment of models. Human feedback helps refine the algorithms, verify their accuracy and suggest improvements. This collaborative approach enhances the user's experience by tailoring recommendations and results to individual preferences.[12]

This concept is very important to this work since the study platforms are dependent on HITL to provide the user with valuable feedback, tailored to the user's own preferences, lifestyle and conditions. Without this personalization, lifestyle assistants could not improve in a significant way the feedback returned to the user.

2.2 Technologies Used

In this subsection, different tools used throughout the work are explored, stating the reasons why they were chosen aswell as describing them and how they fit the work that needed to be done.

Xamarin

Xamarin is one of the tools used in the project. To build and integrate the PPA easily with the platforms that were studied, a mobile application is the logical step, providing easy and seamless integration and access to a user. By building a mobile application that can be executed in both Android and iOS is an invaluable asset that makes it extremely useful to this work. Serving as a binder to different features like the Dialogflow chatbot and the IoT device listing service, it provides with clean graphical interfaces for the user and an easy and straightforward development. Xamarin.Forms allows developers to create user interfaces in XAML with code-behind in C#. These interfaces are rendered as performant native controls on each platform. It is a cross-platform framework that enables programmers to design user interfaces and develop mobile applications that work both on Android and iOS.[14]

Dialogflow

One of the main principles of a Personal Privacy Assistant is to assist the user by advising him on how to improve its security posture and be an all around better data subject, conscious of his rights. To achieve this, one of the key technologies that can help is a chatbot. A chatbot will provide with the natural language and understanding so that the user can feel like his worries and questions can be answered and are a priority. In this work, Dialogflow is used for this purpose, as it allows the construction of a functional and conceptual chatbot that can be upgraded and updated to fit whatever the specific needs are, in this case, the chatbot provides answers to user's queries regarding their privacy and how they can become better protected. Diagloflow as a technology is a cloud-based conversational AI platform developed by Google. It is a natural language understanding platform that makes it easy to design and integrate a conversational user interface into your mobile app, web application, device, bot, interactive voice response system and so on.[15] Dialogflow uses machine learning and natural language processing (NLP) to understand user input and reply to it in a way that is human-like. The Dialogflow API was explored and used to bring the chatbot funcionality to the PPA mobile application, being integrated with the developed Xamarin.Forms application.

Python

Python's own documentation tells us that "Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms."[16] This is exactly why it is an important factor in the development of the practical work in this dissertation. Python and the Flask framework allows for the creation of simple and easy to use scripts that interact with different APIs and are capable of receiving, storing and transmiting information in a concise way. Flask is a web framework and by being imported into a Python script, permits the development of web applications in an easy way.

Arduino

Arduino is an open-source electronics platform based on easy to use hardware and software.[17] The arduino boards have different uses and for the development of this work, an Arduino board is used as a simulation of an IoT device. The Arduino MKR 1000 comes with Wi-Fi access that allows it to connect to servers and send relevant information. The prototyping is quick and the extensive documentation available online facilitates the development. This work utilizes Arduino, a flexible open-source hardware platform. It consists of a microcontroller board and a development environment that supports the design of interactive electronic projects. The development board used was an Arduino MKR 1000 that provides crucial Wi-Fi access. This allows for the communication with the Python server.

2.3 State-of-the-Art

2.3.1 GDPR

One of the main focus of this dissertation is pertaining to data privacy and security. GDPR is one of the most well known privacy and security law worldwide. The General Data Protection Regulation, commonly referenced as GDPR, was enacted by the European Union on May 25, 2018. Even though this law was passed by the European Union (EU), the obligations are imposed onto all the organizations, despite their location, that target or collect data related to people in the EU.[18]

The roots of GDPR can be traced to the 1950 European Convention on Human Rights, which states "Everyone has the right to respect for his private and family life, his home and his correspondence". As technology rapidly evolved, there was an urging need to apply modern protections and in 1995 the European Data Protection Directive was passed, establishing minimum data privacy and security standards. But technology kept evolving and morphing into the data-driven world it is today, with the evolution came the need for an updated directive to address new challenges. In 2016, GDPR was passed in the European Parliament and as of May 25, 2018, all organizations were required to be compliant.[18]

Any organization that processes personal data, offer goods or services to EU citizens or residents must comply with GDPR, since they fall under the regulation's scope. Violating the GDPR comes with significant fines that can reach up to 20 million euros or 4% of global revenue, plus data subjects have the right to seek compensation for damages.[18] GDPR also defines different legal terms and here are some of the most important terms:

- Personal data - Any information that relates to an individual who can be directly or indirectly identified. This includes names, email addresses, location information, ethnicity, gender, biometric data, religious beliefs, web cookies and political opinions are all types of personal data.
- Data processing - Any action performed on data, such as collecting, recording, organizing, structuring, storing, using, erasing.
- Data controller - Person who decides why and how personal data will be processed.
- Data processor - Third party that processes data on behalf of a data controller.

GDPR emphasizes seven core principles on data protection that are outlined in Article 5.1-2[18], these are:

- Lawfulness, fairness and transparency - Processing must be lawful, fair and transparent to the data subject.

- Purpose limitation - You must process data for the legitimate purposes specified explicitly to the data subject when you collected it.
- Data minimization - You should collect and process only as much data as absolutely necessary for the purposes specified.
- Accuracy - You must keep personal data accurate and up to date.
- Storage limitation - You may only store personally identifying data for as long as necessary for the specified purpose.
- Integrity and confidentiality — Processing must be done in such a way as to ensure appropriate security, integrity and confidentiality (e.g. by using encryption).
- Accountability — The data controller is responsible for being able to demonstrate GDPR compliance with all of these principles.

The General Data Protection Regulation places significant emphasis on the data controllers accountability. It is required to demonstrate that it is compliant, as it is stated: "If you think you are compliant with the GDPR but can't show how, then you're not GDPR compliant." There are different ways to be accountable and show compliance, such as designating data protection responsibilities, maintaining detailed data documentation, implementing security measures and having Data Processing Agreement contracts with third parties handling data. And in certain conditionns, appointing a Data Protection Officer (DPO).[18] There are three conditions under which it is required to appoint a DPO:

- You are a public authority other than a court acting in a judicial capacity
- Your core activities require the monitoring of people on a systematic and large scale
- Your core activities are large-scale processing of special categories of data listed under Artice 9 of the GDPR or data relating to criminal convictions and offenses mentioned in Article 10.

It is possible to desginate a DPO even if it is not required. Their basic tasks would involve a deep understanding of the GDPR and how it applies to the organization, advising people, conducting data protection training, audits and monitoring compliance.

Data security is considered as a key aspect of the GDPR, it is necessary to implement the appropriate technical and organizational measures to safeguard personal data. These technical and organizational measures may include two-factor authentication for employees, encryption for data storage and staff training on data privacy policies. It is also very important to know that organizations must report any data breaches to the affected data subjects within 72 hours to avoid penalties

GDPR also requires organization to think of data protection "by design and by default".[18] This means that the implementation of the appropriate measures and the integration of the data protection principles must be embedded throughout the development process by minimizing data collection and enhancing security.

The GDPR outlines several lawful bases for data processing, including:

- The data subject gave you specific, unambiguous consent to process the data.

- Processing is necessary to execute or to prepare to enter into a contract to which the data subject is a party.
- You need to process it to comply with a legal obligation of yours.
- You need to process the data to save somebody's life.
- Processing is necessary to perform a task in the public interest or to carry out some official function.

Consent under the GDPR must be freely given, specific, informed and unambiguous.[18] Organizations must obtain clear and plain language consent, allowing data subjects to withdraw their consent at any given time. Additionally, there are specific rules that apply when the processing of data is of children under 13.

Data subject have numerous privacy rights that are recognized by the GDPR to empower them and give more control over the data. These rights include:

- The right to be informed
- The right of access
- The right to rectification
- The right to erasure
- The right to restrict processing
- The right to data portability
- The right to object
- Rights in relation to automated decision making and profiling.

2.3.2 Personalized Privacy Assistants

PPAs are personalized assistants that help guide a user throught the difficulties of managing personal data an exercise the data subject's rights. A key work done by Carnegie Mellon University [1] highlights the importance of Personal Privacy Assistants. Privacy of the data is a crucial aspect when it comes to protecting people's personal data and giving them more control over how their data is collected and used. Most individuals do not take or have time to read through data privacy policies, something that becomes even more complex by the ever increasing number of smartphones and IoT devices around us aswell as the number of applications used everyday. This lack of control has resulted in a sense of loss of control and trust over their data among users. With PPAs, users can learn about nearby IoT resources, their data collection and use procedures and associated privacy settings. A key component of this infrastructure is the creation of IoT Resource Registries (IRRs), where resource owners can promote the availability of IoT resources and the data practices associated with them, allowing the user to feel more in control and informed about the data that is being collected about him.

Information privacy is a key aspect in regards of protecting people's personal data and giving them actionable choices about how their data is collected, processed and used. However, few individuals take the time to read privacy policies and even less actually end up exercising their rights, which can be further complicated by the proliferation of smartphones and IoT devices which makes everything more cluttered and confusing for the average user.

A new scalable paradigm that gives users back control over their data is a crucial aspect to help solve this issue. Users can learn about nearby IoT resources, their data collection and use procedures and any associated privacy settings with the aid of PPAs [1]. By leveraging machine learning, these PPAs create models of users' privacy expectations and preferences, enabling them to selectively inform users about the data practices they care about and assist them in configuring their privacy settings.

A key component of this infrastructure is the creation of IRRs, where resource owners can promote the availability of IoT resources and the data practices associated with them. Web portals and resource templates are part of the infrastructure, which is meant to make it easier for resource owners to fill out IRR entries. IRRs promote the data collection and use procedures used by registered resources, allowing PPAs to find them and educate their users about the procedures and options involved.

The functionality of unified permission management allows users to select the apps they want to install on their smartphones and to manage the permissions that enable those apps to function. Users, however, frequently lack awareness of and control over the resources they use and interact with technologies that they did not deploy. Users find it challenging to exercise their right to notice and choice due to this lack of awareness. An infrastructure that enables users to find nearby IoT resources and their data practices is required to solve this problem. Additionally, the infrastructure should enable users to access settings for configuring privacy options like opt-in and opt-out settings, as well as information about the data collected by IoT resources and how it is used.

It is suggested a three-part IoT privacy infrastructure to address this key issue: PPAs for IoT, IRRs and Policy Enforcement Points.

Any number of actors may participate in the deployment and management of IoT resources thanks to the infrastructure's open and distributed architecture. IoT systems may be deployed by various parties for a variety of purposes, such as security, marketing and public health monitoring, including businesses, cities and even homeowners. The infrastructure is required to inform users about data collection processes and to give them some degree of control over these processes as connected devices designed to collect potentially sensitive data are deployed.

Owners of IoT resources have two significant financial incentives to contribute to the infrastructure. Initially, to adhere to current and impending regulations like the EU's GDPR, which requires the deployment of infrastructures like the one suggested. Second, IoT resources can be promoted using the infrastructure. IoT resource owners can publish and share descriptions of their IoT resources, including the methods they use for data collection, through an IoT resource registry. Owners of resources can also promote control options that let users limit how their data is used, such as the ability to erase data, limit the retention period, opt in or out and more.

The Internet of Things Personalized Privacy Assistant is a smartphone application [19] that directs users to nearby IoT resources and services. It generates a privacy notice for each resource and retrieves resource listings from IRRs based on the user's location. The PPA provides a list of the resources that are available and explains to the user how they work, who owns them and how they handle data. The authors believe that over time, IoT PPAs will be able to learn a user's privacy preferences and alert them to practices that matter to them. If any privacy settings are offered, they can also assist users in configuring them. PPAs have the discretion to choose what information about nearby IoT resources to display to users and when. They can identify inconsistencies between a user's expectations of privacy and the policies of the resources they use. Such inconsistencies can alert specific users and semi-automatically modify the privacy settings that are available to them. Where such settings are available, PPAs could also semi-automatically configure privacy settings on the user's behalf. A prediction is made that user-configurable privacy settings will become more common over time, in part due to new laws like the GDPR. PPAs create models of the data collection and use practices that various users expect and the data practices they are familiar with.

Owners of IoT resources or data collectors must disclose various privacy settings to their users in order to abide by laws like the GDPR. Policy enforcement functionality that can save user preferences for privacy settings and enforce them is required to enact these settings. This is possible through IoT resources that already have built-in policy enforcement capabilities or through third-party policy enforcement capabilities, like a Policy Enforcement Point (PEP).

For instance, users of cameras with facial recognition capabilities might be given the choice to disable the feature at certain times of the day or in certain places. According to the article, some IoT resources will have built-in policy enforcement capabilities, while others will need external policy enforcement capabilities, like a PEP. Based on user-configurable privacy settings promoted by IoT resources via IRRs, the PEP would manage the collection and use of user data. Typically, the privacy settings would be presented as APIs, such as opt-in or opt-out APIs.

It is possible to describe the infrastructure as owners of IoT resources list their resources in an IRR and provide descriptions of those resources using predefined templates. Then, through APIs that communicate with the PEP enforcing the settings, users can use their PPAs to find resources nearby and customize privacy settings. All pertinent parameters, including the URL for the PEP's API, are included in the IRR entry for a resource. The PEP makes sure that users' chosen privacy settings, such as preventing facial recognition when a user has opted out, are properly applied to data streams coming from the resource. Even a basic generic PEP supporting opt-in and opt-out functionality can give users some control over how their data is collected and used by IoT technologies, despite plans for an extensible collection of privacy controls.

This proposal marks a significant change in how data is managed, especially in relation to the efforts to safeguard user privacy. The main idea is to give users more control over their personal data. Users can decide exactly what information they allow to share and what should stay private, giving them more control. This challenges the usual way data is collected, which often makes users feel like they have no say in the matter. However, in the proposal developed in this dissertation, the introduction of a chatbot system aswell as intermediary servers are what set the difference. This chatbot keeps users engaged and involved. It changes the paradigm by providing users with a more interactive way to communicate, learn and make decisions. When this approach is observed closely, it not only gives users more control but also takes on the problems of data misuse and unfair data collection practices. Additonally, by combining the ideas from CMU with the new concepts in this work, it is being created a link between established research and new ways of tackling the issue.

3 Study Platforms

In this chapter it is provided a comprehensive analysis of the architecture of ISABELA and BATINA platforms. Official resources and academic papers are explored, aimed at gaining a better understanding of how these platforms operate and the role played by the different features and frameworks used. Later, it will be addressed the potential privacy problems affecting these platforms, aswell as possible solutions and mitigations that go along with the developed proposal.

3.1 ISABELA

ISABELA stands for "IoT Student Advisor and Best Lifestyle Analyser," according to the acronym. [20] ISABELA is a socially aware HiTLCPS (Human-in-the-loop Cyber-Physical System) advisor system that combines different sensing data, including physical, behavioral and social network data.

It uses smartphones, smartwatches and IoT boxes as sensing mechanisms in an unobtrusive and passive way and chatbots or interactive agents, complete the circle. Over the course of two real-world trials that lasted four weeks, ISABELA was tested on 30 students from Escuela Politécnica Nacional in Ecuador and 10 students from the University of Coimbra in Portugal. A Natural Language Processing module that uses OSN (Online Social Networks) as a source of context-aware data is part of the system, which is built on open-source Internet of Things solutions, technologies and standards as well as interactive agents.

The contributions of ISABELA include an in-depth look at the implementation of a socially conscious HiTLCPS advisor system, the successful fusion of HiTLCPS and OSNs, a dataset created after a month of data collection, implementation details and outcomes of three machine learning models and the identification of open research avenues. Overall, ISABELA offers a thorough and integrated approach to human-centric IoT and CPS, taking into account social context and interactions and offering significant potential for additional research and development in this field.

Although it was created with students in mind to help them perform their best in the classroom, it can be applied to any type of user in a variety of situations. The system's goal in the student scenario is to use students' states and behaviors to infer their potential academic performance, identify students who are failing and suggest potential improvement strategies. The system also aims to identify students who are succeeding and motivate them to keep going or even get better. The ISABELA CPS (Cyber-Physical System) deduces the user's emotional state by combining data from various sources, including OSNs. The OSN-based NLP module, which is essential for determining emotional state, as well as the overall ISABELA system architecture are all covered in detail.

Orchestration

The Internet of Things encompasses more than just actual electronic-based objects, it also includes software-based things like agents and even actual people. "Human-based sensors" gather data from people to gain insights into their emotions and mood by treating human-originated activities from social media as raw data.

Natural Language Processing methods are being used as "translators" to interpret that collected raw data and convert it into values that can offer information and valuable knowledge in order to make sense of this data. The ISABELA NLP module's focus is on sentiment and emotion analysis.

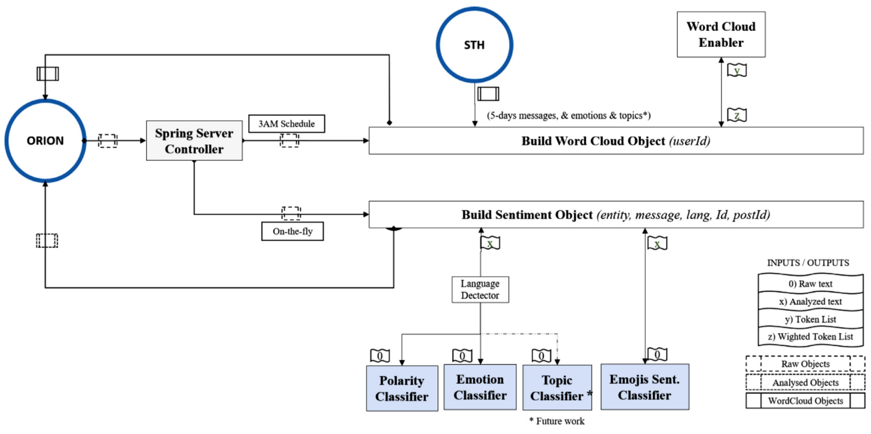


Figure 3: ISABELA Architecture Orchestration

NLP techniques are used to gather information about users' moods by inferring sentiment and emotions from texts posted by ISABELA users on well-known social media sites like Facebook and Twitter. Sentiment analysis gives a text a value in the range of -1 to 1, with 1 denoting extremely positive mood/emotional state and -1 denoting extremely negative mood/emotional state. Emotion analysis goes one step further by categorizing the text's contents according to a range of emotions, such as fear, satisfaction, annoyance, boredom, excitement, happiness, etc.

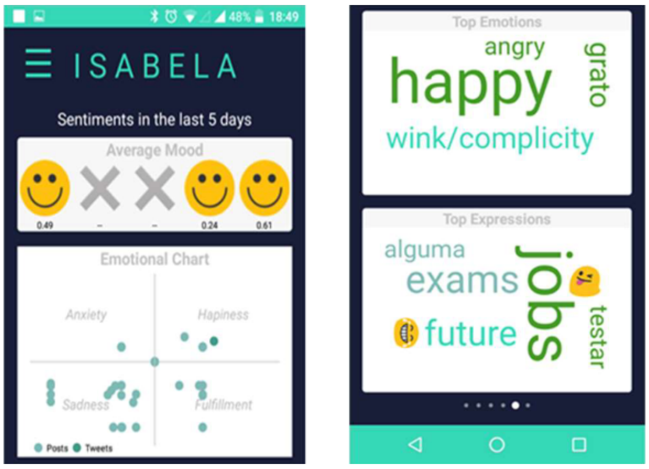


Figure 4: Sentiment Analysis Visualization

Text that has been preprocessed with linguistic techniques like tokenization, speech tagging and lemmatization is the input to the NLP system. By dividing the sentence into phrases that each contain just one opinion, the presumption that each sentence contains a single opinion can be expanded. Tokenization separates out each text element, whereas POS Tagging determines how grammatical each element is. In terms of emotions, adjectives are thought to be more illuminating. Lemmatization reduces the complexity of text analysis by restoring all elements to their original state.

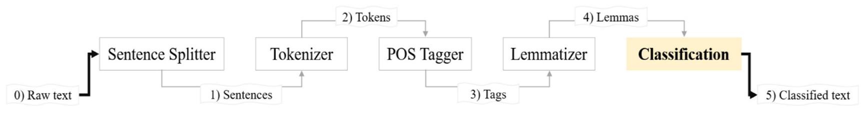


Figure 5: ISABELA Input Processing

Different techniques are applied in the classification unit to assign sentiment scores or emotion tags to the sentence. However, complex design and computations are needed to analyze sentiment and emotions in a text and this has not yet been adequately addressed in the literature. Complexity like sarcasm, noisy texts, emojis and slang must be handled by the system.

