UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF A MOBILE NUTRITIONAL ASSISTANT FOR SUPERMARKETS WITH THE TECHNOLOGY DIALOGFLOW

LUIS ALFONSO CABALLERO HERNÁEZ

JUNIO 2019

TRABAJO DE FIN DE GRADO

Título:	Desarrollo de un Asistente Nutricional Móvil para Supermercados con la tecnología DialogFlow
Título (inglés):	Development of a Mobile Nutritional Assistant for Supermarkets with the technology DialogFlow
Autor:	Luis Alfonso Caballero Hernáez
Tutor:	Carlos Ángel Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF A MOBILE NUTRITIONAL ASSISTANT FOR SUPERMARKETS WITH THE TECHNOLOGY DIALOGFLOW

Luis Alfonso Caballero Hernáez

Junio 2019

Resumen

El uso de Internet como medio de búsqueda de información se ha extendido hasta el punto de que las personas han reemplazado la enciclopedia tradicional por la utilización de aplicaciones o Google. Esto se debe a la facilidad y velocidad que ofrece Internet. Es mucho más fácil encontrar lo que se busca escribiéndolo en Internet que buscándolo en un libro. Internet da al usuario la capacidad de escribir las palabras específicas para encontrar las respuestas deseadas.

Este hecho no se escapa de la vista de los negocios. En un mundo muy afectado por la crisis económica, el universo empresarial busca maneras de gestionar los recursos reduciendo costes y ofreciendo mejores servicios. La solución está en la informatización y la robotización.

La robotización está a la orden del día. Cada día se impone su avance en el mundo virtual y su presencia destruirá millones de empleos aunque no hay motivo por el que alarmarse ya que a su vez generará muchos otros.

Se ha vuelto común encontrar chatbots en varias páginas web. Son una forma sencilla, rápida y barata de atender a los usuarios y están orientados hacia cualquier tema, desde la atención pública hasta los asistentes de compras. El único riesgo que corren es el de no poder satisfacer las necesidades del cliente y el de mostrar una imagen muy poco personal y cercana. Este TFG esta orientado en esa línea, donde se tratará de corregir ese riesgo.

El objetivo de este proyecto es programar un chatbot en una aplicación de Android que ayuda y asesora al usuario sobre qué alimentos son mejores para ellos. Para llegar hasta ahí, este proyecto se ha dividido en dos partes. Primero, analizar las formas en que se pueden programar los chatbots. Segundo, integrar la solución adoptada a una aplicación Android de una forma visual y atractiva para el usuario.

Keywords: Chatbot, bot, asistente virtual, nutrición, compra, supermercado, aplicación android

Abstract

The use of the internet as a means of information search has been spread until the point that people have replaced traditional encyclopedia by applications or Google. This is due to the ease and speed that the Internet offers. It is much easier to find what you are looking for by typing it on the Internet than looking for it inside a book. The internet gives the user the capacity to write the specific words to find the answer.

This insight does not escape from the view of business. In a world hugely hit by the economic crisis, business universe looks for strategic ways to spend less money offering better services. The solution is in computerization and robotization.

Robotization is the order of the day. At every moment it is imposing around the virtual world and its presence will destroy millions of jobs, but there is no reason to be alarmed because, in turn, it will generate many more.

It has become usual to find chatbots in multiple web pages. They are a simple, fast and cheap way to attend the users, and they are orientated to any kind of subject, from public attention to shopping assistants. The main risk here is to not satisfy the user's needs and to show an impersonal and far away image. And in this line this project addresses to eliminate or at least mitigate this risk.

The objective of this project is to program a chatbot in an Android application who helps and advises the users of what food is better for them. To get there, this project has been divided in two parts. First one, analyze the different ways chatbots can be programmed. Second, integrate the adopted solution into a Android application with an attractive and visual interface.

Keywords: Chatbot, bot, virtual assistant, nutrition, shopping, supermarket, android application

Agradecimientos

A mis padres, mis hermanos y mi familia, por haber sido el mayor apoyo durante mi etapa académica, porque siempre confiaron en mi y sin ellos no lo hubiese conseguido.

A mis amigos, por ser la fuente de desahogo y descanso que una ingeniería requiere, porque sin sus ánimos no habría sido posible.

A todos los profesores y profesoras de la politécnica que durante estos años han influído en mi aprendizaje.

A la Universidad Politécnica de Madrid en general, que apuesta por sus alumnos.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

Nutrition is becoming important among the population. In actual society food products are loosing their own natural side, and we can find more variety. The importance given to healthy eating is getting higher. This has been caused by multiple factors.

First of all, day to day we can see more and more natural products out of their natural season. It is not rare to see strawberries in summer or watermelon in winter. Even the outer side of the natural products looks brighter and colorful. The use of preservatives, dyes and pesticides is a common practice in farming. For example, the use of nitrates in meats and other products can preserve food while harming health since they are carcinogenic [5]. People are taking care about the use of these artificial products that in most of the cases can produce health problems.

Secondly, with the offer increase and food products diversity, consumers may be confused about which foods to choose. Now you can see similar products whith different composition, from gluten free breads to milk without lactose. In Spain, 25% of population suffers from any food allergy or intolerance, and the most usual issues are fructose, gluten and lactose intolerance [6]. The ignorance of what is being bought and consumed is growing.

In third place, in a society with more than 52% of overweight population in Spain, concern about healthy food is becoming a necessity, and this is not a unique problem of Spain, it also extends to the rest of Europe and many parts of the world. In the next figure we can see the obesity distribution by autonomous communities.

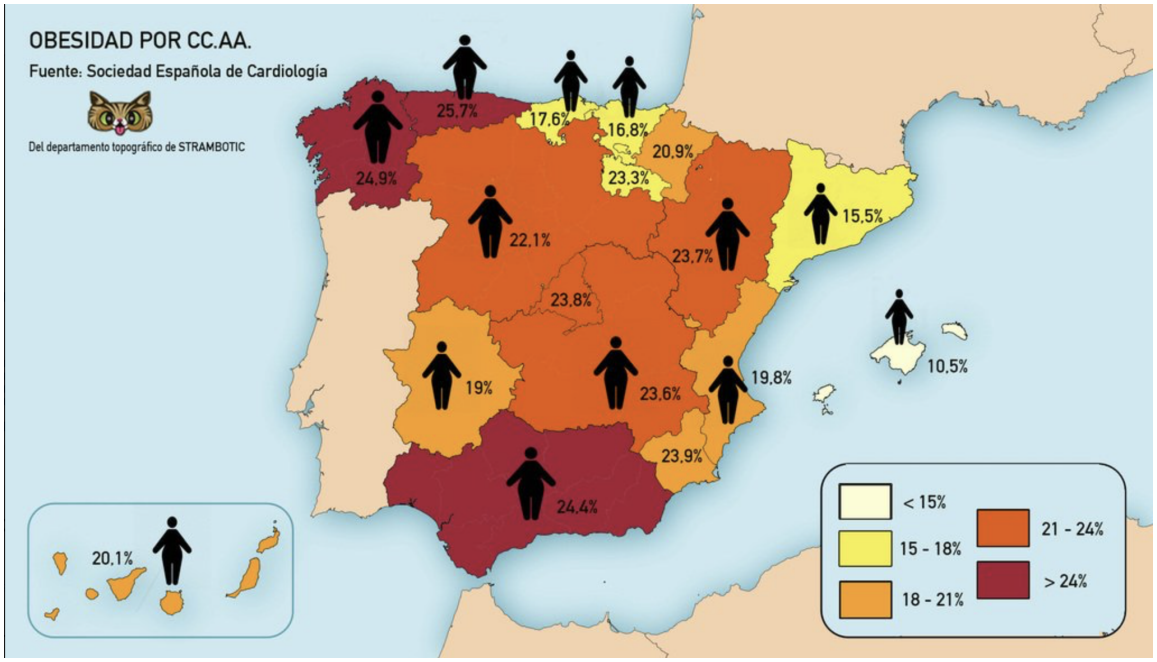

Figure 1.1: Obesity distribution in Spain [1]

By last, people are looking for new ways of feeding. In Spain, 7.8% of people are declared vegetarian and 30-40% of them are vegan [7]. But this is not the only reason why people look for specific ingredients. Alimentary intolerance are like allergies that force people to look for new ways of feeding. In some places this is not easy and people have to be very careful with what they buy. This is why people have to see all the ingredients tags of each product to know what they are eating.

People's awareness about healthy food and lifestyle is a fact, and we can see it in the social networks. In Instagram, Youtube or any other platforms we see referring people that carry out this life form and they are followed by millions of people. This image showed by these 'instagramers' is desired by many people and they find in an application the way to execute this lifestyle.

With the higher use that people make of new technologies, the first way to see all this information about food products is in their own smartphones. Everybody uses them to access all kind of information in just ten seconds. You have many options, from going to internet and search what you need to know to downloading an app about it.

Nowadays every application programmers look for simplicity and effectiveness. The success of applications is achieved with friendly and easy to use interfaces and not with complicated and convoluted apps. The use of chatbots is a frequent trend in many web pages and applications. This is due to the facilities that a chatbot provides users, who are the ones that need to have the information about something in a easy and quickly way. Just typing or saying their intents, users can get the information that they want avoiding waiting long queues and searches by the employees.

Internet has brought the products and services directly to the home, and this produces that the number of users increases exponentially. Businesses can not manage the high number of petitions, and chatbot applications are the perfect solutions to this problem.

