Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: **Sergi Girabet Sala**

Titulació: Doble Grau en Enginyeria Informàtica i ADE

Títol de Treball Final de Grau: **Development of a voice recognition clocking-in system within the business environment using an Alexa skill**

Director/a: **Jordi Planes Cid / Daniel Galan Vilella**

Presentació

Mes: Juny

Any: 2023

Acknowledgments

First of all, I would like to express my sincere gratitude to my esteemed tutors, Jordi Planes and Daniel Galan, who served as the directors of this project, for their invaluable support and guidance throughout its development. Moreover, I would like to express my gratitude to SEMIC Group for not only providing me with a chance but also presenting me with an extraordinary opportunity to undertake my final degree project alongside them In particular, I wish to express my appreciation to Bianca and the Innolab department team for placing their trust in me and my abilities to undertake a project of this magnitude.

I would also like to extend heartfelt thanks to my entire family and friends, especially Belén, who have been a fundamental pillar on my journey, providing unwavering support and helping me overcome obstacles during moments of tension due to the significance of this final project.

Last but not least, thanks to Xavier Portilla, the founder of the Spanish Alexa community, and Andy Whitworth, a contributor from the Alexa developer community, for answering my questions related to the development of the Alexa skill and providing me with useful information and documentation for better development of the skill.

This experience has been enriching and would not have been possible without the collaboration and support of all these individuals and entities. I am deeply grateful for the invaluable support I have received throughout this process, and I will forever be indebted to each and every one of them.

Index

Index of Figures

Index of Tables

Abstract

This document presents a comprehensive exploration of the integration of a voice clocking-in-in system into SEMIC Group's business operations, focusing on the development of an Alexa skill capable of recognizing and distinguishing the voice of the workers to facilitate clocking-in. The introduction provides the context and motivation behind the study, along with the objectives to be achieved. The SEMIC Group and its registration system is introduced.

The document delves into the business case for implementing a voice clocking-in-in system, analyzing existing clocking-in-in systems and highlighting the potential benefits for SEMIC Group. Voice assistants and their history are examined, emphasizing the fundamental technologies such as artificial intelligence, machine learning, and natural language processing.

The impact of voice technology in the market is explored, considering worldwide usage of voice assistants and the implications for consumers and the overall market. The entry of new AI technologies and barriers to voice technology adoption are also discussed.

Specifically, the document focuses on the Alexa voice assistant, its functioning, and the Alexa Skill Kit, including voice interaction models and skill development workflows. The integration of the clocking-in API with the Alexa skill is explained, along with the prototype development process and tools used.

The cost of implementing the clocking-in-in system is analyzed, taking into account hardware and software costs as well as human costs. Finally, the document concludes with key findings and future perspectives for the integration of voice technology into SEMIC Group's operations.

Overall, this document provides a comprehensive analysis of the integration of a voice clocking-in-in system into SEMIC Group, offering insights into the potential benefits, implementation process, and cost considerations.

Keywords: Voice Assistant, SEMIC, Artificial Intelligence, Natural Language Processing, Alexa, Alexa Skills Kit, Alexa Developer Console, API, Django, MySQL, PostgreSQL, Heroku.

List Of Abbreviations

Alexa Developer Console	ADC
Alexa Skill Kit	ASK
Alexa Voice Service	AVS
Amazon Web Services	AWS
Application Programming Interface	API
Artificial Intelligence	AI
Automatic Speech Recognition	ASR
Compound Annual Growth Rate	CAGR
Computer Vision	CV
Coronavirus	COVID-19
Data Base	DB
Deep Learning	DL
Extensible Markup Language	XML
Hypertext Transfer Protocol Secure	HTTPS
Information and Communications Technology	ICT
Instituto Nacional de Estadística	INE
JavaScript Object Notation	JSON
Knowledge representation and reasoning	KRR
Machine Learning	ML
Machine learning algorithms	MLA
Natural Language Processing	NLP
PriceWaterhouseCoopers	PWC
Servicios Microinformática S.A	SEMIC
Simple Object Access Protocol	SOAP
Software Development Kit	SDK
Uniform Resource Locator	URL

1. Introduction

Surely you have heard of Alexa, Google Assistant, Cortana, Siri, etc., or even use them daily. These new technologies, mostly known as digital or virtual voice assistants, are software that will allow you to interact with them through voice commands to help you with various tasks. Not so long ago, we all would have thought that it would be unimaginable to be able to use a technology like this called Artificial Intelligence (AI), but times change and evolve faster than the human consciousness itself. [1]

Did you know that almost 50% of searches are expected to be voice searches? This data project good opportunities for voice-based marketing strategies, which will no longer be based on positioning some keywords but on solving questions and doubts of users. [2]

1.1. Context and Motivation

During these years studying the double degree of Computer Engineering and Business Administration and Management, I have had the opportunity to learn from many subjects that have allowed me to have knowledge of many different fields.

In addition to this knowledge, this last year I have been working with the company Servicios Microinformática S.A (SEMIC), and this has opened an even wider range, as I have been able to understand and apply many of the things given in these last years of the career.

Because of this, my company tutor, Daniel Galan, asked me to improve the company's personnel clocking-in system, since it was a project that he had in mind for some time, and, knowing that I liked challenges and new technologies, he did not hesitate to ask me about it.

So, my tutor offered me this idea in which I would have to automate the current clocking-in system of the company by one that was by voice clocking-in, in which employees would not have to enter the company's website or put their finger to clock in. The idea seemed to me an interesting challenge that I had never heard or seen before, and I didn't think twice about it.

1.2. Objectives

The main objective of this project is to implement a voice clocking-in system and to analyze the cost that this would have, in case SEMIC wanted to implement it. So, to understand and achieve this objective, I have divided the project into different objectives.

The main objectives related to the bachelor's degree in computer engineering in this project are the following:

- Understand the new voice recognition technologies and how they work, as well as deepen in one of these technologies called Alexa and understand how it works.
- Design, develop and test a voice recognition system thanks to the tools that the Alexa voice system and Alexa Skills Kit offers.
- Integration of this voice recognition system with the current clocking-in system used by SEMIC.

The main objectives related to the bachelor's degree in business administration and management in this project are the following:

- To understand and analyze the impact of speech recognition technologies on the market as well as the consumers who make use of these technologies.
- Calculate the estimated cost for the implementation of this new voice clocking-in system at SEMIC and all that it implies: hardware support, system maintenance and the hours worked in it, among others.

2. SEMIC Group

2.1. SEMIC Group History

SEMIC was founded in 1982 by Oscar López, Alvaro Perera, and Xavier Puertas and it currently has 10 offices in Lleida, Barcelona, Girona, Manresa, Madrid, Sevilla, Valencia, Zaragoza, Andorra, Canarias, and it has also more than 130 service points.

Figure 1: SEMIC logo. [Courtesy of SEMIC, used with permission]

Since it was founded, starting with only one store in Lleida, SEMIC has undergone incredibly significant growth. It has opened offices in the cities mentioned, and it increased the workforce to more than 400 employees. In recent years, it has managed to significantly increase its turnover, from € 66 million in 2018 to € 96,6 million in 2022. *[Courtesy of SEMIC, used with permission]*

2.2. What is SEMIC Group?

SEMIC is a global provider of Information and Communication Technologies (ICT) solutions and services that help companies and public administration to incorporate technology in a natural way.

The SEMIC group is part of SLAKSEMIC, S.L., a holding company which in turn has other companies, but nevertheless, at the end of 2022, the company Econocom [3] acquired a majority stake in the SEMIC group.[4] It is now called SEMIC Econocom and consists of two companies (Econocom and SLAKSEMIC, S.L.) that operate independently but are integrated into a single information system. Although they are

different companies, employees can conduct operations through either of them, but they can only be part of one of them.

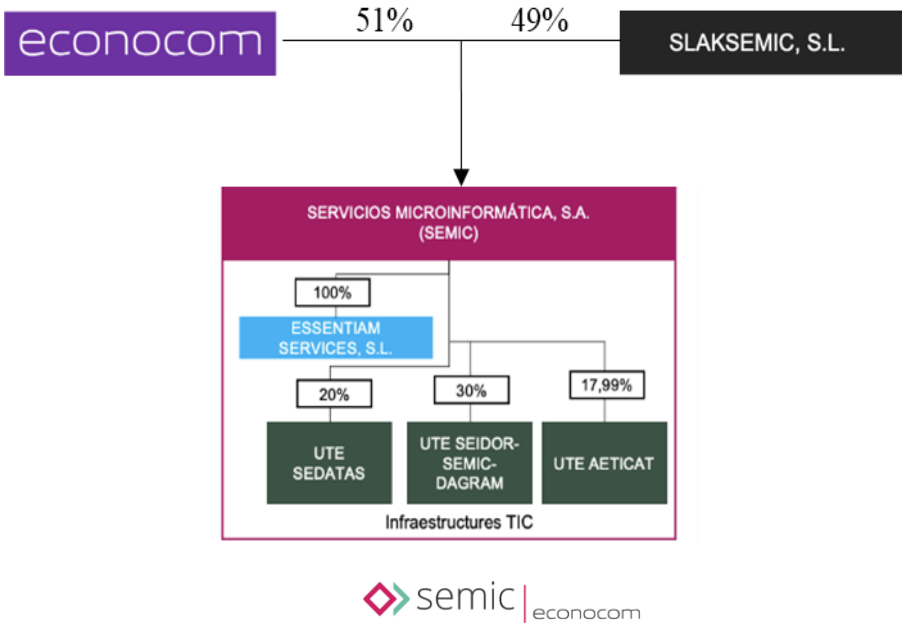


Figure 2: Structure of the new SEMIC Econocom company formed in 2022. [Courtesy of SEMIC, used with permission]

SEMIC is the original company. Before SlakSEMIC was formed, it contained all its services. Currently it contains commercial and prepaid areas and groups.

SEMIC also owns the company Essentiam Services S.L., which is responsible for the manufacture, purchase, rental, wholesale, and retail sale, marketing, import, and export of computer hardware, software, peripherals, accessories, and consumables of all kinds.

Apart from that, SEMIC is also involved in three different Temporary Joint Ventures[1] (TJVs), as can be seen in **(Fig.2)**.

[1] One of the mechanisms most used by companies in Spain. This allows the union of two or more companies for a specific period to provide a particular service or product to take advantage of the experience of both.

2.3. Business Model

SEMIC follows a business model that combines digital innovation, contractual services, and a strong commitment to sustainability. By adopting digital channels and e-commerce, they enhance customer value, while their subscription-based services offer flexibility and convenience. In addition, SEMIC's dedication to sustainability is evident in its reconditioned product offerings and use of sustainable energy sources, exemplifying its commitment to environmental responsibility. *[Courtesy of SEMIC, used with permission]*

2.3.1. Digital (omnichannel)

SEMIC has had to transform and apply new tools due to the changing times. That is why they have implemented a new way of contact and sales to keep customers satisfied. Through e-commerce SEMIC has been able to bring more value to customers than through traditional, well-known sales channels.

2.3.2. Contractual (Services)

SEMIC has implemented the subscription model for the different services it offers. A good example would be the Cybersecurity department, which offers complete monitoring of customer devices and infrastructure in exchange for a monthly subscription. Another example is the JustPrint! department. With mps contracts, customers no longer must pay for printers and copies but are offered the possibility to rent them in exchange for a monthly subscription.

2.3.3. Sustainability (Commitment)

SEMIC is fully committed to the sustainability of the products it offers. That is why not all the products SEMIC offers are no longer firsthand. To be environmentally friendly, products are reused and restocked as reconditioned products at a more economical price. This also applies to printer contracts, where customers can choose a previously used printer and thus extend its service life, instead of always offering them new products. Such is the importance SEMIC attaches to this that the electricity the company consumes comes only from renewable sources.

2.4. Focus

SEMIC specializes in two key areas: Healthcare, Education and Public Administration, as well as Innolab, a team dedicated to driving innovation and optimizing operations within the company. With a formidable team of professionals, SEMIC offers a wide range of services in the first area, while constantly pushing the limits of technology and advances in the second. *[Courtesy of SEMIC, used with permission]*

2.4.1. Health, Education and Public Administration

Thanks to the wonderful team of professionals that SEMIC has, it can provide all kinds of services in these sectors. Specialized in schools, sale of laptops, licenses, printers, infrastructure maintenance, among others. SEMIC also participates in public tenders to expand its business by highlighting its differentiation and reliability as a company.

2.4.2. Innolab (Permanent Innovation)

SEMIC has a team focused on constantly improving and optimizing the company through the implementation of innovative technologies. They work to solve any bottlenecks that may arise in the company's operations, which helps SEMIC to continue growing as a business.

2.5. SEMIC's Registration System

SEMIC's registration system consists of two different systems:

- **Fingerprint reader located at the office doors**: This system consists of putting the fingerprint or a code provided by the Human Resources team, which serves both for the entry, breaks and exit of the staff (**Fig.3)**.

Figure 3: Fingerprint reader located for clocking located at the office doors of SEMIC. Own Source

- **Web page with a simple interface**: Once the employee has been previously identified, the employee will access a new screen (**Fig.4)** where he will be able to mark the time of entry and exit, the time of breaks as well as the total hours worked during the week.

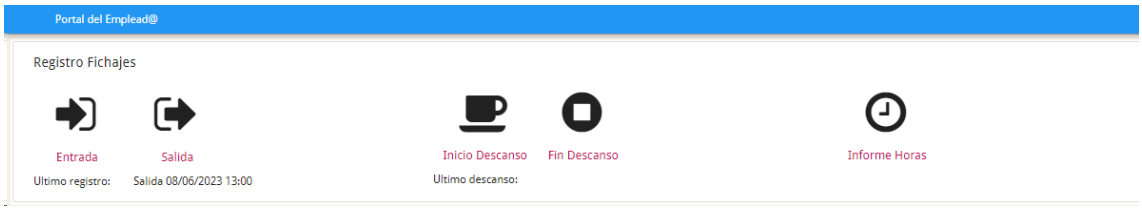


Figure 4: SEMIC's registration system interface opened from the web browser. Own Source.

Simple, isn't it? Why would it be necessary to change it if we only have to enter a page every day, enter our corporate email and password as employees, and press the button when we want to carry out the relevant actions? Why would it be necessary to change it if we have a fingerprint reader right at the door with the option of entering a code in case of an error in the reading? The answer is simple, to optimize the time and comfort of the workers. *(For more information, please refer to section 3, Business Case).*

At the end of this project, we will see if it has really been worthwhile to develop a clocking-in-in system with these new technologies, who knows, maybe in the not-too-distant future this methodology will be the most used and we will normalize it in our day-to-day life.

3. Business Case

As companies strive to optimize their operations and improve efficiency, one area that often requires attention is time tracking and management. Traditional time clocks systems, such as fingerprint readers and web page interfaces, can be time-consuming, error-prone, and lack the flexibility to accommodate remote or flexible work arrangements. However, emerging technologies such as voice-activated clocking-in systems present a new solution that can bring significant advantages to both companies and their employees. In this business case, we will examine the most popular clocking-in systems, the benefits, and costs of implementing a voice-activated clocking-in system, compared to traditional clocking-in systems, and make a recommendation on the most effective solution for our organization.

3.1. Known Clocking-in Systems

Nowadays there are different clocking-in systems that allow employees to clock in and out of the company. The most popular or currently used clocking-in systems have therefore been summarized. [5], [6], [7]

Manual clocking-in systems: These systems require workers to manually record their clock-in and clock-out times on a paper timesheet or punch card. The data is then usually manually entered into a computer system for processing.

Electronic clocking-in systems: These systems use electronic devices, such as swipe cards, key fobs, or biometric scanners, to record when workers arrive and leave work. The data is typically automatically transmitted to a computer system, which can then be used to calculate pay and track attendance.

Mobile clocking-in systems: These systems allow workers to clock in and out using a mobile device, such as a smartphone or tablet. Workers can typically use an app or mobile website to clock in and out, and the data is transmitted to a computer system for processing.

Web-based clocking-in systems: These systems allow workers to clock in and out using a web-based interface. Workers can typically access the interface from any internet-connected device, such as a computer or mobile device.

GPS clocking-in systems: These systems use location data to track when workers arrive and leave each job site. They are commonly used in industries such as transportation and delivery.

Contactless clocking-in systems: These systems use RFID or NFC technology to record when workers arrive and leave work. Workers can simply wave an RFID or NFC-enabled device, such as a card or wristband, near a reader to clock in and out.

Voice-activated clocking-in systems: These systems use voice recognition technology to allow workers to clock in and out by speaking a passphrase or command.

3.2. What Can a Voice Clocking-in System Offer to SEMIC?

SEMIC can benefit from a voice-activated clocking-in system. This system can bring several advantages to the company and its employees. In this section, we will discuss the advantages of using a voice-activated clocking-in system compared to the current clocking-in systems currently available to SEMIC, the well-known fingerprint and web-based clocking-in systems explained previously.

Advantages of the Voice Clocking-in System over the Fingerprint Reader

A voice-activated clocking-in system eliminates the need for employees to touch a shared device such as a fingerprint reader, reducing the risk of virus transmission. The voice-activated system can provide additional security and prevent time theft by verifying employee identity using voice recognition technology, which is difficult to forge, or copy compared to a fingerprint. In addition, it can be used by employees who may have difficulty using the fingerprint reader due to physical disabilities or injuries and reduce the potential errors that these clocking-in systems often have with employee fingerprints.

Advantages of the Voice Clocking-in System over the Web Page Interface

The voice-activated system eliminates the need for employees to navigate to a web page, log in, and manually input their clocking-in data, reducing the time and effort required for time tracking. The voice-activated system can provide real-time data on employee time tracking, enabling managers to view and manage employee time data in real-time. The voice-activated system can integrate with other HR or payroll software, providing a seamless end-to-end solution for time tracking and management.

Overall, implementing a voice-activated clocking-in system can provide several advantages over the existing fingerprint reader and web page clocking-in systems, including contactless operation, convenience, security, accessibility, time savings, accuracy, transparency, and integration. These advantages can benefit both the company and its employees by improving time tracking and management processes, reducing errors and time theft, and improving overall efficiency and productivity.

4. Voice Assistants

Voice assistants use a combination of AI and natural language processing (NLP) to understand and respond to user requests. [2] *(For more information, please refer to section 5, Fundamentals).*

When a user speaks to a voice assistant, the device converts the voice into text through automatic speech recognition (ASR). This text is then analyzed by NLP algorithms designed to understand the meaning and context of the user's words. [2]

Once the voice assistant has understood the user's request, it uses AI algorithms to determine the appropriate response. This can range from a simple answer to a question asked or it can involve performing a more complex task, such as creating an event for you, a reminder, a shopping list, or even playing your favorite songs with a simple question (e.g., assistant[2], play my favorite song). [2]

In addition to ASR and NLP, voice assistants also use machine learning algorithms (MLA) to improve their performance over time. These algorithms allow assistants to learn from user interactions and adjust their responses accordingly. This means that the more you interact with your voice assistant, the better it will understand and respond to your requests.[2]

After having done brief research on the most used voice assistants, the question arose in SEMIC whether we could take advantage of these new technologies in any process of our company, so we came up with the possibility of automating SEMIC's registration system by taking advantage of the benefits offered by a virtual voice assistant.

[2] *Assistant means any of the above, Siri, Alexa, etc.*

4.1. History of Voice Assistants

As we have explained before, voice assistants are software agents that can interpret human speech and respond through synthesized voices. Apple's Siri, Amazon's Alexa, Microsoft's Cortana, and Google Assistant are the most popular voice assistants and are integrated into smartphones or dedicated home speakers.

People have wanted to talk to computers almost from the moment the first computer was invented. Just a few decades ago, the idea of holding meaningful conversation with a computer seemed futuristic, but the technology to make voice interfaces useful and widely available is already here. Several consumer-level products developed in the last few years have brought inexpensive voice assistants into everyday use, and more features and platforms are being added all the time. The number of services that support voice commands is growing rapidly, and Internet-of-Things device manufacturers are also building voice control into their products.

Eliza was the name by which the first device capable of speech and conversation was known, developed in 1964. In 1994, IBM launched Simon, the first smart phone, paving the way for digital assistants as they are known today. [8]

Among the modern models, Apple's Siri assistant has been around the longest, released as a standalone app in 2010 and bundled into iOS in 2011. Microsoft followed shortly thereafter with Cortana in 2013. Amazon launched Alexa with its Echo-connected home speaker in 2014, and Google's Assistant was announced in 2016 along with its home speaker and is also embedded in the Google app for Android based smartphones. Each assistant has its own unique features, but the core functions are the same. Voice assistants differ from earlier voice-activated technologies in that they can respond to a much larger number of commands and questions. This is because they are always connected to the Internet; each interaction is sent back to a central computing system that analyzes the user's voice commands and provides the assistant with the proper response. Earlier voice-activated devices relied on a smaller set of "built-in" commands and responses. Recent advances in natural language processing, also known as computational linguistics, has allowed voice assistants to create meaningful responses quickly.[9]

As personal computers have grown cheaper and more powerful, and people have created more and more online text to be analyzed, scientists have used that text to train voice assistants to listen and respond to our requests in more natural and meaningful ways. Voice assistants can parse requests phrased in several different ways and interpret what the user is most likely to want.[9]

5. Fundamentals

In recent years we have heard a lot about words like Artificial Intelligence (AI), Machine Learning (NL), Deep Learning (DL) or Natural Language Processing (NLP).

We have seen a substantial change in the way we apply these concepts, first with movies in which robots integrated AI, then with music recommendations, or when we use navigation systems; until the latest novelty, the so-called chatGPT that we have heard so much about in recent months.