The NLP outcomes may be attatched to complete documents (for sentiment based on documents), to specific sentences (for sentiment based on sentences) or to particular entities (for aspect-based sentiment). In order to create the most basic classification unit, a collection of lexicons are used as linguistic resources. Lexicons are books that list the emotions or sentimental values associated with the things that are supposed to stand in for a written language.

The intrinsic ”goodness,” ”neutrality,” or ”badness” of a token is expressed by its valence. Arousal expresses the level of excitement attached to the token. These scores can be added together to provide a more detailed understanding of a person's emotions and mood.

FIWARE Platform

The ISABELA system makes use of a number of Generic Enablers (GE) from the FIWARE project, which were created with clearly defined Application Program Interfaces (APIs) aimed at particular IoT-related issues. All different kinds of IoT devices were managed by the IDAS GE to deal with the sensing state. The platform was linked with a variety of IoT devices found inside ISABELA sensing boxes using the UltraLight agent. Data on temperature, humidity, noise and luminosity in the user environment can be collected using the ISABELA boxes. These boxes were meant to be put in the student scenario's classrooms and homes in order to assess how much the environment affected the outcomes of the students.

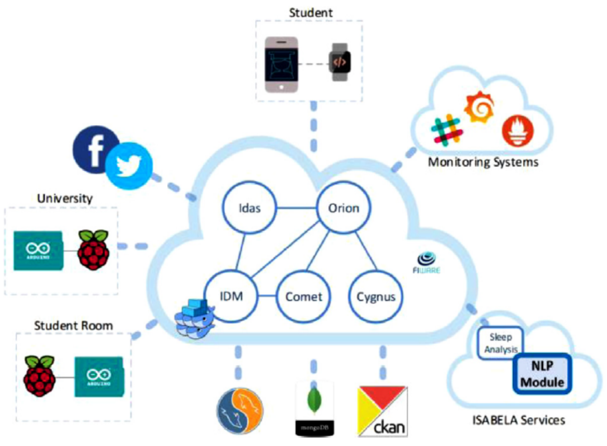


Figure 6: ISABELA Architecture

NGSI Information Model

The FIWARE project adopts the NGSI9/10 information model to create a dynamic ecosystem. This model is built on entities and attributes, where each entity has a type and is represented by JSON-formatted attributes. The ORION generic enabler is used to manage all entities in the ISABELA Android application, which all use the NGSI representation. It is possible to create, delete, retrieve and update existing entities using this API. By serving as a broker, ORION enables external systems to subscribe to data from entities with particular rules and attributes. Using Apache Flume, the Cygnus module is the connector in charge of persisting specific data sources across multiple storages.

Short-Term History (STH-Comet) GE

Typically, databases do not offer APIs for applications to retrieve data. Because of this, the Short-Term History or STH-Comet GE offers a RESTful API with historical-query capabilities and aggregated methods. As a result, the Comet GE connects to the database and retrieves the requested data each time the ISABELA application or the NLP module needs to access historical data.

Security and Privacy

For HiTLCPS systems, security and privacy are essential requirements. The Identity Management Generic Enabler, also known as Keyrock or IDM, was used to secure communications between GEs, IoT devices and applications. Keyrock enhances devices, users and applications with authentication, security and authorization policies. Additionally, all data handled by the Android application is anonymized in order to comply with all EU privacy laws.

Sleep Analysis and NLP Module

Two services are offered by ISABELA: a sleep analysis module and an NLP module. Based on information gathered by smartphone sensors, the former part calculates users' sleep duration. The latter employs natural language processing to identify expressed sentiments and emotions using text data gathered from OSNs. The ”OSN-based Natural Language Processing” section focuses on the latter module.

Data Processing and Anonymization

The application gathers information about how the phone is used, including calendar data, contact data, SMS/call data, screen lock data, Wi-Fi data, Bluetooth data, alarm data and app usage data. To protect user privacy and maintain confidentiality, the collected data is anonymized and hashed. Replaced with computed values and ID numbers, respectively, are Wi-Fi SSID and GPS location values. When the app is uninstalled, the persistence database on the smartphone is where all of the user and chatbot's messages are stored and are deleted. The user's Facebook account is used as the authentication method to prevent unauthorized access to the mobile application.

3.2 BATINA

BATINA is a modern approach to education that makes use of IoT, mobile devices and artificial intelligence. BATINA aims to offer assistance for online theoretical lessons and individualized instruction, even in large-class settings. BATINA provides a real-time tool to adapt lessons based on student needs and brings the classroom closer to the student, according to their unique profiles.

A chatbot, motivation levels and feedback to professors and students are just a few of the new functionalities that BATINA offers as a starting point. It was created using the FIWARE Back-End system and can be used to advertise courses and jobs. BATINA offers in-class questions for top students, challenging subjects and failing students and is accessible through students' phones and a professor's dashboard. The answers are saved as passive data and the OAuth2.0 system uses 2-step authentication and end-to-end encryption to protect student information.

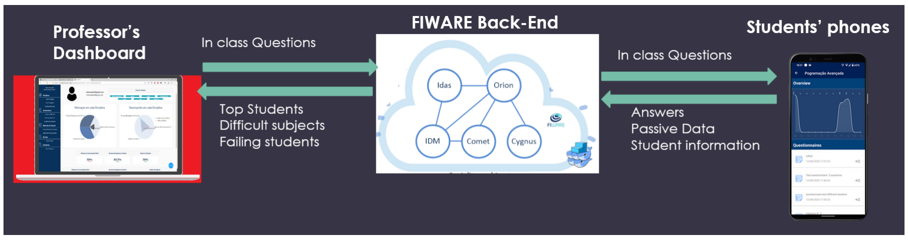


Figure 7: BATINA Architecture

Professors can create courses for each of their lectures in BATINA or split a lecture into two courses. BATINA is intended to represent real-world courses or classes. Professors can easily create, manage and add resources for each individual course using the dashboard. Students can easily add new courses by copying and pasting the access code into the BATINA application, getting access to all the information and materials from their courses.

Questionnaires provided by the BATINA system are a great way to assess how well students comprehend a subject. All participants in the course are informed of the creation of the questionnaires, which are linked to that course. Professors can specify the day and time the questionnaire will be made available for completion. There must be at least one question in each questionnaire, but there is no maximum limit. The questions can be presented in the current order, randomized or pooled. The professor can design questions that require the student to present an image or include mathematical expressions, define support materials for each question and make an image available to the students.

Every time a new survey is made for one of their courses, students are automatically notified and they can only respond to the survey during the designated time. The amount of time left until the questionnaire is accessible is displayed in a countdown. Once a questionnaire has begun, the student cannot stop it and can only proceed through it. To increase reliability, the solutions are kept locally in case the Internet connection is lost. The number of students who responded to the survey and their responses can both be monitored in real time by professors. Additionally, a success rate for each question is shown on the screen and updated with each response. A report for each course with aggregated data can also be generated after the completion of a questionnaire, but an instant report containing information about each student's results is generated instead. Through their personal reports, students can review their test results, examine the correct answers they provided and view the specifics of each question.

Within the mobile App, BATINA offers a direct line of communication as well. Students can ask a question to the course owner directly through this channel and professors will be notified by email and can respond on the dashboard. As soon as the question is cleared up, the student will also be informed. The last step taken by BATINA is the implementation of a chatbot that automatically informs the students and prompts them with information about the questionnaires on which they performed poorly.

4 New Proposal, Implementation and Tests

This chapter is dedicated to the development of the Personal Privacy Assistant. It aims to provide a comprehensive understanding of the project by discussing its context and introducing the implementation of the initial prototype. Through the use of relevant code snippets and practical examples, the functionality of the prototype is explained aswell as the analysis of its intended objectives in a clear and concise manner.

4.1 Work Context

In previous chapters the benefits of a PPA were enumerated, analyzed and explored. The part dedicated to the work done by CMU regarding this topic is invaluable in describing the usefulness and the enriching capabilities that a PPA can have to bolster the confidence of the user regarding their safety and data privacy. This work is intended to be integrated with the two platforms that were studied, BATINA and ISABELA to provide the users of those platforms with a chatbot that can provide answers to their questions regarding security and privacy aswell as giving access to important information about which IoT devices are present nearby and are collecting data about the user, all of this can improve a user's trust in the platform, giving a sense of transparency and more control over the data.

This proposal focuses on two intermediary servers that allow for an anonymization of personal data, while at the same time allows for secure connections to happen between the Dialogflow API and the IoT devices. The PPA mobile application is the glue of the system, connecting with every component and interacting with the user. The intermediary servers handle all the user queries, data requests and data masking that further increase and enhance privacy protection. All of this signifies that these servers will act as needed buffer that ensure that user data is not directly accessible by the platforms, enhancing overall user privacy and data security.

Specific technologies were used to create the mobile application based on the needed features. The chosen technology is called Xamarin.Forms, it helps make the app work on different types of phones (Android and iOS). This resulted in a practical PPA for mobile devices, which easily integrated with the mobile applications of the two studied platforms. Inside the application, it was integrated a chatbot, created using Google's Dialogflow along with its API. This chatbot acts like a virtual assistant and helps users with questions about privacy. It is fully integrated with the mobile application. The chatbot functions like a knowledgeable helper that answers users' questions. The application also communicates with two Python servers, both created using a framework called Flask. One of this servers keep track of nearby IoT devices, allowing users to be aware of them, and the other mediates the connection between the Dialogflow API and the chatbot inside the mobile application.

The practical implementation was then passed along to the two colleagues who are actively engaged in the development of these platforms. In addition to delivering the practical implementation, the two didactic sheets serve as introduction and as guides to the work developed. These sheet available in the Appendix section. By combining the practical implementation with the didactic sheets, the intention to equip the development team with a comprehensive toolkit for the successful implementation and integration of this proposal.

The image below provides a clear and concise architectural overview of the implementation intended for this practical part of the work. The main and most important component is the user and its smartphone that will have the Xamarin application. This application provides the user with a chatbot that was developed using Dialogflow, tailored to its specific use case and the connection is done via a proxy implemented using Python. The IoT device communicates with a Flask server, that serves as an IoT Device Listing Server, registering all the IoT devices that communicate with it. The mobile application communicates with this server retrieving the list of devices that are present nearby. These intermediary servers allow for a better segmentation on the data plane, allowing the handling of data requests in a organized way aswell as the ability to anonymize the information being exchanged. These notions are crucial to be able to understand this ecossytem.

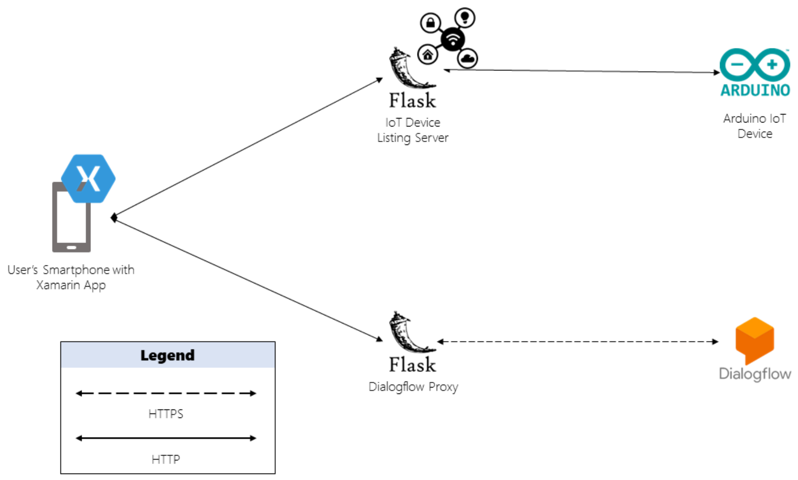


Figure 8: Architecture

4.2 ISABELA and BATINA's Privacy Concerns

Platforms such as ISABELA and BATINA, socially-aware Human-in-the-Loop Advisor Systems, leverage various data sources, including online social networks (OSN) data, environmental data and personal mobile device data, to provide users with valuable advice and insights. While this system has the potential to offer significant benefits, it also raises specific privacy concerns that require attention.

Online Social Network Data

ISABELA's reliance on OSN data introduces concerns regarding user privacy. OSNs often contain personal information, social connections and emotional states. The platform's ability to access and analyze this data necessitates safeguards to protect user privacy. Incorporating OSN data is essential for ISABELA in order for it to provide relevant and useful advice and insights. However, it is crucial to protect the privacy of this information, as it may contain sensitive user information.

Data Security

Given that platforms such as these utilize such broad data sources, including personal mobile device data. Unauthorized access or data breaches could result in severe privacy infringements. Personal mobile device data may include location records, communication histories aswell other sensitive and personal information. By failing to adequately secure and anonymize this data, this could expose users to privacy breaches and misuse.

Improper security measures can come in many different forms. Sensitive data might, for instance, be transmitted or stored in a way that can be easily intercepted by attackers if the system is not properly encrypted [21]. Similar to this, if the system does not have sufficient access controls, unauthorized users may be able to access sensitive data through hacking or other means.

Inference of Personal Information

These platforms analytical capabilities may unintentionally deduce highly personal information about users, even without explicit disclosure. This raises concerns regarding user consent and data transparency. While ISABELA's data analysis is key for generating tailored advice, it might inadvertently unveil information about users' emotions, behaviors or preferences that they did not explicitly share. This clearly demonstrates the importance of transparent data processing and that obtaining informed user consent is crucial.

User Consent and Control

Users of ISABELA and BATINA may be anxious and concerned about how their data is used and what their level of control over the information they share really is. Establishing clear, concise and user-friendly consent is essential for an increased trust in the system. Users should have complete visibility into how their data is collected, processed and employed by the platforms. This empowers users with control over their data, including the ability to opt in or out of specific data collection practices. It is a crucial aspect, when it comes to privacy, to respect user preferences.

In general, users have many concerns regarding data ownership and control [22]. Clients want to know who owns, controls and has access to the personal information that is being gathered about them. Participants may be less likely to use the platform or trust it with their sensitive information if they believe that their data is being used or shared without their permission.

The terms of service and privacy policy agreements that users must accept in order to use the platform are one issue relating to data ownership. Users may become confused and distrustful as a result of these agreements' complexity and difficulty in understanding. Users might accept these terms in order to use the platform, even if they are not fully aware of what doing so might mean.

Data Retention

Platforms which utilize HITL and Machine Learning to provide better services have a deep reliance on historical data for analysis and advice generation. This may raise questions regarding data retention. Long-term storage of sensitive information needs to be contemplated by well-defined data retention policies. To provide advice based on historical data, platforms such as BATINA and

ISABELA may need to retain user information over extended periods. However, this raises concerns about data storage practices aswell as posing security and privacy risks.

Third-Party Integration

As these platforms depend on external APIs [23] and services to collect and process user data, third-party integration raises security and privacy concerns. The FIWARE platform is used on BATINA and ISABELA to provide with a faster and easier deployment. However, with the benefits come the downsides and these third-party services are created by independent companies and frequently integrated using APIs, exposing the system to a number of security risks. It is possible that these third-party services contain flaws that hackers could use to access user data without authorization. Furthermore, if these services are not adequately protected, they might be attacked, resulting in data breaches and the disclosure of private user information.

4.3 ISABELA and BATINA's Privacy Improvements

After the previous analysis, it was demonstrated that systems such as ISABELA and BATINA encounter numerous privacy challenges due to their data-intensive nature. To ensure user trust and adherence to privacy regulations, several measures can be implemented to enhance privacy protection and mitigate risks. In this section, it is explored different approaches one can make to improve the privacy posture on these platforms.

Data Anonymization and Minimization

The implementation of robust data anonymization techniques to ensure that personally identifiable information (PII) is not exposed is crucial in these platforms. Additionally, by minimizing data collection to include only the essential information for it to keep providing the service and the key insights. By anonymizing data it safeguards user identities, making it challenging to trace information back to individuals.

User Control

Giving users access to their data and the ability to choose how it is used can help reduce users' worries about data ownership and control [24]. These platforms can provide users with a variety of controls over their data, including the ability to delete or export their data. Users may feel more in control of their personal information thanks to this feature, which also promotes trust. Platforms can also give users the option to control what information is gathered and shared, such as by allowing them to reject certain data collection methods.

Before collecting or sharing sensitive data, platforms can also get explicit consent. Users' consent must be explicit and informed before their data can be used for a particular purpose, according to this. Platforms should give users the option to opt-out if they are uncomfortable with the data collection practices and should be transparent about what data is being collected and how it will be used.

Transparency and Explicit Consent

The development of user confidence and trust in platforms like BATINA and ISABELA depends heavily on transparency. People are interested in what information is being gathered, how it is used and who has access to it. [25] Transparency can only be attained by outlining privacy policies and terms of service in clear, concise language. These policies should explain the types of data collected, the purposes for which the data is collected and processed and who has access to the data.

Platforms and services can give users fine-grained control over their data [26] in order to ensure transparency. Users may have the option to decide what information they want to share, who can access it and for what uses. Platforms can also offer users simple and understandable privacy control options and settings.

Additonally, platforms like the ones described must adhere to regulations [27] in order to guarantee that user data is handled responsibly and legally. Regulations like the General Data Protection Regulation (GDPR) force platforms to obtain explicit user consent before collecting any data, it also mandates that the platforms should be transparent about the data collected aswell as giving users access to their data.

It is a must to take action to in order to adhere to data protection and privacy policies, including setting up a procedure for dealing with user data requests like requests for data deletion or access, conducting routine audits to ensure compliance and more. Platforms can also give users tools to manage their data, like opt-out choices or detailed controls over what information is gathered and how it is used.

Platforms can prove to their user base that they are truly committed to safeguarding user privacy and data by adhering to relevant regulations. In addition to financial penalties and legal repercussions, breaking these rules can harm the platform's reputation and lose users' trust.

Data Retention and Deletion

Establishing clear data retention policies that clearly define the duration for which user data will be retained is a key aspect in building trust. As an example, by implementing automatic data deletion procedures when data is no longer needed for advice generation is a great way to improve on this aspect. Transparent data retention policies demonstrate a commitment to privacy.

Data deletion is a crucial component of privacy protection and a personal virtual assistant for privacy can assist users in securely deleting all of their personal data from these platforms. The virtual assistant can assist users who want to delete their accounts by walking them through the procedure step-by-step and making sure that all necessary measures are taken to erase their data.

Encryption

For the purpose of preventing unauthorized access to user data, encryption [28] is a crucial tool. Encryption techniques can be used by platforms like BATINA and ISABELA to protect data while it is in transit and at rest. To ensure that data transmitted between a user's device and the platform's servers is secure, HTTPS protocols can be used to encrypt web traffic. In order to ensure that only the intended recipient can access the contents of the messages, end-to-end encryption can also be used for messaging.

Data kept on servers or in databases can also be secured using encryption. The Advanced Encryption Standard (AES), which transforms plain text data into ciphertext that can only be decoded with a secret key, is one example of a cryptographic algorithm that can be used to encrypt data. By encrypting the data, even if a threat actor were to obtain it, they would be unable to read it without the encryption key, which is only available to authorized parties.

It is crucial to remember, though, that encryption is not a perfect solution for all privacy and security issues. Inadequate encryption implementation can occasionally lead to new vulnerabilities and encryption might not be capable of repelling specific types of attacks like phishing or social engineering. However, encryption is a vital tool that can help safeguard user data and therefore must to be used as a component of a thorough security plan.

New proposal

This dissertation seeks to explore and propose an innovative method for enhancing user privacy and bolstering trust in these kinds of systems. The primary goal is to provide users with a straightforward, practical and concise means to exercise their rights and influence system operations. The proposed solution centers on the use of intermediary servers that allow an anonymization of information aswell as the creation secure connections between the PPA embedded in the application and the broader system environment. This intermediary-server can handle user queries, data requests and data masking to further enhance privacy protection, this means that it will act as buffer in ensuring that user data is not directly accessible by the platforms, enhancing overall user privacy. By granting users the ability to gain precise insights into data collection activities in their immediate surroundings, such as IoT devices nearby that are actively collecting information, aswell as equipping them with information to reinforce their security and privacy.

An application or piece of software that can assist users in safeguarding their security and privacy while utilizing different online platforms and services is known as a virtual PPA [1]. These virtual assistants typically function by being attentive on a user's online activity, making privacy setting suggestions, enabling two-factor authentication, assisting with data deletion procedures and informing users of any unusual activity on their accounts.