It has been calculated that robotization will destroy more than 75 millions jobs at the same time that it will produce more than 133 million new jobs. Machines will take charge of the 52% job positions. Right now 30% of activity is already automated. In this stage is where chatbots take place [8].

1.2 Project goals

After have analyzed the current situation about people's concerns and needs around the nutrition, and the position towards which applications are turning, we realize that the way to solve the problems of the actual situation starts at focusing in this sector, the use of internet and smartphones to change the way they get informed about the food products, and the best place to start is from the supermarket itself.

Having said that, the first objective of this project is to develop an Android application that provides users an easy way to access to food products information and to get recommendations about healthy food. It should be not only a home use application, but also a shopping use application. The steps to get there are the following:

- Overview about people eating habits. As seen in the previous section, to make this application it is necesary to know what are the population's habits and the use they make of Internet.
- Study of supply and demand in this sector. Day to day nutritional applications are rising in number. As we said, people concerning about healthy lifestyles is increasing. To be competitive we have to know in which the sector moves.
- Analyze and deepen the use of chatbot applications in our society. After seen the

applications, we will analyze the virtual assistants used on them.

- Develop an Android application implemented with a trained chatbot to recommend users about food. Once we are clear about the previous objectives we can face the beginnig of the application.
- Thinking about the different technologies that there are to make a powerful app, analyzing if we take existing data from internet or if we make a new data center with our own data. The main disadvantage of this option is that it is necessary a long time to get a complete system full of information.
- Validate and review the correct functioning of the app. When the application is finished, tests have to be done to verify the working of the application. The application will be given to close people in order to have users testing the app and providing feedback.
- Future work. After the first phase of the project there must be a back view to perfect and improve not only the application but also the entire project and see if objectives are met.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in the document. The structure is as follows:

- ***Chapter 1:*** Overview of the importance and the necessity of this project, analyzing the current situation. The main objectives will be presented.
- ***Chapter 2:*** Here we expose the different chatbots technologies and alternatives for mobile applications and programming such as databases and scanning code ways.
- ***Chapter 3:*** Explanation about the architecture and design of the Android application and its integration with DialogFlow, databases and barcode scanning.
- ***Chapter 4:*** Use case description about one normal use of the application.
- ***Chapter 5*** Here we find a resume of the conclusions of this project and the next steps to continue.
- ***Appendix A*** This appendix reflects, quantitatively or qualitatively, on the possible impact of this project.

- ***Appendix B*** This appendix details an adequate budget to bring about the project.

CHAPTER 2

Enabling Technologies

2.1 Analysis and annotation

Nowadays [**?**] technology is a powerful way to access all kind of information. Internet is reaching everywhere and everyday billions of people use their smartphones to get the answer to many questions. In recent years, in Spain, the number of users with smart phone's applications has risen until the 27,7 million of users [9]. In food and nutrition the scene does not change and when people go to the supermarket happens the same. Why should this scenario be different?

You can find from applications that show you the ingredients' information of a product to applications that make a meal calendar for loosing weight. Now is common to see in all these applications chatbots that make easier to get information and reduce the cost of employees salaries, replacing them by bots.

Let's make an analysis about the different options we can find in the nutritional world and what these applications can do for consumers.

2.2 Nutritional applications

In the different applications stores we can find a wide variety of apps related to nutrition and food. Analyzing them, we can see that there are two big groups so here we have to distinguish between these modalities:

- **Applications oriented to physical activity and change of habits:**
  - MyFitnessPal: This application monitors about calories. It provides different exercises and diets in function of users objectives. You can also scan barcodes of foods to have information about them [10].
  - Plixi: Oriented to nutrition profesionals, this app calculates body fat with an awesome visual interface [11].
  - Fabulous: Motivate me: For people who are looking for a change in their habits and life style. This app is oriented to motivate users offering ways to avoid bad manners and to convert you in a healthy person [12].

- **Applications oriented to consumer and dietary:**
  - Vegaffinity: This app is a great help for vegan and vegetarian community. It offers recipes and foods to alleviate the deficiencies that a vegan or vegetarian diet can cause a person [13].
  - Healthifyme: With this app you can mark your goals with the help of trainers that a part of offering diets, they offer a yoga guide [14].
  - Manzana Roja: You can make your own diet with this app. It collects information about user characteristics to make a customized diet [15].

As days go by, this market is still growing up. New apps are combining both diet and exercises to make a complete system of prevention and healthy lifestyle. Depending on the platform, many of these apps are programmed in Swift and others in Android (Java). For our application we are going to use Java.

2.3 Chatbot technologies

As commented previously, chatbots are an economic alternative for companies. They allow saving staff costs and offer users a better services. Nowadays many applications includes

this technology, not only for salary savings, but also because it is a way to make apps easier to use and accessible.

In chatbots world we can find different alternatives. You can program your own chatbot, but now we can find multiple ways to make advanced chatbots with very litter complexity. Here we have some of them:

- Chatfuel: Chatbot generator for Facebook Messenger and Telegram. You need to be a Facebook member. This chatbot is very simple to use [16].
- Botsify: This chatbot has been designed by Facebook users. It allows users to make a list with all the services that can be done for you. You can also add images to make this chatbot more attractive and easier [17].
- Facebook Messenger: This alternative is a little bit more advanced than the others because some of the functions must be configured by the user [18].
- Beep Boop: Hosting platform for developers who want to create their own chatbot for Facebook Messenger or Slack [19].
- Telegram Bots: If Telegram is famous for something, it is because of its chatbots. Telegram made public its chatbot code API. From that moment users have created many different chatbots [20].
- DialogFlow: This is a Google technology chatbot. It is very simple to use, and it is based in entities and intents. Later we will make an overview about this chatbot to learn about it with more detail [2].

This is the chatbot we have chosen to make this app because it has an intuitive interface that makes the work more entertaining and enjoyable.

Now that we have taken view at the different alternatives, we are going to deepen in the technologies we are going to use for this project.

2.4 Technologies used

2.4.1 Introduction

In this chapter the technologies to be used for the development of the application will be detailed.

First of all, we will look inside DialogFlow to discover its capabilities and functioning. We will see the Google Console for DialogFlow and all the tools we can find there.

Finally we will take a look into our Android Studio Project, paying attention in the libraries and tools used.

2.4.2 DialogFlow

DialogFlow is a Google virtual assistant with whom you can create complete and intuitive conversational flows. It's based in entities and intents that allows users to establish basic conversational rules to simulate real scenarios. A simple example would the following: A user wants to know where to eat (intent), chatbot responses with a restaurant (context). After this brief introduction we will pass to see its operation in more detail.

Figure 2.1: DialogFlow Logo

Components

- **Intents**: We can define the intents, as its own name says, as the user's intention. To create seamlessly a complete cahtbot, and we need to define a high number if intents, so that all the possible questions of the user will be answered. If user intents are not clear, the chatbot should have defined answers for that situation.

  An ideal chatbot is that one which is able to answer all the user intents avoiding the feeling of being talking with a machine. User should be wondering if there is a robot in the other side of the screen or it is a real person, or he can ask directly the chatbot.

  DialogFlow console offers the possibility to add multiple user sentences for one intent, in order to cover all possibilities. After that, you have to define the chatbot responses. You can write several answers for the same intent and the chatbot will take one of

them randomly.

The last component of intents are the actions and parameters. They are used to extracting relevant information for the user in order to make the chatbot know the specific intent of the user. The action can be used to identify the conversation flow to make later, in Android Studio, different intents depending on the action established.

For example, if the user wants to eat something, the chatbot has to know what kind of food the user wants, so 'kind of food' would be a parameter.

- **Entities**: Entities are made to identify groups of words to make easier the way to understand the intents.

  Inside an entity you can make synonyms for the same word in order to complete the chatbot vocabulary and for a better understanding of the user's intent. In the DialogFlow console the group of synonym is called entity type, and each synonym is an entity entry. You can add as many synonyms as you want. The more synonyms there are, the better will work the chatbot. The key is to cover the greatest possibilities.

  Continuing with the previous example, an entity could be 'kind of food', and inside that entity we could include vegetables and meat, and inside vegetables we could add multiple synonyms as garlic, tomatoes, pumpkins, etc.

  DialogFlow contains multiple system entities inside its structure. A few examples are locations, dates, times, etc. These entities are already provided and user has to do nothing with them, they are automatically detected.

- **Contexts**: One context is the status point of a user intention. In the DialogFlow console context are used to take the user from one intention to another. This is the way DialogFlow has to continue in a certain flow so chatbot can respond not only one user intent, but also recreate a conversation with more than one inter locution.

  For example, if one user wants to go to the cinema, he would ask the chatbot 'I want to go to the cinema'. The intent of that context could be called 'Want cinema', for example. Chatbot would ask next 'Which cinema do you want to go' so it would take the user to a new context 'cinema selection'.

  Anyway, for this situation, you could program your chatbot to ask for a parameter in the first intention, 'cinema' so there would be no need to pass from one intent to another. This shows that DialogFlow is very flexible to do many things in various ways.

- **Events**: Events are used to auto-complete one intent with data from Google. For example if you want to add a joke answer to your chatbot you could include the event

Get Joke.

- **Fulfillment**: This interesting component of DialogFlow serves to make dynamic responses. Some example objectives in the use of the fulfillment are to get information from a database, implement rules or place ordered lists.
- **Training and Analytics**: This feature helps you to get real world data in order to add more information to the given responses.
- **Client Access Token**: This token is used to identify the agent. Every Agent of every DialogFlow account has one token associated. This token is later used to make integrations with Android Studio in order to connect with the app.