In this section we will see the basics of these terminologies mentioned earlier in the project, deepening into the different applications of AI that are present in an intelligent voice assistant.

5.1. Artificial Intelligence

The first thing that comes to most people's minds when they hear the word AI is usually robots. This is because popular movies and books often feature human-like robots wreaking havoc on Earth. However, the opposite is true.

AI is founded on the idea that the human intellect can be described in a way that makes it simple for a computer to duplicate it and carry out activities of any complexity. AI aims to emulate cognitive processes in humans, and in fact, is the emulation of human intellect in devices that have been designed to behave and think like humans. At its simplest form, AI is a field, which combines computer science and robust datasets, to enable problem-solving.[10]

5.1.1. Machine Learning

Self-driving cars, assistants that instantly translate from one language to another or personalized shopping suggestions. Complex tasks that were once a pipe dream are now possible thanks to ML, a discipline that allows computers to learn by themselves and perform tasks autonomously without the need for programming.[11]

In his book `On Intelligence´, published in 2004, Jeff Hawkins defined intelligence as "the ability to predict the future, for example, the weight of a glass we will lift or the reaction of others to our actions, based on patterns stored in memory (the memory-prediction framework)".[11]

ML, through algorithms, gives computers the ability to identify patterns in massive data and make predictions (predictive analytics). ML allows computers to perform specific tasks autonomously, without the need for programming. The term was first used in 1959. However, it has gained prominence in recent years due to the increase in computing power and the data boom.[11]

Voice assistants use MLA to learn from user interactions and improve its performance over time. For example, voice assistants can learn the user's preferences, habits, and patterns of behavior to provide more personalized and relevant responses.[11]

5.1.2. Deep Learning

DL is based on a set of algorithms related to ML and whose real-world applications are becoming increasingly tangible (prediction of business results, evolution of virtual assistants, analysis of medical images, etc.). has attracted the attention of companies for its ability to get the most out of AI. You're going to unlock your mobile phone, but it's not one of those fingerprint-based ones. Instead, the camera recognizes your face, identifies you as the owner and unlocks for use. To finish off that process, seemingly simple to your eyes, the phone had to learn to recognize variations in your facial expressions and it did so thanks to a deep learning system. This is just one of the applications of this technology.[12]

DL is based on ML so that, from a large amount of data and after numerous layers of processing with algorithms, a computer ends up learning on its own and performing tasks similar to those of human beings performing human-like tasks, such as image identification, speech recognition, or prediction, in a progressive way.[12]

What does this mean? For example, voice assistants use DL techniques to perform complex tasks such as speech recognition, natural language understanding, and image recognition. DL models allow voice assistants to recognize patterns and features in data and make more accurate predictions.[12]

5.1.3. Knowledge Representation and Reasoning

Knowledge representation and reasoning (KRR) is a way of teaching computers to store and use knowledge, like people do. In the case of voice assistants, it means that voice assistants have a way of organizing information in a way that it can easily find and use it to answer questions.[13]

For example, if you ask to any of the most knowns voice assistants, "What is the capital of France?", it will look for information about France and its capital in a "database" of information that it has stored. This database, called a knowledge graph, has information about different things like movies, music, geography, and more.[13]

Using KRR, voice assistants can find the information it needs to answer your question by searching the knowledge graph. It can also use reasoning and logic to combine different pieces of information to give you a complete answer.[13]

5.1.4. Computer Vision

Computer vision (CV) enables computers and systems to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.[14]

CV works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.[14]

CV trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.[14]

CV is used in industries ranging from energy and utilities to manufacturing and automotive – and the market is continuing to grow. [14]

Voice assistant devices with screens use CV to recognize and analyze visual content, such as images and videos. It allows voice assistants to provide visual responses and interact with users in a more immersive way.

5.1.5. Natural Language Processing

NLP brings together two disciplines as apparently distant as linguistics and AI Today, this field of computer science, which consists of transforming natural language into a formal language — such as programming — that computers can process, is constantly evolving and its applications are growing.

If you have ever asked Alexa or Siri for the time, you will have realized that you do not always have to ask the question in the same way. You can ask "what time is it?" or "can you tell me the time?" and in both cases receive an appropriate response. The same is true of Google's automatic translator, which detects the nuances between different words depending on the context. These examples, and many more, have something called NLP behind them.[15]

So, basically NLP is concerned with giving computers the ability to understand text and spoken words in much the same way human beings can thanks to statistical, ML and DL models.[16]

Voice assistants use NLP to understand the user's spoken or written commands, and to generate appropriate responses. It allows voice assistants to interpret and analyze the user's intent, context, and sentiment to provide relevant and accurate responses.

6. Impact of Voice Technology in the Market

From the personal assistants in our mobile phones, to the profiling, customization, and cyber protection that lie behind more and more of our commercial interactions, AI touches almost every aspect of our lives. And it's only just getting started.

6.1. Worldwide Voice Assistants in Use

In 2020, the size of the Voice Assistant Industry in units was 4.1B and is projected to grow 24% in 2021. From 2019 to 2023 the Voice Assistant Industry growth is projected to average 28% per year (**Fig. 5)**. These estimates were made before the Coronavirus (COVID-19) pandemic. The short-term impact of COVID-19 on the Voice Assistant market growth will likely be Beneficial because an increasing number of consumers, children, and employees stay indoors. The demand for new smart devices increases for the activities at home; but the supply may face difficulties, since factories are operating at a lower capacity. The long-term impact of COVID-19 on the Voice Assistant market growth beyond the COVID-19 pandemic will likely be Medium because much of the supply chain for smart devices are facing supply constraints as factories are operating at much lower capacity resulting in component shortages.[17]

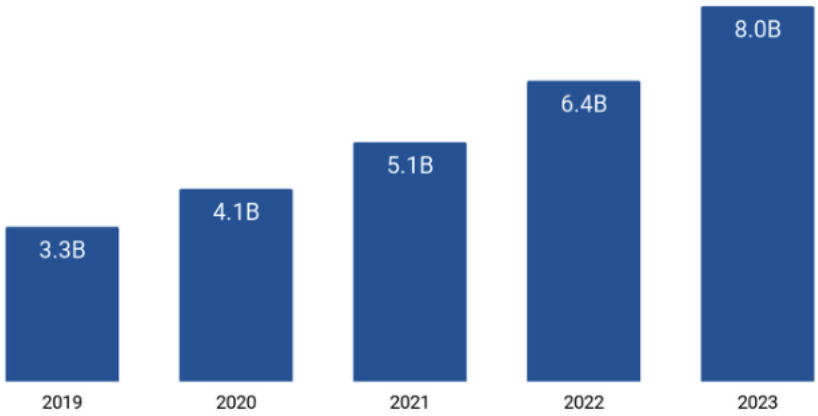


Figure 5: Worldwide Voice Assistants in Use (Units), 2019 – 2023
Source: https://www.t4.ai/industry/voice-assistant-market-share

6.2. Consumers

According to a 2018 PriceWaterhouseCoopers (PWC) survey, more than 90% of consumers aged 18-64 sought information about voice technology, and a 72% used it at least once. A new era for voice assistance has begun due to this massive consumer adoption of voice technology for convenience and home improvement. [18]

According to the National Statistics Institute (INE), in 2022, almost four out of every ten people aged 16 to 74 (39.1%) used home automation devices or services in the home. 23.5% used virtual assistants in the form of a smart speaker or App (such as Alexa, Google Home, Siri, Cortana...), 16.3% used connected appliances and 11.8% used energy management systems (lights, sockets, thermostats, etc.).[19]

By gender, 40.1% of men and 38.1% of women used home automation devices or services. By age, the highest use was in the 35-44 age group, with 47.0%. With regard to the reasons for not using them, 22.5% stated that they were unaware of this type of device and the remaining 77.5% were aware of them but did not use them.[19]

If we compare this with the 2020 data - in the press release of the 2021 study these data do not appear - we can see that the use of home automation devices or services in the home grew by 10.1% and those using virtual assistants in the form of smart speakers or Apps grew by 5.6%.[20]

By usage, Alexa was the most popular assistant in Spain. According to Statista data from 2021 (the last year for which full figures are available), 48.7% of users used Amazon's assistant. It was followed by Google's assistant, with 45.1%, and Apple's assistant, with 31.7%.[21]

6.3. All In The Market

Several consultancies specializing in technology and strategy, such as Gartner and SearchEngineWatch, have already published forecasts about the future of the voice assistant in our lives. [2]

- By 2024, 50% of all Internet searches will be voice-based.
- According to Gartner, in the next few years, 30% of all searches will be conducted using a device without a screen.
- Voice search queries are longer than regular text-based searches and tend to be three to five words in length, Campaignlive explains.
- According to LocationWorld, 240% of adults now use mobile voice search at least once a day.
- 20% of adults use mobile voice search at least once a month according to TheWebIndex.
- 20% of searches on a mobile device are voice-based.
- 25% of queries on Android devices are voice-based.
- As reported by Citrusbits, 55% of teens use voice search daily.
- SearchEngineWatch reports that voice-based searches using a mobile phone are 3 times more likely to be location-specific.

6.3.1. Natural Language Processing

This section is intended to show and demonstrate the different analyses and predictions for the coming years, but also to see where we are now.

To approach it more accurately and correctly, we'll start by looking at Gartner's Hyper Cycle for AI in 2022. This cycle identifies the must-know innovations in the area of technology and AI, beyond the AI we use every day to refine previously static business applications, devices and productivity tools.

(**Fig.6)** If we look at NLP, we see that it is in the Trough of Disillusionment, which means that there are users experimenting with the technology, but there is still a lot of impatience, and this is replacing the initial enthusiasm for the potential value. Performance problems, slower than expected adoption or the inability to generate financial returns in the expected time led to unfulfilled expectations and disillusionment

sets in. However, we can also see from the graph that the Plateau of Productivity will not be achieved until after 5 to 10 years.[22]

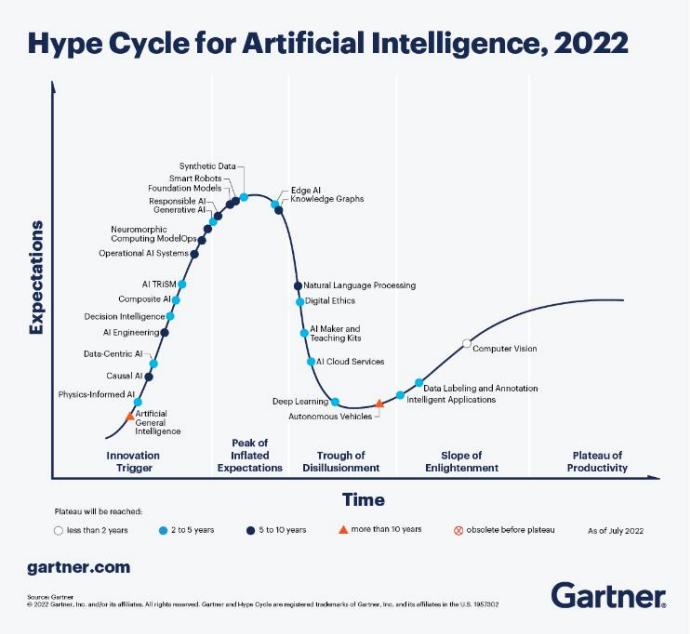


Figure 6: Hype Cycle for AI, 2022. Source: https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2022-gartner-hype-cycle.

The latest research study "Natural Language Processing (NLP) Market: Global Industry Trends, Share, Size, Growth, Opportunity and Forecast 2023-2028" by IMARC Group[3], finds that the global natural language processing (NLP) market size reached US$ 18.1 Billion in 2022. Looking forward, IMARC Group expects the market to reach US$ 74.3 Billion by 2028, exhibiting a growth rate (CAGR) of 26.2% during 2023-2028.[23]

6.3.2. Voice Assistants Market

The Voice Assistant Market Size was valued at USD 3.5 billion in 2021. The voice assistant market industry is projected to grow from USD 4.59 Billion in 2022 to USD 30.72 billion by 2030, exhibiting a compound annual growth rate (CAGR) of 31.2% during the forecast period (2022 - 2030). The development of voice-based AI

[3] The International Market Analysis Research and Consulting Group (IMARC Group) is a leading advisor on management strategy and market research worldwide.

technologies, the rising popularity of voice-enabled devices, and the growing emphasis on customer engagement are all expected to drive growth in the voice assistant market.[24]

According to the "Cisco Annual Internet Report", the number of devices connected to Internet networks will more than triple the world's population by 2023. According to the report, there will be approximately 3.6 networked devices per capita by 2023, up from 2.4 devices per capita in 2018. It also projects that there will be 29.3 billion networked devices by 2023, up from 18.4 billion in 2018. Each year, several new devices in different form factors with enhanced capabilities and intelligence are developed and adopted in the market. As the number of connected devices increases, this will drive the adoption of voice assistant application solutions.[25]

6.3.3. Barriers to Voice Technology Adoption

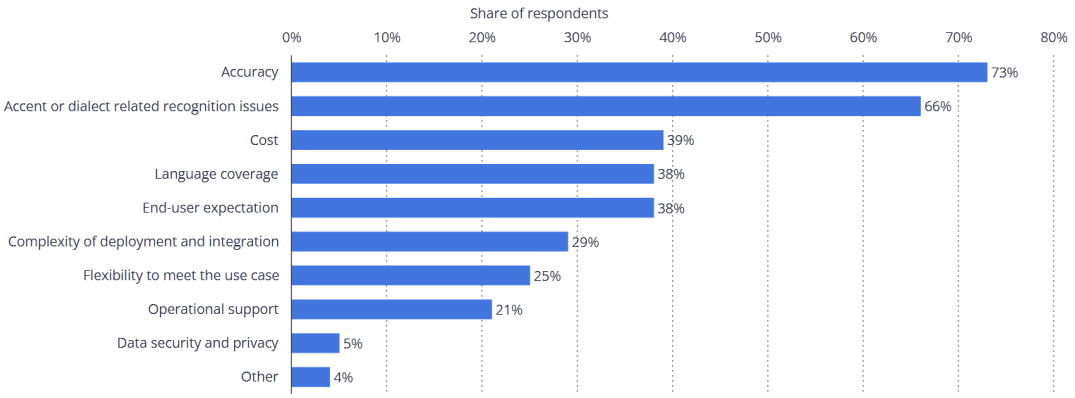


Figure 7: Barriers to voice technology adoption worldwide as of 2020, [Courtesy of SEMIC, used with permission]

(**Fig. 7**) We can see that accuracy and accent, or dialect-related recognition issues are the most significant barriers to voice technology adoption, with 73% and 66% of respondents respectively citing them as challenges. These findings suggest that voice technology providers need to continue investing in improving the accuracy and inclusiveness of their speech recognition algorithms to cater to users with diverse accents and dialects.

Cost, language coverage, and end-user expectations are also key factors that could affect the adoption of voice technology, with 39%, 38%, and 38% of respondents respectively identifying them as barriers. Providers will need to ensure that their voice technology is affordable, supports a wide range of languages, and meets the expectations of end-users in terms of functionality and ease of use.

The complexity of deployment and integration, flexibility to meet the use case, and operational support are also significant barriers to voice technology adoption, albeit to a lesser extent, with 29%, 25%, and 21% of respondents respectively citing them as challenges. Voice technology providers should aim to provide solutions that are easy to deploy and integrate into existing systems, flexible enough to meet diverse use cases, and backed by robust operational support.

Data security and privacy are cited as a relatively minor concern, with only 5% of respondents identifying them as barriers. However, as voice technology becomes more pervasive, it is essential for providers to address these issues proactively to gain and maintain user trust.

Overall, the graphic suggests that voice technology providers should prioritize accuracy, inclusiveness, cost, language coverage, and meeting end-user expectations to overcome the main barriers to adoption. Additionally, providers should aim to provide solutions that are easy to deploy, flexible, and backed by robust operational support.

6.3.4. The Entry of New AI

OpenAI recently gave us access to the most intelligent AI chatbot. The launch of ChatGPT not only sparked the internet, but it also sparked discussion regarding the potential applications and constraints of AI. However, we already employ virtual helpers in a variety of ways in our daily lives. Can the outstanding smartness of ChatGPT push users from virtual assistants? Or could be a future-proof collaboration? [26]

It was hard to stay away from the hype behind virtual assistants 10 years ago. Virtual assistants were everywhere in award-winning and popular movies; smartphones and laptops came readily equipped with Siri or OK Google; and the ability to have a direct,

back-and-forth conversation with a robot added an unprecedented layer of personalization and familiarity with technology. The hype was real, as Amazon reportedly sold 100 million Alexa-equipped devices by 2019, leading to significant expansion and hiring for the development of Technology.[26]

However, Amazon's hardware department, including Alexa, was a major target for job cuts in 2022. Microsoft's Cortana had a small share of the virtual assistant market due to Google and Apple's advantage in pre-installed features. Even successful companies in space are realizing the trend could be on its way out. OpenAI's ChatGPT and Whisper, a voice recognition software, are available for companies to integrate with their own apps. Microsoft also announced the integration of its AI-powered chatbot with its Bing mobile app. [27]

7. Alexa

Alexa is the voice-controlled virtual assistant created by Amazon. It was launched in November 2014. Alexa was inspired by the computer voice and conversation system aboard the Starship Enterprise in science fiction television series and films, starting with `Star Trek: the Original Series and Star Trek: The Next Generation'.[28], [29]

The name Alexa was chosen because the 'x' in her name is a complicated consonant and thus also easy for the virtual assistant to recognize. In June 2015, Amazon announced Alexa Found, a program that would invest in companies that manufacture voice control skills and technologies. In 2016, the Alexa Prize was announced to encourage technology and its creation. In January 2017, the first Alexa conference took place in Nashville, Tennessee, an independent gathering of the global community of Alexa creators and fans. In May 2018, Amazon announced that Alexa will be included in the 35,000 Lennar Corporation homes built this year. In November 2018, Amazon announced that Alexa would be available in Spain through Echo devices (Echo, Echo Dot, Echo Plus) and the Alexa app. [28], [29]

7.1. Functioning

When you ask Alexa a question, what you're doing is communicating with a cloud-based service. Amazon has designed the Alexa Voice Service (AVS) to mimic real conversations, but you're using intuitive voice commands to get this service to perform specific tasks. "Alexa" is simply the "wake word" that alerts the service to start listening to your voice. For most devices, you just have to say the wake word to get a response.[30]

According to Amazon's Developer site, AVS lives in the cloud. Amazon's AVS is an intelligent voice recognition and natural language understanding service. The service can be used to voice-enable any connected device that has a microphone and speaker. That's why you're starting to see Alexa in headphones and other devices. "Alexa is always getting smarter with new capabilities through ML," Amazon's Developer site reads.[30]

While Alexa is the official name for Amazon's voice assistant, you can change this wake word to "Amazon," "Computer," or "Echo." That's a useful feature, especially if your name or your partner's or roommate's name happens to be Alexa or something that sounds similar.[30]

At first, the assistant was only linked to smart speakers created by Amazon, however, within a few months its software development kit (SDK) was opened for other manufacturers and developers to work on. Since then, the assistant has been included in many devices, from wall clocks to microwaves.[30]

Thanks to the company's expansion capacity, Alexa has become one of the reference assistants.

Having explained a brief introduction on what Alexa is, we will go to see what we can do with it, and that is that Alexa's functions depend on two key elements: [31]

- For one part, you find the voice commands that it integrates in series, and with which we can make a wide variety of requests. On the other hand, you find skills, which are add-ons you can install to add even more functionality to Alexa.

  As for voice commands, Alexa can be asked many kinds of questions, like asking her for information about famous people, curiosities, putting music, time... among others. You can also set alarms, set reminders, and start stopwatches. Obviously, Alexa integrates with Amazon to allow you to make voice purchases and inform you about the status of shipments.

- Then we find Alexa Skills, which, as I mentioned earlier, is equivalent to third-party applications that are available for Alexa. With these you can expand the wizard's ability, adding more custom voice commands or different sources to make the already included features more complete. Globally, there are tens of thousands of different skills, and we can even create our skill, which is, in fact, what this project is based on.

7.2. Alexa Skill Kit

The Alexa Skills Kit (ASK) is a software development framework that enables you to create content, called skills. With ASK, you can build custom skills that make it easier to interact with Alexa and get her to do things for you, like playing music, answering questions, controlling your smart home devices, or ordering groceries. ASK provides a simple, user-friendly way for developers to create these skills, without requiring deep knowledge of computer programming. The kit includes templates and code samples to help get you started, and a testing console to help you ensure that your skill works correctly before it's released to the public.[32], [33]

Overall, ASK is a great platform for anyone who wants to create new and exciting voice-based experiences for Alexa users, without needing extensive technical expertise. With its comprehensive documentation and community support, it's a great choice for anyone looking to get started with Alexa development.[32], [33]

You will need an Amazon Developer account to use ASK and create Alexa skills. Once you have created your Amazon Developer account, you can access the ASK and start building skills for Alexa.

7.2.1. Voice Interaction Models

Every Alexa skill has a voice interaction model that defines the words and phrases that users can say to Alexa to make the skill do what they want. Alexa supports two types of interaction models:[34]

Pre-built voice interaction model – Alexa defines the set of utterances for each skill type for you.

Custom voice interaction model – You define the phrases or utterances that users can say to interact with your skill.