Users looking to protect their online privacy and security may find virtual PPAs to be a useful tool. These assistants can support users in making knowledgeable decisions about their online behavior and taking proactive measures to protect their personal information by offering real-time monitoring, direction and recommendations.

4.4 Prototype

The MainPage and DeviceList pages were the two main parts on which the development process was concentrated. The application's main hub, the MainPage, offers users a simple-to-use and interesting interface. The chatbot quickly responds with pertinent information or answers when users input their queries or questions. A seamless user experience is guaranteed by the user interface's simplicity.

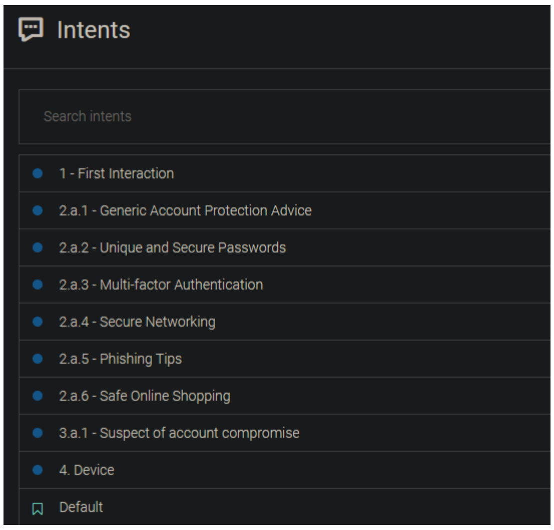
This application's interaction with the natural language processing platform Dialogflow is a standout feature. The program receives a code from Dialogflow when a user asks a question like

”Can you list me devices nearby?” This code serves as a trigger, causing the program to automatically switch to the DeviceList page. This behavior can be seen on the following code snippet, with the NavigateTo() calling the page that displays the list of IoT devices.

```
if (chatText == "**deviceList**")
{
    await NavigateTo();
}
return chatText;
```

The DeviceList page is intended to provide users with a list of the devices nearby. Users can quickly explore and assess the alternatives available to them by utilizing the application’s ability to gather and display the pertinent device information using the data obtained from Dialogflow.

The Dialogflow component was developed by creating multiple intents that can be further personalized and tailored to fit a certain user experience and context. Multiple intents were created to give a big range of tips that a user can adopt in order to increase their security posture and help safeguard their information. This can be seen on the following figure.



Intents

Search intents

- 1 - First Interaction
- 2.a.1 - Generic Account Protection Advice
- 2.a.2 - Unique and Secure Passwords
- 2.a.3 - Multi-factor Authentication
- 2.a.4 - Secure Networking
- 2.a.5 - Phishing Tips
- 2.a.6 - Safe Online Shopping
- 3.a.1 - Suspect of account compromise
- 4. Device
- Default

Figure 9: Display of the created Dialogflow intents that respond to user inquiries made on the Xamarin mobile application chat

Dialogflow integration is accomplished by deploying a Flask server that serves as an intermediary between your application and the Dialogflow API. Flask is a popular Python web framework that makes it simple to create online applications and APIs.

To begin, you must set up and configure a Dialogflow agent with different intents so it can respond to user’s inputs. Google’s Dialogflow is a natural language platform that allows developers to create dialog based interfaces for a variety of applications. In this specific case, it is used to respond to user’s questions about how one can improve their security and privacy posture.

The Flask server works as an intermediary for the Xamarin application and the Dialogflow API. It receives user messages from your app and forwards them to Dialogflow for processing. The server then receives the Dialogflow answer and transmits it back to the application. It is also important to note the significance of the intermediary Python servers, the possibilities that these servers offer range from providing a better security, by allowing a segmented and isolated way for connections to occur aswell as the anonymization and the possibility of data masking is also a crucial aspect that must be understood.

A streamlined and effective user experience is guaranteed by the combination of Xamarin.Forms, Dialogflow integration and the MainPage-DeviceList page structure. This application meets the needs of customers looking for rapid and precise information about nearby devices by offering a user-friendly chatbot interface and a dedicated device listing area.

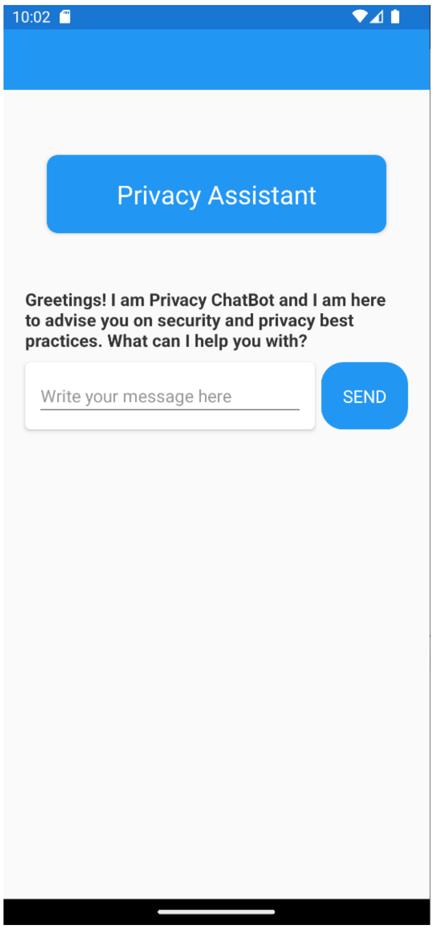
The MainPage component is used primarily to create a connection with the DialogFlow chatbot inside the context of a Xamarin.Forms application. This connection is established using RestSharp, allowing the API to work between the Xamarin.Forms application and the DialogFlow API. This can be seen in the following code.

```
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);
```

The MainPage component is an essential building piece in the Xamarin.Forms application. This creates a connection with the DialogFlow chatbot. This critical connection is established using the RestSharp library, which is smoothly integrated with Xamarin. The following code demonstrates the implementation details, emphasizing the connection between the Xamarin.Forms application and the DialogFlow Proxy, allowing for advanced and personalized conversational features.

```
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;
        if (chatText == "**deviceList**")
        {
            await NavigateTo();
        }
        return chatText;
    }

    return null;
}
```

Users are presented with an intuitive graphical interface within the application that displays the chatbot's response while also providing a dedicated area to input messages for the chatbot to respond to. The accompanying image depicts a visual representation of this dynamic user experience.



Greetings! I am Privacy ChatBot and I am here to advise you on security and privacy best practices. What can I help you with?

Figure 10: User Experience of the Main Page

The DevicesPage is essential for displaying an organized set of IoT devices collected from a Python Flask server. The DevicesPage rapidly gathers device information via API calls to the Python server and presents it to the user after the data is deserialized. This essential feature ensures that users can easily access and manage IoT devices within the application.

The image below depicts the graphical interface within the application and it serves as a visual illustration of the user-friendly display of IoT devices on the DevicesPage.

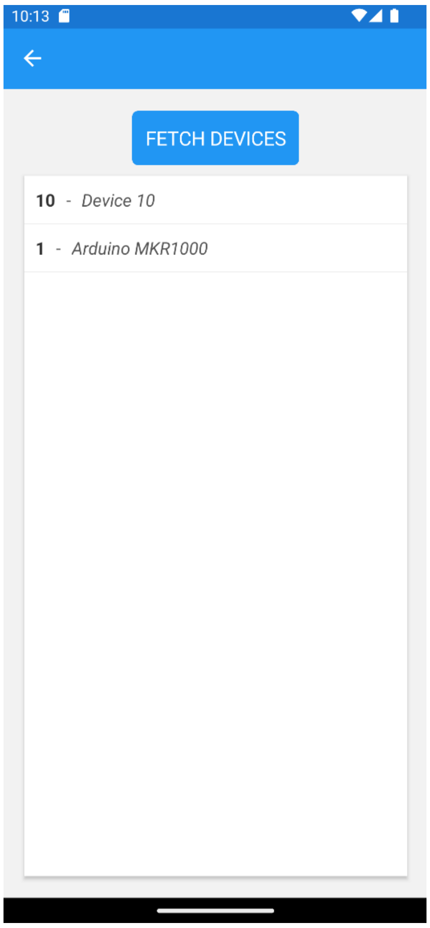


Figure 11: User Experience of the Device Listing Page

The Device Listing Service is a Flask server that listens for POST requests from nearby IoT devices. When these devices send data, the Flask server saves that data for safekeeping. Whenever an application requires access to the registered devices, this intelligent Flask server comes into action, quickly returning the entire list of devices to the Xamarin application, so that information can be then displayed to the user. The refresh of new information can be done via the "Fetch Devices" button on top of the canvas. This will force a new GET request to the Flask server. This routine can be observed with the following.

```
private async void FetchDevicesButton_Clicked(object sender, EventArgs e)
{
    await FetchAndDisplayDevicesAsync();
}
```

In this configuration, the Flask server serves as an important middleman, allowing IoT devices and the Xamarin application to communicate. It ensures that the data from the devices is captured and immediately available. The Xamarin application uses, as it does on the MainPage, a RestSharp connection to establish the communication with the Flask server. This can be seen with the following piece of code.

```
private async Task FetchAndDisplayDevicesAsync()
{
    var options = new RestClientOptions($"{iotProxy}")
    {
        RemoteCertificateValidationCallback = delegate {
            return true;
        }
    };
    // Create a new RestSharp client
    var client = new RestClient(options);
    var request = new RestRequest();
    var response = await client.ExecuteAsync(request);

    if (response.IsSuccessful)
    {
        // Deserialize the JSON
        List<Device> devices = JsonConvert.DeserializeObject<List<Device>>(response.
    Content);
        DevicesListView.ItemsSource = devices;
    }
}
```

Finally, adressing one of the most important parts of the development is the Dialoflow chatbot. The development of a Dialogflow chatbot is simple and powerful, to make this prototype work and function by giving some advice and tips to the user on how he can better safeguard his data a few intents (described above) were created and they were the following.

- First Interaction - This is the first interaction, when a user starts by saying "Hey" or "Hello".
- Generic Account Protection Advice - Brief generic introduction to some tips, while also allowing the user to further explore more options.
- Unique and Secure Passwords - Informs the user on latest password security standards.
- Multi-factor Authentication - Advocates for the use of MFA and listing the benefits.
- Secure Networking - General but essential tips on how to protect oneself on public networks.
- Phishing Tips - Tells the user on how to detect and protect himself from falling victim to a phishing attack.
- Safe Online Shopping - Provides advice on how the user can protect himself when shopping online.
- Suspect of Account Compromise - Gives information about how a user must handle a situation in which he suspects his account might already have been compromised.
- Device - Used for the transition between Xamarin application's pages.

4.5 Testing

This part is going to be dedicated to the execution of different tests that can be used to guarantee a satisfactory functioning of the different applications and components developed throughout.

Unit and Integration Testing

Starting with unit testing, the different components were tested in isolation and integrated whenever it is deemed necessary, assuring that these components indeed work as they are intended and the connections that are supposed to be established function properly and without any problems.

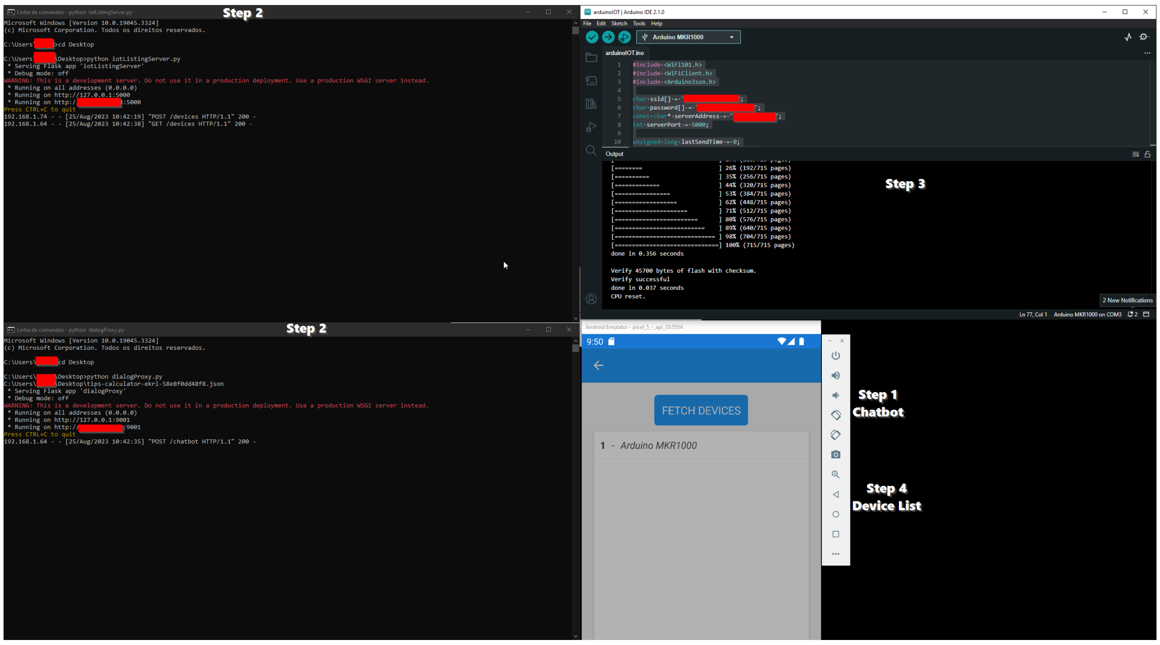
Component	
PPA Xamarin mobile application	✓
Device Listing Python server	✓
Dialogflow Proxy Python server	✓
Arduino Module	✓

Table 1: Results from Unit testing

For the integration testing, the components of the overall system were executed. Both Python servers (Device Listing and Dialogflow Proxy) were activated and running, aswell as executing the code developed for the Arduino module and for the Xamarin application. By using the chatbot to ask for nearby devices, this allows the application to communicate with Dialogflow, receiving the response, it then uses that response to switch the page and fetches the list of devices on the Device Listing Python server. The steps for this integration test are the following.

1. Run Xamarin Application - Execute Xamarin mobile application
2. Start the Device Listing Service and Dialogproxy python scripts
3. Upload the code to the Arduino module
4. On the Xamarin application, use the chatbot with a query like ”Hello! Can you list nearby devices?”
5. The Device List page will appear, clicking on the Fetch button, the Arduino module will be displayed

The overall result of the integration test can be visualized right below.


Linha de comandos - python iotListingServer.py
Step 2
arduinoIOT | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino MKR1000
arduinoIOT.ino
Output
Step 3
Linha de comandos - python dialogProxy.py
Step 2
Android Emulator - pixel_5_-_api_33:5554
9:50
FETCH DEVICES
1 - Arduino MKR1000
Step 1
Chatbot
Step 4
Device List
2 New Notifications
Ln 77, Col 1 Arduino MKR1000 on COM3

```
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\[redacted]>cd Desktop

C:\Users\[redacted]\Desktop>python iotListingServer.py
 * Serving Flask app 'iotListingServer'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://[redacted]:5000
Press CTRL+C to quit
192.168.1.74 - - [25/Aug/2023 10:42:19] "POST /devices HTTP/1.1" 200 -
192.168.1.64 - - [25/Aug/2023 10:42:38] "GET /devices HTTP/1.1" 200 -
```

```
#include·<WiFi101.h>
#include·<WiFiClient.h>
#include·<ArduinoJson.h>

char·ssid[]·=·"[redacted]";
char·password[]·=·"[redacted]";
const·char*·serverAddress·=·"[redacted]";
int·serverPort·=·5000;

unsigned·long·lastSendTime·=·0;
```

```
[========                      ] 26% (192/715 pages)
[==========                    ] 35% (256/715 pages)
[=============                 ] 44% (320/715 pages)
[================              ] 53% (384/715 pages)
[==================            ] 62% (448/715 pages)
[=====================         ] 71% (512/715 pages)
[========================      ] 80% (576/715 pages)
[==========================    ] 89% (640/715 pages)
[============================= ] 98% (704/715 pages)
[==============================] 100% (715/715 pages)
done in 0.356 seconds

Verify 45700 bytes of flash with checksum.
Verify successful
done in 0.037 seconds
CPU reset.
```

```
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\[redacted]cd Desktop

C:\Users\[redacted]\Desktop>python dialogProxy.py
C:\Users\[redacted]\Desktop\tips-calculator-ekrl-58e8f0dd48f8.json
 * Serving Flask app 'dialogProxy'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:9001
 * Running on http://[redacted]:9001
Press CTRL+C to quit
192.168.1.64 - - [25/Aug/2023 10:42:35] "POST /chatbot HTTP/1.1" 200 -
```

Figure 12: Overview of the unit and integration test done on the components of the Personal Privacy Assistant

Functional Testing

In this section, it is done a plethora of functional testing processes for the Xamarin mobile application and Python Flask servers developed. By subjecting the application to different well-defined test cases, it is possible to examine its response to various inputs, interactions and scenarios, aiming to validate its compliance with the established functional requirements. The functional testing phase offers an opportunity to not only detect defects and discrepancies but also to confirm the application's ability to deliver a seamless user experience.

Functional Test	Expected Outcome	Real Outcome	
Xamarin application connection to Dialogflow	The application should successfuly receive the response from Dialogflow and display its contents	The application successfuly receives the response and displays it accordingly	✓
Xamarin application connection to Device Listing Server	The Xamarin application should successfully retrieve any data contained inside the Device Listing Python Server and display a list with all the devices	The application successfuly retrievess the device list and displays it accordingly	✓
Xamarin application switches correctly between the Chatbot page and the Device Listing page	The Xamarin application should successfully switch between the chatbot and device listing page by receiving a response from Dialogflow and return to the chatbot page by using a button	The application successfuly switches between the two aforementioned pages	✓
Dialogflow Proxy Python server connection to Google and Mobile Application	Dialogflow Proxy Python server should successfuly establish a connection to the Xamarin application aswell as correctly communicating the Dialogflow Google API	The server correctly communicates with both ends	✓
Device Listing Python server connection from Arduino to Mobile Application	Device Listing Service Python server should successfuly respond to GET requests from the Xamarin application by sending JSON formatted information and receive POST requests from the Arduino module containing the JSON information	The server correctly communicates with both ends and the data is transmitted correctly	✓
Encrypted communication between Xamarin application and both Python servers	Communication should be encrypted between the Xamarin application and both Python servers to maintain data privacy and security	The server does not encrypt data in its current state	✗
Arduino communication with Device Listing Python server	Arduino should successfuly connect to the Wi-Fi and send timed POST requests to the Device Listing Python server containing the IoT module information	Data is transmitted successfuly over Wi-Fi from the Arduino to the Python server	✓

Table 2: Detailed description of the functional tests pertaining to the practical development ecosystem of the dissertation

Non-Functional Testing

This section digs deeper into the analysis of non-functional tests, which are to be conducted on a desktop computer featuring the following specifications:

- CPU: i5 12400F (6 core @ 2.5GHz to 4.4GHz)
- RAM: 16GB of DDR4 - 2133 MHz

The non-functional testing is centered around three distinct categories as follows:

- Load Test
- Stress Test
- Capacity Test

The assessment was conducted on the Device Listing Python server using a set of specific tests. These assessments were done using the Locust tool, designed to facilitate comprehensive performance evaluations of the server.

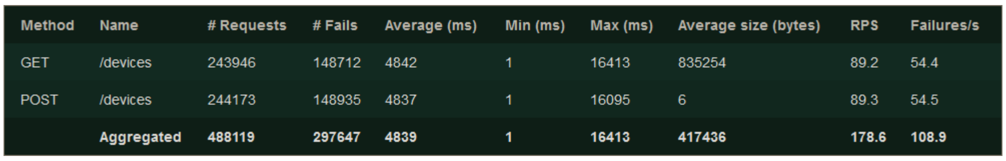
To facilitate this process, Locust was installed through Python's 'pip install' command. Furthermore, a simple script was developed to contextualize the ensuing GET and POST requests that are going to be tested. The script is displayed below.

```
from locust import HttpUser, task, between

class MyUser(HttpUser):
    # Waiting time between user actions
    wait_time = between(1, 3)

    @task
    def test_add_device(self):
        device_data = {
            'DeviceId': '1',
            'DeviceName': 'StressTestDevice'
        }
        response = self.client.post('/devices', json=device_data)
        assert response.status_code == 200
        assert response.json() == {'success': True}

    @task
    def test_get_devices(self):
        response = self.client.get('/devices')
        assert response.status_code == 200
        assert response.json() == []
```

To test it, a set of parameters needed to be set. The parameters such as the maximum number of users and the spawn rate can be seen on the list below.