After seeing the components we are able to understand the DialogFlow basic working. In the next figure the basic structural flow of DialogFlow is shown.

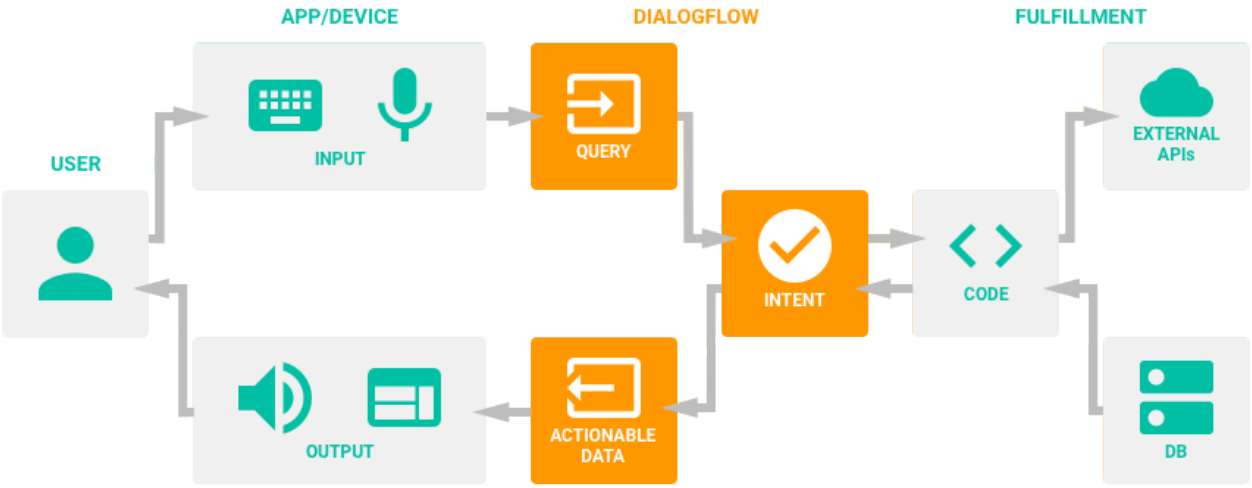

Figure 2.2: DialogFlow Structure [2]

Let's make a quick summary of the previous figure. A user interact to speak with the DialogFlow chatbot. The chatbot starts to listen and user uses the input devices to transmit the voice or text message. After indentifying the query, the intent is understood by the chatbot so it rescues the necessary information to make the response from the fulfillment databases and APIs. Finally, chatbot produces the response, that is received by the user through the output devices.

2.4.3 Android

Once we have seen DialogFlow operation we pass to the application programming analysis.

In 2017 the most used mobile Operating System was Android with 84.5% of market share, followed by iOS with 14.5%. To understand these differences between one operating system and another we have to focus in Samsung. The South Korean giant is the cause of this consequence and it is because its mobile phones have been able to compete in the glory moment of Apple. From 2010, when Samsung Galaxy S was the second most sold phone in the world behind the first iPhone, to 2014, Samsung devices have been the most sold phones in market. Here we see the evolution, expressed in millions:

OS	2012	2015	2019
Android	69.7	80.7	90
IOs	20.9	17.7	11
Symbian	1.2	0.5	0
RIM	3.5	0.5	0
Microsoft	3	1.1	0

Table 2.1: Main Smartphone IOs Evolution [3]

As we can see, the secondary OS are disappearing in benefit of Android. Even iOS is going down year by year. More than 1300 producers make Android devices with over than 24 thousand different models, Android is unstoppable.

This fact does not escape from our sight and it is the reason why Android is going to be the chosen programming language for this project. In this way we will reach more users. In addition, the subsequent adaptation to other platforms is not complicated.

Java and XML

Android is mainly based in Java. This object-oriented programming language developed by Sun Microsystems is one of the most demanded languages. One of the reasons is because once the program is created it can be running on any platform. It is also a simply language and is very similar to other languages as C or C++, multiple sentences are common for all of them. Other well known Java characteristics are that is a dynamic language, strong, safe, strongly typed and with a high performance, with multiple programs to execute it.

On the other side we have XML. This mark language based on labels runs in parallel with Java in Android applications and is used in the graphic interfaces of the Android

Figure 2.3: Android Studio Languages

applications.

There is an indicator of the popularity of programming languages made by TIOBE, who determines month by month, the most popular programming language and compares the result with the same month of the past year. These are the results of May 2019:

May 2019	May 2018	Programming Language	Ratings	Change
1	1	Java	16%	-0.38%
2	2	C	14%	+0.24%
3	3	C++	8%	+0.43%
4	4	Python	7.8%	+2.64%
5	6	Visual Basic .NET	5.2%	+1.07%
6	5	C#	4%	-0.42%

Table 2.2: Most Popular Programming Languages [4]

The table shows that Java and C are the most popular language programming, fighting for the first place every month. This aspect further reinforces the fact that Android has been chosen as the development environment.

Android Studio

The best advantage of using Java and XML to develop an Android app is Android Studio.

This very fast-compilation Google program offers a high variety of tools with which create an app is converted in a dynamic and fluid task. In fact, Android Studio has replaced Eclipse as official Integrated Development Environment (IDE) and it has been made specifically for Android applications' development.

We can find both Java and XML included on a native way on Android Studio. Now it is even possible to use Kotlin language in Android Studio to develop applications, and you can convert your Java applications to Kotlin and vice versa, but in this project we are focused on Java and XML. Both languages have different functions on Android Studio.

XML is part of the project design and it is used to declare layouts. But in Android Studio Layouts are not only xml files, but also have a visual interface in where you can place buttons, textViews, EditText or many other components in the screen of your applications, and the xml code will be automatically written. These quick and friendly features are gaining importance in Android Studio everyday.

On the other hand, Java provides the logic of the app. Both are connected, Activities are programmed in Java, where we create the classes, methods and functions that use as objects the different components of the application, identified by labels written (or placed) in the XML layouts.

As I said before, Android Studio has been improved a lot, and now it is not necessary to chop all the code to create the different components of the app as buttons, texts or images. Now you just have to drag them from a menu and place them where you want. This makes the process of creating an app much more agile. In the latest version of Android Studio it has been included the navigation editor, where you can see a graphic which represent all the different activities of your app and their internal connections. This new feature allows to see all the application flow from one window, which is very interesting. This last characteristic reminds to Swift code in Xcode, the IOS language IDE where you can program the activity change and flow from an editor like this.

Regarding the simulation of the applications, Android Studio offers the Android Virtual Device Manager (AVD).

This tool is used to generate a simulator, with all the different characteristics of an Android device, to execute any application avoiding the need of connecting a real device to run our apps. Here you can also modify the existing devices in case you have already created one. These virtual devices can be whatever is available on the market, from tablets to watches.

Now we are going to define some basic components for the functionality of an Android

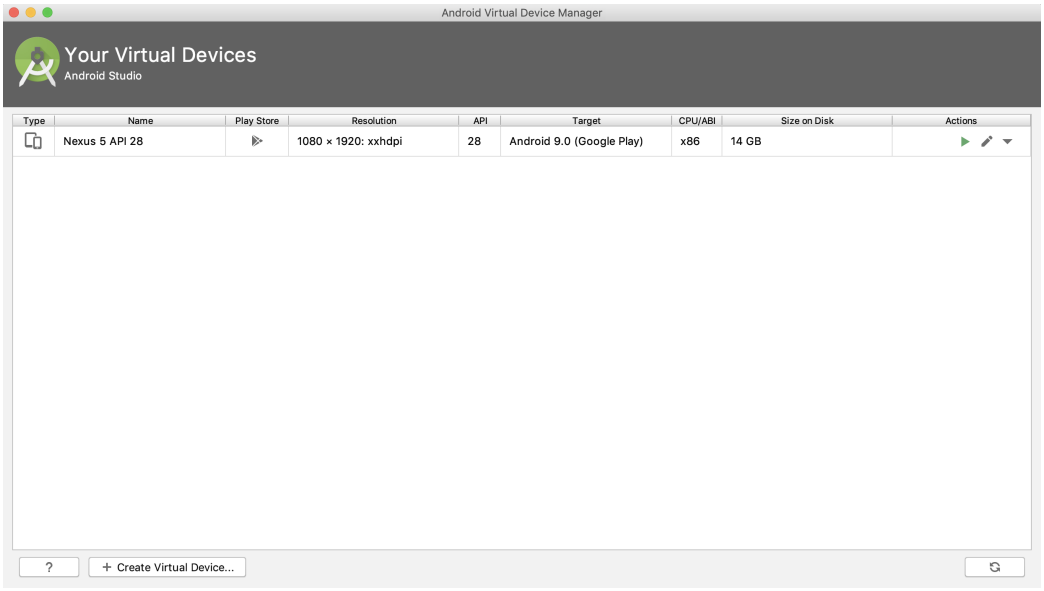

Figure 2.4: AVD Manager

application:

- Android Manifest: This XML file is where we describe the components of our application such as the activities, services, permissions, etc. It is compulsory to have this file in our application root directory for a correct compilation. For this project we will have to add some permissions such as internet, camera and recording audio access, write external storage permissions or read external storage permissions, and here is where all these permissions have to be added.