7.2.1.1. Pre-built voice interaction model

The pre-built voice interaction model gives you a set of predefined utterances that users say to interact with your skill. For example, to control cloud-connected devices, the user simply says, "Turn on the lights," or "Turn off the television." The skill accepts the *turn on* and *turn off* requests and responds when it has satisfied the request. ASK offers pre-built voice interaction models for different skill types, such as the smart home and music pre-built models. When you choose the pre-built voice interaction model, ASK defines the utterances and requests, called *intents*[4], for you. You just need to code your skill to respond to the predefined intents. To develop a skill with the pre-built voice interaction model, you design:[34]

- The name Alexa is used to identify your skill, if applicable. The user speaks this name, called the *invocation name*[5], when initiating a conversation with your skill.
- The skill logic to fulfill the predefined intents.

7.2.1.2. Custom voice interaction model

The custom voice interaction model gives you the most control over the user experience. You design the set of words and phrases for every action your skill can perform to deliver an entirely custom voice experience. Custom skills give you flexibility and control over the skill design and code. You can integrate voice, visuals, and touch interactions into custom skills.[34]

[4] An intent represents the user's intention or goal when interacting with a skill. It is the action or request that a user wants the skill to perform. For example, in a weather skill, a user might say "Alexa, ask weather skill for today's forecast" or "Alexa, ask weather skill if it will rain tomorrow". In this case, the intent is to get information about the weather, specifically the forecast for today or tomorrow.

[5] An invocation name is the phrase or word that users say to activate a skill. It should be unique, easy to remember, and relevant to the skill's purpose. When a user says the invocation name followed by a request, Alexa knows which skill to activate and how to handle the request. For example, if the invocation name for a weather skill is "My Weather", a user might say "Alexa, ask My Weather for the forecast in New York City". In this case, "My Weather" is the invocation name.

When you design a custom voice interaction, you define:[34]

- The *intents* the skill can handle. Each intent invokes specific skill functionality. Intents can have arguments, called *slots*[6], that collect variable values that your skill needs to fulfil the user's request.
- The utterances users say to invoke the intents. For example, the user might say, "Plan a trip to Hawaii." The mapping of utterances to intents forms the voice interaction model for your skill. There can be a many-to-one mapping of utterances to intents because the model must define every way a user might communicate the same request to your skill.
- The name Alexa is used to identify your skill. The user speaks the invocation name when initiating a conversation with your skill.
- (Optional) The visual elements and touch interactions for Alexa-enabled devices with a screen.
- The skill logic to fulfil your custom intents.

7.2.2. Types Of Skills

There are several types of skills you can create using the ASK. The categories of skills are not mutually exclusive, and many skills could fall into multiple categories depending on their specific features and focus.[35]

- **Custom skills:** These are skills that you create from scratch to provide users with a unique and personalized experience. When you build a custom skill, you define your own interaction model, or voice use interface, rather than relying on one provided by Alexa. They can be designed to perform any type of task, such as providing personalized information, playing games, education, fitness or

[6] Developers can also define slot types, which are variables that represent specific pieces of information that a user might include in their request. Slots provide context to the intent and help the skill understand what information the user wants to provide. For example, in a weather skill, a user might say "What's the temperature in {city}?" In this case, "city" is a slot, and the user's response would provide the value for that slot.

controlling smart home devices. You can also use Alexa Presentation Language (APL) to add visuals and tap interactions for Alexa enabled devices with a screen.

- **Apps and websites skills:** These are skills that let users add a voice interface to access existing content in their Android and iOS apps and websites.
- **Automotive skills:** These are skills that let users control features of their vehicle remotely, such as turning on the engine, lock or unlock the door, defrost the windshield. You can make the interaction more secure with voice codes.
- **Flash briefing skills:** These are skills that provide users with quick updates on news, weather, and other topics. Users can customize their flash briefing to include the sources and topics they care about.
- **Smart home skills:** These are skills that allow users to control their smart home devices using voice commands. With a smart home skill, users can ask Alexa to turn on the lights, adjust the thermostat, or control other compatible devices.
- **Video skills:** These are skills that enable users to access video content on devices with a screen, such as the Amazon Echo Show or Fire TV. With video skills, users can ask Alexa to play a movie or TV show, or control playback using voice commands.
- **Music, Radio, Podcast skills:** These are skills that allow users to access music and other audio content using voice commands. With this skill, users can ask Alexa to play a specific song or artist, create a playlist, or control playback using voice commands. Users can also:
    - Submit their audio catalog metadata to Amazon so that Alexa can match voice requests with their content.
    - Integrate their streaming service to enable access to music, radio, and podcasts from their catalog.
    - Voice interactions enable users to shuffle, loop, start, stop, pause, and search for audio content.
- **Game skills:** These are skills that allow users to play games using voice commands. With a game skill, users can ask Alexa to start a game, provide instructions, and control gameplay using voice commands.

- **Smart Properties skills:** These allow users to develop property skills to reflect your brand, such as Alexa Smart Properties for healthcare, hospitality, residential, and senior living properties.

- **List skills:** These allow users to develop skills that read and update a user's Alexa lists, such as the Alexa Shopping list and the Alexa To-Do list or create custom lists.

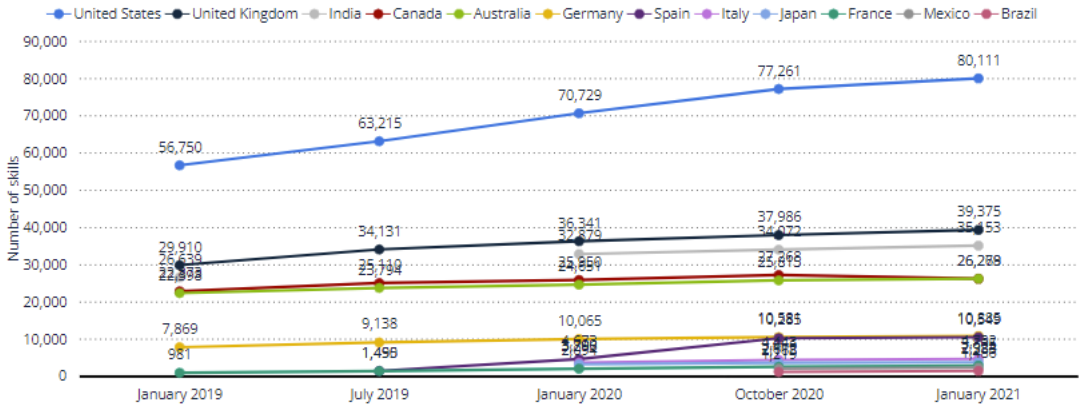


Figure 8: Amazon Alexa total skills 2019 – 2021, by country [Courtesy of SEMIC, used with permission]

As we can see in (**Fig.8),** United States and United Kingdom have the highest number of Alexa skills, with over 80,000 and 39,000 skills respectively as of January 2021. India, Canada, and Australia also have a considerable number of skills, with over 35,000, 26,000, and 10,000 skills respectively as of January 2021. Other countries such as Spain, Italy, Japan, France, Mexico, and Brazil have a smaller number of skills, but are also experiencing growth in the number of skills available. The data suggests that Alexa skills are becoming increasingly popular worldwide, with more developers creating new skills for the platform.

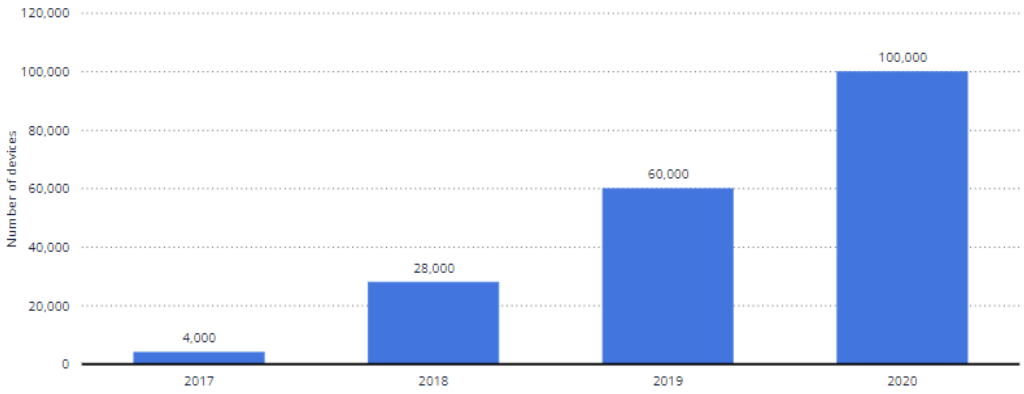


Figure 9: Amazon Alexa, number of supported smart home devices worldwide 2020. [Courtesy of SEMIC, used with permission]

In addition to the increasing number of Alexa skills, it is also notable that the number of supported smart home devices for Amazon Alexa has been steadily increasing over the years. (**Fig.9)** As of 2017, Alexa supported 4,000 smart home devices, and this number grew to 28,000 in 2018, 60,000 in 2019, and 100,000 in 2020. This indicates that not only are more developers creating Alexa skills, but also that more smart home device manufacturers are integrating their products with the Alexa platform. Overall, this trend suggests that Alexa is becoming an increasingly popular platform for both smart home device control and voice-based interactions.

7.2.4. How Does a User Access Skill Content?

A user accesses content in a skill by asking Alexa to *invoke*[7] the skill. Alexa is always ready to invoke new skills. When a user says the wake word, "Alexa," and speaks to an Alexa-enabled device, the device streams the speech to the *Alexa service*[8] in the cloud. Alexa recognizes the speech, determines what the user wants, and then sends a request to invoke the skill that can fulfil the request. The Alexa service handles the speech recognition and natural language processing. Your skill runs as a service on a cloud

[7] The act of beginning an interaction with a particular Alexa ability. For example, if a customer wants to wake Alexa to use the Horoscope skill, Alexa, ask Horoscope for today's reading. Alexa then follows up after the invocation and asks, what horoscope sign would you like?

[8] The cloud-based voice service that powers Alexa-enabled devices made by Amazon or other manufacturers. You can give Alexa new abilities by creating your own cloud-based service that accepts requests from Alexa and returns responses.

platform. Alexa communicates with your skill by using a request-response mechanism over the *HTTPS* interface. When a user invokes an Alexa skill, your skill receives a *POST request*[9] containing a *JSON body*[10]. The request body contains the parameters necessary for your skill to understand the request, perform its logic, and then generate a response.[36] *(For more information, please refer to section 10.1, Purpose, and functionality of the skill).*

The following diagram (**Fig.10**) shows the voice-activated processing flow to invoke a skill with the Alexa service.

Figure 10: How custom skill works Source: https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html#how-does-a-user-access-skill-content.

In addition to voice interaction, skills might include complementary visuals and touch interactions.

7.2.5. Skill Development Workflow

After knowing what type of skill, we want to develop and how a user can interact with the skill, we must follow and familiarize ourselves with the Alexa skill development terminology.[37]

[9] A POST request is a type of HTTP request that is used to submit data to a server to create or update a resource.

[10] A JSON body is a type of request or response body that is formatted in JSON

To develop any type of Alexa skill, we need an Amazon developer account, using an existing Amazon account to sign in or creating a new Amazon developer account. We should follow the Alexa skill development workflow (**Fig.11**) to create our skill.[38]

Figure 11: Alexa skill development workflow Source: https://developer.amazon.com/es-ES/docs/alexa/ask-overviews/alexa-skills-kit-glossary.html

- **Design a skill**

This step involves designing the user experience of your skill, including the *conversation Flow*[11], user interface, and intents and slots. You will need to identify the problem your skill will solve or the functionality it will provide, and then define the user experience in terms of how users will interact with your skill.

- **Build a skill**

In this step, you will develop the backend logic and any supporting *APIs*[12] needed to handle user requests. This typically involves writing code to handle the different intents and slots defined in the design phase, as well as connecting to any external services or data sources needed by the skill.

[11] The conversation flow is the sequence of interactions between the user and an Alexa skill. It refers to how the user and the skill communicate with each other during a session. In other words, it's the way in which the skill prompts the user for information or input, and how the user responds to those prompts. Designing an effective conversation flow is a critical part of creating an engaging and useful Alexa skill. A well-designed conversation flow should be intuitive and easy to follow, while also allowing for flexibility in the user's responses. This involves considering all the possible ways a user might interact with the skill and planning out the corresponding responses that the skill should provide.

[12] An API, or Application Programming Interface, is a set of protocols, routines, and tools for building software applications. It specifies how software components should interact and communicate with each other, allowing different systems to work together seamlessly. In simpler terms, an API acts as a bridge between different software applications, enabling them to share data and functionality. It defines a set of rules and protocols that developers can use to access a specific service or resource provided by another application, without having to know the underlying technical details.

- **Test a skill**

Once you have built your skill, you will need to test it to ensure it works as expected. This can involve using Amazon's developer console, as well as physical devices or testing tools, to test the functionality of your skill and ensure it meets your expectations.

- **Certify & publish a skill**

Before your skill can be made available to the public, it must be certified by Amazon to ensure it meets certain quality and security standards. Once your skill has been certified, you can publish it to the Alexa Skills Store, where users can discover and use it.

- **Monitor a skill**

After your skill has been published, you will need to monitor it to ensure it continues to work as expected and to identify any issues or areas for improvement. This can involve monitoring usage metrics and user feedback, as well as making updates or revisions to your skill as needed.

Overall, these steps are critical to the success of your Alexa skill and may require revisions and iterations throughout the development process to ensure a high-quality, user-friendly skill.[37]

7.3. AWS Lambda

AWS Lambda is an Amazon cloud service that allows you to run code without having to own and manage servers. This service executes code only when needed, and scales automatically, from a few requests per day to millions per second. You only pay for the computation time consumed. With AWS Lambda, you can run the code for virtually any type of application or service, without the need to manage anything. AWS Lambda runs the code on a computer infrastructure and performs all the necessary resource management, including server and operating system maintenance, capacity provisioning and automatic scaling, monitoring, and logging of the code.

These Lambda functions are used to build new Alexa skills and are the default way when creating a skill to host all the skill code. It is the simplest way to build a skill, hosting it in the Amazon cloud, as the developer is responsible for coding, but not for managing the execution of the skill. The AWS Lambda environment takes care of running it in response to Alexa's voice interactions and manages the computer resources. [39]

7.3.1. Skills Hosting with AWS Lambda

Using AWS Lambda to host the skill removes some of the complexity surrounding the development of such a service. Some of the advantages it provides are as follows:

- When using AWS Lambda to host an Alexa skill, the developer does not need to worry about the infrastructure or maintenance of the server environment. The developer only needs to focus on the code that implements the skill's functionality.
- No need for an SSL certificate.
- No need to verify that requests are coming from the Alexa service. Access to run the skill back-end is controlled by permissions within AWS.
- AWS Lambda executes code only when needed and handles resource scaling, so there is no need to continuously provision or run servers.
- AWS Lambda functions can be easily tested and deployed using tools like the AWS Command Line Interface (CLI) or the AWS Management Console, which streamlines the development and deployment process.
- Alexa encrypts communications with AWS Lambda using TLS.
- For most developers, the free resources provided by Lambda are sufficient to host an Alexa skill. One million free requests per month are available. This plan is available indefinitely and does not expire.

AWS Lambda also supports several programming languages, including Node.js, which is what will be used in this project, Python and Java. [40], [41]

8. State of Art

In the previous section, we have seen the technology that Alexa offers us to create skills. In this section we will see different tools that other companies offer us to create them.

But, before making the comparison, (**Fig.12**) shows which voice assistant has had the greatest impact on the global market. The score in this graph was scored between -250 and 250, provided by industry professionals, based on aspects such as breadth of use and degree of adoption, among others.

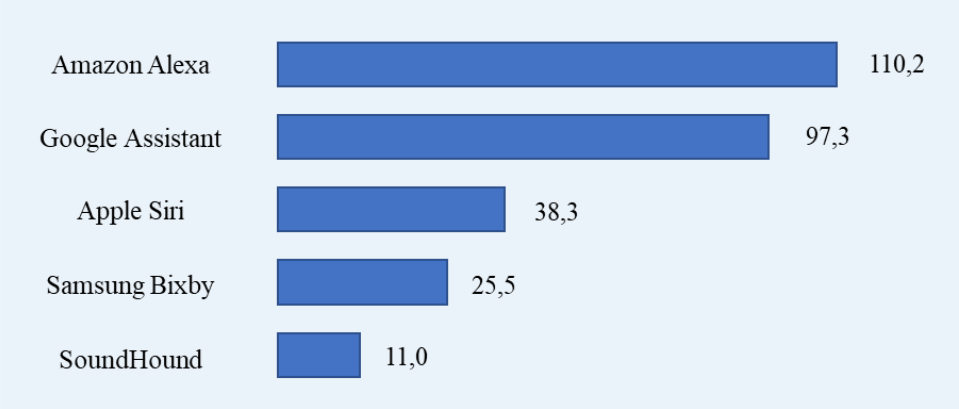


Figure 12: Voice assistants with the greatest impact on the market. ranking of voice assistants according to the score obtained in the "Voice Platforms Impact Rating, 2020". Source : https://es.statista.com/grafico/22578/clasificacion-de-los-asistentes-de-voz/

In Spain, according to *Statista*, Alexa topped the ranking of the leading assistants in the national market, with 48.7% of users, closely followed by Google's assistant with a percentage of 45.1%. The podium was completed by Siri, developed by Apple, with 31.7%. [42]

8.1. Siri

SiriKit is the platform that Apple offers for creating skills, which was first introduced in June 2016, as part of the iOS 10 software release.[38], [43], [44]

Advantages:

- **Tight integration with iOS:** SiriKit is intimately integrated with the iOS operating system, which enables it to provide iOS apps a fluid and natural voice-based user experience.
- **Advanced NLP:** SiriKit is known for its advanced natural language processing capabilities, which allow it to understand and respond correctly to complex and conversational voice commands.

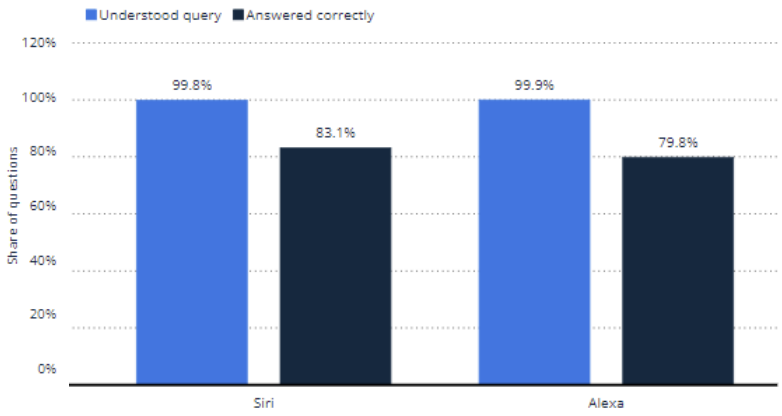


Figure 13: Share of questions answered and understood correctly by selected digital assistants as of 2019. [Courtesy of SEMIC, used with permission]

(**Fig.13**): As we can see, Alexa has a higher accuracy rate in understanding user queries with an accuracy rate of 99,9%, while Siri had an accuracy rate of 99,8%. In terms of answering queries correctly, Siri scored 83,1% accuracy compared to Alexa's 79.8%. This suggests that in general Siri has a more advanced NLP.

- **Privacy and security:** Apple is renowned for placing a high priority on privacy and security, and SiriKit was created with privacy in mind. For example, SiriKit only processes voice data on the device, and does not store or transmit voice data to Apple's servers.

Disadvantages:

- **Limited developer community:** Compared to Amazon Alexa Skills Kit, SiriKit has a smaller developer community, which may result in less developer resources and support.
- **Limited devices support:** Only Apple devices running iOS 10 or later, macOS Sierra or later, watchOS 3 or later, and tvOS 10 or later are compatible with SiriKit. This limits the reach of SiriKit-enabled apps to users of Apple devices only.

8.2. Cortana

In 2017, Microsoft introduced the Cortana Skills Kit as a platform for developers to create skills for Cortana. Cortana Skills Kit allowed developers to create skills that could integrate with Cortana's virtual assistant and provide a more personalized experience for users.[45]

Cortana Skill Kits have continued to evolve over time, adding new features and development tools to help developers create high-quality skills. While Cortana has not gained the same popularity as other virtual assistants, Cortana's skill development platform remains a viable option for those who want to create applications and services that interact with users through voice commands.[45] [46], [47]

Advantages of Cortana Skill Kits:

- **Native integration:** Cortana is integrated into the Windows operating system, meaning users don't have to download a separate app to use it.
- **Flexibility:** The Cortana Skills Kit provides better support for apps that are integrated with Microsoft products, such as Office 365.
- **More data available:** Cortana can provide more personalized assistance since it can access more user data than Alexa.

Disadvantages of Cortana Skill Kits:

- **Limited availability:** Cortana Skill Kits is only available on Windows devices, which limits its reach.

- **Limited devices and platforms:** The Cortana Skills Kit is focused primarily on Microsoft products and services, which can be a limitation for developers who want to target a broader range of devices and platforms.

8.3. Google Assistant

Google Assistant was first announced at Google's developer conference, Google I/O, in May 2016. It was initially released as part of Google's chat app, Google Allo, and later expanded to other Google products such as Google Home and Android phones.[48], [49] [50], [51]

Advantages:

- **Large user base:** Google Assistant is available on a wide range of devices, including Google Home smart speakers, Android smartphones and tablets, and third-party devices. This means that Google Assistant-enabled apps and services can potentially reach a large audience of users.
- **Advanced NLP:** Google Assistant is known for its advanced natural language processing capabilities, which allow it to understand and respond to complex and conversational voice commands.