- Number of users (Peak concurrency): 2000
- Spawn rate (users started/second): 1

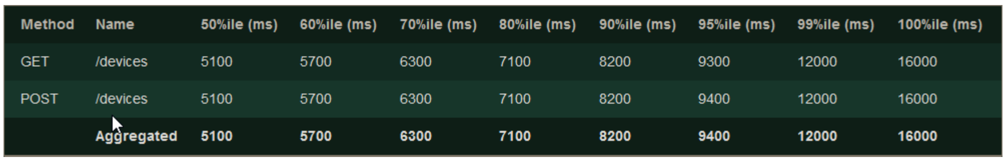
In this next image displays the statistics pertaining to the requests. The statistics are divided by the type of request (either GET or POST) and then giving insights at how many requests were sent and how many failed. The average response time in ms was about 4800 ms (4.8 seconds) in both of the request types, with the minmum response time being 1 ms and the maximum response time around the 16000 ms (16 seconds) mark. It is important to note that average response time is high due to the number of tests done, with the number of people being simulated, the response time increases sharply thus increasing the average response time. The number of failures is also explained by the increasing number of requests being simulated.



Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/devices	243946	148712	4842	1	16413	835254	89.2	54.4
POST	/devices	244173	148935	4837	1	16095	6	89.3	54.5
	Aggregated	**488119**	**297647**	**4839**	**1**	**16413**	**417436**	**178.6**	**108.9**

Figure 13: Requests Statistics

Here it is digged deeper and it is possible to observe a more detailed view regarding the response time statistics. In this table it is possible to analyze the response time in miliseconds for different percentiles and divided by request method.



Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/devices	5100	5700	6300	7100	8200	9300	12000	16000
POST	/devices	5100	5700	6300	7100	8200	9400	12000	16000
	Aggregated	**5100**	**5700**	**6300**	**7100**	**8200**	**9400**	**12000**	**16000**

Figure 14: Response Time Statistics

This graphic gives a clear vision towards the increment in the number of users over time, it starts with 1 user making requests and proceeds with unitary increments to the number of users until it finally reaches the number configured on the beginning of the test (2000 users). The y-axis on the graphic represents the number of people while the x-axis the time.

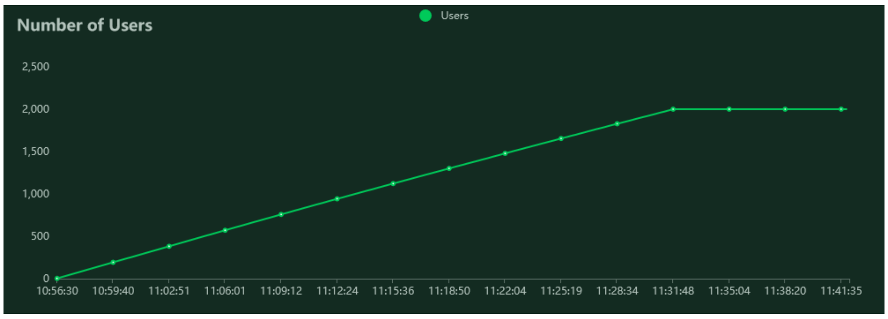


Figure 15: Number of users

On this graphic, it is possible to observe the impact on response times to the requests to the number of users making them. This also illustrates perfectly, the explanation above about the average response time being so elevated, as it is evidenced below, this is due to the increment in the number of requests being made.

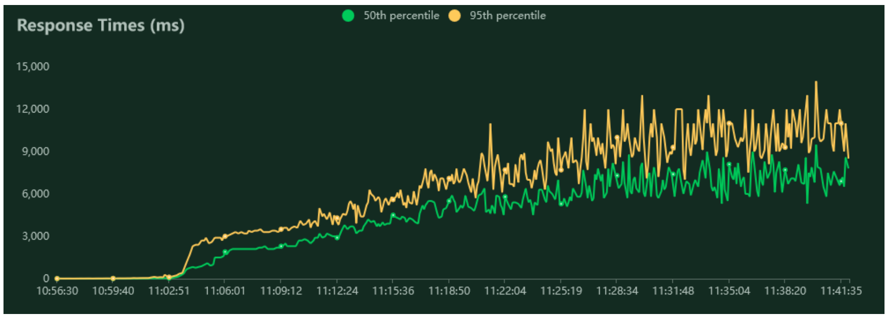


Figure 16: Response times

As it is possible to observe on the graphic above and even more clearly on the graphic below, the reponse time begins to take a performance hit around the mark of 420 users, this marks the start of an increased response time.

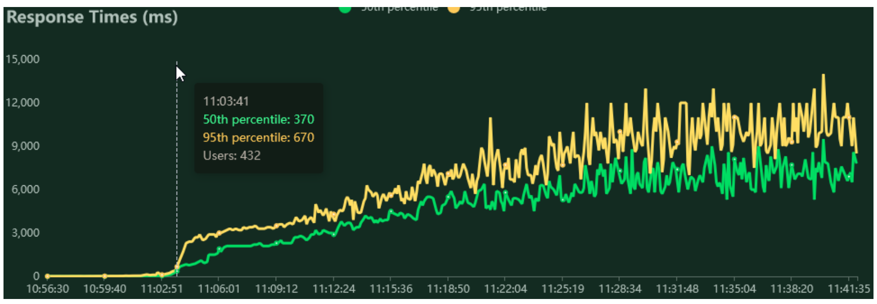


Figure 17: Response times to request begin increasing around the mark of 420 users

On the graphic displayed below, an inference can be made between the number of request per second and the number of failures. This also supports the explanation above about the high number of failures, the addition of more simulated users causes a rise in the number of request failures.

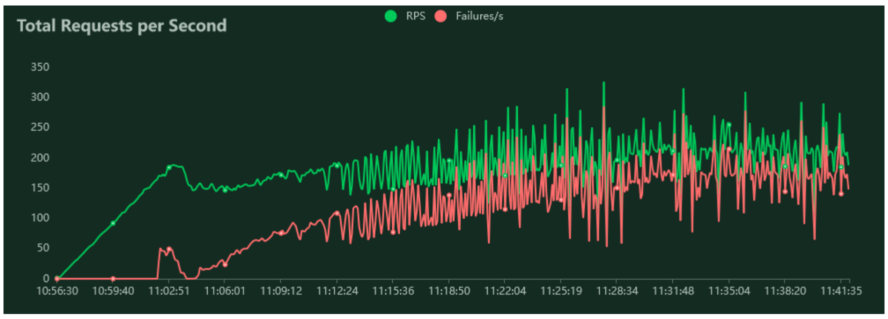


Figure 18: Requests per second

The requests start to show failures at approximately 347 concurrent users, with a cumulative request rate of 160 requests per second.

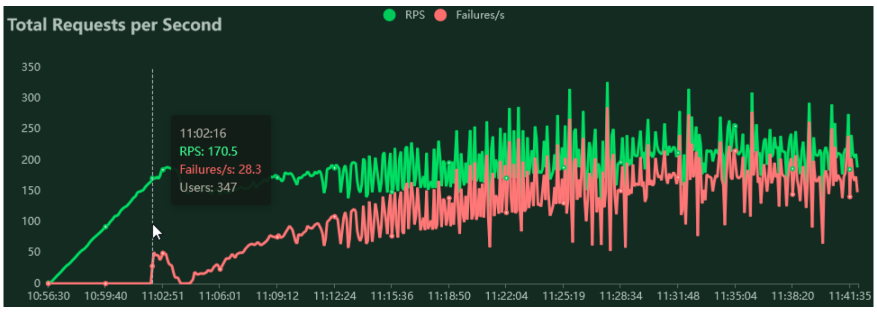


Figure 19: Request begin to fail at roughly 347 users

It is not possible assess the non-functional aspect of the Dialogflow Proxy Python server due to limitations within the trial version of the Google API. As a result, the connection flow experiences packet drops, as these are being rejected by Google.

4.6 Further Development and Didactic Sheets

The proposed system and its accompanying prototype represent a step towards a practical implementation for future integration into the ISABELA and BATINA platforms. The intermediary servers, along with the prototype mobile application serve as foundational work that can be integrated and further tailored to fit the necessities of both platforms, while at the same time providing security and privacy capabilities, such as the anonymization of personal data. The didactic sheets included in Appendix B play a crucial role in supporting not only the current project but also in fostering a deeper comprehension of the work undertaken. These educational resources facilitate the work for the colleagues responsible for the continued evolution of ISABELA and BATINA.

5 Conclusion and Future Work

5.1 Conclusion

As it was written in the beginning of this dissertation, the ever-evolving landscape of technology has brought to light, with an ever-increasing importance and force, the key and crucial aspect that is data privacy and security. With the emergence of cutting edge spyware like Pegasus, they serve as messengers and reminders of the urgent need to improve and safeguard personal data from unauthorized access and violations of privacy rights. Privacy is one of the foundational pillars of a modern society and the increase of technology in our daily lives, such as AI and IoT, has been a warning call for many people, most of them worrying about the enormous amounts of data that are collected each and everyday about all of us.

New paradigms are needed to give more power to the user and restore some control over the data. Technologies and concepts such as PPAs appear as potential solutions in alleviating growing concerns as these allow for a shift on the control over data. This was one of the main and focus topics, the exploration and analysis of a PPA, developed using Xamarin to create a chatbot-based assistant that offers users personalized recommendations and guidance on how they can protect themselves and their personal information. Drawing inspiration from Carnegie Mellon University, this work tried to replicate the usefulness of these assistants applied to two specific platforms, ISABELA and BATINA. This user centric approach attempted to show the usability of these types of tools in promoting the protection of personal information.

Throughout the research conducted, prevalent security and privacy concerns were analyzed and examined, applying that knowledge to the specififc case of the two platforms and exploring their own privacy and security concerns aswell as different measures that could be applied to increase the trust and confidence of a user in regards to the use of their data. One of the main contributions was the demonstration of the potential integration of a chatbot-based PPA with applications that can answer user's inquiries and offer invaluable tips about how a user can enhance their security and privacy posture.

In conclusion, the work developed throughout this academic investigation attempts to contribute to the ongoing efforts of creating privacy-aware and empowered digital citizens, by giving more control and knowledge to the user, while at the same time simplifying the process, which can offer invaluable insights and teachings into the effectiveness of PPAs aswell as making room for more research and different paradigms. As the evolution of technology advances, the utilization of frameworks and tools like PPAs become increasingly crucial in ensuring that users are in control and empowered in regards to how and which data is collected, processed and stored. This foundational research and practical development shows that there is a potential to revolutionize the way people understand and perceive their privacy rights and the active practice of data protection.

5.2 Future Work

Throughout this dissertation it was attempted to display the usability and accessibility of incorporating a PPA into multiple applications and it has shown a considerable development. The present effort has established a solid basis, but there are many opportunities for additional innovation that build on its successes. In this section dedicated to the Future Work it is explored the major areas where the framework presented in this work might be extended and improved with new features.

Future research should focus on improving the contextual chatbot experience. The chatbot may fully integrate with the intricacies of the host application by adding a greater contextual awareness of it. This entails integrating the chatbot into the ecosystem of the application and allowing the chatbot to get a deep understanding about the methods for processing, using and storing data. Through real-time communication with the chatbot, data subjects are empowered to actively exercise their data rights thanks to this immersive method. In addition to enhancing user experience, the resultant integration gives data subjects direct control over their data privacy, increasing their sense of control and empowerement over their data. The chatbot and application union is regarded as one of the main and key aspects to be developed as it can improve and give an impressive immersion and confidence on the end-user.

The Device Listing Service and Dialogflow Proxy may receive more improvements, with an emphasis on enhancing security controls using powerful encryption methods and serving as an intermediary agent that filters communication and connections. This also allows for a development of anonymization capabilities that further increases privacy. A more transparent and informed user experience may also be achieved by giving users comprehensive contextual information about each module. The capability to change various configurations on Internet of Things (IoT) modules is also included in this empowerment, which is in line with the ideas outlined in the Carnegie Mellon University article. A more complete and user-friendly system may be achieved by strengthening the Device Listing Service's security and user-centricity.

Despite the fact that it has strived in doing a professional and thorough job of introducing the idea of a PPA, there is still much room for improvement in terms of capabilities, user control and integration. The work developed does improve the current paradigm and attempts to display it on practical and easy terms while also laying the groundwork for future innovation and development in the fields of data privacy and application interaction.

Bibliography

References

[1] Anupam Das et al. “Personalized Privacy Assistants for the Internet of Things: Providing Users with Notice and Choice”. en. In: *IEEE PERVASIVE COMPUTING* 17.3 (2018), pp. 35–46. DOI: 10.1109/MPRV.2018.03367733.

[2] *What is IoT (Internet of Things) and How Does it Work? - Definition from TechTarget.com.* en. URL: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT (visited on 07/02/2023).

[3] *Internet of Things and Privacy - Issues and Challenges.* en-AU. URL: https://ovic.vic.gov.au/privacy/resources-for-organisations/internet-of-things-and-privacy-issues-and-challenges/ (visited on 07/02/2023).

[4] Pardis Emami-Naeini et al. “Privacy Expectations and Preferences in an IoT World”. en. In: *Thirteenth Symposium on Usable Privacy and Security* (July 2017).

[5] OAIC. *What is privacy?* en. Last Modified: 2023-03-10T16:37:29+11:00. Mar. 2023. URL: https://www.oaic.gov.au/privacy/your-privacy-rights/your-personal-information/what-is-privacy (visited on 07/02/2023).

[6] CSRC Content Editor. *privacy - Glossary — CSRC.* EN-US. URL: https://csrc.nist.gov/glossary/term/privacy (visited on 07/17/2023).

[7] Amy Beloume. *The Problems of Internet Privacy and Big Tech Companies.* en, sv. Section: News. URL: https://thesciencesurvey.com/news/2023/02/28/the-problems-of-internet-privacy-and-big-tech-companies/ (visited on 09/01/2023).

[8] Muhammad Bin Haris, Mohammad Bin Yahya, and Muhammad Bin Ibrahim. *Security and Privacy Issues in Internet of Things (IoT).* English.

[9] *IBM — What is Machine Learning?* en-us. URL: https://www.ibm.com/topics/machine-learning (visited on 07/03/2023).

[10] *GDPR considerations when training machine learning models — Qwak’s Blog.* URL: https://www.qwak.com/post/gdpr-considerations-when-training-machine-learning-models (visited on 07/03/2023).

[11] *TELUS International — What is Human-in-the-loop?* en. URL: https://www.telusinternational.com/glossary/human-in-the-loop (visited on 07/09/2023).

[12] Robert Koch. *Human-in-the-Loop: What is it and why it matters for ML.* en. May 2022. URL: https://www.clickworker.com/customer-blog/human-in-the-loop-ml/ (visited on 07/09/2023).

[13] *Humans in the Loop: The Design of Interactive AI Systems.* en. URL: https://hai.stanford.edu/news/humans-loop-design-interactive-ai-systems (visited on 07/09/2023).

[14] profexorgeek. *What is Xamarin.Forms? - Xamarin.* en-us. July 2021. URL: https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms (visited on 07/19/2023).

[15] *Dialogflow Documentation.* en. URL: https://cloud.google.com/dialogflow/docs (visited on 07/19/2023).

[16] *The Python Tutorial.* URL: https://docs.python.org/3/tutorial/index.html (visited on 07/20/2023).

[17] *What is Arduino?* en. 2018. URL: https://www.arduino.cc/en/Guide/Introduction (visited on 07/20/2023).

[18] GDPR. *What is GDPR, the EU's new data protection law?* en-US. Section: GDPR Overview. Nov. 2018. URL: https://gdpr.eu/what-is-gdpr/ (visited on 07/22/2023).

[19] *IoT Privacy Infrastructure.* URL: https://www.iotprivacy.io/discovering-iot (visited on 09/01/2023).

[20] J. Fernandes et al. "An Integrated Approach to Human-in-the-Loop Systems and Online Social Sensing". en. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* Paris, France: IEEE, Apr. 2019, pp. 478–483. ISBN: 978-1-72811-878-9. DOI: 10.1109/INFCOMW.2019.8845278. URL: https://ieeexplore.ieee.org/document/8845278/ (visited on 06/17/2023).

[21] *Cryptographic Failures Vulnerability - Examples & Prevention.* en-GB. Running Time: 595 Section: Vulnerability Prevention. June 2022. URL: https://crashtest-security.com/owasp-cryptographic-failures/ (visited on 07/30/2023).

[22] *Data Ownership: Considerations for Risk Management.* URL: https://www.isaca.org/resources/isaca-journal/issues/2020/volume-2/data-ownership (visited on 07/30/2023).

[23] *Third Party APIs — 6 Risks of Consumption — Akana.* en. URL: https://www.akana.com/blog/third-party-apis (visited on 07/30/2023).

[24] https://www.facebook.com/tresorit. *Data Access Control: How to Keep Data Safe and Users Happy at the Same Time?* en. Nov. 2022. URL: https://tresorit.com/blog/data-access-control/ (visited on 07/30/2023).

[25] Shannon Flynn. *How To Ensure Data Transparency - DZone.* en. 2022. URL: https://dzone.com/articles/how-to-ensure-data-transparency-and-why-its-import (visited on 07/30/2023).

[26] Timothy Morey, Theodore "Theo" Forbath, and Allison Schoop. "Customer Data: Designing for Transparency and Trust". In: *Harvard Business Review* (May 2015). Section: Analytics and data science. ISSN: 0017-8012. URL: https://hbr.org/2015/05/customer-data-designing-for-transparency-and-trust (visited on 07/30/2023).

[27] *5 Reasons Why Compliance Is Important for a Business — Indeed.com Canada.* en. URL: https://ca.indeed.com/career-advice/career-development/why-compliance-is-important (visited on 07/30/2023).

[28] IBM. *What is encryption? Data encryption defined — IBM.* en-us. URL: https://www.ibm.com/topics/encryption (visited on 07/30/2023).

Appendix

Appendix A - Prototype Code

This appendix provides the code developed for each one of the applications. The main codes are the Xamarin mobile application code, the Dialogflow Proxy, the IoT Device Listing server and the Arduino code for the Arduino MKR 1000 board.