- build.gradle: This script file defines the compiling configurations. It is based in Groovy and DSL. Here we add all the dependencies and libraries that we need for our app. Later in a couple of sections we are going to see all the dependencies used for this project.

2.4.4 Firebase

It has been a difficult decision to choose which database to use. First of all we though about using an existing database from which read all the data. But then we started to think about making our own database with the data we want to have. As we said before, in the beggining this database will be half empty and it will be increasing day by day. We have also made the option to allow the users to add new elements to the database in order to create a strongly structured database.

Once we decided to make our own database we tried to decide which options were better. First we made a local database, but this was not the best idea if we wanted to make a powerful application. So finally we discovered Firebase.

Firebase [21] is a real time database developed by Google. Actually it is more than just a database. With this tool you can make an entire development platform to develop your applications.

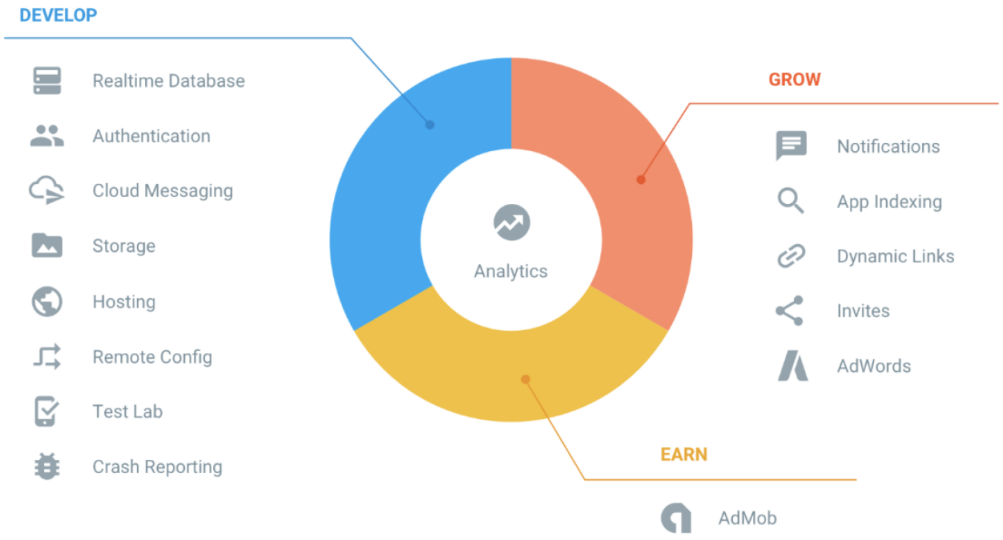

Figure 2.5: Firebase elements

The list of things you can do with Firebase is very large. Its main characteristics are the following:

- Analitics: Free app that provides app usage and monitoring.
- Development: There are too many alternatives inside development characteristic. The most remarkable is the real time database, web hosting or cloud firestore.
- Stability: Very interesting characteristic which consists in the Test Lab, where you can test your iOS and Android applications.
- Earn: With Google Admob you have marketing scope.

For our application we are going to use the Realtime Database. Later, in the architecture section, we will see Firebase with more detail.

2.4.5 Libraries

In this section we are going to explain all the libraries used for this app. To add these libraries you have to open the build.gradle file of our Android project and add them into the dependencies section. The way to add a library inside the dependencies of the build.gradle is as follows:

"implementation 'library.name'

So here is where all the libraries are included. Every Android applications have some default libraries, but you have to add some more to increase the functionality of your app. Most of the libraries are based in APIs, application programming interfaces, so they include many subroutines, methods, communication protocols and others.

Figure 2.6: Libraries

Support Libraries

- Android AppCompat Library: This library is added to use APIs from older platforms that are not available. It is also useful to add APIs that are not part of the framework APIs.
- Android Constraint Layout Solver Library: A solver for Constraint Layout that allows you to position and size all components however you want inside the layout.
- Design Library: Provides APIs to support adding material design to an app.

- Annotation Libraries: This library allows users to include annotation metadata.
- v13 Library: Add support for the Fragment user interface pattern, which represents the behavior of an Activity inside the application.

Inlcuded Libraries

For this project it has been necessary to define libraries for adding some new features to the application. Now I am going to enumerate them and make a short description about them, and then, in the architecture epigraph, a more detailed view will be seen.

- API.AI SDK: We need to add this library for speech recognition and natural language processing. It is a basic pillar for the integration with DialogFlow. The way to include it into our project is adding the following line in the dependencies of the build.gradle: implementation 'ai.api:sdk:2.0.7@aar'
- API.AI Libai: This library is necessary for the class that implements the AIListener interface, which process responses from the API.AI. It is also important for the correct working of DialogFlow in our application.
- Code GSON: It's a dependency that we have to add for the integration between DialogFlow and our Android app and is basically used to convert from Java to JSON.
- Commons IO: As it happens to the previous library, commons has to be added for the integration with the chatbot.
- Zxing: For barcode scanning we have used the Zebra Crossing library, that allows a device with a built-in camera to scan barcodes or QR codes. We use this library to allow the barcode scanning in our app in order to read the food products.

In android there are countless libraries that can be added. We could mention many more but the list is very extensive. The ones we have mentioned are the most remarkable libraries, but they are not all that we have used during the development of the application.

CHAPTER 3

Architecture

3.1 Introduction

In this chapter we will cover the design phase of this project, as well as the implementation details related to its architecture.

Firstly we will present an overview of the project with all the components that are part of the development, and secondly we will show the every component, and we will describe them one by one and in much more depth. This is intended to offer the reader a general view of this project and its architecture.

3.2 Architecture

This application is supported by three basic pillars. In the first pillar we have the application, programmed in Android Studio with Java and XML. This pillar is directly connected with the second one, DialogFlow. The application has communication with our third pillar, a Firebase database so when the user wants to know the information of a product he asks for it to the chatbot and, scanning the barcode, you obtain the information of the product.

These three pillars are the 'front-end' part of the applications. In the 'back-end' part we include all the libraries and dependencies included in the application, mentioning separately the Zebra Crossing library. With this library it is possible to develop the barcode and QR scanner inside the application. In the next figure we can see the architecture diagram.

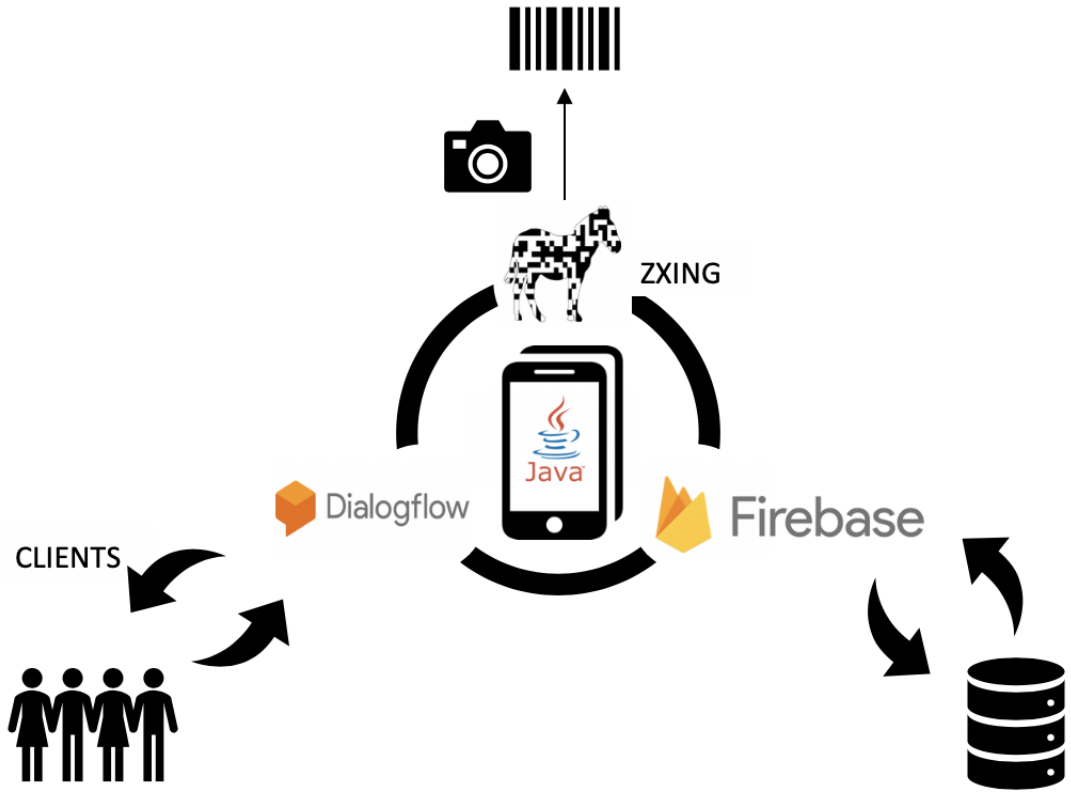

Figure 3.1: Architecture

In summary, we have an Android application programmed using Android Studio integrated with DialogFlow and communicated with Firebase which uses, among other libraries, Zxing to make the barcode scanning.

3.3 Overview

Now we are going to show and explain in detail all the modules involved in our app, and we will see their basic functioning, how they appear and the actions to communicate between

them.