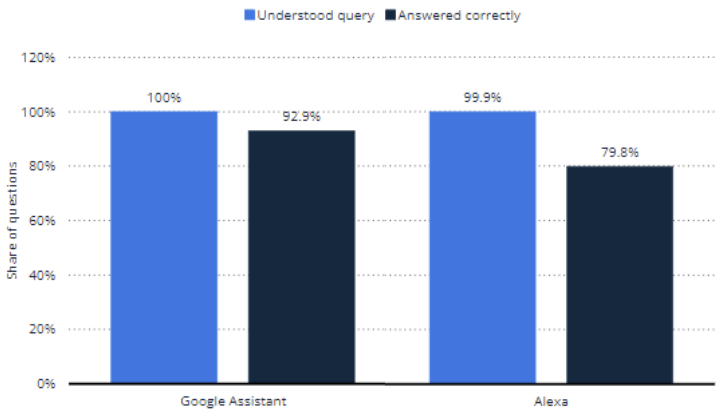


Figure 14: Share of questions answered and understood correctly by selected digital assistants as of 2019. [Courtesy of SEMIC, used with permission]

(**Fig.14**): As we can see, Google Assistant has a higher accuracy rate in understanding user queries with a perfect accuracy rate of 100%, while Alexa had an accuracy rate of 99,9%. In terms of answering queries correctly, Google Assistant scored 92,9% accuracy compared to Alexa's 79.8%. This suggests that in general Google Assistant has a more advanced NLP.

- **Integration with Google services:** Google Assistant is tightly integrated with a wide range of Google services, such as Google Maps, Google Calendar, and Google Search, which can provide additional functionality and data for Google Assistant-enabled apps.

Disadvantages:

- **Limited developer community:** The developer community for Google Assistant SDK is not as large as that of Amazon Alexa Skills Kit, which means that there may be fewer resources and support available for developers.

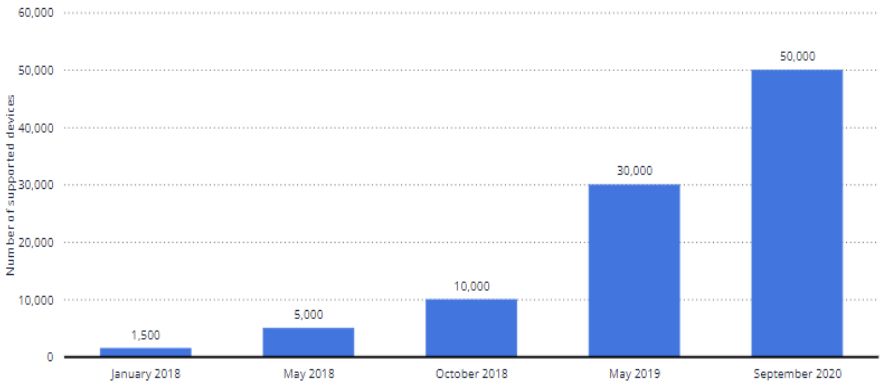


Figure 15: Google Assistant, number of supported smart home devices worldwide 2017 - 2020. [Courtesy of SEMIC, used with permission]

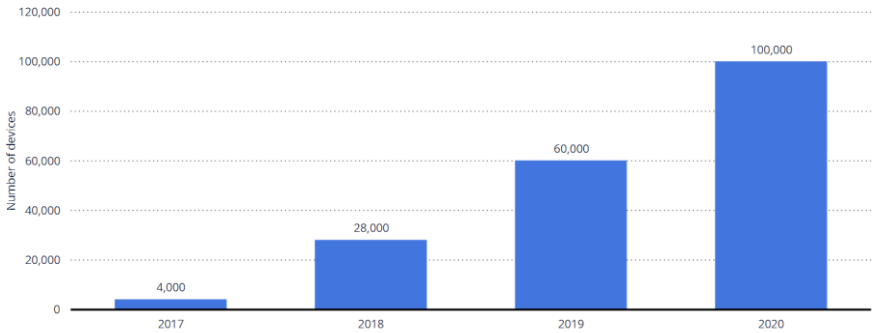


Figure 16:Alexa, number of supported smart home devices worldwide 2017-2020. [Courtesy of SEMIC, used with permission]

(**Fig.15 and Fig.16):** Amazon Alexa has a larger number of supported smart home devices than Google Assistant. In 2017, Alexa supported 4000 smart home devices while Google Assistant only supported 1500. By 2020, Alexa supported 100000 smart home devices while Google Assistant supported 50000. While Google Assistant's support for smart home devices has grown significantly over the years, it still lags the number of devices supported by Alexa. This may be due, in part, to the limited developer community for Google Assistant, which could result in fewer resources and support available for developers compared to the Alexa Skills Kit.

- **Limited Ecosystem:** Google Assistant is not as tightly integrated with the Amazon ecosystem as Alexa, which can limit its functionality on Amazon devices and services.

9. The Project

Having seen and understood the different sections above, it is time to develop the skill of the project and analyze the different challenges that have arisen while I was conducting the different phases of the project.

After several talks with my company mentor, we decided to divide the development of the skill into distinct phases (**Fig.17**).

In the **first phase**, we will define the purpose and functionality of the skill we want to implement, which must allow to clock in SEMIC workers. We will go into detail about what information needs to be collected from workers to customize the experience for each user, how they can invoke the skill, and how they can interact with it. Thanks to this information we will understand how an Alexa developer transfers all this information to be able to create the skill itself.

We will transfer all this information and begin developing the clocking-in skill based on the insights gained earlier.

In the **second phase**, we will create an API with the functionality of allowing employees to clock in and out, considering all the necessary scenarios for storing this record in a database and retrieving the information when needed.

In the **third phase**, we will integrate this API into the Alexa skill. This integration will enable us to access the API's logic and perform clock-ins and clock-outs using voice commands within the company.

Figure 17: Phases of the project. Own Source

10. First Phase. Analysis Of the Alexa Skill

This first phase will cover the purpose and functionality of the skill, the design and development process, including skill creation in the Alexa Developer Console (ADC), building the interaction model, writing code, and testing the skill.

10.1. Purpose and Functionality of The Skill

First, to be able to build our skill that allows SEMIC's employees to clock in using their voice, we have to determine the value of this skill. To do that, we must understand and know the needs of our client, in our case, the SEMIC employees themselves.

We don't just have to convert a clocking-in application into a voice experience. We also must think about the problem we are trying to solve as a way to improve our customers' experience. Does it provide our customer with a more efficient service than the one they currently have? Is it simpler?

To solve these questions, we need to keep in mind these four steps when designing a skill, which we will answer one by one:

1. **What is the purpose of our skill?**

The main purpose of our skill is to allow SEMIC workers to clock in using their voice, in a simple and efficient way.

2. **How can employees invoke the skill?**

Testing phase

In the testing phase, which is when the skill is not yet published, these steps must be followed for a user to be able to invoke the skill:

SEMIC workers will need to verbally say a specific phrase for the skill to recognize the command and proceed with the check-in.

This is done through the invocation name, which in our case will *be "Alexa, abre fichaje SEMIC."* When Alexa recognizes this voice command, it will activate and ask the user

what they would like to do. It is at this point that the user, using the utterances defined by the skill developer, will respond and the clocking-in process will begin *(see section Invocation Name and Interaction Model for more information)*.

Another correct option is to combine the invocation name with the utterances defined by the developers to call a specific utility of our skill. This allows for timesaving by making a single call instead of two, as in the first case.

The following utterances have been defined:

- **Input Utterances:**

These intents have been added so that the user directly invokes the functionality of the skill.

"Alexa, abre fichaje SEMIC **para fichar"**

"Alexa, abre fichaje SEMIC **para entrar"**

"Alexa, abre fichaje SEMIC **para entrar ahora"**

"Alexa, abre fichaje SEMIC **para entrar al trabajo"**

"Alexa, abre fichaje SEMIC **para entrar a trabajar"**

These intents have been added so that the user first invokes the skill with the invocation name and then says the utterance. They have also been implemented in this way so that by the time the skill is published it will not be necessary to say the invocation name.

"entrar"

"fichar"

- **Output Utterances:**

These intents have been added so that the user directly invokes the functionality of the skill.

"Alexa, abre fichaje SEMIC **para desfichar"**

"Alexa, abre fichaje SEMIC **para salir**"

"Alexa, abre fichaje SEMIC **para salir del trabajo**"

"Alexa, abre fichaje SEMIC **para salir de trabajar**"

"Alexa, abre fichaje SEMIC **para salir ahora**"

These intents have been added so that the user first invokes the skill with the invocation name and then says the utterance. They have also been implemented in this way so that by the time the skill is published it will not be necessary to say the invocation name.

"marcharme"

"irme"

"desfichar"

"salir de trabajar"

"salir del trabajo"

The utterance "exit" has not been added as this is restricted to when a user wants to exit the skill, which would cause the user to not accept our intent to clock out and exit the skill directly.

Skill published

When the clocking-in SEMIC skill has been published, it will no longer be necessary to use the invocation name. The Alexa Developer Console provides us with Intent Launch Phrases, where we can assign specific launch phrases to allow users to interact directly with specific intents in a custom skill without needing to indicate the skill name. Alexa allows us to add a maximum of five phrases for each intent. [52]

It is for this reason that in the test phase the utterances have been defined without the use of the invocation name. In this way they will be used again for this section once the skill is published.

3. **What can workers do with the skill?**

Workers can use the skill to clock in and out of work by using their voice.

4. **What information do we need to collect to personalize the experience for each worker?**

To personalize the experience for each worker, we will need to train their voice. This can be achieved through a feature called Alexa Voice Training. The training process will take place within the Alexa application developed by Alexa. Upon completion, an Alexa Voice Profile will be created, enabling Alexa to recognize the user's voice and identify their name, among other things. It's worth noting that there is currently no limit to the number of Alexa Voice Profiles that can be created on an account.[53]

10.2. Design and Development

10.2.1. Creation of the Skill

To create a skill, we first need to have a developer account. Once we have that account created, we can start using their tool, called ADC. [54]

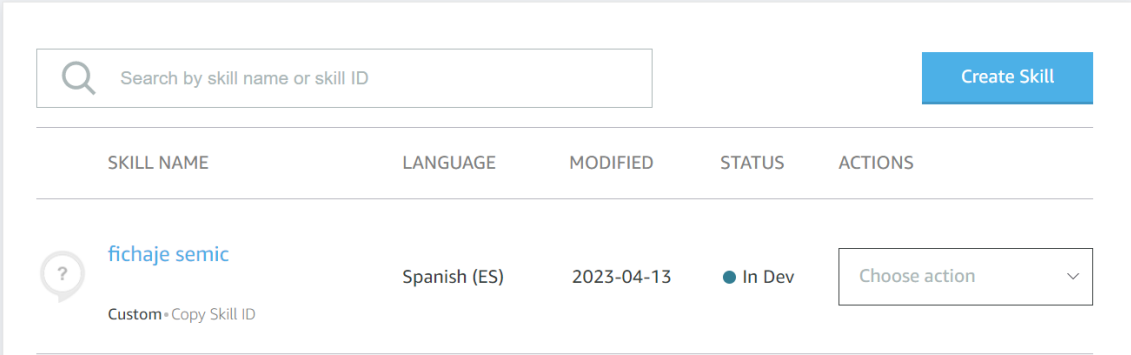


Figure 18: Alexa Developer Console. Own source

The skill called "fichaje semic", which can be seen in (**Fig.18**), is the one that has been created for this project, and the one that will be analyzed and developed during this and the next chapter.

When creating the skill, we will encounter different sections that allow us to choose and customize various parameters to create a skill that suits our needs.

1. **Name, Locale:** In this section, we need to choose the name of the skill, which can be the same name we want to invoke the skill with *(for more information, please*

refer to section 10.2.2.1.1., Invocation). We also need to select the primary locale, which in our case is Spanish (Spain).

2. **Experience, Model, Hosting service:** In this section, we will choose the experience we want to incorporate into our skill. In some cases, Alexa provides predefined phrases that users can say based on the type of skill we want to create. Since there is no predefined experience that suits our skill's purpose, we have selected "Other". For the model, we have opted for "custom" for the same reason mentioned earlier. Finally, Alexa asks us how we want to host the service for our skill. We have chosen "Alexa-hosted (Node.js)" (**Fig.19**) for the convenience provided by Alexa, as they offer their own skill hosting and developer console for development. It's important to note that there are limits on calls and storage when choosing this option.

   Another option would be to choose "provision your own," where we would have to provision our own endpoint and backend. For future considerations, we will study the use of this option, since SEMIC has its own storage server where we could provision our backend. However, for now, we have decided to use the first option for ease and speed.

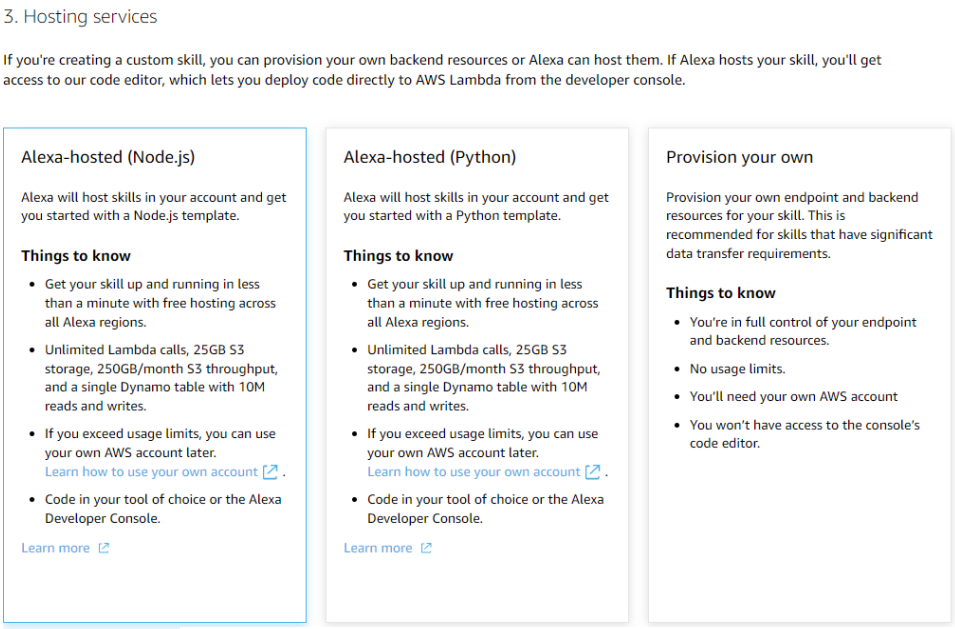


Figure 19: Alexa custom skill hosting services. Own Source

3. **Templates:** In this section, we are introduced to templates shared by the Alexa community or the public GitHub repository. In our case, we have decided to start from scratch as there was no skill that could assist us.

4. **Review:** Finally, we review our previous selections collected in this section, and if everything looks good, we click the button to create our skill.

Once the skill is created, having chosen the hosting option provided by Amazon, we will have three scenarios to work with. The first one is Build, followed by Code, and finally, Test.

10.2.2. Alexa Developer Console

10.2.2.1. Build

This part of ADC (**Fig.20**) is where we develop the frontend of our skill, that is, the part of the skill closest to the user, where the main aspects of the skill that will directly affect the user are defined and configured.

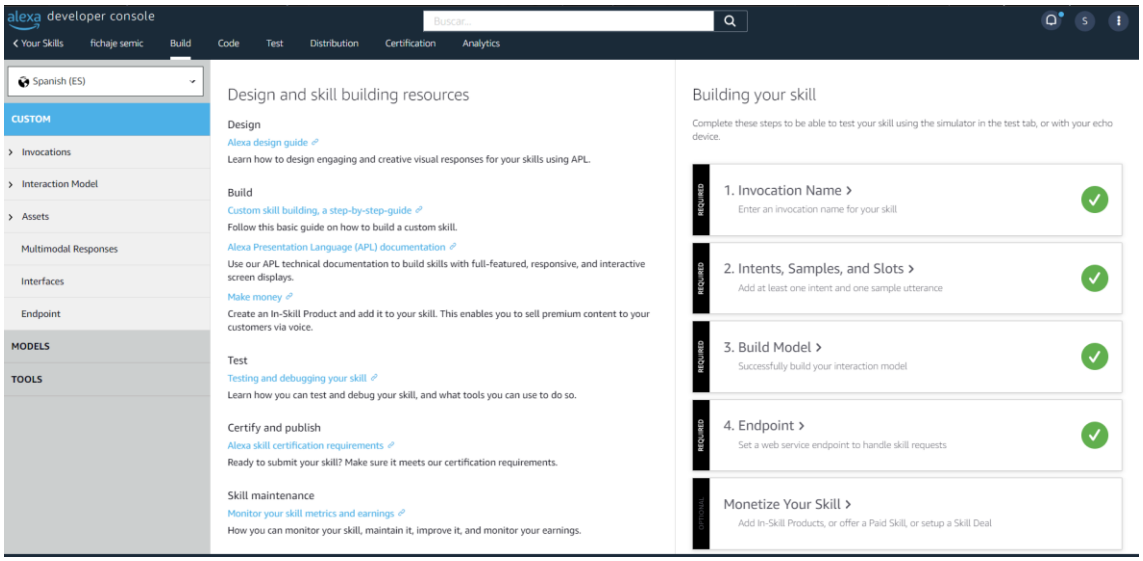


Figure 20: Alexa custom skill hosting services. Own Source

10.2.2.1.1. Invocation Name

When initiating a skill, the user must speak certain words called Invocation Name for the Alexa assistant to recognize which type of skill to activate and start the interaction to gather the subsequent information.

This Invocation Name should consist of two or more lowercase words with spaces between them. In the case of the skill developed in this work, named *"fichaje SEMIC"*, the Invocation Name is also *"fichaje SEMIC"*. To start interacting with the skill, the user would say, *"Alexa, open fichaje SEMIC"*. It is possible to replace the launch phrase "open" with other alternatives such as *"lanza," "activa," "comienza,"* or *"carga"*.

It's important to note that the Invocation Name does not necessarily have to match the skill's name. Additionally, there are certain words that are prohibited when choosing this opening phrase for a skill. Some of these words are the ones used to launch skills, such as *"abre," "lanza," or "activa,"* as well as wake words like *"Amazon," "Alexa,"* and *"Echo"*. These words are prohibited because pronouncing them alerts the service to start listening. Other prohibited words include *"ordenador", "skill"* or *"app"*. [55]

10.2.2.1.2. Interaction Model

The interaction model is the way in which the worker interacts with the assistant, and through this model, the dialogues that the assistant can receive from the worker are defined. To initiate a skill, it is always necessary to start the phrase with a wake word, which alerts the system to listen to the worker. By default, this word is "Alexa."

Following the wake word, there will be a launch word, which can be terms like *"abre", "pide a", "empieza", "comienza", "inicia"* or *"lanza"* among many others. After that, the invocation name can be provided (**Fig.21**).

Figure 21: Statement to invoke the skill. Own Source

To request the skill to perform a specific functionality it is capable of, it is necessary to follow the previously mentioned elements of an utterance. By pronouncing an utterance, the user informs the assistant about what they want to do. Once the Automatic Speech Recognition (ASR) translates the user's audio into text, Alexa searches for the corresponding intent in the voice model that matches that utterance.

An intent represents the user's intention, and once the system identifies the intent corresponding to the uttered utterance, Alexa can perform the appropriate actions to fulfill the user's need. Each intent can have as many associated utterances as desired by the skill developer, allowing the user to request the same action in many different ways, as shown in **Fig.22**.

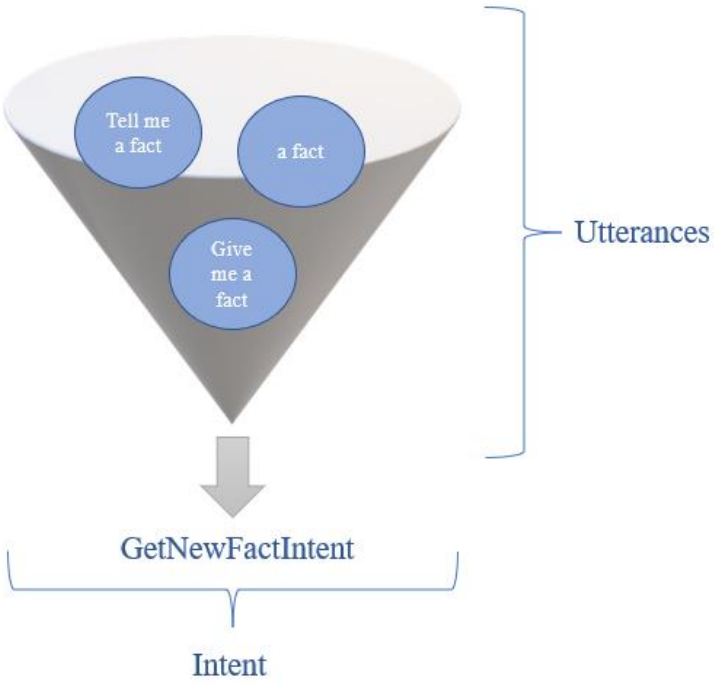


Figure 22: Mapping of utterances in an intent. Own Source

By default, a newly created skill already has a set of intents called built-in intents, which are as follows:

- **AMAZON.CancelIntent:** Responds to words and phrases indicating that the user wants to cancel the current interaction. This intent can be used by the application to clear slot values and other attributes before ending the interaction with the user. Some common utterances used to trigger this intent are *"cancela"* or *"olvídalo"*.
- **AMAZON.HelpIntent:** Responds to words and phrases indicating that the user needs help while interacting with the virtual assistant. When invoked, the skill can be configured to provide information about its capabilities. Common utterances for this intent include *"ayuda"*, *"ayúdame"*, or *"necesito ayuda"*.
- **AMAZON.StopIntent:** Responds to words and phrases indicating that the user wants to stop processing the current intent and end the interaction with the skill. Some common utterances for this intent are *"para"*, *"apágate"* or *"salir"*. Unlike AMAZON.CancelIntent, this intent concludes the interaction with the skill.