Xamarin Mobile Application

MainPage.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="PPrivacyAssistant.MainPage">

    <StackLayout Padding="20" Spacing="10">
        <Frame BackgroundColor="#2196F3" Padding="20" CornerRadius="10" Margin="20,
    40">
            <Label Text="Privacy Assistant" HorizontalTextAlignment="Center"
    TextColor="White" FontSize="24"/>
        </Frame>

        <ScrollView VerticalScrollBarVisibility="Always">
            <StackLayout x:Name="ChatContainer" Spacing="10">
                <Label x:Name="AnswerLabel" FontSize="16" TextColor="#333333"
    FontAttributes="Bold"/>
            </StackLayout>
        </ScrollView>

        <Grid VerticalOptions="End">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <Frame Grid.Row="0" Grid.Column="0" BackgroundColor="#FFFFFF"
    CornerRadius="5" Padding="0">
                <Entry x:Name="InputQuery" Placeholder="Write your message here"
    PlaceholderColor="#999999" Margin="10" FontSize="16" BackgroundColor="Transparent
    " TextColor="#333333"/>
            </Frame>
            <Button Grid.Row="0" Grid.Column="1" Text="Send" BackgroundColor="#2196F3
    " TextColor="White" CornerRadius="20" FontSize="16" HeightRequest="40"
    WidthRequest="80" Clicked="SendButton_Clicked"/>
        </Grid>
    </StackLayout>
</ContentPage>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
using Xamarin.Forms;
using RestSharp;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

namespace PPrivacyAssistant
{    public partial class MainPage : ContentPage
    {
        private static readonly string ChatBotProxyAddress = $"http
    ://192.168.1.64:9001"; // 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;
                if (chatText == "**deviceList**")
                {
                    await NavigateTo();
                }
                return chatText;
            }

            return null;
        }
        public static async Task NavigateTo()
        {
            await ((NavigationPage)App.Current.MainPage).Navigation.PushAsync(new
    DevicesPage());
        }
    }
}
```

DevicesPage.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="PPrivacyAssistant.DevicesPage"
             BackgroundColor="#F2F2F2">
    <StackLayout Spacing="10" Padding="20">
        <Button Text="Fetch Devices"
                Clicked="FetchDevicesButton_Clicked"
                HorizontalOptions="Center"
                BackgroundColor="#2196F3"
                TextColor="White"
                FontSize="18"
                CornerRadius="5"
                HeightRequest="50"/>

        <ListView x:Name="DevicesListView"
                  BackgroundColor="White"
                  SeparatorColor="#CCCCCC">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout Orientation="Horizontal" Padding="10">
                            <Label Text="{Binding DeviceId}"
                                   FontAttributes="Bold"
                                   VerticalOptions="CenterAndExpand"
                                   FontSize="16"
                                   TextColor="#333333"/>
                            <Label Text=" - "
                                   FontSize="16"
                                   VerticalOptions="CenterAndExpand"
                                   TextColor="#666666"/>
                            <Label Text="{Binding DeviceName}"
                                   FontAttributes="Italic"
                                   VerticalOptions="CenterAndExpand"
                                   FontSize="16"
                                   TextColor="#555555"/>
                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

DevicesPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Net.Http;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Xamarin.Forms;
using RestSharp;

namespace PPrivacyAssistant
{
    public partial class DevicesPage : ContentPage
    {
        private static readonly string iotProxy = $"http://192.168.1.64:5000/devices
    ";
        public DevicesPage()
        {
            InitializeComponent();
        }
        public List<Device> Devices { get; set; }

        private async Task FetchAndDisplayDevicesAsync()
        {
            var options = new RestClientOptions($"{iotProxy}")
            {
                RemoteCertificateValidationCallback = delegate {
                    return true;
                }
            };
            // Create a new RestSharp client
            var client = new RestClient(options);
            var request = new RestRequest();
            var response = await client.ExecuteAsync(request);

            if (response.IsSuccessful)
            {
                // Deserialize the JSON
                List<Device> devices = JsonConvert.DeserializeObject<List<Device>>(
    response.Content);
                DevicesListView.ItemsSource = devices;
            }
        }

        private async void FetchDevicesButton_Clicked(object sender, EventArgs e)
        {
            await FetchAndDisplayDevicesAsync();
        }
    }

    public class Device
    {
        public int DeviceId { get; set; }
        public string DeviceName { get; set; }
    }
}
```

Dialogflow Proxy

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

project_id = "tips-calculator-ekrl"
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = r"C:\Users\jpnan\Desktop\tips-
    calculator-ekrl-58e8f0dd48f8.json"
print(os.environ["GOOGLE_APPLICATION_CREDENTIALS"])

app = Flask(__name__)

def parse_object(query_result):
    response = {}
    response['query_text'] = query_result.query_text
    response['intent'] = query_result.intent.display_name
    response['confidence'] = query_result.intent_detection_confidence
    response['fulfillment_text'] = query_result.fulfillment_text
    return response

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.query_result)
    responses.append(responseObject)
    return responses

@app.route("/chatbot", methods=['POST'])
def chat_bot():
    body = request.json
    responseBody = detect_intent_texts(body["session_id"], body["query"], body['
    language_code'])
    responseHelper = {"responses": responseBody}
    return app.response_class(
        response=json.dumps(responseHelper),
        status=200,
        mimetype='application/json'
    )

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

IoT Device Listing Server

```
from flask import Flask, jsonify, request

app = Flask(__name__)

devices = []
```

```python
@app.route('/devices', methods=['POST'])
def add_device():
    # Get device data from request
    device_data = request.get_json()

    # Get Device ID and Device Name
    device_id = device_data['DeviceId']
    device_name = device_data['DeviceName']

    # Add device to list
    devices.append({'DeviceId': device_id, 'DeviceName': device_name})

    # Return success message
    return jsonify({'success': True})

@app.route('/devices', methods=['GET'])
def get_devices():
    # Send devices
    return jsonify(devices)

if __name__ == '__main__':
    app.run(host="0.0.0.0",port=5000)
```

Arduino Code

```cpp
#include <WiFi101.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>

char ssid[] = "Vodafone-FAA207";
char password[] = "FmSUvR86EMEa3bgg";
const char* serverAddress = "192.168.1.64";
int serverPort = 5000;

unsigned long lastSendTime = 0;
unsigned long sendInterval = 60000;

WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Connect to Wi-Fi
  while (WiFi.begin(ssid, password) != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    delay(1000);
  }

  Serial.println("Connected to Wi-Fi");

  // Send device data to Flask server
  sendData2Server();
}

void loop() {
  if (millis() - lastSendTime >= sendInterval) {
    // Send device data to Flask server
    sendData2Server();

    // Update the last send time
    lastSendTime = millis();
  }
}

void sendData2Server() {
  // Create JSON payload
  DynamicJsonDocument jsonDoc(128);
  jsonDoc["DeviceId"] = 1;
  jsonDoc["DeviceName"] = "Arduino MKR1000";

  String payload;
  serializeJson(jsonDoc, payload);

  // Connect to the server
  if (client.connect(serverAddress, serverPort)) {
    Serial.println("Connected to server");

    // Make a POST request
    client.println("POST /devices HTTP/1.1");
    client.println("Host: " + String(serverAddress));
    client.println("Content-Type: application/json");
    client.print("Content-Length: ");
    client.println(payload.length());
    client.println();
    client.println(payload);

    // Wait for the server's response
    while (client.connected()) {
      if (client.available()) {
        String response = client.readStringUntil('\n');
        Serial.println(response);
      }
    }
  } else {
    Serial.println("Connection to server failed");
  }

  client.stop();
}
```

Appendix B - Didactic Sheets

First Didactic Sheet

In this section it is displayed a comprehensive learning resource tailored for Professor Jorge Sá Silva. This sheet explores Xamarin application development and its integration with Dialogflow, being created to provide students with a good understanding of how these two powerful technologies work and how they integrate with one another.

Universidade de Coimbra
Faculdade de Ciências e Tecnologia

DEPARTAMENTO DE ENG. ELECTROTÉCNICA E COMPUTADOPRES

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)

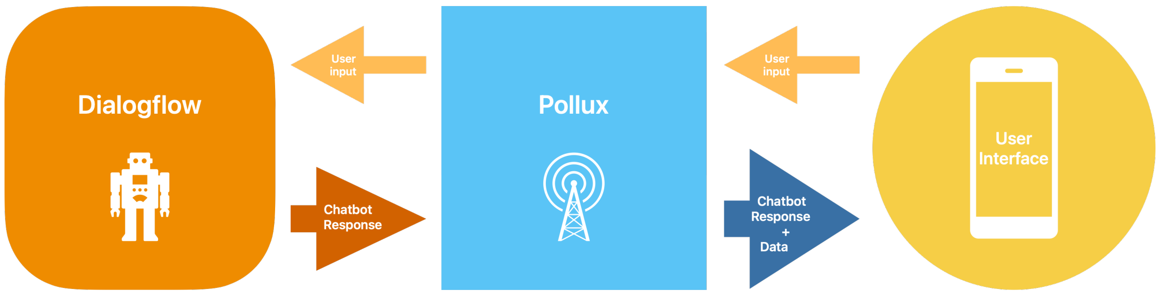


Fig. 1 - Simple architecture diagram

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

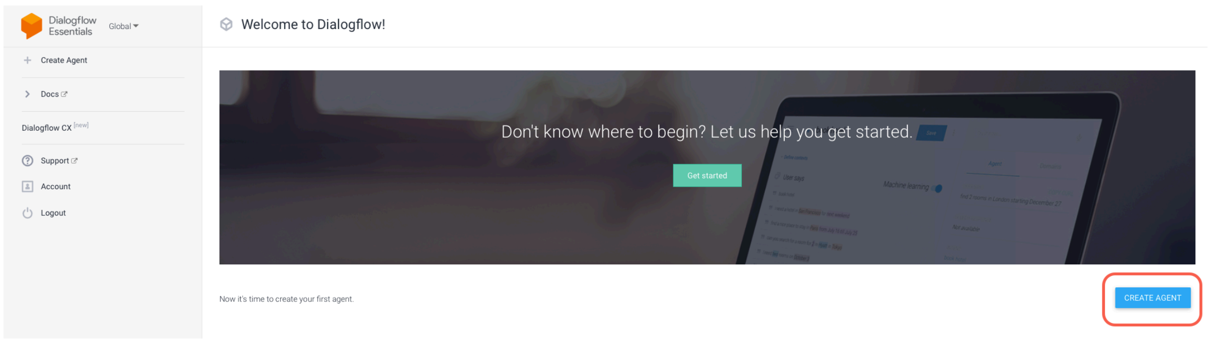
To be able to develop and run your server (Pollux) you need to install python and flask. To make it simple you should use JetBrains IDE for python - PyCharm. You can download it here: https://www.jetbrains.com/pycharm/

Note: if you don't have python installed on your machine, download it and install it from https://www.python.org/downloads/

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.

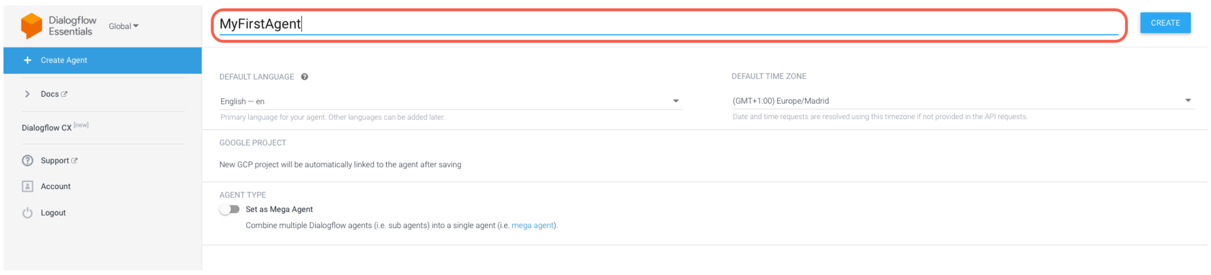


Welcome to Dialogflow!

Don't know where to begin? Let us help you get started.

Now it's time to create your first agent.

Click the button "Create Agent"



MyFirstAgent

DEFAULT LANGUAGE

English — en

Primary language for your agent. Other languages can be added later.

DEFAULT TIME ZONE

(GMT+1:00) Europe/Madrid

Date and time requests are resolved using this timezone if not provided in the API requests.

GOOGLE PROJECT

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

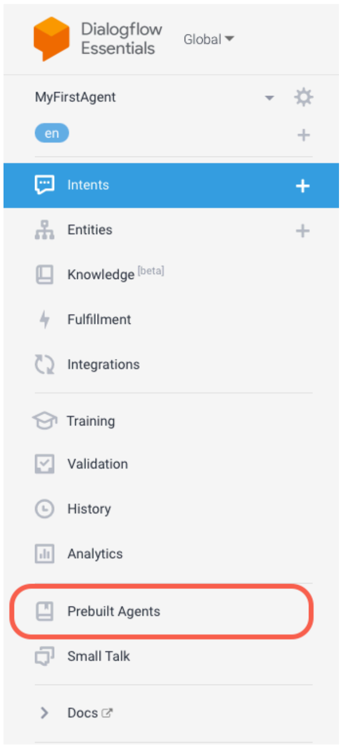
AGENT TYPE

Set as Mega Agent

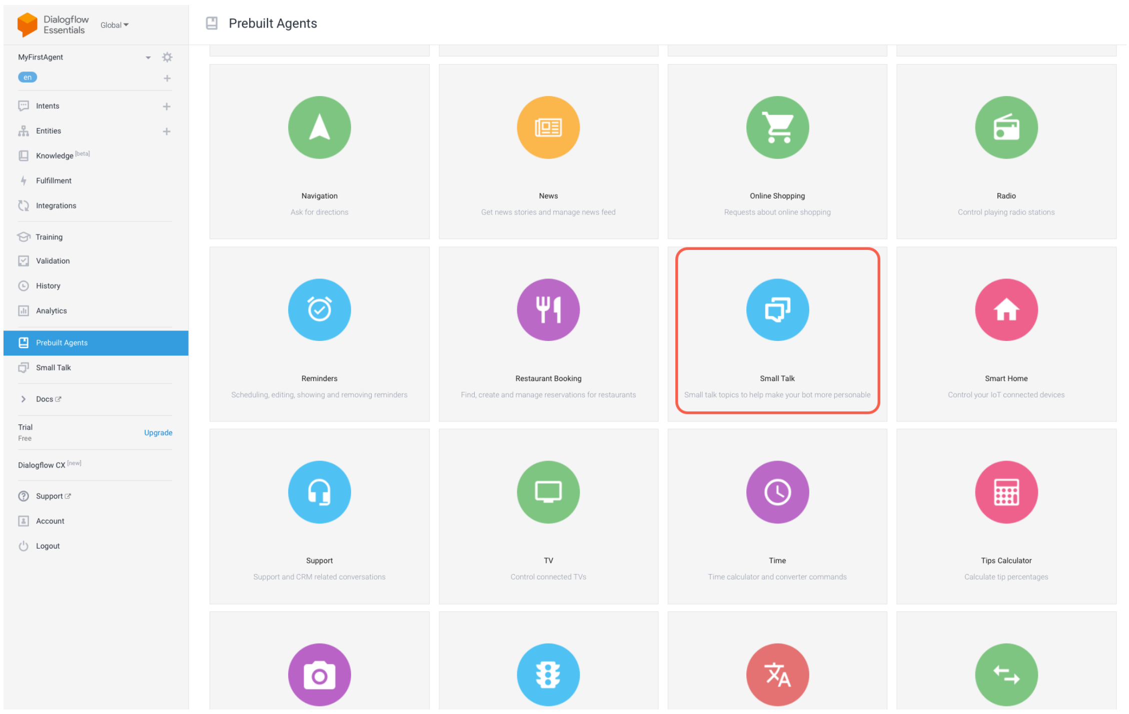
Combine multiple Dialogflow agents (i.e. sub agents) into a single agent (i.e. mega agent).

Give your agent a name and click "Create"

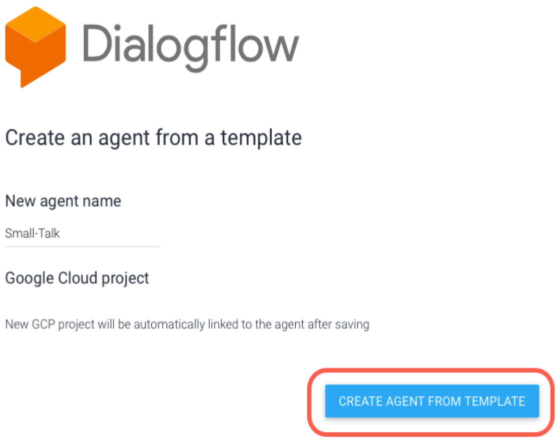
3. Exercise: Dialogflow Agent



On the sidebar, click the "Prebuilt Agents"


Dialogflow Essentials
Global
MyFirstAgent
en
Intents
Entities
Knowledge [beta]
Fulfillment
Integrations
Training
Validation
History
Analytics
Prebuilt Agents
Small Talk
Docs
Trial
Free
Upgrade
Dialogflow CX [new]
Support
Account
Logout
Prebuilt Agents
Navigation
Ask for directions
News
Get news stories and manage news feed
Online Shopping
Requests about online shopping
Radio
Control playing radio stations
Reminders
Scheduling, editing, showing and removing reminders
Restaurant Booking
Find, create and manage reservations for restaurants
Small Talk
Small talk topics to help make your bot more personable
Smart Home
Control your IoT connected devices
Support
Support and CRM related conversations
TV
Control connected TVs
Time
Time calculator and converter commands
Tips Calculator
Calculate tip percentages

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



Create an agent from a template

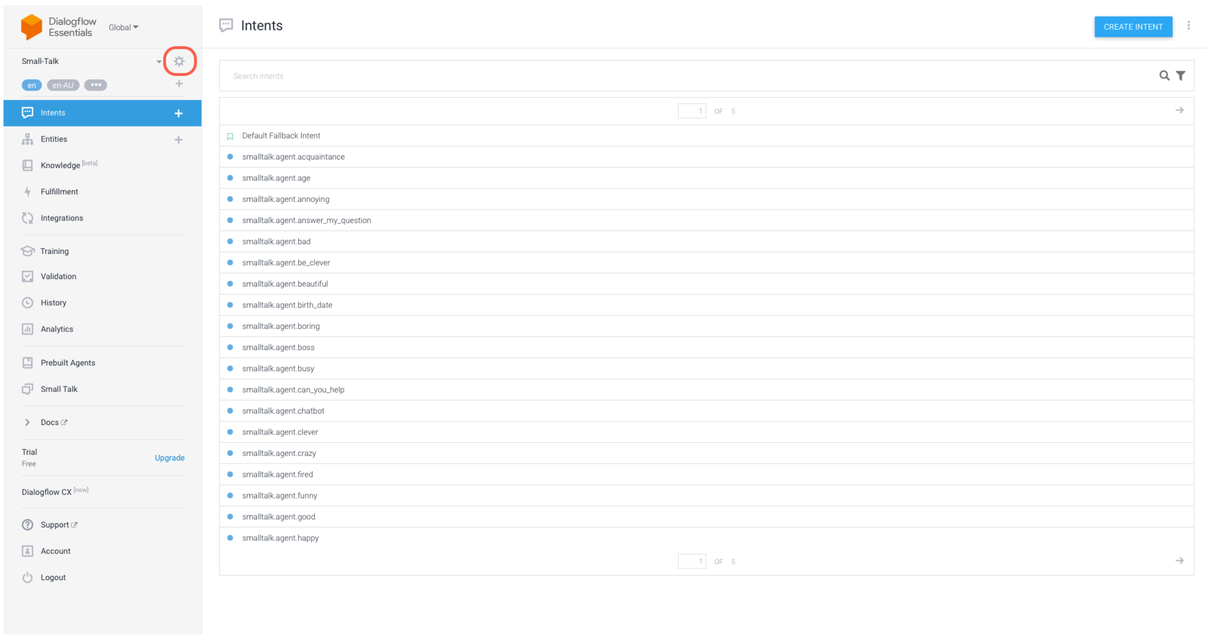
New agent name

Small-Talk

Google Cloud project

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

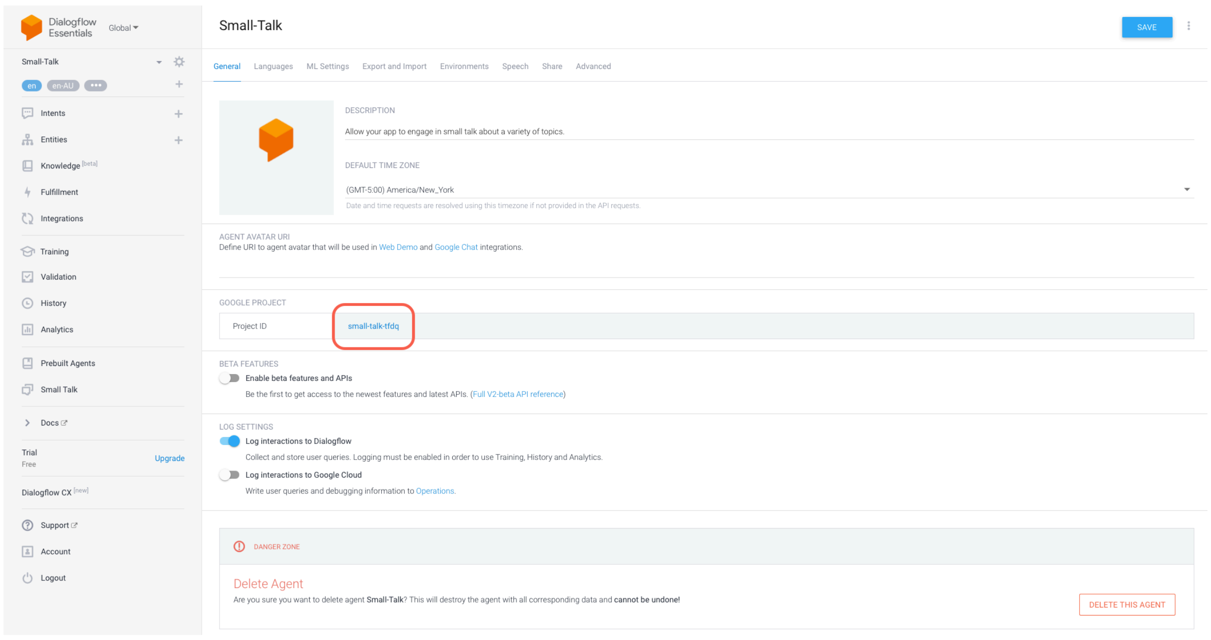
Leave the name on default and create your new agent from template



Intents

Default Fallback Intent
smalltalk.agent.acquaintance
smalltalk.agent.age
smalltalk.agent.annoying
smalltalk.agent.answer_my_question
smalltalk.agent.bad
smalltalk.agent.be_clever
smalltalk.agent.beautiful
smalltalk.agent.birth_date
smalltalk.agent.boring
smalltalk.agent.boss
smalltalk.agent.busy
smalltalk.agent.can_you_help
smalltalk.agent.chatbot
smalltalk.agent.clever
smalltalk.agent.crazy
smalltalk.agent.fired
smalltalk.agent.funny
smalltalk.agent.good
smalltalk.agent.happy

Next open your agent settings


Dialogflow Essentials · Global

Small-Talk · en · en-AU

Intents · Entities · Knowledge [beta] · Fulfillment · Integrations · Training · Validation · History · Analytics · Prebuilt Agents · Small Talk · Docs · Trial Free · Upgrade · Dialogflow CX [new] · Support · Account · Logout

Small-Talk · SAVE

General · Languages · ML Settings · Export and Import · Environments · Speech · Share · Advanced

DESCRIPTION
Allow your app to engage in small talk about a variety of topics.