It is possible to define three different levels. First of all we have the external modules. Here we include Firebase database. Here we will have the data from all the food products with their associated barcodes. Secondly we can define the intermediates modules. DialogFlow can be considered an intermediate module because in this project we have done many actions with it inside the application development and is the cohesion between the internal and external modules. Finally, we have the internal modules with our Android Application, developed in Android Studio with the Java language and all the libraries used.

3.3.1 External Modules

As we commented in the enabling technologies section, Firebase is a new development platform powered by Google that allows to develop multiplatform applications. For this project we have been interested in the real time database provided in Firebase. Thanks to this database, changes can be visualized live in the applications.

3.3.1.1 Firebase Connection

The integration of Firebase database with the Android Application it is not difficult, and it can be done in few steps. Now we are going to define those steps.

First of all, it is necessary to create a new project inside Firebase. In this project is where we are going to generate our own database with all the food products. After creating the project we have to connect it with our Android Studio project. Firebase has an automated way to connect both projects in which we have to follow the steps provided in the Firebase Assistant inside Android Studio. This is the fastest way to make the integration, but not the recommended one.

The non-automated form is the one we have selected for this project. After creating the Firebase project we have to provide our application ID, which is the package name. If we do not remember it we can find it in the app/build.gradle. In this case the project name is 'es.upm.dit.feedmefy'. Once we have given the ID we click in Register App in the Firebase console.

After doing that, Firebase will produce the configuration file for Android. Its name is google-services.json. We have to download it and add it to the project path into the module-app level. In the project build.gradle we have to make sure that we have the Maven repository, and we add the following line code for the Google Services plugin: classpath

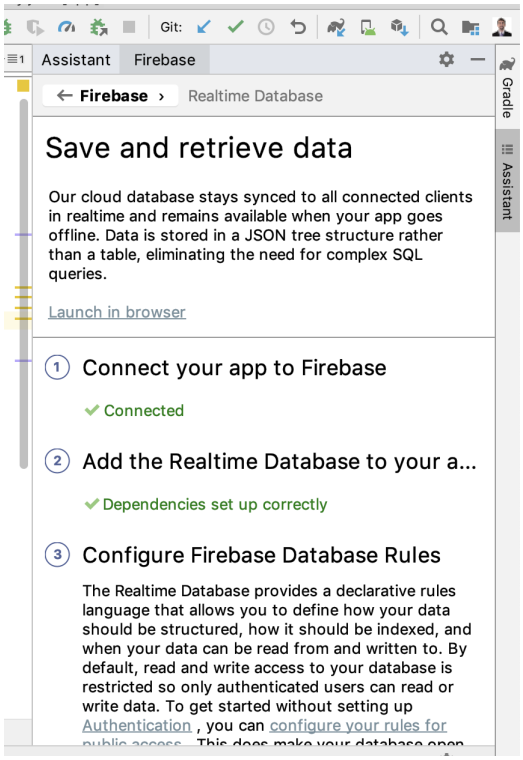

Figure 3.2: Firebase Assistant

'com.google.gms:google-services:4.2.0'

Now in the app build.gradle we have to add the Google Play services Gradle plugin. Here we also have to add the dependencies. The main dependence is com.google.firebase:firebase-core. It is possible to add more dependencies to gain more functionalities, but it is not necessary. After this step we have to sync the application. There is so many libraries that we can add in the dependencies. In the next table we show the main libraries.

With Firebase is possible to check if the database is connected with the Android Studio application and from Android Studio we can do the same with the Firebase Assistant that we have showed before.

Finally, we have to mention that it is necessary to have internet access in our application. The way to do this is by adding the following line in the Android Manifest:

```
<uses-permission android:name="android.permission.INTERNET" />
```

We can connect with the database without this line, but then we will not be able to read or save data from our application.

Service	Dependence
AdMob	com.google.firebase:firebase-ads:17.2.0
Analytics	com.google.firebase:firebase-core:16.0.8
Authentication	com.google.firebase:firebase-auth:16.2.0
Cloud Storage	com.google.firebase:firebase-storage:16.1.0
Cloud Firestore	com.google.firebase:firebase-firestore:18.1.0

Table 3.1: Android Libraries for Firebase products [**?**]

3.3.1.2 Database Reference

To initialize the Firebase database connection and the database query from the Android activity we have to create the database reference and then a child. Child can be defined as the directory who points to the database reference and save all the information. It is the JSON key where everything is added. This is the code for doing that.

```
myDatabaseReference = FirebaseDatabase.getInstance().getReference();
myChild = myDatabaseReference.child(paso_bar);
```

In the first line we create a reference of the database. We could make a reference to something inside the structrure of the database if we add the name of it. With the second one we can point to the desired part of the inner structure of the database.

To start the real time database we have to make a listener over the child inside the onStart method. With this listener we will take data from the external database live. The way to include this feature is the following:

```
myChild.addValueEventListener(new ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

String texto = dataSnapshot.getValue().toString();
textoFire.setText(texto);
}
});
```

3.3.1.3 Save and retrieve data

In this project we have developed both actions. We can read data from the database, and we can write on it. We have decided to give the users the possibility of enriching the database by adding new food products. This will be showed in the selected use case.

First we are going to see how we can read data from the database. The first step is to take our database reference and create a child pointing to the scanned barcode. Once we are pointing to the desired barcode we can read the information of that product. In the following example we show the name of the product and some information about it.

```
myDatabaseReference.child(paso_bar).addListenerForSingleValueEvent(new
    ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if(dataSnapshot.exists()){
String alimentos = dataSnapshot.child("alimento").getValue(String.class);
System.out.println(alimentos);
alimento.setText("Alimento: " +alimentos);
String informaciones = dataSnapshot.child("descripcion").getValue().toString
    ();
System.out.println(informaciones);
informacion.setText("Informacion: " +informaciones);
}
}
});
```

As we can see, we listen for value events thanks to the ValueEventListener. In this case we read the value once, and then we remove it because we do not expect any change in the value. This listener will receive a data snapshot with the data at the specific location. After reading the data we set the TextView text that will show the information in the screen, and we write on it this data.

Now we are going to see how to write data. First of all the users will have to fill the information with the product data in all the EditText provided on screen. Then, with that information, they click the 'add' button and the product will be uploaded. In the following code we provide the OnClick method, that is called when the mentioned add button is clicked:

```
add.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
String code = codigo.getText().toString();
String name = nombre.getText().toString();
String info = informacion.getText().toString();
if(!TextUtils.isEmpty(code)){
if(!TextUtils.isEmpty(name)){
if(!TextUtils.isEmpty(info)){
Barcode barcode = new Barcode(name, info);
bbdd.child("BARCODE").child(code).setValue(barcode);
Toast.makeText(ContribuirActivity.this, "Alimento anadido", Toast.
    LENGTH_LONG).show();
}
else Toast.makeText(ContribuirActivity.this, "Debes anadir informacion",
    Toast.LENGTH_LONG).show();
}
else Toast.makeText(ContribuirActivity.this, "Debes anadir un nombre", Toast
    .LENGTH_LONG).show();
}
else Toast.makeText(ContribuirActivity.this, "Debes anadir un codigo de
    barras", Toast.LENGTH_LONG).show();
}
});
```

For this example we just take the information and the name from the product. First we take the text from the EditText. A check is made that all fields have been filled in and then we create the Barcode object and upload it with the setValue method.

3.3.1.4 Database structure

Finally, we are going to see the internal structure of the database. It is classified by the barcode numbers. Every element of the database is referenced by this number and inside it we find the different sub-elements that defines each product. Here there is an example of the JSON code of one product:

```
"87157260" : {
"NotaFeed" : "D",
"alimento" : "ketchup",
"descripcion" : "ketchup Heinz. 0.1 g Grasas en cantidad baja 0 g Acidos
    grasos saturados en cantidad baja 22.8 g Azucares en cantidad elevada
    1.8 g Sal en cantidad elevada",
```

```
"marca" : "Heinz"
}
```

As we can see, the elements that compose one food product, identified by its barcode, are the following: 'NotaFeed', which shows an evaluation about the product where A is the highest qualification and E the worst one, 'alimento' which shows the kind of product, 'description' where we can see the nutritional information about the product and 'marca' which shows the product brand.

3.3.2 Intermediate Modules

As we said before, in this modules' classification we classify DialogFlow [2] as an intermediate module. We are going to see how it is integrated in the application, some example intents used in the application and the actions' differentiation.

3.3.2.1 DialogFlow integration

First of all, we have to say that 'Feedme' has been the selected name for the chatbot. We have decided to implement the chatbot through verbal communication because it is more intuitive and faster than writing. Because of that it has been necessary to give recording audio permissions to the application. But this is not the only permission we have to add to integrate DialogFlow. It is an external module and it is necessary to have an Internet conexion, so we have given the application internet access permissions to connect with DialogFlow. These have been include in the Android Manifest thanks to these two lines:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
```

As we said before in the second chapter, it is necessary to add some libraries to implement speech recognition and natural language processing features in the app. This library is the API.AI SDK library. We have added it in the build.gradle app. After that our app is ready to use the speech recognition.

Now we are going to take a look to the Java class. To make the integration with DialogFlow the class needs to implement AIListener in order to process information from API.AI. We have created an instance of AIConfiguration and have written our own token from DialogFlow. Here we also select the language:

```
final AIConfiguration config = new AIConfiguration("0
    c63e74fffaa4d8ba0fc90f0415b0094",
AIConfiguration.SupportedLanguages.Spanish,
AIConfiguration.RecognitionEngine.System);
aiService = AIService.getService(this, config);
aiService.setListener(this);
```

The aiService object, created at the beginning of the class, is used to get the voice of the user from the microphone. It is configured, among other things, to be activated when user pushes the button:

```
findViewById(R.id.empezarvoz_button).setOnClickListener(new View.
    OnClickListener() {
@Override
public void onClick(View v){
aiService.startListening();
}
});
```

According with the intent of the user, the chatbot gives one response or another. To make this we check the action of the AIResponse result inside the onResult method provided by the AIResponse model.