Following there is a list of the customized intents available in the fichaje SEMIC skill, apart from those mentioned above, which, as explained above, come by default.

- **EnterWorkIntent:** When an utterance is mapped in this intent, the first thing the skill does is an HTTPS request to Alexa's *PersonProfile API* to try to collect the name of the worker who is speaking. If it was possible because he has a voice profile created and permissions granted, and, therefore, Alexa recognizes his voice, the name of this worker will be sent to the SEMIC *clocking-in API* and his entry time will be collected. If it was not possible, a voice message will be sent to human resources to set up his voice profile.

- **LeaveWorkIntent:** When an utterance is mapped in this intent, the first thing the skill does is an HTTPS request to the Alexa *PersonProfile API* to try to collect the name of the worker who is speaking. If it was possible because he has a voice profile created and permissions granted, and therefore Alexa recognizes his voice, the name of this worker will be sent to the SEMIC *clocking-in API* and his departure time will be collected. If it was not possible, a voice message will be sent to human resources to set up his voice profile.

- **UnhandledIntent:** Responds to words and phrases that the user has said and the skill has not been able to process because the developer has not taken them into account in its custom intents.
- **ProfileError:** Responds when there has been a failed attempt to connect to an API that the developer is using due to a server problem or a lack of permissions from the user.

The creation and modification of the voice model can be carried out from the ADC, in the Build tab. There, you will find a tool called JSON Editor, which translates everything that is done graphically into a JSON file. To develop the voice model locally, you can shape this interaction model by modifying that JSON file, in which the invocation name and the different intents with the utterances mapping to that intent is defined.

```
{
    "name": "EnterWorkIntent",
    "slots": [],
    "samples": [
        "para entrar ahora",
        "para fichar",
        "para entrar",
        "entrar",
        "fichar",
        "para entrar al trabajo",
        "para entrar a trabajar"
    ]
},
{
    "name": "LeaveWorkIntent",
    "slots": [],
    "samples": [
        "marcharme",
        "irme",
        "para salir",
        "para salir del trabajo",
        "para desfichar",
        "desfichar",
        "salir de trabajar",
        "para salir ahora"
    ]
}
```

Figure 23: EnterWorkIntent and LeaveWorkIntent utterances. Own Source

In **Fig. 23**, you can see the two entries of our voice model in this JSON file. Each entry consists of three fields, corresponding to the *intent name*, the *slot* definitions for that intent, and the different *utterances* indicating that intention. It can be observed that each utterance, or sample utterance, corresponds to a different way of expressing the same intention. If the user executing the skill uses any of these different formulations, the skill would translate it into the *EnterWork* or *LeaveWork* intent. In our case, slot definitions were not necessary, which is why we see [ ]. The utterances you can observe are defined in section *10.1, Purpose, and functionality of the skill*, and will be used by SEMIC workers to perform the relevant actions in the clocking-in skill.

10.2.2.1.3. Permissions

Alexa skills might require personal information from the customer to provide relevant information in skill responses or to complete transactions. We can configure our skill to request permission from the customer for specific information. As we have configured our skill in this way, then when a user first enables our skill, Alexa asks the user to go to the Alexa app to grant permission to obtain this specific information.[56].

Figure 24: Permissions granted in the configuration of the fichaje semic skill in the Alexa application. Own Source

In our case, it was necessary to enable *full name* and *personalization* options as seen in the **Fig.25** and **26**. In this way, we can collect the names of workers who have created a voice profile and give them a personalized response to their voice requests.

Apart from giving the corresponding permissions in the Alexa application, it has been necessary to *activate the personalization* in the permissions section of the ADC, as can be seen in **Fig.25**. This will make our skill allow customization, which is necessary to collect the names of SEMIC employees.

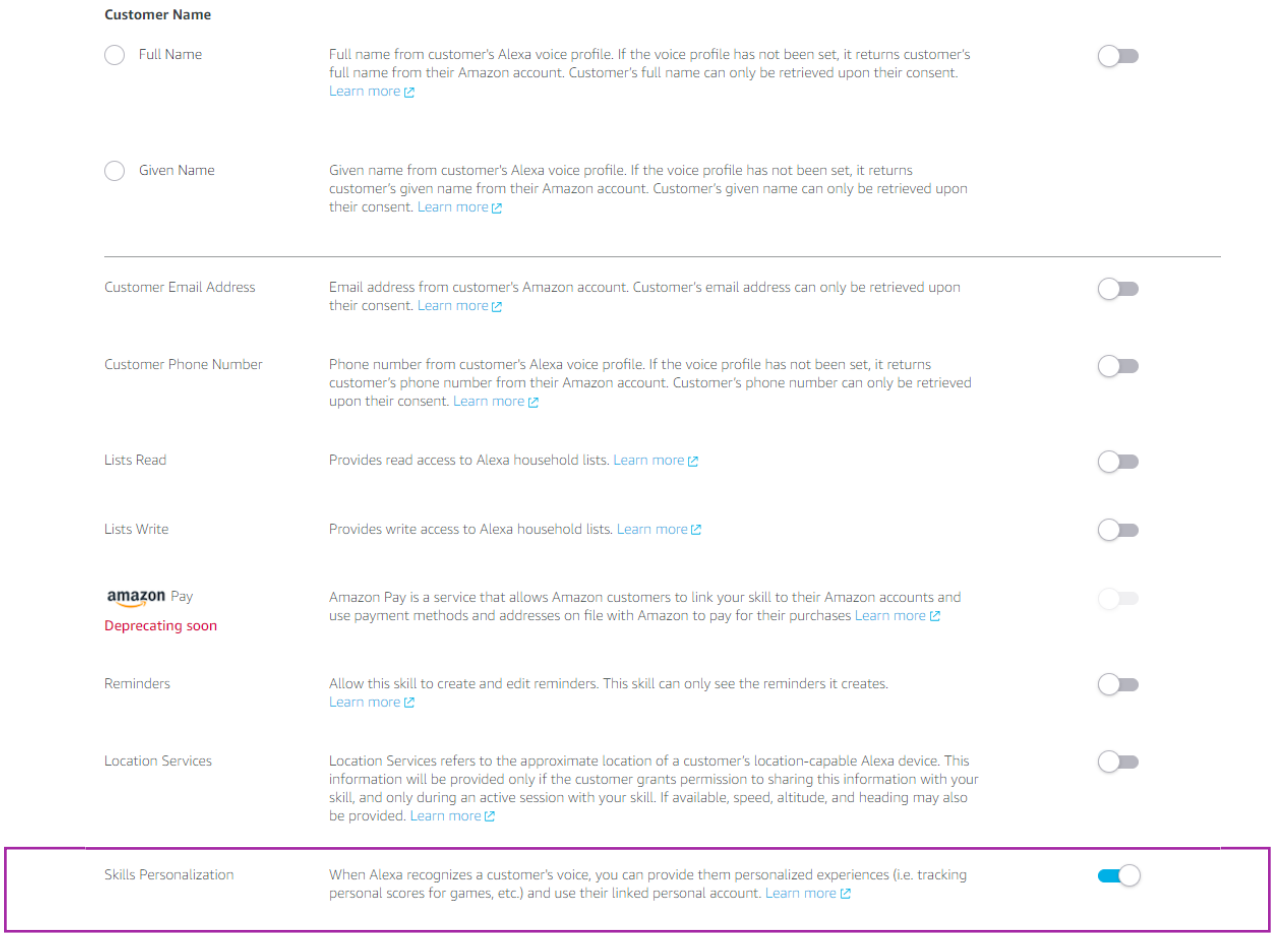


Figure 25: Skill Personalization granted in the permissions configuration of the fichaje semic skill in the Alexa Developer Console. Own Source

Finally, in the test section of our ADC, it has also been necessary for each employee, once they have created a voice profile in the Alexa application and allowed skill personalization in the previous step, to allow that Alexa can accesses to the *Profile Personalization* and the *Full Name* of the profile, as can be seen in **Fig.26**. In this way, we can collect the names of workers who have created a voice profile and give them a personalized response to their voice requests.

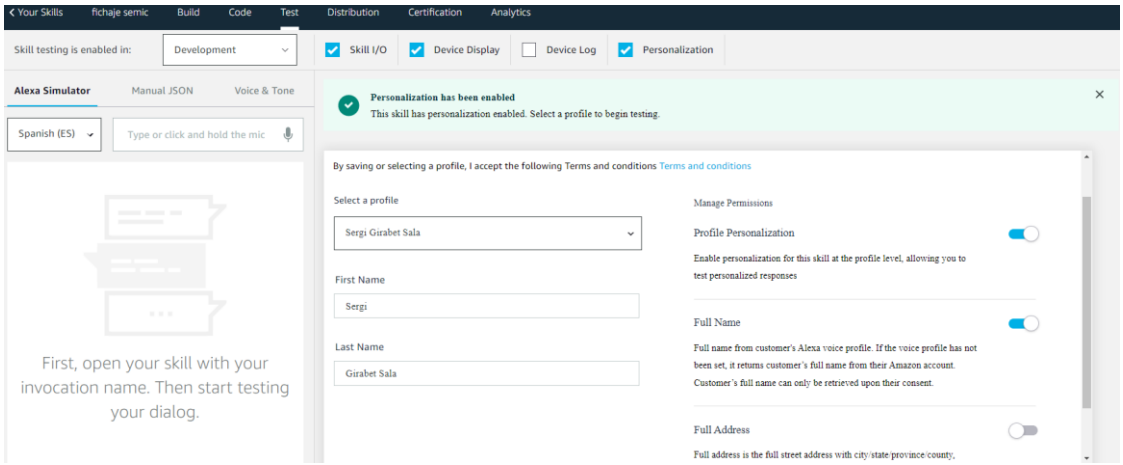


Figure 26: Allowing Profile Personalization and Full Name in the Test configuration of the fichaje semic skill in the Alexa Developer Console. Own Source

Once these different steps have been completed, fichaje SEMIC skill will be able to recollect the necessary information to be able to recognize an employee by his/her voice.

10.2.2.2. Code

In the Code window of the ADC, developers are provided with a code editing tool to create and implement the various functionalities they want the skill to have. This tool allows modifications to the entire backend of the skill, which is hosted in the Amazon cloud, specifically for Alexa-hosted skills like ours. In this type of skill, the backend of the skill is stored using AWS Lambda, a service provided by Amazon Web Services (AWS). *(For more information, please refer to section 7.3, AWS Lambda).*

By default, Alexa provides us with the following files: *package.json, local-debugger.js,* and *index.js.* The first one contains all the necessary information and dependencies that Alexa needs to make the code work, in our case, these are shown in **Fig.27**. The second one includes a debugger to facilitate code debugging, although Alexa warns us that it is deprecated and no longer maintained. The last file contains all the Intents we have defined and the necessary logic to use our skill.


```
{
  "name": "fichaje-semic",
  "version": "1.2.0",
  "description": "fichaje semic skill",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Amazon Alexa",
  "license": "Apache License",
  "dependencies": {
    "ask-sdk-core": "^2.7.0",
    "ask-sdk-model": "^1.19.0",
    "aws-sdk": "^2.326.0",
    "axios": ">=0.21.2",
    "xml2js": "^0.4.23"
  }
}
```


Figure 27: Alexa package.json file. Own Source.

As also shown in **Fig.28**, additional files have been added to separate different functionalities of the code, making it more understandable and scalable. However, it is not mandatory, meaning that the original structure could be maintained, and all the functionalities could be placed in *index.js*. Nevertheless, it is not a highly recommended practice.

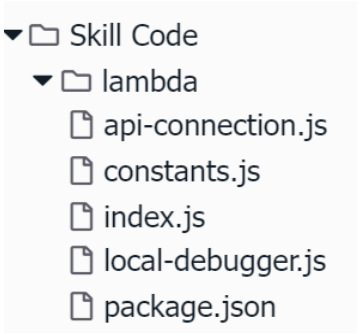


Figure 28: Alexa Code followed structure for a better understandability and scalability. Own Source.

The *constants.js file*, as seen in **Fig.29**, contains the default messages that we, as developers, have configured for Alexa's responses to user utterances.

```
module.exports = {

    WELCOME_MSG:  'Bienvenido a la aplicación de fichaje de SEMIC! Que quieres hacer?',
    WHAT_DO_YOU_WANT: 'Puedes preguntar por fichar o desfichar',
    PROFILE_NOT_RECOGNIZED: 'No puedo reconocer tu voz, contacta con el equipo de recursos humanos para que te puedan sincronizar conmigo',
    ERROR: 'Parece que algo ha ido mal',
    NAME_AVAILABLE: 'Acabas de entrar al trabajo: ',
    LEAVE_WORK: 'Acabas de terminar tu jornada laboral: ',
    ENTER_WORK: 'Acabas de empezar tu jornada laboral: ',
    NAME_MISSING: 'Me parece que no tengo tu nombre guardado, te lo puedes terminar de configurar en la app de Alexa',
    HELP_MSG: 'Puedes usar esta skill diciendo una de estas frases: fichar, desfichar',
    NOTIFY_MISSING_PERMISSIONS: 'Porfavor, activa los permissos de perfil de persona el la aplicación de Alexa.',
    UNHANDLED: 'Esta skill no acepta este tipo de peticiones. Porfavor, pregunta para entrar o salir de la empresa.',
    API_FAILURE: 'Ha habido un error con la API de perfiles de personas de Alexa, porfavor, intentalo otra vez.',
    GOODBYE_MSG: 'Hasta luego!',
    STOP: 'Adiós!'
}
```

Figure 29: Constants.js file containing the configured messages for responding the users utterances. Own Source.

Finally, the *api-connection.js* file contains the functions that will call our clocking-in API. *(For more information, refer to section 12.2, "Clocking-in API Call to the Alexa skill").*

In order for Alexa to retrieve data from different files and process it in *index.js*, it is essential to use *module.exports* in the respective files and include *require('./file_name')* in the *index.js* file. This allows Alexa to seamlessly gather information from other files.

10.2.2.2.1. Person Profile API

Once we have added all the necessary permissions mentioned in the previous point, we can utilize the Person Profile API provided by Alexa. This API allows us to differentiate individual users who have configured a *voice ID*, indicating that they have created a voice profile and activated the required permissions.

When a skill request is recognized for a specific speaker, it includes a *context.System.person* object with the *personId* identifier. We can use this object to access the user's profile information and personalize the skill's responses based on that information. [57]

With this capability, we can make use of the *getUpsServiceClient()* service to make calls related to the user's profile. Specifically, we are interested in the *getPersonsProfileName()* method, which will return the name of the user's profile who is making the skill call. [58]

By obtaining the user's profile name, we can customize our responses accordingly.

10.2.2.2.2. Enter and Leave Work Intents

In this section, we will explore the implementation of the Alexa Person Profile API and the corresponding modifications made in the initial code to enable its usage. Specifically, we will examine the *EnterWorkIntent*, *LeaveWorkIntent*, *ProfileError*, and the *skillBuilder* configuration.

Fig.30 showcases the code for *EnterWorkIntent*, which bears resemblance to *LeaveWorkIntent* as its main objective is to capture and identify the user speaking to Alexa.

```
const EnterWorkIntent = {
    canHandle(handlerInput) {
        return (
            handlerInput.requestEnvelope.request.type === 'IntentRequest' &&
            handlerInput.requestEnvelope.request.intent.name === 'EnterWorkIntent'
        );
    },
    async handle(handlerInput) {
        const person = handlerInput.requestEnvelope.context.System.person;
        console.log('recieved person', person);
        const consentToken = handlerInput.requestEnvelope.context.System.apiAccessToken;
        console.log('access token recieved', consentToken);

        if (person) {
            const personId = person.personId;
            console.log("Received personId: ", personId);

        } else {
            return handlerInput.responseBuilder
                .speak(messages.PROFILE_NOT_RECOGNIZED)
                .reprompt(messages.PROFILE_NOT_RECOGNIZED)
                .getResponse();
        }

        try {
            const client = handlerInput.serviceClientFactory.getUpsServiceClient();
            const profileName = await client.getPersonsProfileName();
            console.log("recieved person name", profileName);

            let response;
            if (profileName == null) {
                response = handlerInput.responseBuilder.speak(messages.NAME_MISSING)
                    .getResponse();
            } else {
                const speechText = `${messages.ENTER_WORK} ${profileName}`;
                response = handlerInput.responseBuilder.speak(speechText)
                    .getResponse();
            }
            return response;
        } catch (error) {
            if (error.name !== 'ServiceError') {
                const response = handlerInput.responseBuilder.speak(messages.ERROR).getResponse();
                return response;
            }
            throw error;
        }
    }
};
```

Figure 30: EnterWorkIntent implementation. Own source.

In the third phase of our project, we will integrate the Alexa skill with the Clocking-in API developed in the second phase. For more information on the final outcome of the EnterWork and LeaveWork intents after integrating with the Clocking-in API, *refer to point 13.*

To enable our skill to detect and handle custom intents, we have added the intent names to the *`addRequestHandlers()`* method, as shown in **Fig.31**. This method acts as a request handler and is responsible for managing the various interactions and requests our skill can receive.

Additionally, we have made two important configurations related to the usage of the Person Profile API:

1. We have added the error handler: Using the *addErrorHandlers(ProfileError)* method (**Fig.31**), we have included a specific error handler named *ProfileError* (**Fig.32**). This error handler is responsible for managing any errors that may arise when using the Person Profile API. By passing the *ProfileError* function as a parameter, our skill is prepared to handle and respond appropriately to errors related to retrieving user profile information. This allows us to provide customized error messages to the user when issues occur during profile access.

2. We have configured the Alexa API client: By utilizing the *withApiClient(new Alexa.DefaultApiClient())* method (**Fig. 31**), we have set up an Alexa API client. This configuration is necessary to ensure that our skill can properly interact with the Person Profile API and access user profile information when making requests.

These configurations ensure that our skill is prepared to handle custom intents and can interact effectively with the Person Profile API. As a result, our skill can personalize responses and adapt to user preferences and profiles, providing a more personalized and satisfactory experience.

```
const skillBuilder = Alexa.SkillBuilders.custom();

exports.handler = skillBuilder
    .addRequestHandlers(
        LaunchRequest,
        EnterWorkIntent,
        LeaveWorkIntent,
        SessionEndedRequest,
        HelpIntent,
        CancelIntent,
        StopIntent,
        UnhandledIntent
    )
    .addErrorHandlers(ProfileError)
    .withApiClient(new Alexa.DefaultApiClient())
    .withCustomUserAgent('sample/fichaje-semic/v1.2')
    .lambda();
```

Figure 31: addRequestHandlers, addErrorHandlers and ApiClient code implementation

```
const ProfileError = {
    canHandle(handlerInput, error) {
        return error.name === 'ServiceError';
    },
    handle(handlerInput, error) {
        if (error.statusCode === 403) {
            return handlerInput.responseBuilder
                .speak(messages.NOTIFY_MISSING_PERMISSIONS)
                .withAskForPermissionsConsentCard(permissions.NAME_PERMISSIONS)
                .getResponse();
        }
        return handlerInput.responseBuilder
            .speak(messages.API_FAILURE)
            .reprompt(messages.API_FAILURE)
            .getResponse();
    },
};
```

Figure 32: Profile Error Implementation. Own Source.

10.2.2.3. Test

Testing is of vital importance when carrying out any software development project. It allows developers to verify their progress and identify any errors or flaws before adding new functionalities. It is always easier to spot mistakes or bugs in a small section of the code rather than the entire project.

Within the ADC, there is a window that enables developers to test the skill they are creating. They can interact with Alexa by using the microphone to speak or by typing input via the keyboard. The skill's responses are returned through the speakers in Alexa's

voice, simulating the final skill's behavior, and the corresponding text is also displayed on the screen.

In our case, by integrating Alexa's Person Profile API to detect the voice of the user who is communicating, our skill will only respond by voice, it is for this reason that only the microphone that incorporates the ADC test has been used. It is also worth mentioning that it can be tested by speaking directly with the Alexa device, although we do not have the tools provided by the Alexa test itself, such as the “device log”, in which we can keep track of what the skill does to be able to provide us with this voice clocking-in.

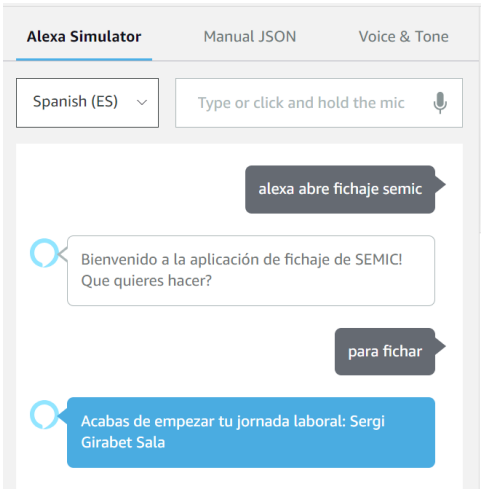


Figure 33: Proof of EnterWorkIntent in the Alexa test. Own Source.