DEFAULT TIME ZONE
(GMT-5:00) America/New_York
Date and time requests are resolved using this timezone if not provided in the API requests.

AGENT AVATAR URI
Define URI to agent avatar that will be used in Web Demo and Google Chat integrations.

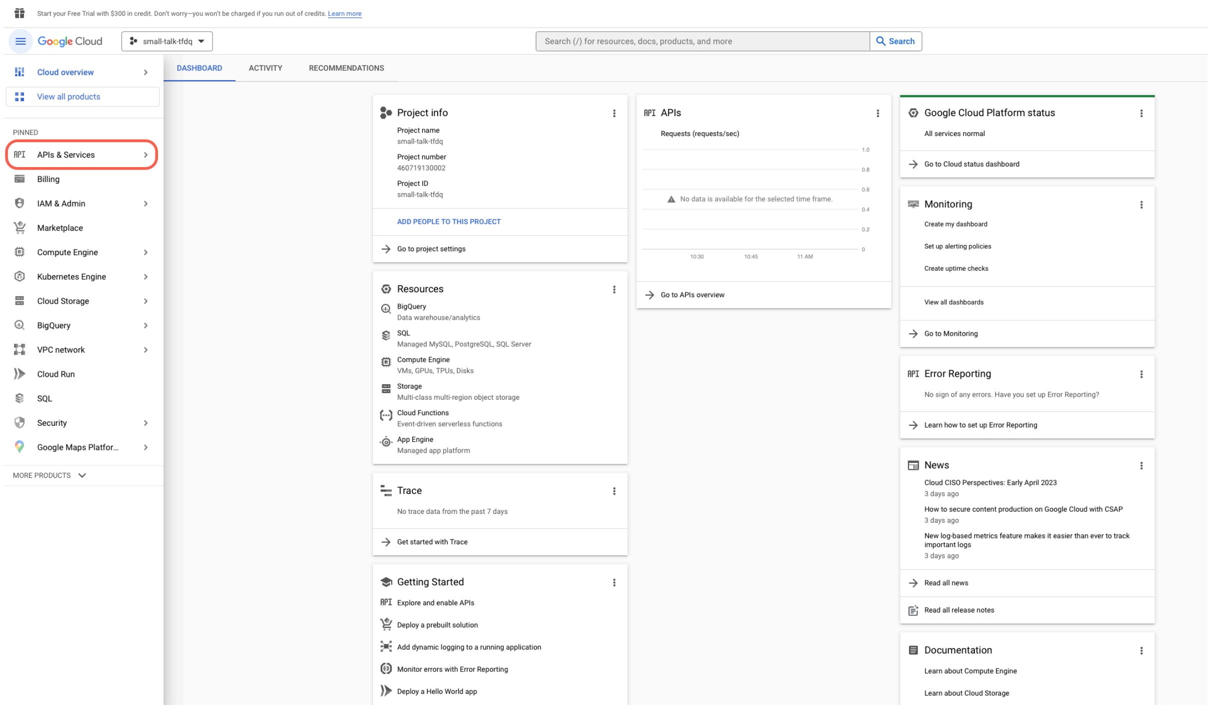
GOOGLE PROJECT
Project ID: small-talk-tfdq

BETA FEATURES
Enable beta features and APIs
Be the first to get access to the newest features and latest APIs. (Full V2-beta API reference)

LOG SETTINGS
Log interactions to Dialogflow
Collect and store user queries. Logging must be enabled in order to use Training, History and Analytics.
Log interactions to Google Cloud
Write user queries and debugging information to Operations.

DANGER ZONE
Delete Agent
Are you sure you want to delete agent **Small-Talk**? This will destroy the agent with all corresponding data and **cannot be undone!**
DELETE THIS AGENT

Click on your Project ID


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

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 Platfor... · MORE PRODUCTS

DASHBOARD · ACTIVITY · RECOMMENDATIONS

Project info
Project name: small-talk-tfdq
Project number: 460719130002
Project ID: small-talk-tfdq
ADD PEOPLE TO THIS PROJECT
Go to project settings

APIs
Requests (requests/sec)
No data is available for the selected time frame.
Go to APIs overview

Google Cloud Platform status
All services normal
Go to Cloud status dashboard

Monitoring
Create my dashboard
Set up alerting policies
Create uptime checks
View all dashboards
Go to Monitoring

Resources
BigQuery – Data warehouse/analytics
SQL – Managed MySQL, PostgreSQL, SQL Server
Compute Engine – VMs, GPUs, TPUs, Disks
Storage – Multi-class multi-region object storage
Cloud Functions – Event-driven serverless functions
App Engine – Managed app platform

Error Reporting
No sign of any errors. Have you set up Error Reporting?
Learn how to set up Error Reporting

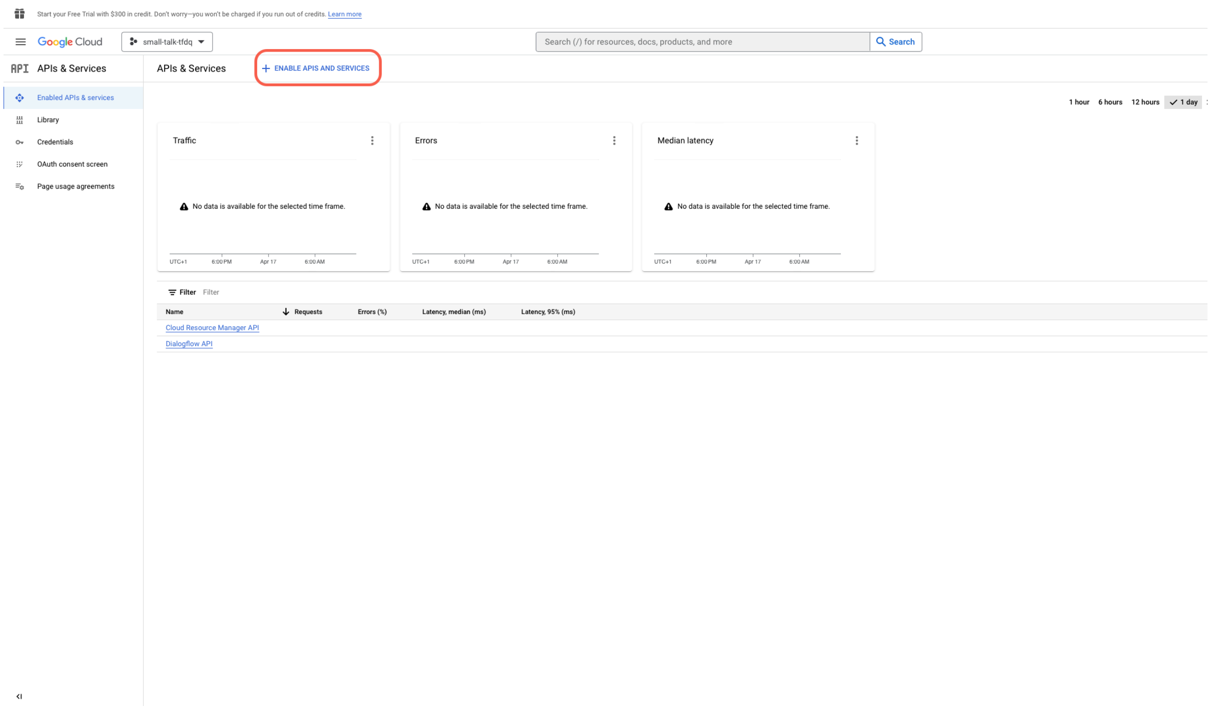
Trace
No trace data from the past 7 days
Get started with Trace

News
Cloud CISO Perspectives: Early April 2023 – 3 days ago
How to secure content production on Google Cloud with CSAP – 3 days ago
New log-based metrics feature makes it easier than ever to track important logs – 3 days ago
Read all news
Read all release notes

Getting Started
Explore and enable APIs
Deploy a prebuilt solution
Add dynamic logging to a running application
Monitor errors with Error Reporting
Deploy a Hello World app

Documentation
Learn about Compute Engine
Learn about Cloud Storage

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

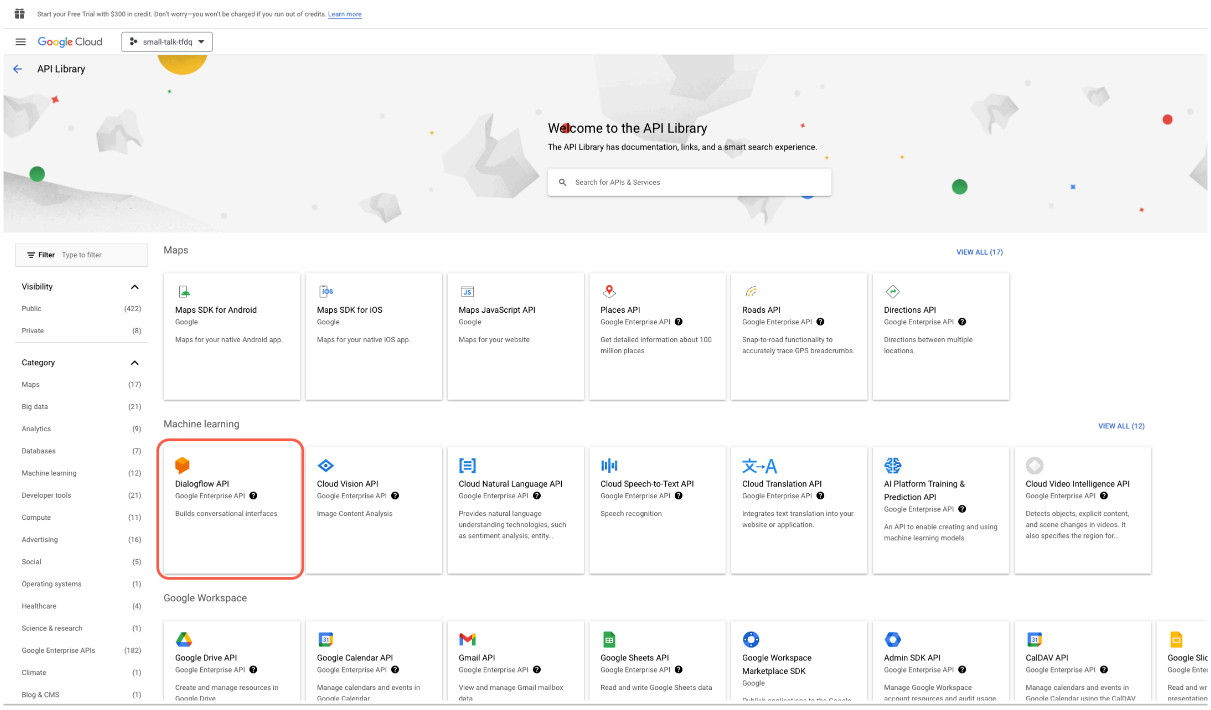


Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more

No data is available for the selected time frame.

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

Go to "Enable API and Services"



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

Welcome to the API Library

The API Library has documentation, links, and a smart search experience.

Visibility	
Public	(422)
Private	(8)
Category	
Maps	(17)
Big data	(21)
Analytics	(9)
Databases	(7)
Machine learning	(12)
Developer tools	(21)
Compute	(11)
Advertising	(16)
Social	(5)
Operating systems	(1)
Healthcare	(4)
Science & research	(1)
Google Enterprise APIs	(182)
Climate	(1)
Blog & CMS	(1)

Maps

- Maps SDK for Android — Google — Maps for your native Android app.
- Maps SDK for iOS — Google — Maps for your native iOS app.
- Maps JavaScript API — Google — Maps for your website
- Places API — Google Enterprise API — Get detailed information about 100 million places
- Roads API — Google Enterprise API — Snap-to-road functionality to accurately trace GPS breadcrumbs.
- Directions API — Google Enterprise API — Directions between multiple locations.

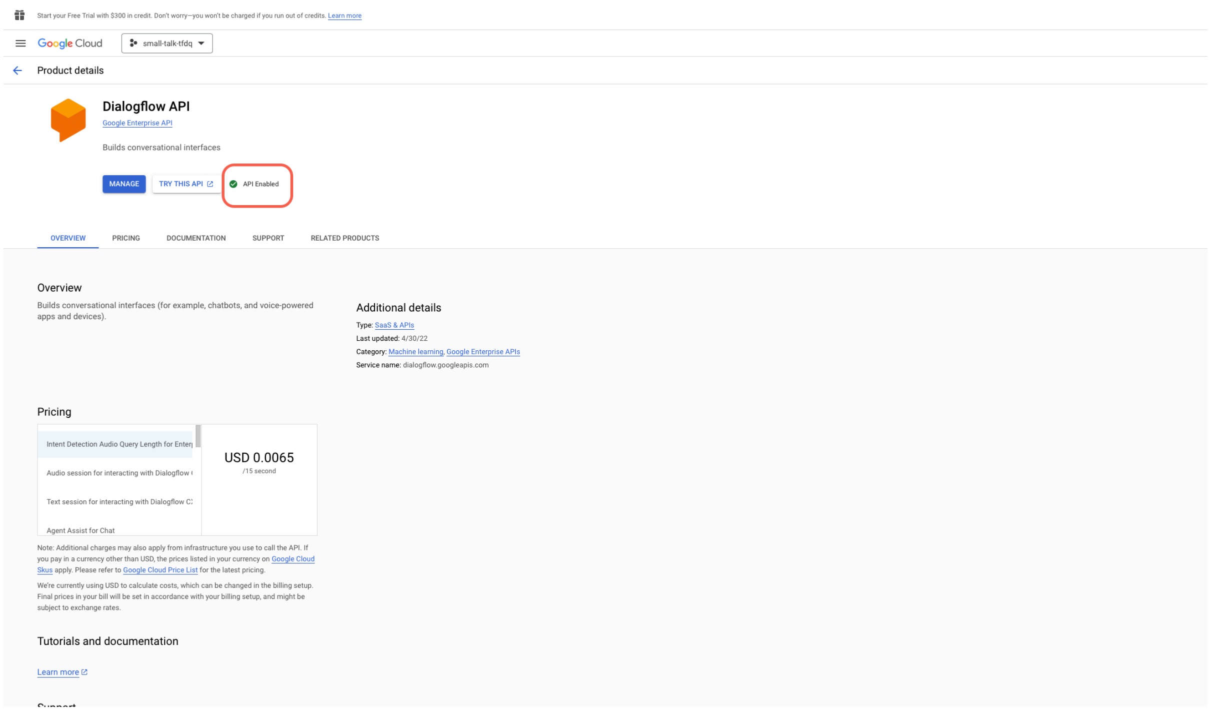
Machine learning

- Dialogflow API — Google Enterprise API — Builds conversational interfaces
- Cloud Vision API — Google Enterprise API — Image Content Analysis
- Cloud Natural Language API — Google Enterprise API — Provides natural language understanding technologies, such as sentiment analysis, entity...
- Cloud Speech-to-Text API — Google Enterprise API — Speech recognition
- Cloud Translation API — Google Enterprise API — Integrates text translation into your website or application.
- AI Platform Training & Prediction API — Google Enterprise API — An API to enable creating and using machine learning models.
- Cloud Video Intelligence API — Google Enterprise API — Detects objects, explicit content, and scene changes in videos. It also specifies the region for...

Google Workspace

- Google Drive API — Google Enterprise API — Create and manage resources in Google Drive
- Google Calendar API — Google Enterprise API — Manage calendars and events in Google Calendar
- Gmail API — Google Enterprise API — View and manage Gmail mailbox data
- Google Sheets API — Google Enterprise API — Read and write Google Sheets data
- Google Workspace Marketplace SDK — Google
- Admin SDK API — Google Enterprise API — Manage Google Workspace account resources and audit usage
- CalDAV API — Google Enterprise API — Manage calendars and events in Google Calendar using the CalDAV
- Google Slic — Google Ente — Read and wr presentation

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

Product details

Dialogflow API

Google Enterprise API

Builds conversational interfaces

Overview

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

Additional details

Type: SaaS & APIs
Last updated: 4/30/22
Category: Machine learning, Google Enterprise APIs
Service name: dialogflow.googleapis.com

Pricing

Intent Detection Audio Query Length for Enter	USD 0.0065 /15 second
Audio session for interacting with Dialogflow (	
Text session for interacting with Dialogflow C:	
Agent Assist for Chat	

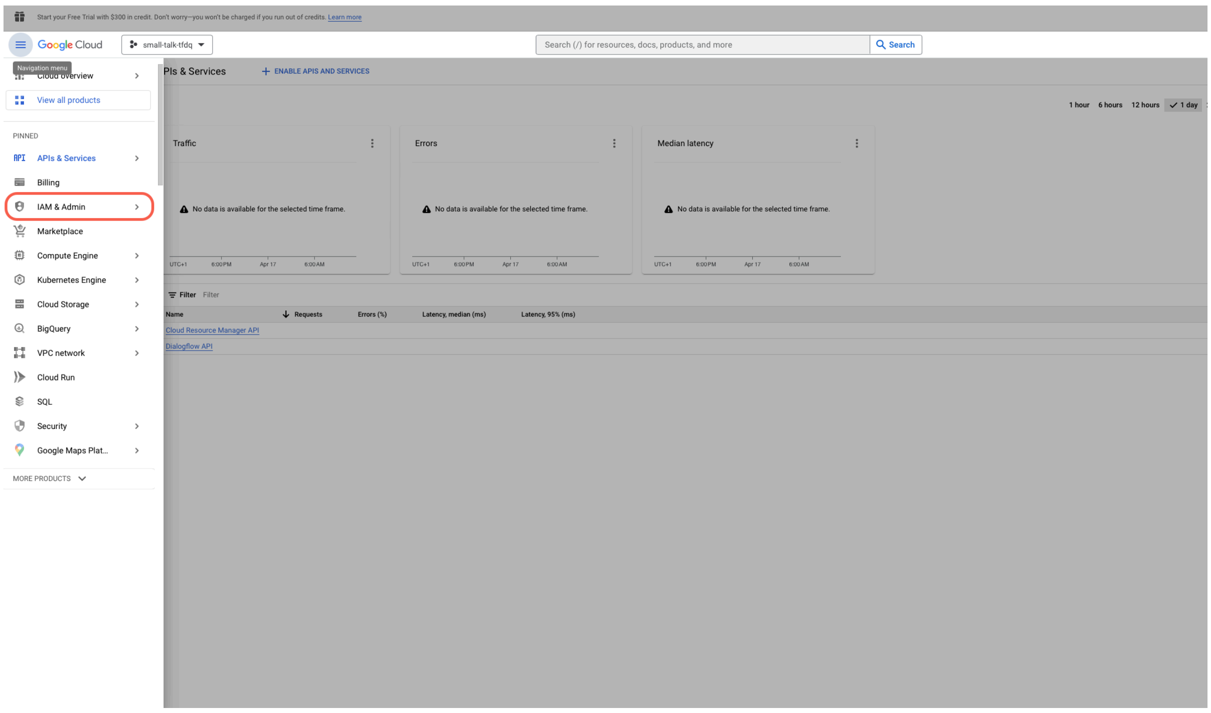
Note: Additional charges may also apply from infrastructure you use to call the API. If you pay in a currency other than USD, the prices listed in your currency on Google Cloud Skus apply. Please refer to Google Cloud Price List for the latest pricing.