```

if(result.getAction().equals("Escanear_alimento")){
Intent escanear = new Intent(VozActivity.this, EscanearActivity.class);
startActivity(escanear);
}
if(result.getAction().equals("abrir_lista_compra")){
Intent lista = new Intent(VozActivity.this, ListaActivity.class);
startActivity(lista);
}
```

As we said, here we check the result variable to see the different parameters that we can get form the DialogFlow. If the intent is to scan food it takes us to the scan screen.

Here we also have a TextView where we write the resolved query from user and the answer given by the chabot.

```
chat.setText("TU: "
+ result.getResolvedQuery()
+ "\nAction: " + result.getAction()
+ "\nParameters: " + parameterString
+ "\nFEEDMEFY: " + result.getFulfillment().getSpeech()
);
```

3.3.2.2 Components

Now we are going to see the different components we have created for this application, starting with the intents and ending with the actions and parameters.

We have provided DialogFlow to be able to respond multiple users' questions, and this has been possible thanks to identify some possible intents. It is very complicated to foresee what is going to be the people's intention. Every person has its own thinking, and he is going to ask different things than other one. That is why we have to delimit the responses, providing the same answer to multiple similar but not the same questions.

The first thing we are going to talk about are the intents. Her we have an enumeration of the main intents with a short description:

- Escanear Alimento: This intent is identified when the users want to scan a barcode. Feedme is able to understand the same intent with multiple sentences thanks to some keywords. When the sentence said by the users contains 'scan' or 'barcode' Feedme will identify this intent.
- Lista Compra: With this intent Feedme will be able to open the Shopping. Like the other intent, this has some parameters to identify the sentences.
- Contribuir: If the users want to add new products to the Database they can ask the chatbot. This inten will take the user to the Contribute Screen.
- Recomendar Sanos: This intent is used to get the 'A' NotaFeed of one food product, which is the healthiest one, and takes a list with all the aliments.

There are more intents, most of them oriented to give natural responses to colloquial questions or sentences pronounced by users.

Another important component is the Action. Actions in DialogFlow are used to provide the response. Then in Android Studio we check the action and depending on which actions

are enabled, the app provides one screen or another. We are not defining them because are strongly interrelated with intents.

To take a fast look to the DialogFlow console, we are going to show what are the parameters when users want to open the scanner.

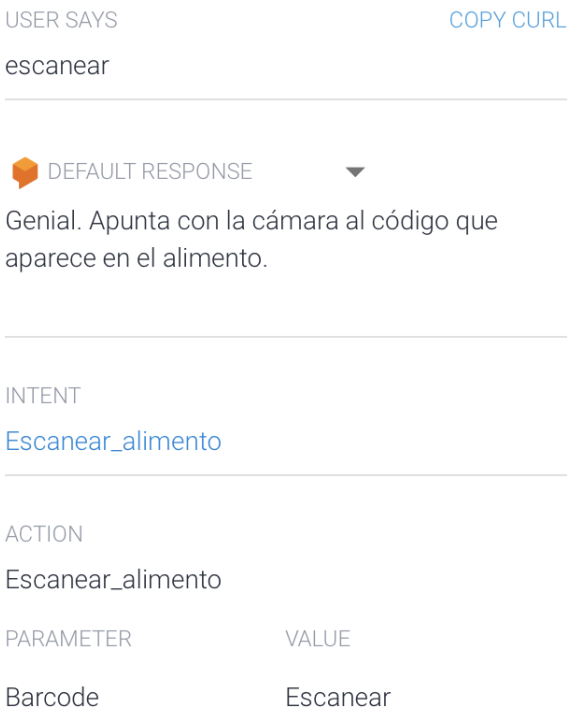

Figure 3.3: DialogFlow Console

Here we can see some components of the DialogFlow working. User has said 'escanear' and DialogFlow has been able to recognize the intent , the action and the required parameters (Barcode). Without any of these components our chatbot would not be able to give a coherent response.

In the next section we are going to see one use case to see the navigation flow between the chatbot and the scanner activity in order to see how DialogFlow works with the different intents and actions.

3.3.3 Internal Modules

In this group we can define the Android application with all the libraries included. We are going to start talking about the Zebra Crossing, in which is supported the barcode scan.

The associated library with Zebra crossing is ZXing, which is an open-source barcode

image processing library implemented in Java, that can be ported to other languages. Allows scanning barcodes and QR codes. It also has the capacity to generate new QR and barcodes. This library is implemented in the barcode scanner activity.

```
public void Escanear(View v){

IntentIntegrator intent = new IntentIntegrator(this);
intent.setDesiredBarcodeFormats(IntentIntegrator.PRODUCT_CODE_TYPES);

intent.setPrompt("Escanear codigo");
intent.setCameraId(0);
intent.setBeepEnabled(false);
intent.setBarcodeImageEnabled(false);
intent.initiateScan();
}
```

We use the IntentIntegrator library, which is inside de ZXing library. In the previous code we create an intent where we will handle a scan result. Then we make all the steps to open the camera and make the scan. It is necessary to give camera permissions for this action.

```
<uses-permission android:name="android.permissions.CAMERA" />
```

3.3.3.1 Shopping List

Inside the internal module we would like to talk about the shopping list.

We have made an action in the DialogFlow console called 'abrir_lista_compra' which detect the sentences when users want to open the shopping list. This action throws an intent to open this screen thanks to the following code:

```
if(result.getAction().equals("abrir_lista_compra")){
Intent lista = new Intent(VozActivity.this, ListaActivity.class);
startActivity(lista);
}
```

This list is programmed to add, safe and delete data from it. The first method we are showing is adding an item to the list.

```
private void addItem() {
String apuntado = apuntar.getText().toString();
if(!apuntado.isEmpty()) {
listaCompra.add(new ElementoLista(apuntado));
adapter.notifyDataSetChanged();
apuntar.setText("");
}
lista.smoothScrollToPosition(listaCompra.size()-1);
}
```

As we can see, we test if there is an item pointed and if it is not in the list we added it and we notify the change to the adapter.

For removing an item from the list we have done the next implementation.

```
private void wantRemoveItem(final int pos) {
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle(R.string.confirm);
String formato = getResources().getString(R.string.confirm_message);
builder.setMessage(String.format(formato, listaCompra.get(pos).getTexto()));
builder.setPositiveButton(R.string.remove, new DialogInterface.
    OnClickListener() {
@Override
public void onClick(DialogInterface dialogInterface, int i) {

listaCompra.remove(pos);
adapter.notifyDataSetChanged();
}
});
builder.setNegativeButton(android.R.string.cancel, null); builder.create().
    show();
```

First of all we create an alert dialog builder, and we set its title and format to show a previous question when a user wants to remove an item from the list. When user confirms he wants to delete another dialog appears indicating that the item has been removed. Then we remove the indicated position from the 'listaCompra' and we make the advise to the adapter with 'notifyDataSetChanged'.

To safe data in the list we have made the next method.

```
private void guardaLista(){
```

```
try {
FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
for(int i = 0; i < listaCompra.size(); i++){
ElementoLista el = listaCompra.get(i);
String linea = String.format("%s;%b\n", el.getTexto(), el.isChecked());
fos.write(linea.getBytes());
}
fos.close();
} catch (FileNotFoundException e){
Log.e("luis", "guardaLista: FileNotFoundException");
Toast.makeText(this, R.string.imposible_escribir, Toast.LENGTH_SHORT).show()
    ;
} catch (IOException e) {
Log.e("luis", "guardaLista: IOException");
Toast.makeText(this, R.string.imposible_escribir, Toast.LENGTH_SHORT).show()
    ;
```

In this method we make sure that when the user marks the check it is still marked when he leaves the application and came back later.

It has been programmed a menu on top right corner of the window. On it you can choose between deleting all the items of the list or deleting just the checked items. The following code shows how it has been done.