10.2.2.3.1. CloudWatch Logs

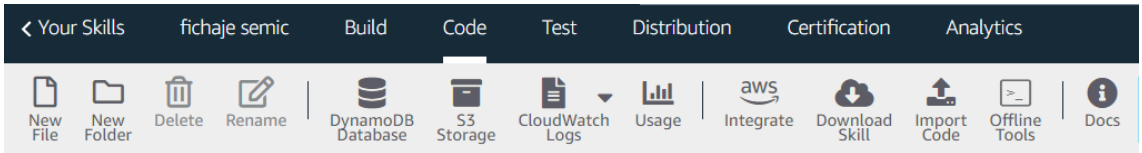


Figure 34: Localization of CloudWatch Logs in the Alexa Developer Console. Own Source

This tool is integrated in the Code part of the Alexa Developer Console (**Fig.34)**. However, it is a useful tool for testing our skill, since by using *console.log("identifier text", variable_name)* you can notice what they contain and that can help us to understand how our skill works and what data it is collecting.

As we click on this tool **(Fig.35)**, we are shown a screen where we can select our log streams **(Fig.36)**, which are just the collection of logs that are generated when the user opens the skill, and the skill interacts with that user. A history is created that can be analyzed to see what useful information the skill has provided.

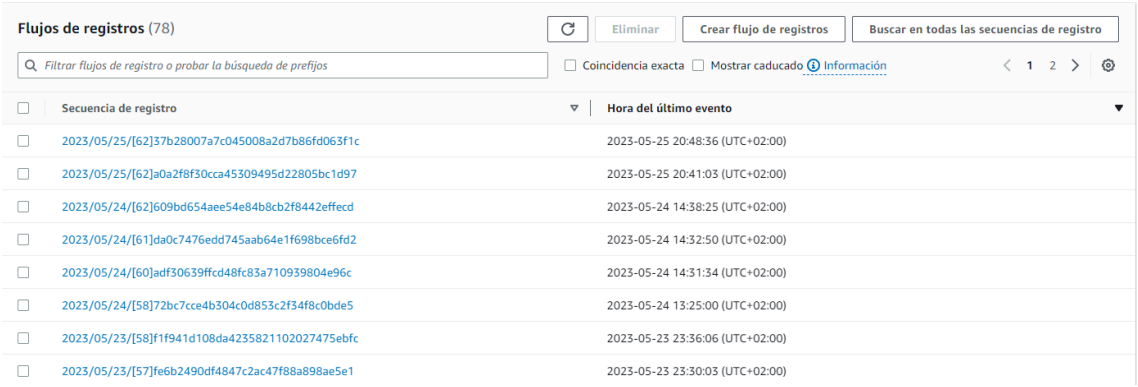


Figure 35: Show the different flows generated when the skill is invoked. Own Source

By clicking on each of the registration flows, we will be able to see in more detail the interactions and steps that our skill has followed to interact with us **(Fig. 36)**.

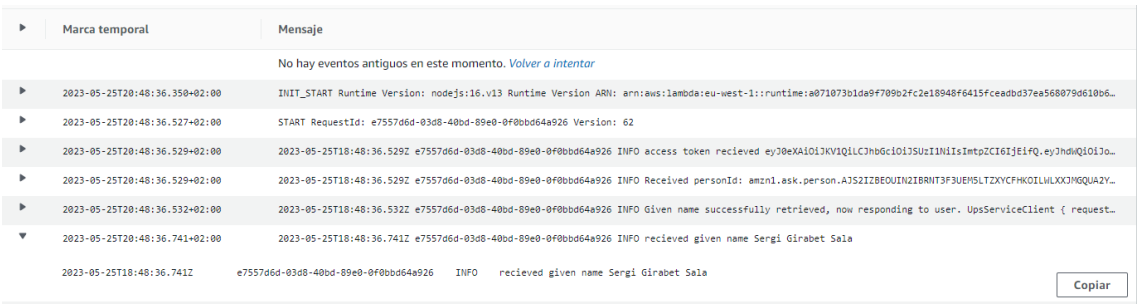


Figure 36: Example where each of the console.log () that we have added to our skill is detailed. You can see how in one of them you can see more detailed response information. Own Source

11. Second Phase. Prototype of a Clocking-in API

In this second phase, we will focus on the creation of the *clocking-in-system-soap*, a clocking-in API. This is a prototype that has been developed specifically for this project since the initial goal was to connect Alexa skill to SEMIC's existing API with all the necessary logic. However, due to security concerns and issues with employee verification, we were unable to access these resources.

Despite this setback, it has been decided to develop a more basic API that simulates the actual behavior of SEMIC's clocking-in API. This includes communication through SOAP[13] protocol, so that in the future, when we have resolved the security issues, it will be easier to switch to SEMIC's production API.

11.1. Used Tools

For the development of this part of the project, i.e., the creation of the prototype, it has been necessary to install these applications. We decided to do it this way because of the ease and speed that these components offer us:

- **Django:** It is a high-level, open-source web development framework used to simplify the creation of complex web applications using the Python programming language. In our case, Django has been used to build the signing API. Django provides many useful features, such as URL routing, database management and the ability to define SOAP web services, using specific libraries and tools to define the methods and operations that will be accessible through the signing API. [59]
- **Visual Studio Code:** It is a popular and easy-to-use source code editor. It is used by developers to write and edit the code of their projects. In our case, Visual Studio Code has been used to develop a web application using Django. Visual Studio Code offers useful features, such as syntax highlighting, code auto-

[13] SOAP stands for Simple Object Access Protocol. It is a messaging protocol commonly used in web services and APIs to exchange structured information over a network.

completion and debugging, which make programming easier and increase productivity. [60]

- **XAMPP:** It is a tool that allows us to easily set up a web server environment on our own computer. It is useful for developing and testing web applications locally before uploading them to an online server. XAMPP includes several essential parts for a web development environment, such as the Apache web server and the MySQL database management system. [61]
- **MySQL:** It is a database management system widely used in web applications. MySQL has been used to retain the data of the inputs and outputs of the workers. Basically, MySQL allows us to store and retrieve information efficiently. In this case, we are storing the name of the person who clocks in and the date they clocked out in the database.
- **Postman:** Postman is a popular tool that is used to test and interact with APIs. It provides a user-friendly interface that allows developers to make requests to APIs, inspect the responses, and perform various testing and debugging tasks. Its flexibility and extensibility allow it to be used effectively for testing SOAP APIs. It provides a convenient and intuitive interface that simplifies the process of constructing, sending, and validating SOAP requests and responses. [62]

These technologies have been used because they offer a powerful and efficient combination for developing web applications. XAMPP provides an easy-to-configure web server environment, MySQL is a reliable and widely adopted database, Django simplifies the development of complex web applications by offering a wealth of libraries and tools, Visual Studio Code is a popular and highly functional code editor and Postman serves as a versatile tool for testing SOAP APIs. Together, these tools have allowed us to create a punch-in API efficiently and with greater productivity.

11.2. Prototype Development

When you create a project in Django, the framework generates a basic structure of files and directories. However, it is we, as developers, who must work on those files to implement the desired functionality in our application. To achieve the desired functionality, the different files included in Django's basic structure have been tweaked.

We have also installed the necessary libraries and packages in the different files for them to work correctly.

11.2.1. models.py

This file defines the classes that represent the data models of our application. Each class defines a table in the database and the fields of that table.

In our case, we have defined a model called *Clocking* that inherits from the class models.Model. This model represents a table in the database called "clocking".

These fields have been added to this model:

- **name:** A *CharField* representing the full name of the worker.
- **timestamp**: A *DateTimeField* representing the timestamp (date and time) of the entry or exit associated with the worker.
- **type**: It is an *IntegerField* representing the type of input and is defined using the choices attribute which provides two options: "in" (0) and "out" (1). It has been defined in this way to differentiate the inputs and outputs of workers.

In **Fig.37** you can see these three fields, which will be stored in our DB created below.

```python
from django.db import models
from django.db.models.fields import CharField

# Create your models here.
class Clocking(models.Model):
    name = models.CharField(max_length=255)
    timestamp = models.DateTimeField()
    OPTIONS = (
        (0, 'in'),
        (1, 'out')
    )
    type = models.IntegerField(choices=OPTIONS)

    def __str__(self):
        return self.name
```

Figure 37: models.py file with its implementation. Own Source

11.2.2. settings.py

In this settings file, the 'DATABASES' section has been modified. By default, Django comes with SQLite3, which is fine for local testing, but we preferred to use MySQL for its performance and scalability, ready for production environments. That's why we've added the MySQL driver using the command *'pip install mysqlclient'* and modified the previously mentioned section with this one here:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'semic_clocking',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

Figure 38: MySQL Database configuration in the settings file. Own Source

Note that when changing the database, you need to make use of these two commands: *'python manage.py makemigrations' and 'python manage.py migrate'*. That's because these commands are necessary to update the database structure according to the changes in your models and ensure that the MySQL database correctly reflects those changes. [63]

In addition, once the database has been configured, it is important to create a superuser to access the Django administration panel and manage our application. To create a superuser, execute the following command in the terminal or command line, from the root directory of your Django project: *'python manage.py createsuperuser'*. The command will prompt you to enter a username, an email address (optional) and a password for the superuser. [64]

Thanks to XAMPP **(Fig.39)**, we are able to deploy this database locally, without using external servers, as it allows us to easily configure a complete local development that includes a web server (Apache), a database (MySQL) and other necessary components to run the application efficiently on our own computer.

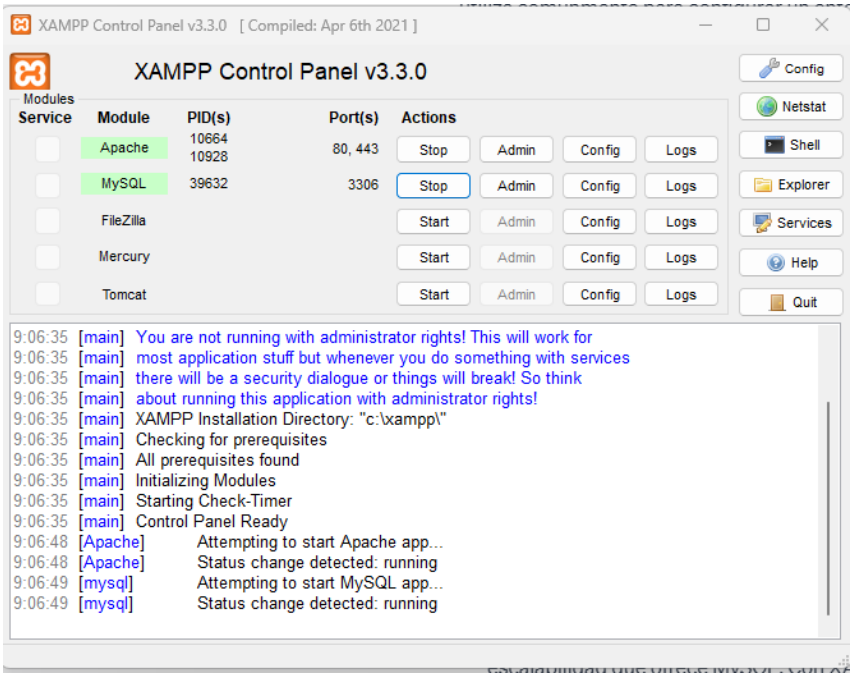


Figure 39: XAMPP Control Panel with Apache and MySQL running actions. Own Source

It is essential that the port must be 3306, as this is where MySQL runs. Also note that this environment is for local testing, in the production environment we will switch to Heroku with a PostgreSQL database *(for more information see 12.1. Deploying Clocking-in API with Heroku).*

Once we have Apache and MySQL running, we must insert the command *'python manage.py runserver'*.

This will give us, as shown in **Fig.40**, a url. If we add the */admin* to this url (our localhost with port 8000), we will be able to access the Django administration panel and administer our application once we have requested our credentials as superuser as can be seen in **Fig.41**.

```
Django version 4.2.1, using settings 'clocking.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 40: python.manage.py runserver response. Own Source

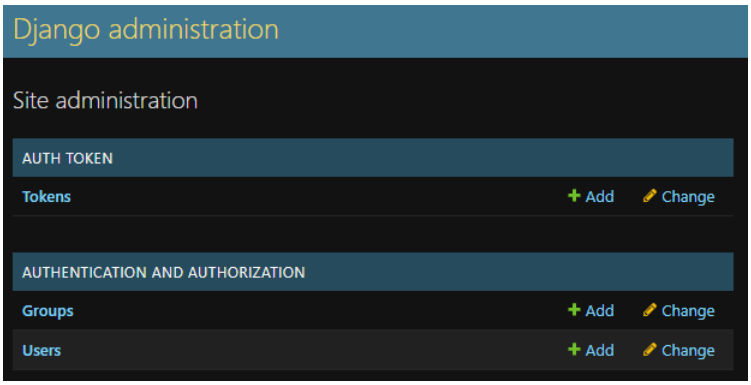


Figure 41: Django Administration Console. Own Source

We have chosen to add an authentication token to make the application more secure when making calls from the Alexa skill. In this way, only users who have this token can be authenticated and make the process of signing in or out of the company. The process is as simple as pressing the add button and a token will be generated automatically.

11.2.3. utils.py

This file contains a function called *checkToken*, as can be seen in **Fig.42**. This function is used to check the validity of an authentication token, created in the Django Administration Console. The purpose of this function is to ensure that a valid token is provided in the calls to the *clockIn* and *clockOut* methods of the *views.py* file. This function plays an important role in the security of operations and allows us to control access to the people who use our application, and thus make our application a little more secure.

```python
from rest_framework.authtoken.models import Token

class Utils:  # Function to check if the given token is valid

    @staticmethod
    def checkToken(token):
        user_with_token = Token.objects.filter(key=token).first()
        if not user_with_token:
            raise Exception(400, "token.invalid")
```

Figure 42: Implementation of the ckeckToken function. Own Source

11.2.4. views.py

It is used to define the functions or classes that handle HTTP requests and generate the corresponding responses. These functions or classes are known as views.

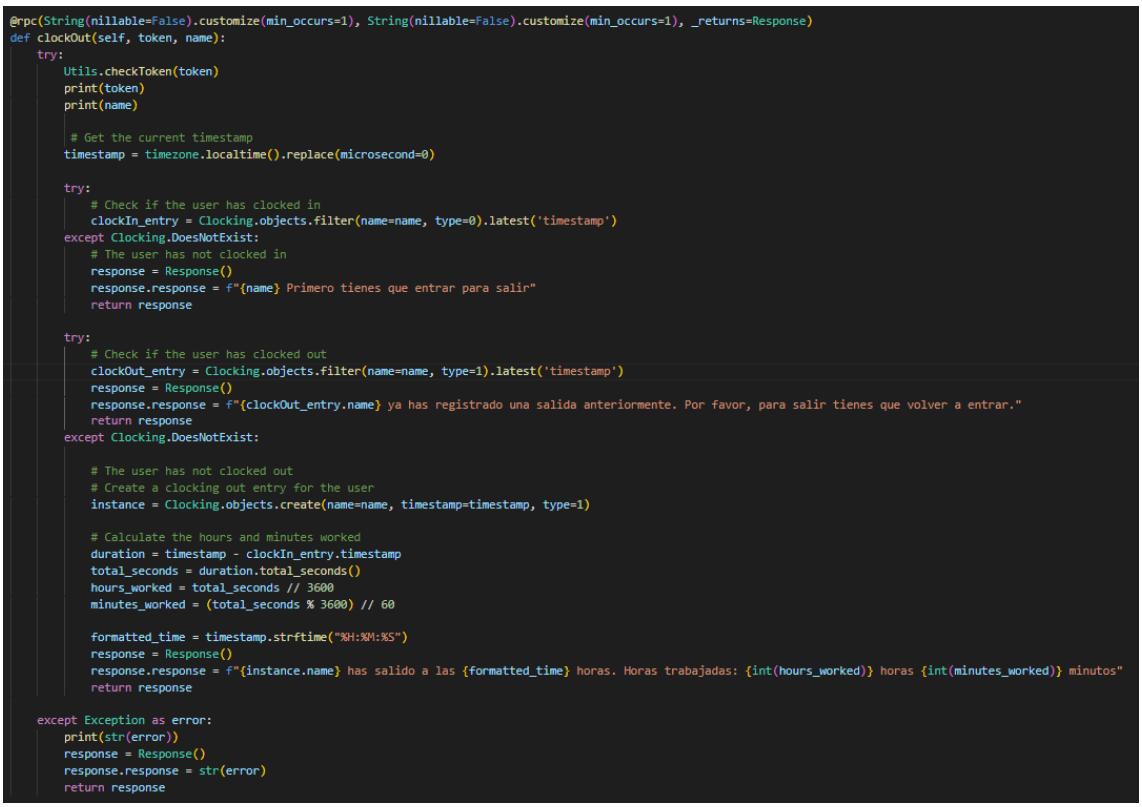
The *ManageClocking-inActionsView* class contains two main methods: *clockIn* and *clockOut*. Both methods are decorated with @rpc, which indicates that they are operations that can be invoked via SOAP. These methods receive certain parameters, such as token and name, and return a Response object.

The *clockIn* method, as can be seen in the **Fig.43,** is used to clock in an employee. First, the user's token is checked using the *checkToken* function of the Utils module. Then, it checks if the user already has an active clock-in without a clock-out. If so, a response is returned indicating that the user has clocked in before. Next, the current timestamp is obtained, and a record is created in the database using the *Clocking model* with the user's name and clock-in time. Finally, a response is generated indicating the clocked-in time and the user who has clocked in.

```python
@rpc(String(nillable=False).customize(min_occurs=1), String(nillable=False).customize(min_occurs=1), _returns=Response)
def clockIn(self, token, name):
    try:
        Utils.checkToken(token)
        print(token)
        print(name)

        try:

            # Check if the user has an active clock-in without a clock-out
            latest_clock_in = Clocking.objects.filter(
                name=name,
                type=0
            ).latest('timestamp')

            if latest_clock_in:
                try:
                    # Check if the latest clock-in has a subsequent clock-out
                    clock_out = Clocking.objects.filter(
                        name=name,
                        type=1,
                        timestamp__gt=latest_clock_in.timestamp
                    ).exists()

                    if not clock_out:
                        response = Response()
                        response.response = f"{name} Ya has hecho check-in anteriormente. Primero debes hacer check-out."
                        return response

                except Clocking.DoesNotExist:
                    pass

        except Clocking.DoesNotExist:
            pass

        # Get the current timestamp
        timestamp = timezone.localtime().replace(microsecond=0)


        # Get the user's name
        instance = Clocking.objects.create(name=name, timestamp=timestamp, type=0)

        formatted_time = timestamp.strftime("%H:%M:%S")
        response = Response()
        response.response = f"{instance.name} has entrado a las {formatted_time} horas"
        return response

    except Exception as error:
        print(str(error))
        response = Response()
        response.response = str(error)
        return response
```

Figure 43: ClockIn method implementation. Own Source

The *clockOut* method, as can be seen in the **Fig.44**, is used to clock out an employee. As in clockIn, the user's token is checked. Then, the current timestamp is retrieved and the user's last clocking in is searched for using the *Clocking model*. If no clock-in token is found, a response is returned indicating that the user must clock-in first. Then, a check is made to see if the user has already clocked out. If so, a response is returned indicating that the user has already clocked out and must clock back in. If not, a clock-out record is created in the database and the duration of the time worked is calculated. Finally, a response is generated indicating the recorded clock-out time and the duration of the time worked by the employee who has clocked out.



```
@rpc(String(nillable=False).customize(min_occurs=1), String(nillable=False).customize(min_occurs=1), _returns=Response)
def clockOut(self, token, name):
    try:
        Utils.checkToken(token)
        print(token)
        print(name)

        # Get the current timestamp
        timestamp = timezone.localtime().replace(microsecond=0)

        try:
            # Check if the user has clocked in
            clockIn_entry = Clocking.objects.filter(name=name, type=0).latest('timestamp')
        except Clocking.DoesNotExist:
            # The user has not clocked in
            response = Response()
            response.response = f"{name} Primero tienes que entrar para salir"
            return response

        try:
            # Check if the user has clocked out
            clockOut_entry = Clocking.objects.filter(name=name, type=1).latest('timestamp')
            response = Response()
            response.response = f"{clockOut_entry.name} ya has registrado una salida anteriormente. Por favor, para salir tienes que volver a entrar."
            return response
        except Clocking.DoesNotExist:

            # The user has not clocked out
            # Create a clocking out entry for the user
            instance = Clocking.objects.create(name=name, timestamp=timestamp, type=1)

            # Calculate the hours and minutes worked
            duration = timestamp - clockIn_entry.timestamp
            total_seconds = duration.total_seconds()
            hours_worked = total_seconds // 3600
            minutes_worked = (total_seconds % 3600) // 60

            formatted_time = timestamp.strftime("%H:%M:%S")
            response = Response()
            response.response = f"{instance.name} has salido a las {formatted_time} horas. Horas trabajadas: {int(hours_worked)} horas {int(minutes_worked)} minutos"
            return response

    except Exception as error:
        print(str(error))
        response = Response()
        response.response = str(error)
        return response
```

Figure 44: ClockOut method implementation. Own Source

The **soap_app** variable, as can be seen in **Fig.45** is creating a SOAP application specifically for handling clocking actions. This application is designed to receive SOAP requests related to *clocking in* and *clocking out.*

Inside the SOAP application, there is a view called **ManageClockingActionsView**. This view contains the logic for processing SOAP requests and generating appropriate responses.

The SOAP application is configured with two protocols: an input protocol and an output protocol. The input protocol, defined as SOAP 1.1, handles the incoming SOAP requests and ensures they adhere to the specified structure. The output protocol, also SOAP 1.1, handles the generation of SOAP responses and allows for different data types to be returned.

After configuring the SOAP application, it is integrated with the Django framework using **DjangoApplication**. This integration allows the SOAP application to seamlessly work within the Django project and leverage Django's features and functionality.

To ensure smooth operation, the **csrf_exempt** decorator is applied to the SOAP application. This decorator exempts the SOAP application from Cross-Site Request Forgery (CSRF) protection, which is a security measure in Django.

In summary, the provided code in **Fig.45** sets up our SOAP application to manage clocking actions. It defines the necessary protocols, integrates it with Django, and ensures smooth operation by exempting it from CSRF protection.