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

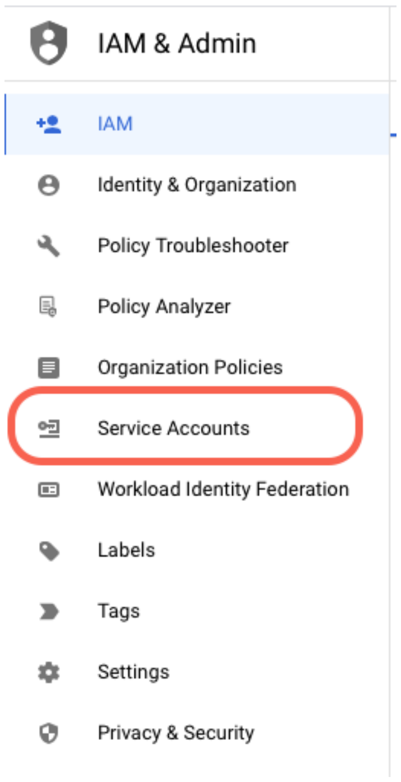
Make sure it is enabled


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

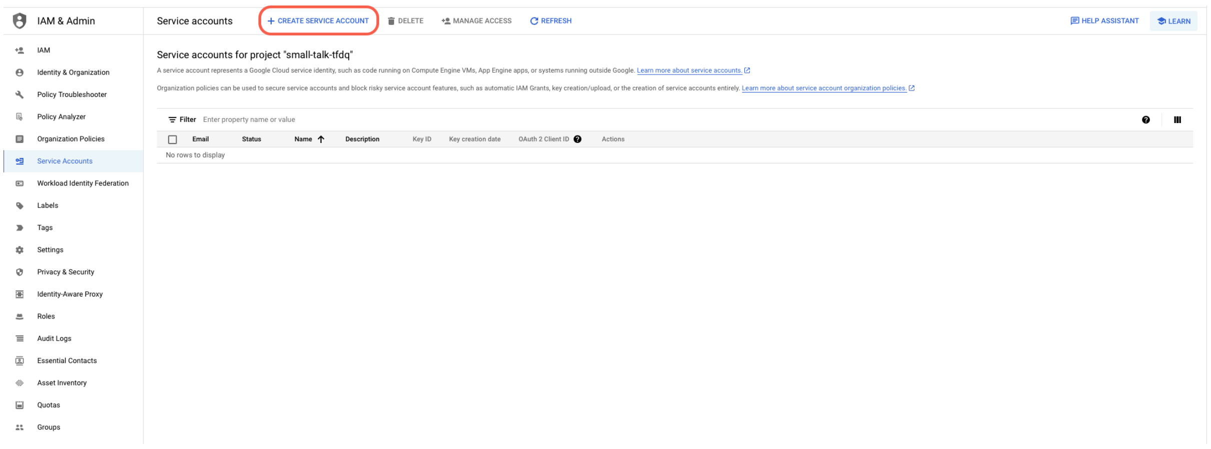
No data is available for the selected time frame.

Name	Requests	Errors (%)	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



Service accounts for project "small-talk-tfdq"

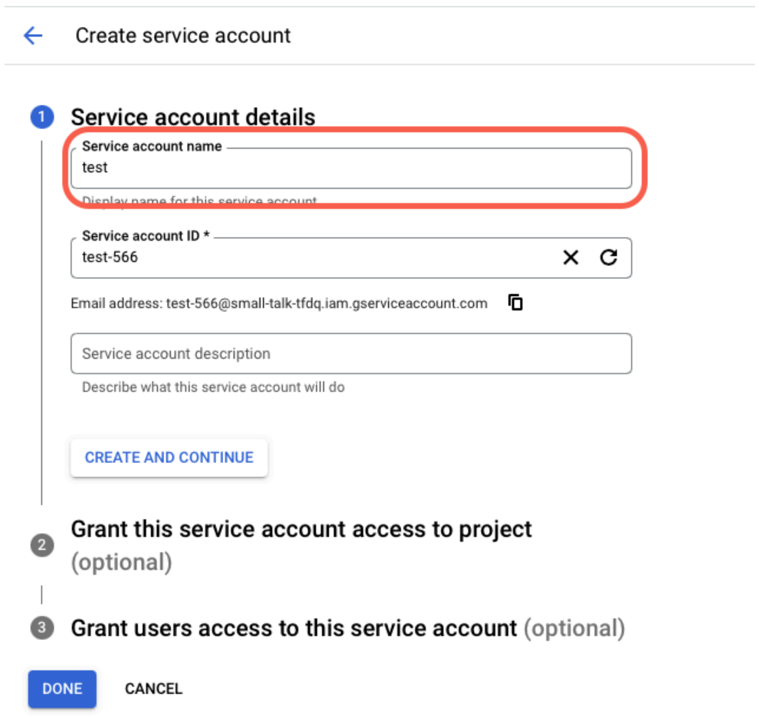
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.

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

	Email	Status	Name ↑	Description	Key ID	Key creation date	OAuth 2 Client ID	Actions
No rows to display								

Create a service account



Create service account

Service account details

Service account name: test

Display name for this service account

Service account ID *: test-566

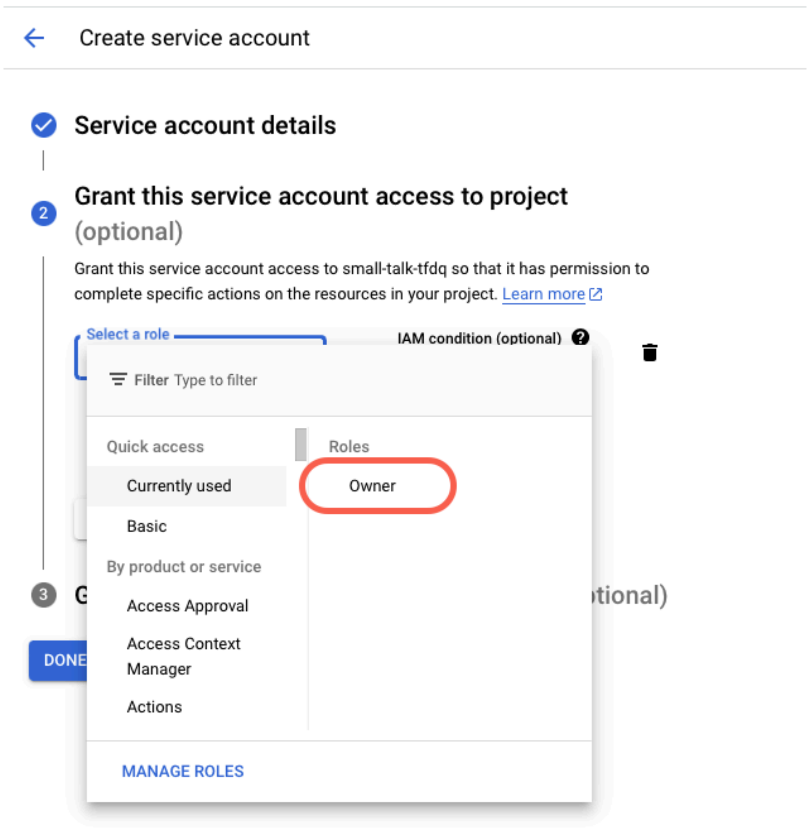
Email address: test-566@small-talk-tfdq.iam.gserviceaccount.com

Service account description

Describe what this service account will do

CREATE AND CONTINUE
Grant this service account access to project (optional)
Grant users access to this service account (optional)
DONE
CANCEL

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



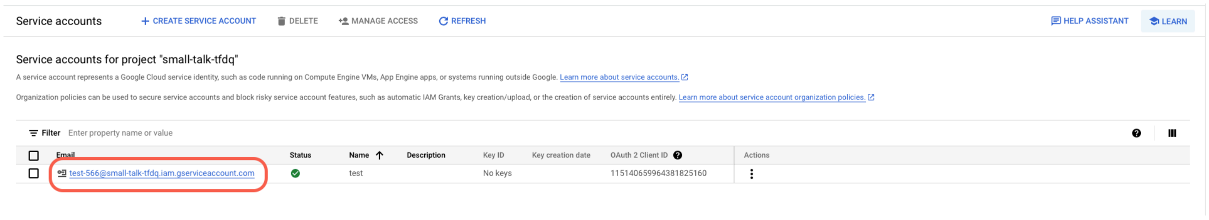
Create service account

Service account details

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

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



Service accounts for project "small-talk-tfdq"

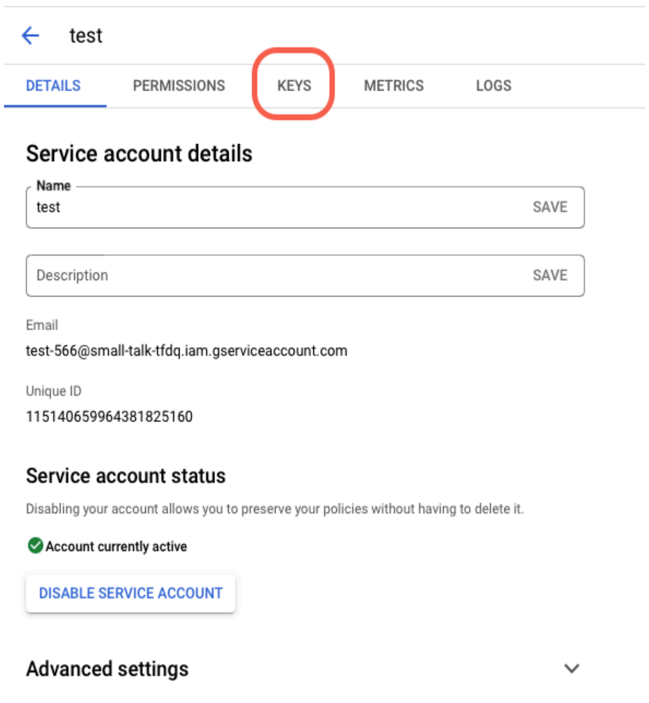
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.

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

Email	Status	Name	Description	Key ID	Key creation date	OAuth 2 Client ID	Actions
test-566@small-talk-tfdq.iam.gserviceaccount.com	✓	test		No keys		115140659964381825160	⋮

Click on your new account



Service account details

Name: test SAVE

Description SAVE

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

Unique ID
115140659964381825160

Service account status

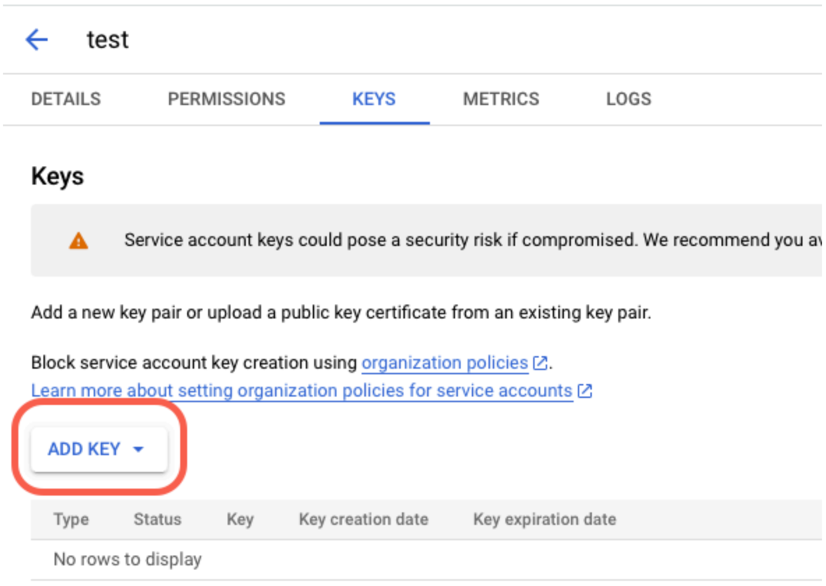
Disabling your account allows you to preserve your policies without having to delete it.

✓ 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 av

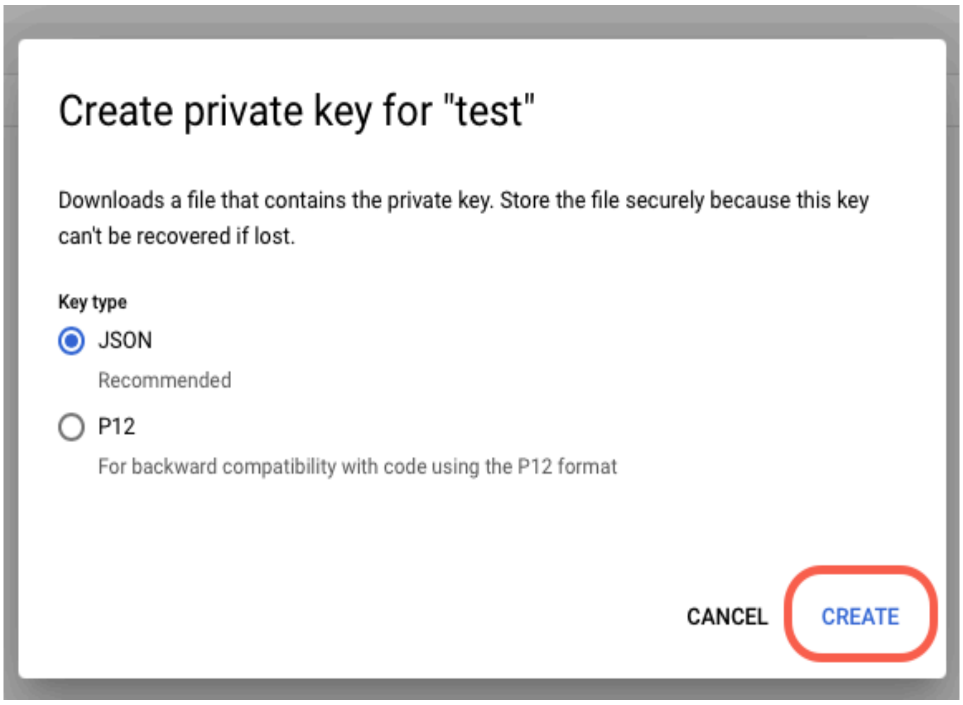
Add a new key pair or upload a public key certificate from an existing key pair.

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

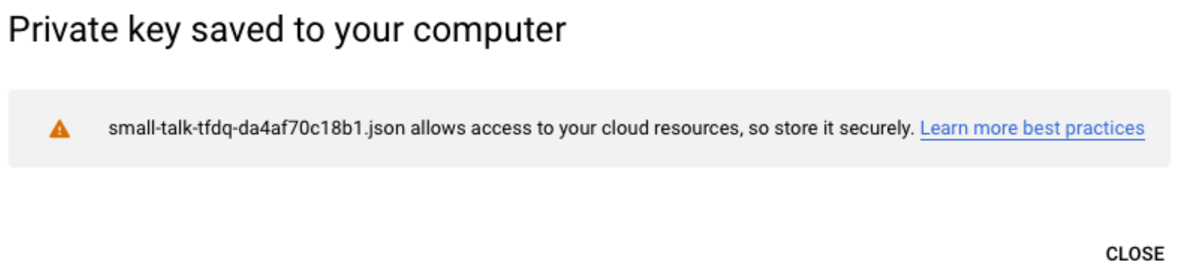

Create private key for "test"

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

A file should be downloaded and a pop up like this should show up

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.

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'
```

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. And lastly you need to route the server like this:

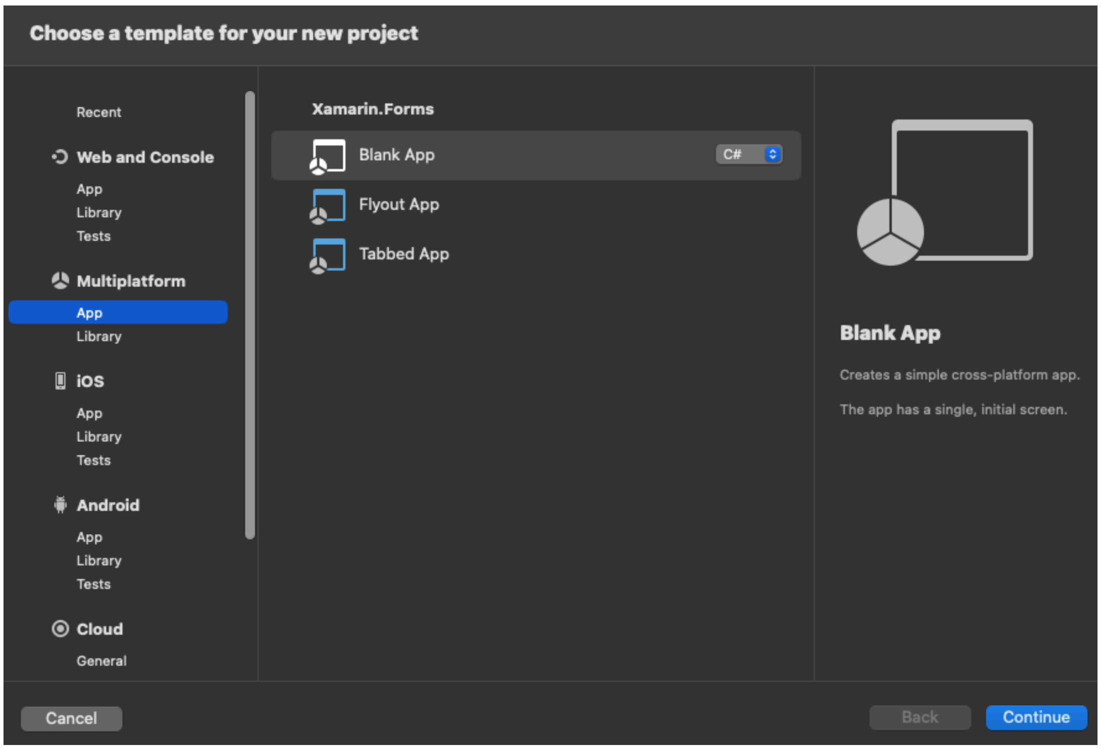
```
if __name__ == '__main__':
    app.run(port=9001, host="0.0.0.0")
```

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.

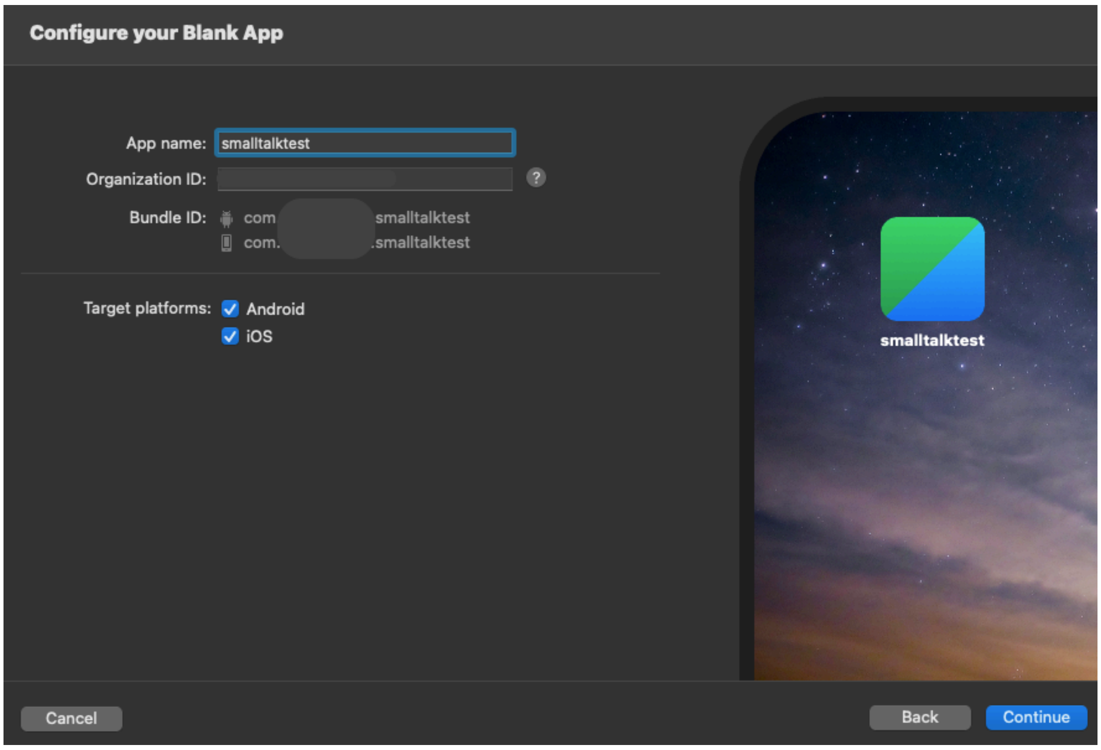


Blank App

Creates a simple cross-platform app.

The app has a single, initial screen.

Select a Multiplatform Blank App in C#



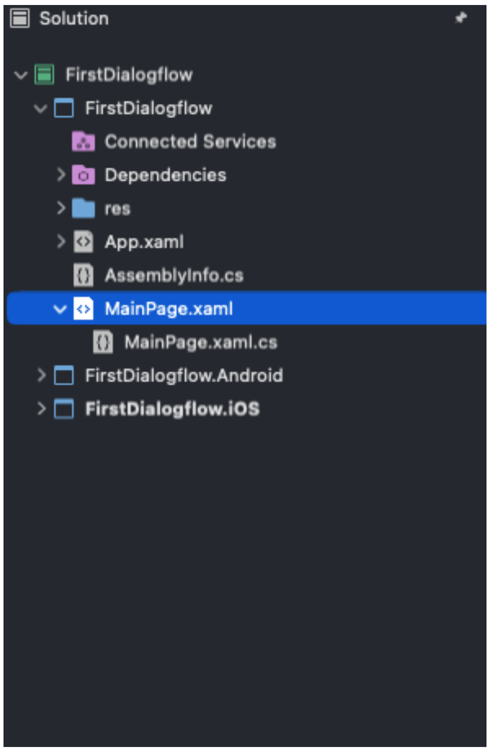
App name: smalltalktest

Organization ID: [redacted]

Bundle ID: com [redacted] smalltalktest
com. [redacted] .smalltalktest

Target platforms: ✓ Android
✓ iOS

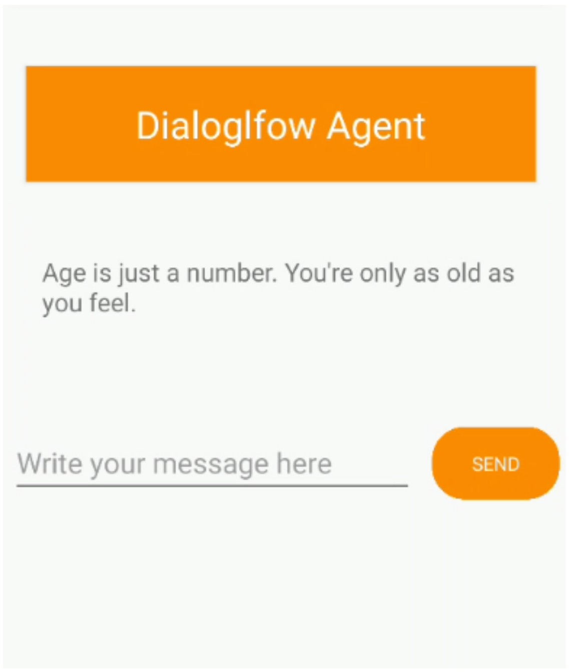
Give your app a name of your choice



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 textbook to receive Dialogflow's responses, an input to send messages and a Send button.

The following code generates the previous layout:

```
<StackLayout>
        <Frame BackgroundColor="DarkOrange"                          Padding="20"
CornerRadius="0" Margin="20, 40">
            <Label Text="Dialoglfow 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>
```

And this snippet creates the desired behaviour:

```
  public partial class MainPage : ContentPage
  {
    private SessionsClient sessionsClient;
    private SessionName sessionName;
    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.

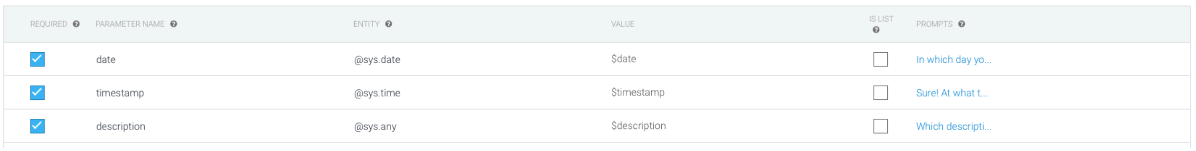
4.5 - Try three other examples of messages and register the corresponding answers.

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 the intent again.



REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
✓	date	@sys.date	$date	☐	In which day yo...
✓	timestamp	@sys.time	$timestamp	☐	Sure! At what t...
✓	description	@sys.any	$description	☐	Which descripti...

New intent parameters

Second Didactic Sheet

This second sheet continues to explore Xamarin application development but develops a Device Listing service created using Python and the programming of an Arduino module that will communicate with the Python server. In this sheet, improvements are made to the Xamarin overall design, while developing a new Python server using the Flask framework and by fostering a creative and challenging environment supported by a set of questions that follow a logical development of the overall ecosystem.

Universidade de Coimbra
Faculdade de Ciências e Tecnologia

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E COMPUTADORES
Redes de Comunicação

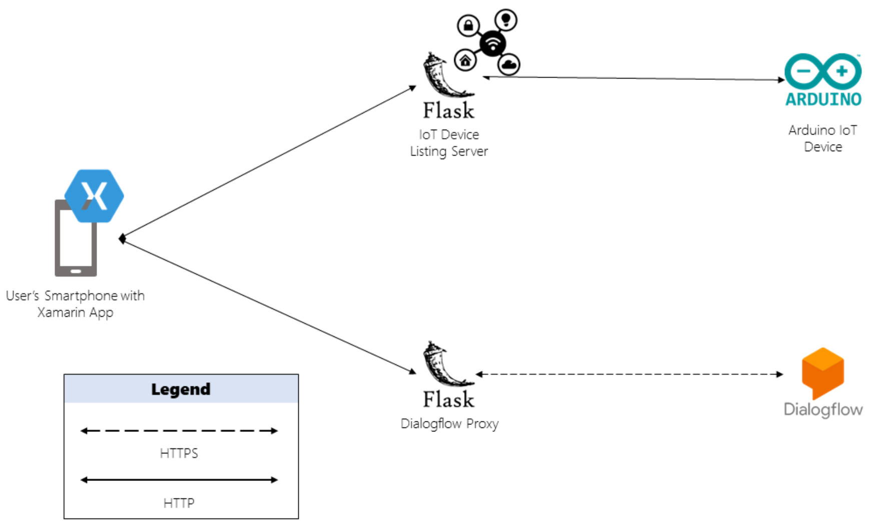
Intermediary Server
@João Antunes

Dialogflow and Xamarin

2023/24

1. Introduction

The primary objective of this assignment is to improve upon our earlier work and advance our chatbot application. In previous work, we looked into conversational AI frameworks, Python servers, Natural Language Processing (NLP), and user interfaces (UIs). We will now concentrate on improving the UI design in this relevant new assignment and incorporating the capability to show a list of IoT (Internet of Things) devices provided by a server. Our goal is to improve the program further while constantly focusing on simplicity and usability.



Assignment Architecture

As you can see in the image above, it is an overview of the global architecture of this assignment. We have at the center the Xamarin.Forms mobile application, the application will interact with the Dialogflow proxy that in turn will communicate with the Dialogflow API, providing the chatbot experience. In this new assignment, we are adding a new component, a IoT Device Listing server, this server is created using the Flask Python's framework, this server will handle request and will serve as a repository for IoT devices that must be listed in the environment.

2. Requirements

The requirements for this assignment are the same as the ones described on the previous assignment, no new tools or frameworks are used since we are building and improving upon the work already done.

Main Page

This code introduces a different feature that will be important to obtain the answer to the second question. Pay attention to the following code, analyze it carefuly and try and catch any differences in the implementation.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
using Xamarin.Forms;
using RestSharp;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

namespace PPrivacyAssistant
{    public partial class MainPage : ContentPage
    {
        private static readonly string ChatBotProxyAddress =
$"http://<IP>: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;
                if (chatText == "**deviceList**")
                {
                    await NavigateTo();
                }
                return chatText;
            }

            return null;
        }
        public static async Task NavigateTo()
        {
            await
((NavigationPage)App.Current.MainPage).Navigation.PushAsync(new
DevicesPage());
        }
    }
}
```

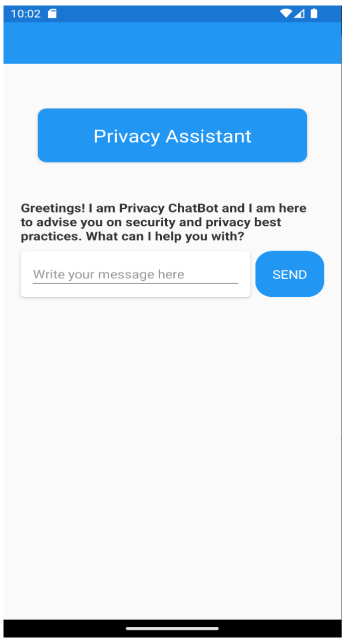
Main Page Styling

The first step in this assignment will be to update the styling of the app. To do this, just copy and paste the following code to the **MainPage.xaml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="PPrivacyAssistant.MainPage">


    <StackLayout Padding="20" Spacing="10">
        <Frame BackgroundColor="#2196F3" Padding="20" CornerRadius="10"
Margin="20, 40">
            <Label Text="Privacy Assistant" HorizontalTextAlignment="Center"
TextColor="White" FontSize="24"/>
```

```
        </Frame>

        <ScrollView VerticalScrollBarVisibility="Always">
            <StackLayout x:Name="ChatContainer" Spacing="10">
                <Label x:Name="AnswerLabel" FontSize="16"
TextColor="#333333" FontAttributes="Bold"/>
            </StackLayout>
        </ScrollView>

        <Grid VerticalOptions="End">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <Frame Grid.Row="0" Grid.Column="0" BackgroundColor="#FFFFFF"
CornerRadius="5" Padding="0">
                <Entry x:Name="InputQuery" Placeholder="Write your message
here" PlaceholderColor="#999999" Margin="10" FontSize="16"
BackgroundColor="Transparent" TextColor="#333333"/>
            </Frame>
            <Button Grid.Row="0" Grid.Column="1" Text="Send"
BackgroundColor="#2196F3" TextColor="White" CornerRadius="20" FontSize="16"
HeightRequest="40" WidthRequest="80" Clicked="SendButton_Clicked"/>
        </Grid>
    </StackLayout>
</ContentPage>
```

This new styling will make the Chatbot page look like the following.



Greetings! I am Privacy ChatBot and I am here to advise you on security and privacy best practices. What can I help you with?

Main Chatbot Page

3. IoT Device Listing Server

A Python file that uses the Flask framework is created. This server will receive a POST request, this will save new IoT devices to be listed and saved. The GET request will return the IoT device list to the mobile application. Use the following code to create a new Python file called **iotDeviceListing.py**.

```
from flask import Flask, jsonify, request
```

First step is to import the needed libraries. If any of these are not recognised, install them using *pip*. Next add the following lines, the **devices = []** will instantiate the list that will be used to save information about the devices.

```
app = Flask(__name__)

devices = []
```

Add the following code to manage the POST requests, this will handle any request to add a new IoT device that must be listed inside the server.

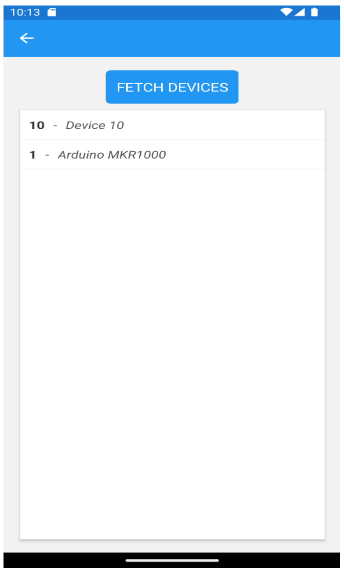
```
@app.route('/devices', methods=['POST'])
def add_device():
    # Get device data from request
    device_data = request.get_json()

    # Get Device ID and Device Name
    device_id = device_data['DeviceId']
    device_name = device_data['DeviceName']

    # Add device to list
    devices.append({'DeviceId': device_id, 'DeviceName': device_name})

    # Return success message
    return jsonify({'success': True})
```

The next code will handle any GET requests. The Xamarin mobile app will send a GET request, this will handle that request providing a list of IoT devices saved inside the server.

```
@app.route('/devices', methods=['GET'])
def get_devices():
    # Send devices
    return jsonify(devices)

if __name__ == '__main__':
    app.run(host="0.0.0.0",port=49152)
```

4. Xamarin App - Device Listing

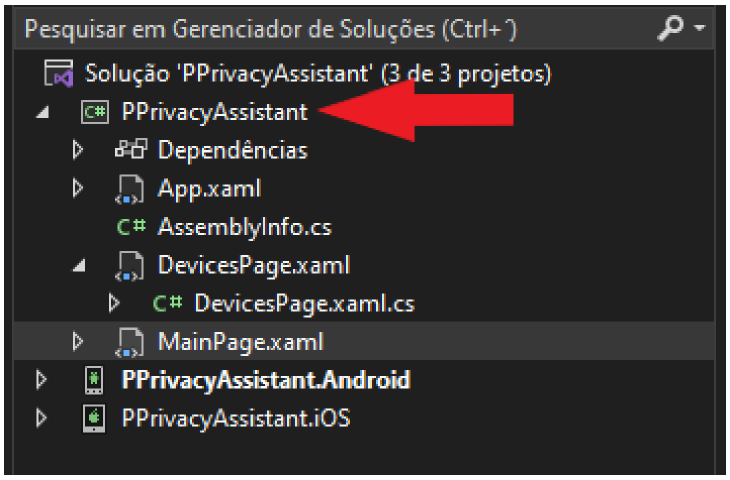
From the code of the Xamarin mobile application that was provided in the previous assignment, you can observe that there is currently a single page for the user, the chatbot page. To have a page where the IoT devices will be listed, a new one is needed. This page dedicated to the listing of IoT devices will look like the following.



IoT Device Listing Page

To do this follow the next steps.

1. Right click with your mouse on the project (as can be seen in the image below).



Add a new page

2. Select **Add** and followed by **New item**.
3. Select the Xamarin.Forms type of file and select **Content Page**.
4. Change the name to **DevicesPage** and **Add**.

After creating this new page, two new files are created. The first one, the one we will change the code right now is the **DevicesPage.xaml.cs**.

DevicesPage.xaml.cs

In this file, the connection between the Xamarin.Forms mobile application and the IoT Device Listing server is created and maintained. This code will send GET requests to the

Python server, saving the information so it can be listed in the new page with proper styling. Analyze the following code, understand what is done and copy it to the file.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Net.Http;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Xamarin.Forms;
using RestSharp;

namespace PPrivacyAssistant
{
    public partial class DevicesPage : ContentPage
    {
        private static readonly string iotProxy =
$"http://<IP>:49152/devices";
        public DevicesPage()
        {
            InitializeComponent();
        }
        public List<Device> Devices { get; set; }

        private async Task FetchAndDisplayDevicesAsync()
        {
            var options = new RestClientOptions($"{iotProxy}")
            {
                RemoteCertificateValidationCallback = delegate {
                    return true;
                }
            };
            // Create a new RestSharp client
            var client = new RestClient(options);
            var request = new RestRequest();
            var response = await client.ExecuteAsync(request);

            if (response.IsSuccessful)
            {
                // Deserialize the JSON
                List<Device> devices =
JsonConvert.DeserializeObject<List<Device>>(response.Content);
                DevicesListView.ItemsSource = devices;
            }
        }

        private async void FetchDevicesButton_Clicked(object sender,
EventArgs e)
        {
            await FetchAndDisplayDevicesAsync();
        }
    }

    public class Device
    {
        public int DeviceId { get; set; }
        public string DeviceName { get; set; }
    }
}
```

The IP Address should be changed to fit your own IP Address, like it was done in the previous assignment.

DevicesPage.xaml

This file (**DevicesPage.xaml**) will handle the UI and the styling for the page. This will display a list of the IoT devices and will provide a **Fecth** button that will send a GET request to the IoT Device Listing server to update the list.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="PPrivacyAssistant.DevicesPage"
             BackgroundColor="#F2F2F2">
    <StackLayout Spacing="10" Padding="20">
        <Button Text="Fetch Devices"
                Clicked="FetchDevicesButton_Clicked"
                HorizontalOptions="Center"
                BackgroundColor="#2196F3"
                TextColor="White"
                FontSize="18"
                CornerRadius="5"
                HeightRequest="50"/>

        <ListView x:Name="DevicesListView"
                  BackgroundColor="White"
                  SeparatorColor="#CCCCCC">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout Orientation="Horizontal" Padding="10">
                            <Label Text="{Binding DeviceId}"
                                   FontAttributes="Bold"
                                   VerticalOptions="CenterAndExpand"
                                   FontSize="16"
                                   TextColor="#333333"/>
                            <Label Text=" - "
                                   FontSize="16"
                                   VerticalOptions="CenterAndExpand"
                                   TextColor="#666666"/>
                            <Label Text="{Binding DeviceName}"
                                   FontAttributes="Italic"
                                   VerticalOptions="CenterAndExpand"
                                   FontSize="16"
                                   TextColor="#555555"/>
                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

Analyze the code above, pay special attention to the button section and the **Clicked=** part.

```
<Button Text="Fetch Devices"
            Clicked="FetchDevicesButton_Clicked"
            HorizontalOptions="Center"
            BackgroundColor="#2196F3"
            TextColor="White"
            FontSize="18"
            CornerRadius="5"
            HeightRequest="50"/>
```

Now just copy and paste the code to the appropriate file.

Arduino - IoT Device

In this section, an Arduino Mkr 1000 will be used to simulate an IoT device. This device will be used to register information on the IoT Device Listing server. Create a new Arduino file and copy the following code:

```
#include <WiFi101.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>

char ssid[] = "<WIFI-SSID>";
char password[] = "<WIFI-PASSWORD>";
const char* serverAddress = "<IoT Device Listing Server IP>";
int serverPort = <IoT Device Listing Server Port>;

unsigned long lastSendTime = 0;
unsigned long sendInterval = 60000;

WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Connect to Wi-Fi
  while (WiFi.begin(ssid, password) != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    delay(1000);
  }

  Serial.println("Connected to Wi-Fi");

  // Send device data to Flask server
  sendData2Server();
}

void loop() {
  if (millis() - lastSendTime >= sendInterval) {
    // Send device data to Flask server
    sendData2Server();

    // Update the last send time
    lastSendTime = millis();
  }
}

void sendData2Server() {
  // Create JSON payload
  DynamicJsonDocument jsonDoc(128);
  jsonDoc["DeviceId"] = 1;
  jsonDoc["DeviceName"] = "Arduino MKR1000";

  String payload;
  serializeJson(jsonDoc, payload);

  // Connect to the server
  if (client.connect(serverAddress, serverPort)) {
    Serial.println("Connected to server");

    // Make a POST request
    client.println("POST /devices HTTP/1.1");
    client.println("Host: " + String(serverAddress));
    client.println("Content-Type: application/json");
    client.print("Content-Length: ");
    client.println(payload.length());
    client.println();
    client.println(payload);

    // Wait for the server's response
    while (client.connected()) {
      if (client.available()) {
        String response = client.readStringUntil('\n');
        Serial.println(response);
      }
    }
  } else {
    Serial.println("Connection to server failed");
  }

  client.stop();
}
```

5. Exercises

1. Add the code listed above and pay attention to the code, analyze it carefully.
2. Create a new Dialogflow intent that will respond to the user query that asks to list the IoT devices. You should pay attention to the code of the Xamarin application and find the word that the application will pick up to make the transition between the Chatbot page and the IoT Device List page.
3. Add some new intents giving some security and privacy advice to a user. Example: If a user asks how it can protect itself on a public network.

4. Create a Python script using the Flask framework that will send a POST request to the IoT Device Listing server. The POST request should include the following information: Device ID, Device Name in JSON format. Tip: Use the **requests** library in Python.

5. Run the Dialogflow Proxy and the new IoT Device Listing Device server and test them with the Xamarin mobile application

6. Update the Arduino code provided above and change the values of the the ssid, password, serverAddress and serverPort to fit your own needs. Re-run the mobile application and test using the Arduino module.

7. Bonus question: Add more information to the IoT Device Listing service besides the Device ID and Name. You should change the appropriate Python server and the Xamarin mobile application to handle this new information.