```
@Override
public boolean onCreateOptionsMenu(Menu menu){
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.opciones, menu);
return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item){

switch(item.getItemId()){
case R.id.borrar_tick:
borrarTick();
return true;
case R.id.borrar_lista:
borrarLista();
return true;
default:
return super.onOptionsItemSelected(item);
```

```
}
}

builder.setNegativeButton(android.R.string.cancel, null);
builder.create().show();
}
private void borrarTick() {

int i = 0;
while (i<listaCompra.size()){
ElementoLista item = listaCompra.get(i);
if (listaCompra.get(i).isChecked()){
listaCompra.remove(i);
}else i++;
}
adapter.notifyDataSetChanged();

}
```

In the first method the menu is created, the second method calls the two last methods, which are made to deleting all the list or just the checked items.

3.4 Design

In this section we are going to talk further about the design of the application. We are going to explain the name, the colors selection and the icons design.

3.4.1 Application Name

First thing we are talking about is the application name. Feedmefy is a name that is fairly included in the context of nutrition applications. Nowadays we find too many apps with names terminated in -ify. Etymologically, the Anglo-Saxon suffix -ify means that is a noun that makes an action. So in this case Feedmefy would mean 'make the feed'. That is why we have chosen this name for the app. Also, this name sounds a little bit like 'Give me five', that is a friendly sentence that in this case, with a chatbot integration that tries to emulate an human behavior, makes all the sense.

3.4.2 Color

We have selected the green color as the main color for the application. The green is a relaxing color that induces to whom contemplates sensations of serenity and harmony. It is related to everything natural, also symbolizing life, fertility and good health. We have decided, thanks to this last fact, to select the green as the corporate color.

3.4.3 Logo and App Icon

Then we are going to take a look to the logo. The logo has been made taking different types of food and mixing it all in the Paint application for Windows and applying a filter later.

All the buttons included in the application have a green color in order to give a healthy and nutritional image.

Finally, I would like to mention the app icon. It is a very simple icon that in most of cases these kinds of icons are the ones that call more attention to the users. It has been done with the Word Art of the Word application of Microsoft.

Figure 3.4: Logo

Figure 3.5: App Icon

3.4.4 Background

Continuing with the same color's composition we have decided to put a background image with green tones and whose image represents the shelves of a supermarket. In every screen of the app we have selected a background that shows fruits and vegetables. Then I have lowered the contrast and brightness in order to allow the rest of the components can be

visualized properly.

CHAPTER 4

Case study

4.1 Introduction

In this chapter we are going to describe a selected use case. The objective is to visualize and learn all the Android application features involved in this case, using a large part of the characteristics and modules in order to see if the goals have been achieved.

For this use case we are going to establish a user with a young profile as the actor who is shopping in the supermarket, and he is trying to decide between two similar products. He decides to use Feedmefy because he wants to obtain some characteristics from both food products, so he takes his smartphone and runs the application.

4.2 Information Page

When the user opens the application he will find the main page. First of all we are going to base that the user wants to know how he can use the app because it is his first time using the app, so he pushes to 'ACERCA DE' button to open some basic information that will help him. Once he has read the information he will go back to the main page. In the

following figure we can observe both screens.

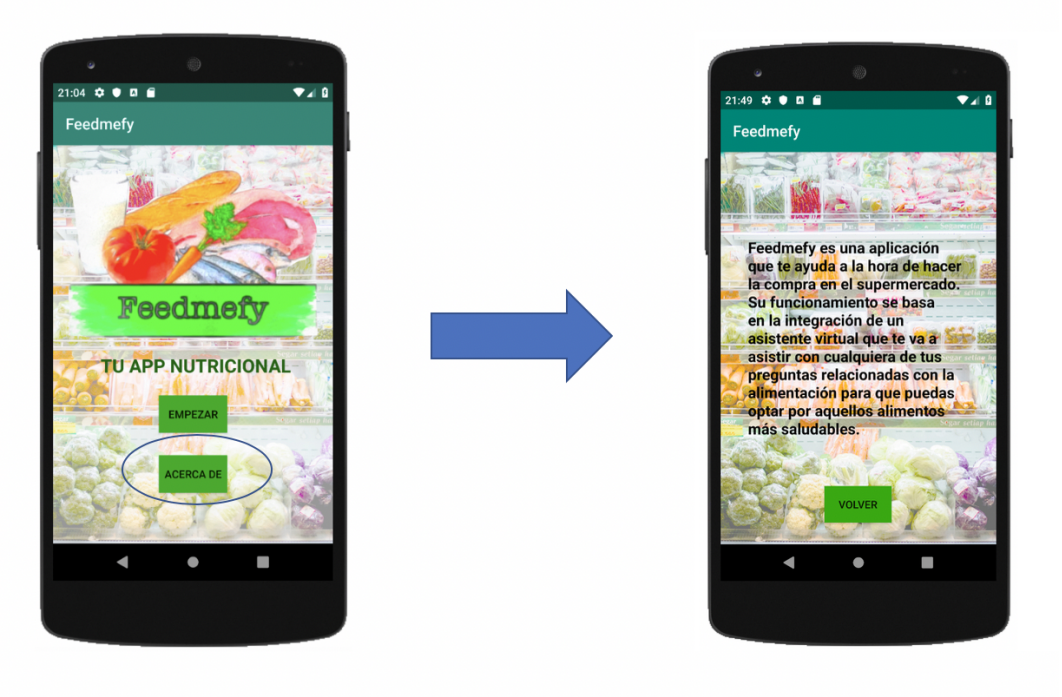

Figure 4.1: General Information Access

After this short information, the user is ready to start using the app.

4.3 Chatbot Access

The previous step to start talking with the chatbot is the selection screen. Here the user has to select 'HABLAR CON FEEDME' if he wants to scan the products in order to obtain the information. Once he has clicked on it he will go directly to the chatbot screen.

4.3.1 Permissions

If it is the first time that the users access to this screen, they will be asked for allowing recording and microphone access permissions. After that, they will be able to see the screen.

In this screen the users will find a microphone button on the center of the sceen. Pushing that button the users will access to the recording recognition. They will know that thanks to a soft beep. After that beep the users will be able to ask for everything that comes to mind.

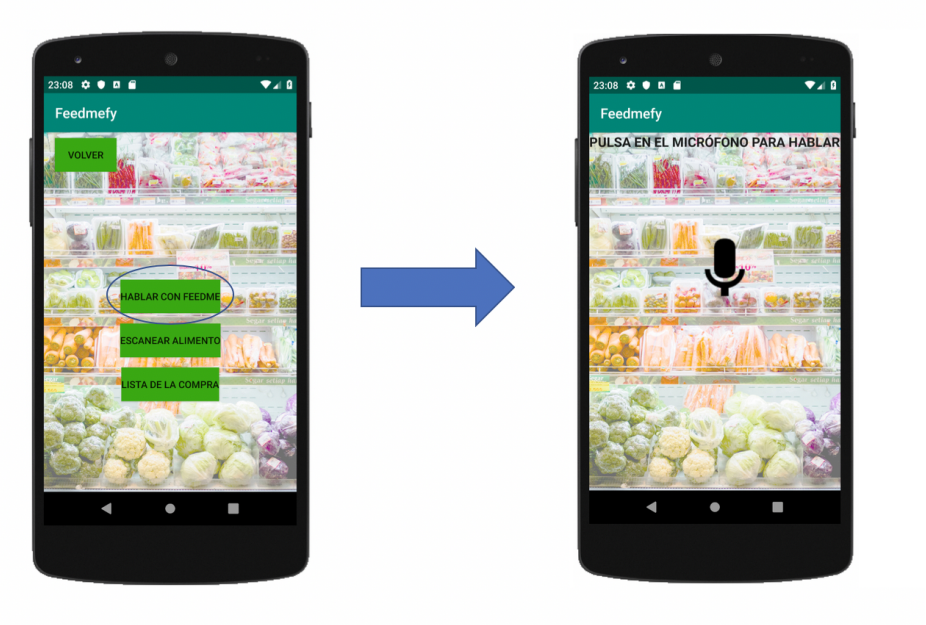

Figure 4.2: Chatbot Accesst

Now we are going to figure that the user does not know how to ask for the barcode scannig. In this situation he only has to ask for help and the chatbot will respond with some actions that user can do among which is the barcode scanner.

Figure 4.3: Help Intent

This user's request is grouped within the intent 'help'. This intent has been formed in DialogFlow console and has many ways to be called. It produces that response that can be read in the screen or listened.

Now the user would know how to request for the barcode scanner so his next action is asking the chatbot for scanning one product.

4.4 Barcode Scanner Request

Once the user knows what to do, he presses the microphone button to start with the scanner petition. As we said before, there are many ways to produce the same intent. This makes easier the communication between the chatbot and the user. Even if the user does not pronounce properly the sentence, the chatbot will be able to understand the intent because it is programmed to detect some keywords that guide him until the goal. If this does not happen and the chatbot can not detect the intent, he opens the default response automatically that induces the user to repeat the wanted request.

When the intent is taken, the chatbot will response and immediately he will open the barcode scanner screen. This is made thanks to an Android intent programmed inside the application that associate the DialogFlow intent's action with the intent of screen change.

In the following figure we can observe the transition.

Figure 4.4: Scanner Screen

In the barcode scanner screen the first thing that user sees is the 'ESCANEAR' button. When the user presses it the camera is immediately open. User will focus on the desired barcode. After that, the barcode numbers are going to be transpassed. A little below there is an EditText where these numbers are written. If by any chance the barcode number is not correct the user can modify himself selecting the EditText and typing the correct one.

Once the barcode number is ready it is time to press the 'BUSCAR' button to get the products information.

4.5 Information Screen

It is assumed that the user has been able to scan the barcode and he is in position to press the 'BUSCAR' button. Anyway, as we have already mentioned, this is not a problem because the user can type it himself. This should not happen, but sometimes the environmental light is not appropriate or the camera quality is not the best.

Now we are close to the end of this use case flow. Making a compilation of the steps that we have been following, we started with a user that wanted to compare two similar products by scanning their barcodes. He has asked the chatbot for how to use the scanner, later he has made a request for opening it and latest he has scanned the barcode.

When that button is pressed the barcode number is transmitted to the next screen. Here we find the barcode number on top, followed by one TextView that shows the product informations. This information is extracted from the Firebase database. Here we have the information screen appearance.

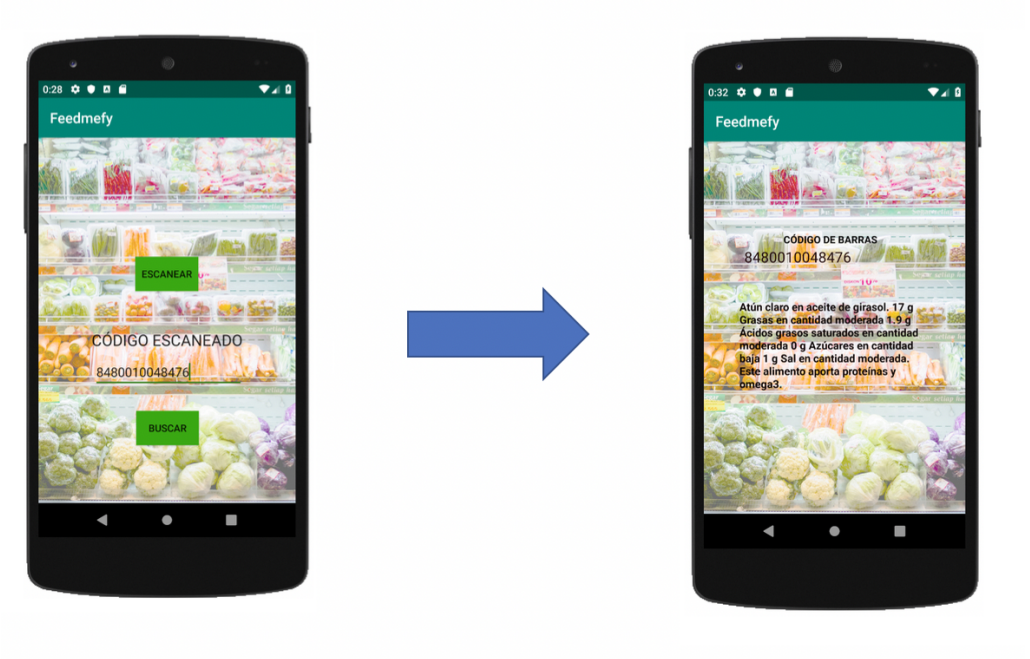

Figure 4.5: Information Screen

4.6 Contribute Screen