```
soap_app = Application(
    [ManageClockingActionsView],
    tns='django.soap.clocking',
    in_protocol=Soap11(validator='lxml'),
    out_protocol=Soap11(polymorphic=True),
)


django_soap_application = DjangoApplication(soap_app)
soap_application = csrf_exempt(django_soap_application)
```

Figure 45: SOAP configuration to manage clocking actions. Own Source.

11.2.5. admin.py

Used to register the Clocking model in the Django admin panel.

The Django admin panel is a built-in web interface that allows administrators to easily manage and manipulate application data. Registering the Clocking model using

admin.site.register(Clocking), as shown in **Fig.46**, enables the administration functionality for that particular Clocking model. model.

```
from django.contrib import admin
from manageClock.models import *

# Register your models here.

admin.site.register(Clocking)
```

Figure 46: Implementation of the admin.py file. Own Source

Once the model is registered in the admin panel, it can be accessed via the Django admin panel URL (**Fig.47**). This allows us, as administrators, to view, create, modify, and delete objects in the Clocking model directly from the web browser, without the need to write additional code.

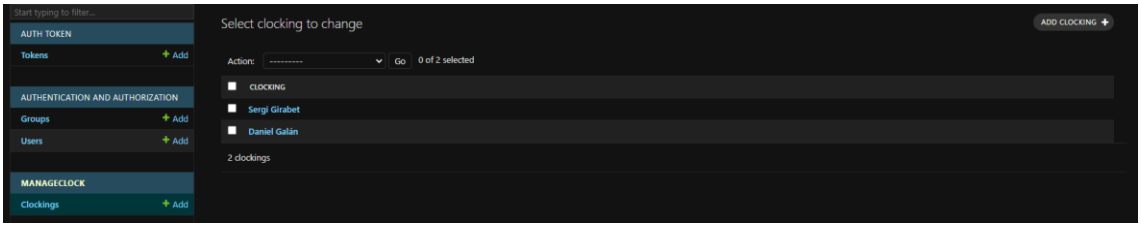


Figure 47: Clocking Panel to view, create, modify and delete clockings. Own Source.

11.2.6. url.py

In this file we have defined the *Clocking_patterns* variable (**Fig.48)**, a tuple that contains a list of URL patterns specific to the clocking feature. It uses the *re_path* function from Django's *urls* module to define the URL pattern.

In this case, the URL pattern is *'^soap_service/'*, which means that any request with a URL path starting with */soap_service/* will be handled by the associated view.

The associated view is referenced *as view.soap_application*. By including these URL patterns in the application's URL configuration, any request with a URL path

matching the defined pattern will be routed to the *soap_application* view, defined in the *views.py* to call the *ClockIn* and *ClockOut* functions.

```
from django.urls import re_path
import manageClock.views as view

Clocking_patterns = ([
    re_path(r'^soap_service/', view.soap_application),
], 'clocking')
```

Figure 48: Implementation of the url.py file. Own Source.

11.2.7. urls.py

In this file we are configuring the URL patterns for the Django project, including the clocking-in functionality and the Django admin interface.

The *re_path('admin/', admin.site.urls)* maps the URL path starting with */admin/* to the Django admin interface. It allows you to access the admin site where you can manage and interact with the project's models and data.

The *re_path(r'^api/clocking/', include(Clocking_patterns))* maps the URL path starting with */api/clocking/* to the clocking functionality. It includes the *Clocking_patterns* from the *manageClock.url* module, viewed previously, which contains the specific URL pattern for clocking.

```
from django.contrib import admin
from django.urls import include, re_path
from manageClock.url import Clocking_patterns
from django.conf import settings

urlpatterns = [
    re_path('admin/', admin.site.urls),
    re_path(r'^api/clocking/', include(Clocking_patterns))
```

Figure 49: Implementation of the urls.py file. Own Source.

11.2.8. wsgi.py

WSGI stands for Web Server Gateway Interface, and it is a specification that defines how web servers communicate with web applications. The WSGI configuration file is responsible for setting up the WSGI application callable that the web server will use to handle incoming requests. Overall, this code sets up the WSGI application for the clocking-in API and makes it available to the web server for handling incoming requests.

```
import os
from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'clocking.settings')

application = get_wsgi_application()
```

Figure 50: Implementation of the wsgi.py file. Own Source.

11.2.9. Testing the clocking-in API with Postman

With this application, we are able to make a call to our API called the *clocking-in-system-soap* to test the Django application itself along with the database configuration.

In our case, a POST request is made to the following URL:

http://localhost:8000/api/clocking/soap_service/

The port 8000 is the default port provided by Django for testing purposes, while *api/clocking/soap_service/* is the URL we have defined in the *urls.py* file to send the necessary information along with the Extensible Markup Language (XML) generated in settings.py for conducting the corresponding tests. **Fig.51** and **Fig.52** show the respective calls to the *clockIn* and *clockOut* methods.

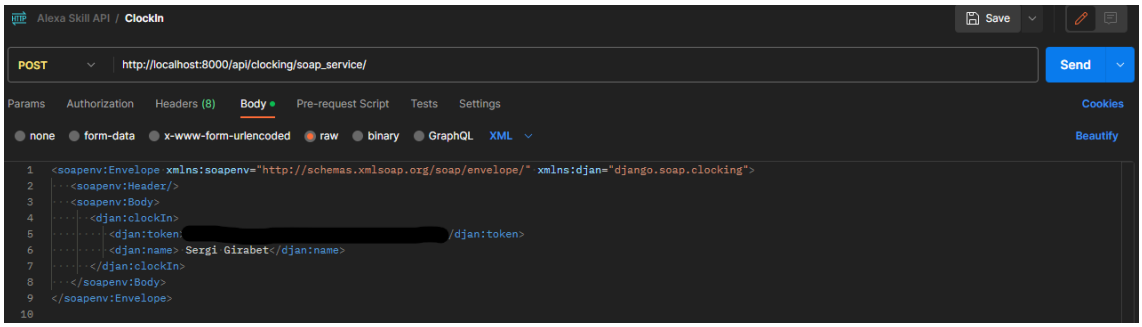


Figure 51: ClockIn call example with Postman. Own Source.

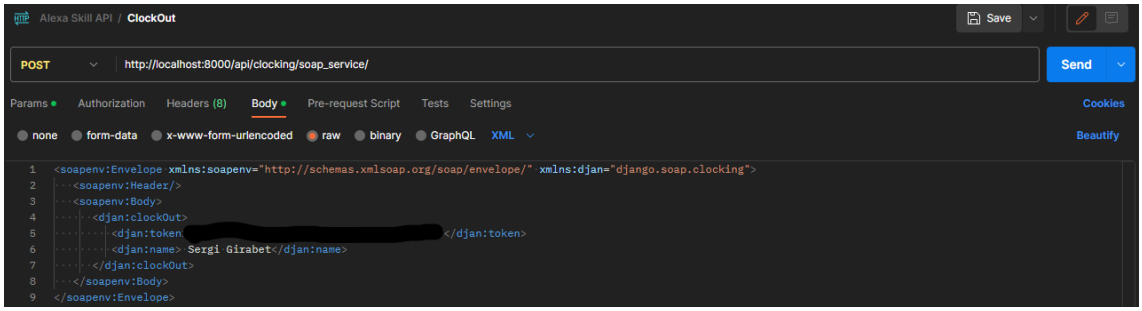


Figure 52: ClockOut call example with Postman. Own Source.

In the figures shown below, the implemented logic of *ClockIn* and *ClockOut* functions of *settings.py* is applied. In **Fig.53**, we can observe that it returns the person and the time when clocking was performed. **Fig.54** indicates that the person has already clocked in and therefore needs to clock out to clock in again. **Fig.55** shows that the person has just clocked out, showing the total hours worked. **Fig.56** indicates that the person has already clocked out and therefore needs to clock in again to perform the clock-out action. Lastly, **Fig.57** shows that the person has tried to clock out before clocking in.

Figure 53: Example of a response from the ClockIn method in Postman that returns the name and time of the entry we have clocked in. Own Source.

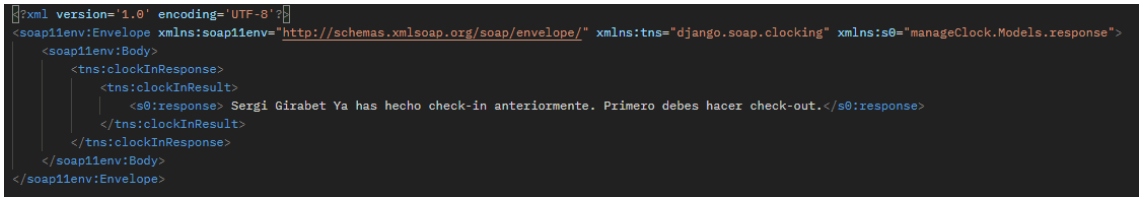


Figure 55: Example of a response from the ClockIn method in Postman returning that we have already clocked in. Own Source.

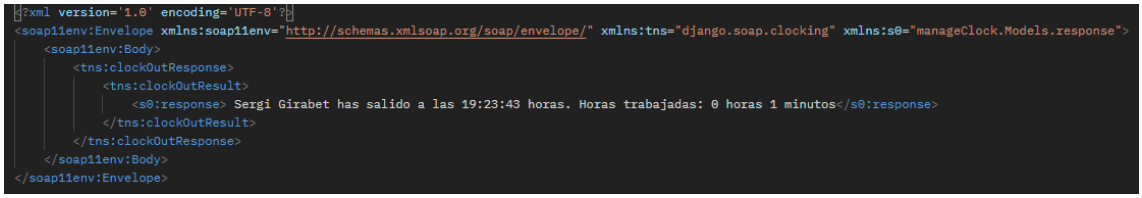


Figure 54: Example of a response from the ClockOut method in Postman that returns the name and the hours we have worked. Own Source.

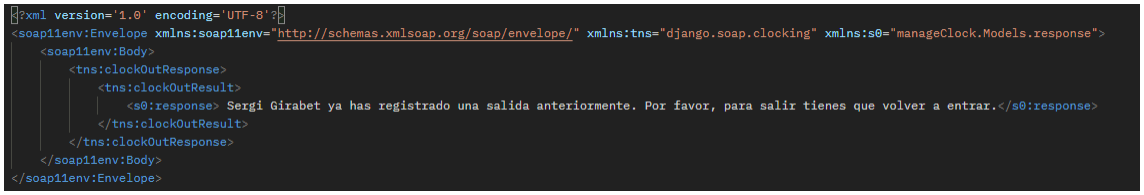


Figure 56: Example of a response from the ClockOut method in Postman returning that we have already clocked out. Own Source.

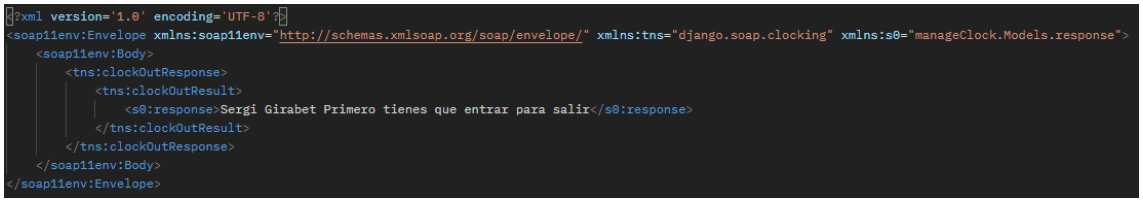


Figure 57: Example of a response from the ClockOut method in Postman returning that we must clock in before clocking out. Own Source.

12. Third Phase. Integration of the Skill with the Clocking-in API

In this third phase, we will explore the deployment of the clocking-in API using the Heroku platform. Heroku offers a convenient and scalable solution for hosting web applications, making it an ideal choice for deploying APIs. We will walk through the step-by-step process of deploying the Clocking-in API to Heroku, ensuring that it is accessible to users over the internet.

Once our API is deployed, we will delve into integrating it with an Alexa skill. By establishing a connection between the Clocking-in API and Alexa via SOAP protocol, we can enable users to interact with the API through voice commands for clocking-in purposes.

Finally, we will conduct final tests to ensure the smooth functioning of our deployed Clocking-in API. Through rigorous testing, we will verify the API's functionality, performance, and responsiveness in a production environment.

12.1. Deploying Clocking-in API with Heroku

Deploying a Django API to Heroku involves the process of making our API accessible on the internet through Heroku's cloud platform. Heroku provides a convenient and scalable environment for hosting web applications, allowing us to reach a wide audience without worrying about server infrastructure.

There is a wide array of platforms available for hosting web services, each with its own set of advantages and considerations. However, after careful consideration, the decision has been made to utilize Heroku for hosting our web service. Heroku stands out for its extensive experience in the field and the convenience it provides in deploying and managing web applications.

Nonetheless, it is important to acknowledge that Heroku is no longer a free application as it used to be. This means that for the purposes of this project, we will solely be utilizing Heroku during the development phase. Once we have confirmed the smooth and efficient operation of our Clocking-in API in conjunction with the SEMIC timekeeping skill, we will explore alternative hosting options that are more cost-effective.

By taking this approach, we can ensure that the project progresses smoothly and efficiently within the development environment while being mindful of potential cost considerations. The evaluation of alternative hosting solutions will allow us to strike a balance between functionality, performance, and cost-effectiveness, ensuring the long-term sustainability and scalability of our web service beyond the development phase. [65]

To deploy our clocking-in API to Heroku, first we need to create an Heroku account and set up a new Heroku application. [66]

Once we are registered and our application, named *fichajesemic*, as can be seen in **Fig.57** is set up, it is time to use the Heroku Command Line Interface (CLI) to link our local code repository to our Heroku application. [67]

It is worth mentioning that Heroku has a deployment option with GitHub[14], in which it is only necessary to link our project and it will do the deploy for us. We have used *SEMIC's GitLab*[15], that's why we will deploy it via CLI, as Heroku doesn't provide the option to upload it directly.

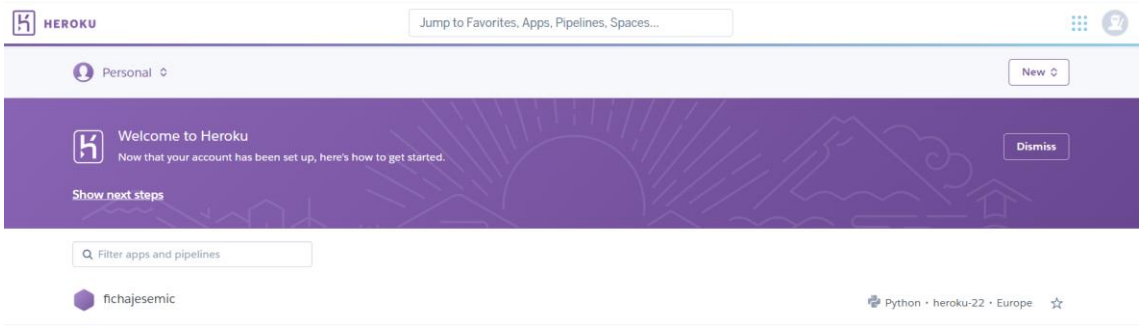


Figure 58: fichajesemic application hosted in the Heroku platform. Own Source.

[14] GitLab is a web-based platform that provides a complete DevOps lifecycle solution for managing source code repositories, tracking issues, and enabling continuous integration and deployment. It offers features such as version control, collaboration tools, project management, and automated testing. GitLab allows teams to work together effectively, making it easier to develop and deploy software applications.

[15] GitHub is a web-based hosting service for Git repositories. It provides a platform for developers to collaborate on projects, manage code repositories, and track changes made to the codebase. GitHub offers features such as version control, issue tracking, pull requests, and code review tools. It is widely used by open-source projects and serves as a hub for sharing and discovering code.

Next, we will need to configure our clocking-in API for Heroku. This involves specifying the necessary dependencies and configurations in a *requirements.txt* file**,** using the command *python -m pip freeze > requirements.txt* and a *Procfile* file in the root of the project. The *requirements.txt* file lists all the Python packages our project depends on, as seen in **Fig.59,** while the *Procfile* specifies the command to run the clocking-in API, as seen in **Fig.58**. [68]

```
asgiref==3.6.0
dj-database-url==2.0.0
Django==4.2.1
django-heroku==0.3.1
django-rest-authtoken==2.1.4
djangorestframework==3.14.0
gunicorn==20.1.0
lxml==4.9.2
psycopg2==2.9.6
pytz==2023.3
spyne==2.14.0
sqlparse==0.4.4
typing_extensions==4.6.2
tzdata==2023.3
whitenoise==6.4.0
```

Figure 60: requirements.txt file with all the necessary dependences to deploy our project in the Heroku platform. Own Source.

```
web: gunicorn clocking.wsgi --log-file -
```

Figure 59: Procfile file with the specific command to run the clocking API in Heroku platform. Own Source.

Heroku also provides a wide range of add-ons and services that allows us to integrate into our clocking-in API, such as databases, caching systems, and logging utilities. We have used the Heroku Postgres add-on (**Fig.60**) to deploy our *PostgreSQL* database. [69]

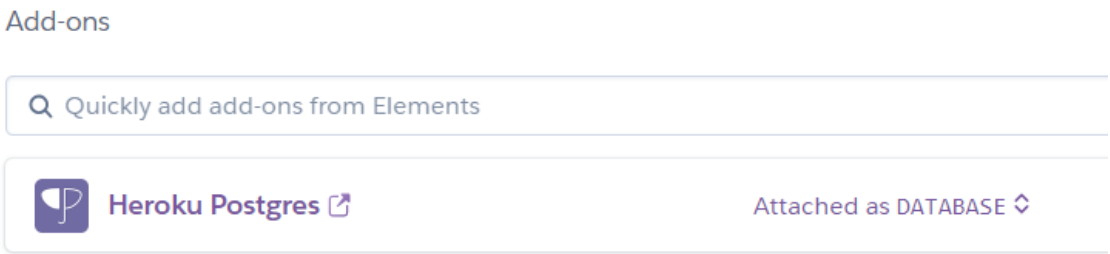


Figure 61: Heroku Postgres add-on. Own Source.

In the second phase of this project implementation, we were using *mySQL* database, but we decided to switch from MySQL to PostgreSQL when deploying our application on Heroku as PostgreSQL brings a host of benefits, such as improved compatibility and advanced features. PostgreSQL prioritizes data integrity, reliability, and scalability, with a strong community and ecosystem support. Heroku seamlessly integrates with PostgreSQL, providing managed databases and simplified deployment. Overall, this switch enhances our application's capabilities and ensures a robust and reliable database solution for our deployed application. To fulfil the required specifications, the DATABASES of the settings.py of our application has been changed to this one:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'd8ua9dkrhff7o5',
        'USER': 'kvxtfrhzyjbvnd',
        'PASSWORD': '[redacted]',
        'HOST':'ec2-54-195-144-105.eu-west-1.compute.amazonaws.com',
        'PORT': '5432'
    }
}
```

Figure 62: DATABASES configuration for running PostgreSQL database in Heroku platform. Own Source.

The data you can see in **Fig.61** is provided by the Heroku Postgres add-on. It is also important to mention that, after having made a database change, it will be necessary to do the *'python manage.py makemigrations'* and *'python manage.py createsuperuser'* again. *(for more information see section 11.2.2. settings.py).*

After configuring our project, we have used the *Git version control system*[16] to push our code to Heroku. Heroku will automatically build and deploy our Django API based on

[16] Git is a distributed version control system that allows developers to track changes to their codebase over time. Version control systems like Git help manage and organize code by keeping a record of every change made to files and allowing developers to collaborate on a project without overwriting each other's work. Git enables developers to work on different branches, merge changes, and revert to previous versions of their code when needed. It helps teams to collaborate efficiently and maintain a history of their codebase.

the instructions in our *Procfile*. The result will give us the URL of our application deployed in Heroku, in our case: *https://fichajesemic.herokuapp.com/*

If we add the path */admin/* to the end of the previous URL we will be able to enter the admin screen of our API. If we add the path */api/clocking/soap_service/* we will be able to see our xml to make the properly API calls.

12.2. Clocking-in API call to the Alexa skill

The *api-connection.js* module acts as a bridge between the EnterWork and LeaveWork Intents and the clocking-in API designed in the second phase of this project. This module is responsible for handling clock-in and clock-out operations seamlessly.

The module provides two main functions: *getClockIn* (**Fig,62**) and *getClockOut* (**Fig.63**). These functions allow users to interact with the clocking-in API and perform the respective actions.

When a user wants to clock in with the proper utterance, Alexa recognizes this utterance and prompts to the *EnterWorkIntent.* Then, this intent calls the *getClockIn* function and pass the profile name of the recognized worker as an input. Behind the scenes, the module constructs a SOAP request, encapsulating the necessary information. This includes the user's profile name and the token generated in the second phase of this project for authentication.

The module then sends the SOAP request to the clock-in endpoint of the API using the popular **axios** library. The request is formatted properly, ensuring it is recognized as a SOAP request, and includes the appropriate headers. The clock-in request is sent to the designated URL where the clocking-in API is hosted.

Once the response is received from the API, the module parses the XML data using the **xml2js** library. This parsing step transforms the XML response into a more manageable data structure. From this parsed data, the module extracts the clock-in response and prepares it for the user.

The *getClockOut* function operates similarly to *getClockIn*. When a user wants to clock out with the proper utterance, Alexa recognizes this utterance and prompts to the

LeaveWorkIntent. Then, this intent calls the *getClockOut* function and pass the profile name of the recognized worker as an input. The module constructs a SOAP request for clock-out, sends it to the corresponding endpoint of the API, and handles the XML response in a similar fashion.

Both functions are designed to be asynchronous, returning promises to allow for non-blocking execution. This means that users can handle the responses asynchronously and continue with their application logic without waiting for the clocking-in operations to complete.

```
module.exports = {
    getClockIn(profileName){
        let data = `
            <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:djan="django.soap.clocking">
                <soapenv:Header/>
                <soapenv:Body>
                    <djan:clockIn>
                        <djan:token>[redacted]</djan:token>
                        <djan:name>${profileName}</djan:name>
                    </djan:clockIn>
                </soapenv:Body>
            </soapenv:Envelope>`;

        let config = {
            method: 'post',
            maxBodyLength: Infinity,
            url: 'https://fichajesemic.herokuapp.com/api/clocking/soap_service/',
            headers: {
                'Content-Type': 'application/xml'
            },
            data: data
        };

        return axios.request(config)
            .then((response) => {
                return new Promise((resolve, reject) => {
                 parseString(response.data, (err, result) => {
                    if (err) {
                        reject(err);
                    } else {
                      const clockInResponse = result['soap11env:Envelope']['soap11env:Body'][0]['tns:clockInResponse'][0]['tns:clockInResult'][0]['s0:response'][0];
                      resolve(clockInResponse);
                    }
                 });
              });
            })
            .catch((error) => {
                console.log(error);
            });
    },
```

Figure 63: getClockIn function implementation. Own Source.

```
getClockOut(profileName) {
    let data = `
        <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:djan="django.soap.clocking">
            <soapenv:Header/>
            <soapenv:Body>
                <djan:clockOut>
                    <djan:token>[redacted]</djan:token>
                    <djan:name>${profileName}</djan:name>
                </djan:clockOut>
            </soapenv:Body>
        </soapenv:Envelope>`;

    let config = {
        method: 'post',
        maxBodyLength: Infinity,
        url: 'https://fichajesemic.herokuapp.com/api/clocking/soap_service/',
        headers: {
            'Content-Type': 'application/xml'
        },
        data: data
    };

    return axios.request(config)
        .then((response) => {
            return new Promise((resolve, reject) => {
             parseString(response.data, (err, result) => {
                if (err) {
                    reject(err);
                } else {
                  const clockOutResponse = result['soap11env:Envelope']['soap11env:Body'][0]['tns:clockOutResponse'][0]['tns:clockOutResult'][0]['s0:response'][0];
                  resolve(clockOutResponse);
                }
             });
          });
        })
        .catch((error) => {
            console.log(error);
        });

},
```

Figure 64: getClockOut function implementation. Own Source.

12.3. Final Tests

To demonstrate the proper functioning of the SEMIC clocking-in skill after integrating the API, we return to the Alexa Developer Console (ADC) in the "Tests" section to test how Alexa responds to our utterances. As you can see in the figures, the behavior is appropriate, and it responds exactly as we expect and know will happen after implementing the API.

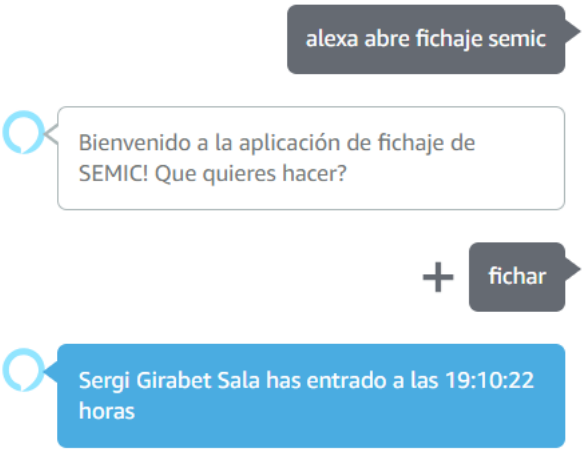


Figure 65: Example of response from Alexa returning the name and the time from a user who clocks in. Own Source.

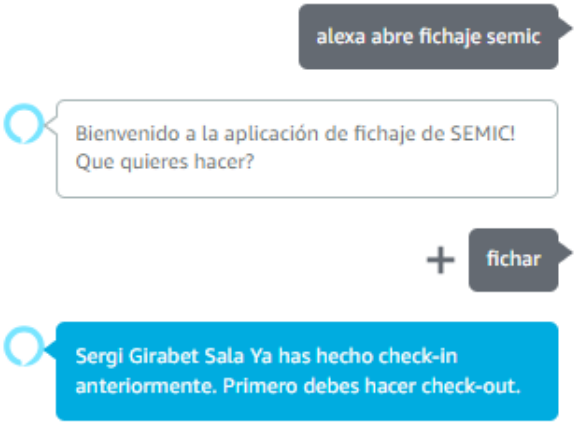


Figure 66: Example of response from Alexa returning that the user must clock out because he/she has already clocked in. Own Source.

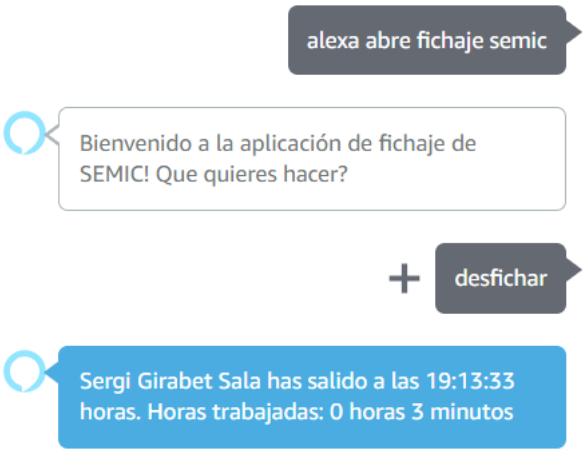


Figure 68:Example of response from Alexa returning the name and the total hours worked from the user who has clocked out. Own Source.

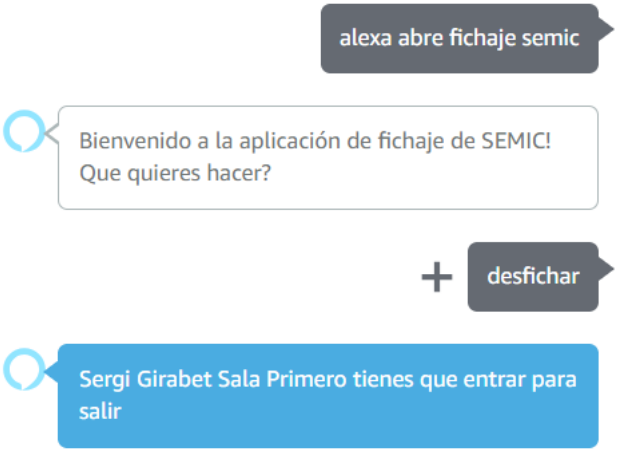


Figure 67: Example of response from Alexa returning that the user has already clocked out. Own Source.

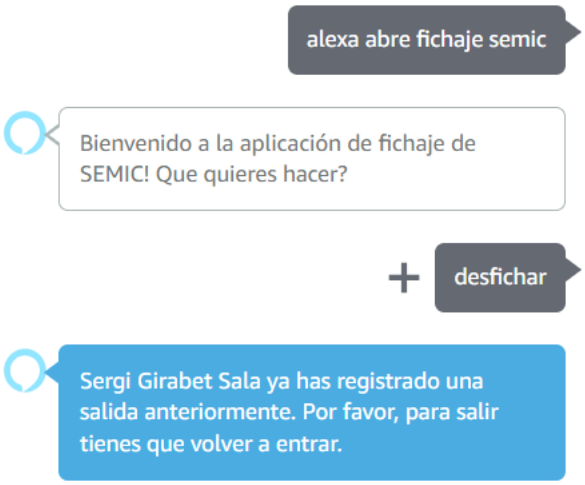


Figure 69: Example of response from Alexa returning that the user must clock in before clocking out. Own Source.

13. Cost of Implementing the Clocking-in System

The costs of implementing a skill of clocking-in with Alexa are of two types: human and technical. The technicians include the software, equipment, devices and all the infrastructure needed to get the system up and running.

13.1. Hardware And Software Cost Criteria

When calculating costs related to hardware, only those investments that I had to make in this project have been considered. In those hardware elements with a useful life longer than the duration of the project, the amortization cost of this equipment has been taken into account instead of the total purchase price. You can see this calculation below:

Element	Purchase price	Service life	Depreciation	Project cost
Laptop Dell XPS 13 9310	1.699,99 €	6 years	7%	119,00 €
Samsung Monitor	499,99 €	6 years	5%	25,00 €
Alexa Echo Dot (3rd Generation)	29,99 €	6 years	20%	6,00 €
Total	-	-	-	*150,00 €*

Table 1: Hardware depreciation cost. Own source.

About the software, the cost of hosting the clocking-in skill and its publication is free, as there are some limits set but these are not exceeded as this skill is intended to be used only in the SEMIC company and this means that these resources are not exceeded. It is also worth mentioning that all the programs used for the implementation of the clocking-in skill that has been mentioned in the previous sections are free of charge.

13.2. Human Costs Criteria

To determine the employee cost estimate, we will utilize a time estimate reflecting the hours dedicated to this project, not only by myself but also by the individuals who have contributed and assisted me in its execution. Here is a table presenting the roles of the individuals involved, their hourly rates, and the number of hours they have dedicated to this project:

Role Type	Hours Worked	Hours of meetings	Total hours
Programmer	800	12	812
Senior Programmer	0	10	10
Head of Department	0	2	2

Table 2: Worked hours depending on the role type. Own Source.

Role Type	Cost per hour	time spent (hours)	Total cost
Programmer	8,33 €	812	6.766,67 €
Senior Programmer	17,19 €	10	171,88 €
Head of Department	20,83 €	2	41,67 €
Total	-	*824*	*6.980,21 €*

Table 3: Cost per hour depending on the role type. Own Source.

13.3. Total Costs

Concept	Cost
Hardware	150,00 €
Salaries	6.980,21 €
Total	*7.130,21 €*

Table 4: Total cost of the project. Own Source

14. Conclusions

First and foremost, we have examined the various employee clocking-in methods that a company can implement. Emphasis has been placed on the clocking-in systems currently used by SEMIC, and improvements have been proposed through the implementation of a voice-based clocking-in system using a virtual assistant, concluding that the implementation of such a voice clocking-in system would benefit both the company and its employees by improving time tracking and management processes, reducing errors and time theft, and improving overall efficiency and productivity.

In the fundamentals section, we have presented the principles of artificial intelligence (AI) and its respective branches, such as machine learning (ML), deep learning (DL), and natural language processing (NLP), among others. This has allowed us to understand the fundamental principles of voice assistants and how they handle user requests.

We have also analyzed and concluded the impact of voice technology in the market, which is beginning to consolidate despite its limited utilization in the business environment. However, there is a strong commitment to enhancing voice technology, aiming for greater precision and broader, more accurate responses. It is also important to consider the emergence of new AIs like ChatGPT. It will be interesting to determine the extent to which these technologies may converge or diverge, although it is still too early to assert this with certainty.

For the development of a voice-based clocking-in system, Alexa has been chosen due to its wide range of tools it offers to developers, quick and cost-effective implementation, and the existence of a robust community of users and developers who provide feedback to improve the numerous utilities Alexa offers daily. We have understood how Alexa can identify user utterances through intents to provide the appropriate response. Additionally, we have comprehended how Alexa can differentiate between different users based on their voices in order to provide a more personalized response to each of them. As a result, we have successfully developed an Alexa skill that employs user utterances to differentiate the voices of individual workers and ascertain the speaker's identity at any given moment.

Moreover, we have expanded our knowledge of creating APIs using SOAP, databases, and programs such as Django, MySQL, PostgreSQL, XAMMP, and Postman. We have also appreciated the speed and efficiency with which an API can be deployed in the cloud through Heroku. It is worth mentioning that integrating the clocking-in API with Alexa's skill has been quite a challenge, but it has ultimately been accomplished.

Lastly, I want to highlight that despite the numerous hours invested in this project, the outcome and the effort expended have been worthwhile. I have learned a great deal about emerging technologies that, who knows, we may become accustomed to using frequently in the near future within the business environment. I only hope that the completion of this project contributes to breaking down the existing barriers between the potential offered by voice assistants like Alexa and businesses.

14.1. Future Perspectives

Although the intended functionality of the skill was completed, there are still tasks that can be improved and developed:

- The prototype of the clocking-in API will be replaced to the SEMIC clocking-in API once we solve the security concerns and issues with employee verification.
- Although we believe that voice recognition is reliable and secure, we need to assess how well Alexa can handle the request and the voice of so many different people. More testing of the *fichaje semic skill* will be conducted with additional individuals, and if necessary, another layer of security will be added through email verification or a PIN to prevent identity theft or mistaken identity.
- The skill will be submitted to Amazon for certification, ensuring that it meets all the minimum requirements established.
- For the time being, only the use of an Alexa will be necessary for clocking-in the company, if this skill is finally implemented, it would be necessary to study how many Alexa devices would be necessary for clocking-in in the company, as well as the study of the key location for this device. It is worth mentioning that there is no limit to the amount of voice recognition that an Alexa device can accept, although it will also be necessary to study how this affects the voice recognition

that Alexa offers. That is why the use of more devices by the company is not ruled out if this project comes to be carried out.

15. References

[1] "https://www.domoticada.com/alexa-google-assistant-siri-asistentes-de-voz/ [5 March 2023]."

[2] "https://www.iebschool.com/blog/futuro-asistentes-voz-business-tech/ [5 March 2023].".

[3] "https://www.econocom.es/ [6 April 2023]."

[4] "https://www.semic.es/ [6 April 2023]."

[5] "https://www.egress-sys.co.uk/news/types-of-clocking-in-systems/ [10 April 2023]."

[6] "https://es.indeed.com/orientacion-laboral/desarrollo-profesional/sistemas-fichar-trabajo [10 April 2023]."

[7] "https://www.thinking-software.com/blog/clocking-in-machines-guide [10 April 2023]."

[8] "https://www.cinconoticias.com/asistentes-de-voz/ [5 March 2023]."

[9] "https://www.researchgate.net/publication/322456429_Alexa_Siri_Cortana_and_More_An_Introduction_to_Voice_Assistants [5 March 2023]."

[10] "https://www.ibm.com/topics/artificial-intelligence [6 April 2023]."

[11] "https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico [1 March 2023]."

[12] "https://www.iberdrola.com/innovacion/deep-learning [1 March 2023}".

[13] "https://en.wikipedia.org/wiki/Knowledge_representation_and_reasoning [6 April 2023]."

[14] "https://www.ibm.com/topics/computer-vision#citation1 [28 February 2023]."

[15] "https://www.iberdrola.com/innovation/natural-language-processing-nlp [1 March 2023]."

[16] "https://www.ibm.com/topics/natural-language-processing [1 March 2023]."

[17] " https://www.t4.ai/industry/voice-assistant-market-share [6 April 2023]."

[18] "https://www.pwc.com/us/en/services/consulting/library/consumer-intelligence-series/voice-assistants.html [8 March 2023]."

[19] "(13) https://www.ine.es/prensa/tich_2022.pdf [15 March 2023]."

[20] "(14) https://www.ine.es/prensa/tich_2020.pdf [15 March 2023]."

[21] "(15) https://es.statista.com/estadisticas/1012695/asistentes-virtuales-de-voz-mas-usados-en-espana/ [15 March 2023]."

[22] "https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2022-gartner-hype-cycle [1 March 2023]."

[23] "(14) https://www.einnews.com/pr_news/618005335/at-26-2-cagr-natural-language-processing-market-size-us-74-3-billion-by-2028-says-imarc-group [6 March 2023]."

[24] "(13) https://www.marketresearchfuture.com/reports/voice-assistant-market-4003 [20 February 2023]."

[25] "(14) https://www.mordorintelligence.com/es/industry-reports/voice-assistant-application-market [20 February 2023]."

[26] "https://www.indianic.com/blog/automation/chatgpt-vs-virtual-assistant-vs-search-engines.html [6 March 2023]."

[27] "https://fortune.com/2023/03/06/microsoft-ceo-satya-nadella-virtual-assistants-dumb-as-a-rock-ai-future/ [6 March 2023]."

[28] "https://medium.com/introducci%C3%B3n-de-amazon-alexa/introducci%C3%B3n-a7c0c33f92b0#:~:text=En%20noviembre%20de%202014%2C%20Amazon,Star%20Trek%CB%90%20The%20Next%20Generation [7 March 2023]."

[29] "https://www.developer.amazon.com/es-ES/alexa [7 March 2023]."

[30] "https://www.digitaltrends.com/home/what-is-amazons-alexa-and-what-can-it-do/ [7 March 2023]."

[31] "https://www.xda-developers.com/best-alexa-skills/ [7 March 2023]."

[32] "https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html [8 March 2023]."

[33] "https://developer.amazon.com/en-GB/alexa/alexa-skills-kit [7 March 2023]."

[34] "https://developer.amazon.com/de-DE/docs/alexa/ask-overviews/voice-interaction-models.html [8 April 2023]."

[35] "https://www.developer.amazon.com/en-US/docs/alexa/ask-overviews/list-of-skills.html [10 April 2023]."

[36] "https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html#how-does-a-user-access-skill-content [13 March 2023]."

[37] "https://developer.amazon.com/es-ES/docs/alexa/ask-overviews/alexa-skills-kit-glossary.html [8 April 2023]."

[38] "https://developer.amazon.com/es-ES/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html [8 April 2023]."

[39] "https://docs.aws.amazon.com/lambda/latest/dg/welcome.html [18 April 2023]."

[40] "https://developer.amazon.com/en-US/docs/alexa/custom-skills/host-a-custom-skill-as-an-aws-lambda-function.html [18 Apil 2023]."

[41] "https://developer.amazon.com/es/blogs/alexa/post/ba9a0041-d842-4908-9f22-96737252dd3e/why-lambda-4-benefits-to-using-aws-lambda-for-your-custom-skill [18 April 2023]."

[42] "https://es.statista.com/estadisticas/1012695/asistentes-virtuales-de-voz-mas-usados-en-espana/ [24 April 2023]."

[43] "https://www.aalpha.net/articles/how-to-integrate-siri-in-third-party-apps/ [11 April 2023]."

[44] "https://www.macstories.net/stories/apple-and-the-alexa-ecosystem/ [11 April 2023]."

[45] "https://blogs.windows.com/windowsdeveloper/2017/05/10/cortana-skills-kit-empowers-developers-build-intelligent-experiences-millions-users/ [8 March 2023]."

[46] "https://searchengineland.com/microsofts-debuts-brilliant-cortana-sdk-strategy-265510 [11 April 2023]."

[47] "https://venturebeat.com/ai/what-alexa-can-and-cannot-do-on-a-pc/ [11 April 2023]."

[48] "https://developers.google.com/assistant/sdk?hl=es-419 [8 March 2023]."

[49] "https://www.techrepublic.com/article/google-assistant-the-smart-persons-guide/ [8 March 2023]."

[50] "https://www.ionos.com/digitalguide/online-marketing/web-analytics/google-home-vs-amazon-echo-a-comparative-study/ [11 April 2023]."

[51] "https://www.tomsguide.com/face-off/alexa-vs-google-assistant [11 April 2023]."

[52] "https://developer.amazon.com/en-US/docs/alexa/custom-skills/understand-name-free-interaction-for-custom-skills.html [17 May 2023]."

[53] "https://www.hellotech.com/guide/for/how-to-make-alexa-learn-your-voice [3 May 2023]."

[54] "https://developer.amazon.com/alexa/console/ask [8 May 2023]."

[55] "https://developer.amazon.com/en-US/docs/alexa/custom-skills/choose-the-invocation-name-for-a-custom-skill.html [10 May 2023]."

[56] "https://developer.amazon.com/es-ES/docs/alexa/custom-skills/configure-permissions-for-customer-information-in-your-skill.html [12 May 2023]."

[57] "https://developer.amazon.com/de-DE/docs/alexa/custom-skills/request-recognized-speaker-contact-information.html [1 June 2023]."

[58] "http://ask-sdk-node-typedoc.s3-website-us-east-1.amazonaws.com/classes/services.ups.upsserviceclient.html [1 June 2023]."

[59] "https://docs.djangoproject.com/en/4.2/ [15 May 2023]."

[60] "https://code.visualstudio.com/ [15 May 2023]."

[61] "https://www.apachefriends.org/es/index.html [15 May 2023]."

[62] "https://www.postman.com/ [2 June 2023]."

[63] "https://docs.djangoproject.com/en/4.2/topics/migrations/ [15 May 2023]."

[64] "https://docs.djangoproject.com/en/3.2/intro/tutorial02/#creating-an-admin-user [20 May 2023]."

[65] "https://blog.back4app.com/es/las-10-mejores-alternativas-de-heroku-en-2022/#Por_que_los_desarrolladores_buscan_una_alternativa_a_Heroku [5 June 2023]."

[66] "https://dashboard.heroku.com/apps [5 June 2023]."

[67] "https://devcenter.heroku.com/articles/heroku-cli#install-the-heroku-cli [5 June 2023]."

[68] "https://www.youtube.com/watch?v=Jb1NcPdN34k&t=49s&ab_channel=AgustinNavarroGaldon [5 June 2023]."

[69] "https://www.youtube.com/watch?v=5d8AQFF0Ot0&t=466s&ab_channel=CodeAura [5 June 2023]."