Finally we have to ask one thing. What happens if the selected product is not in the database? In this situation is in the user's hand to contribute by adding new prodcuts to the database.

The way to do that is by clicking on the 'Contribute' button placed in the bottom of the Information Screen. Other way to access here is going to the chatbot and telling it the intent to make a contribution. It will take the user to the contribution screen automatically.

Now user has reached the end of the flow and he finally can make a shopping list with his favourites products, but that would be other use case. We put an end to this use case.

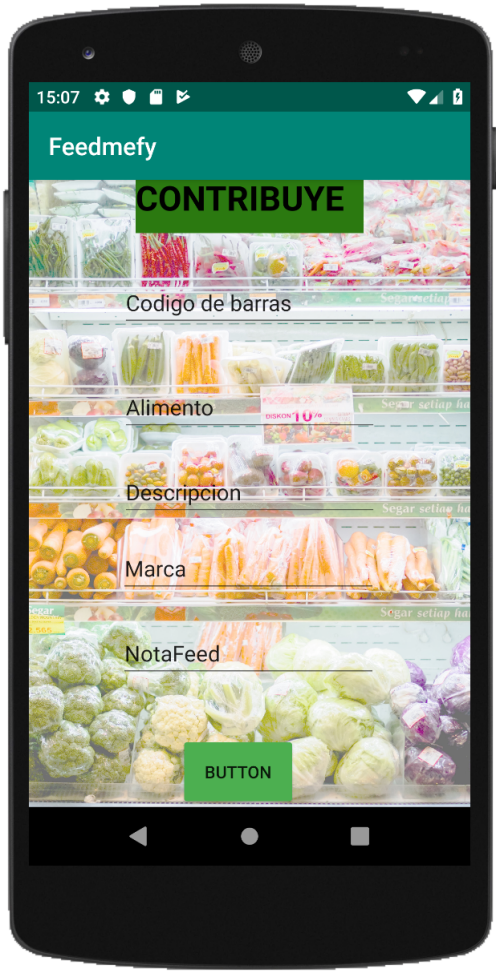

Figure 4.6: Contribution Screen

CHAPTER 5

Conclusions and future work

In this chapter we will describe the conclusions extracted from this project, and the thoughts about future work. We are going to analyze how the process of making the project has been, explaining the difficulties and the strengths during the development.

5.1 Conclusions

We have programmed an Android application which makes users' shopping easier and faster. The basic operation of the application is to offer the user the information about a food product in the supermarket and this functionality has been achieved.

Looking back, the first objective that we set was to get users to have a human and customized attention but saving costs, avoiding to wait long queues to be attended, and this was only possible with automation. This has been done over the years in all areas, and most of the services and web pages have different ways to attend its user in this way, and we thought that in nutrition this cannot be different.

To get that, first of all we made an overview of all the possibilities we had inside automation, and in the programming applications field the chatbots are the solution. After

studying different options inside the chatbots world we discovered DialogFlow. As a summary, we decided to choose DialogFlow because of its simplicity and easy integration with a few programming languages. Its programming environment is so intuitive and with multiple documentation to answer all doubts. [2]

Second, our study analyzed the different apps that are in the applications stores, and we find out that most of them, divided in two big groups, physical activity and nutrtion, offer fears of virtual assistants but with a lack of feels that you are talking to human beings.

That is why this project has been oriented to fight against the applications' impersonality, looking for a way to attend the user needs as if they had a person dedicated to them all day. And we decided to make this application in Android Studio.

By last, and one of the most difficult decisions, was to choose the database from which obtain all the information. Many food databases has been searched and studied. We saw that the main world food databases were Open Food Facts and Open Food Repo, two open source databases where everybody can contribute and use. After many problems trying to integrate one of these databases, we decided to use Firebase database and save there the food information.

Firedatabase, as DialogFlow, is a Google database that is very easy to integrate. The most liked feature was that is in real time so if one food product is changed you can see this change in real time on you application.

Once we have all the elements we started programming the application. During the development we have solved all the components that we did not have in mind in the beginning. The first one was the scanning medium. As the rest of elements, the first thing we did was to find the best way to scan barcodes and QR codes, and we discover Zebra Crossing. After studying the way to integrate it with the Android Studio project we could be able to scan the barcodes of the different products.

By last, it is time to test the application. The main way to prove it has been executing it multiple times and try all the alternatives of the application. It has been given to a sample of selected users in order to try it and to find all the possible errors.

5.2 Achieved goals

Now we are going to resume in the next points all the project process.

- First of all, we have made an intensive analysis to see the applications in the current

market in order to see what is the trend in this sector. Thanks to this analysis we have taken the main idea of the new project.

- After that we have seen the alternatives that we had in all fields involved in the project, starting with the development environment and ending with the storage database. First we thought in an external database but then we opted for our own database.
- Once selected the components we have made an Android Application integrated with DialogFlow in order to have a chatbot to attend the users requests. This chatbot and the Firebase database are both from Google, which makes a strong and powerful integration with Android Studio.
- We have achieved to incorporate the barcode scanning thanks to the Zebra Crossing library for java and it works correctly.
- We have included a Splash Screen and onboarding screens in order to give a more friendly image to the application.
- In conclusion, we have programmed an Android application that meets with the objective to offer to users the capacity to be informed about food products in the supermarket.

5.3 Future work

Now we make an analysis of the future characteristics that we want the application to have, either because they could not be added in time or because they have been devised later.

- The first future objective is to make a sign up screen with a storage of all the users in order to create a specialized space for each user. With this feature users could select their favourite food product and save them, or make valuations about products or posting comments.
- Create a more complete structure about products, including more information and with the possibilities of seeing other user's comments about them.
- Improve the design of the different components and make a more orderly distribution of them with the aim of make a more friendly app.
- Create a button to add the scanned product to the shopping list.
- Add images for every product

- Make a classification with the different supermarkets to classify the food products.

APPENDIX A

Impact of this project

This appendix reflects, quantitatively or qualitatively, on the possible impact.

A.1 Social impact

Technologies have been imposed in current society. Even the least advanced societies, in the so-called third world, the use of the mobile phone and the applications of these devices are used by thousands of citizens. We can say that even in the least developed countries and with the lowest per capita income, any citizen, even in a state of poverty, has a mobile phone.

However, the rational use of these technologies is very different in each society. A high percentage uses these "technologies" in a banal way, they suppose a mere entertainment that absorbs to the citizen of other more transcendental and useful questions and they turn him into a dependent of the use of the mobile. The immediacy with which you achieve any goal: buy, talk with friends, have immediate answers, is making us increasingly dependent.

The responsible consumption of these technologies, and the use of them to make life no more dependent, but healthier, is the goal of this application dedicated to good eating

habits, to the knowledge of the properties of them, and to have that into account when making the purchase, which is the first step of a healthy diet.

A healthy diet improves the people's life quality, a healthy aging, so important in advanced societies in which the life expectancy of the average is over 80 years. These diets help to save on health expenses, since it avoids high percentages of cancers, diseases, cardiovascular diseases, oral diseases and the great plague of the 21st century: obesity, with many associated pathologies.

Throughout these last years, the WHO, and since 2004, in which it adopted a Plan called "World Strategy on Diet, Physical Activity and Health", said Organization has established a series of protocols addressed to governments, manufacturers and civil society. -consumers-, tending to the improvement of the diet with the objective of favoring health.

APPENDIX B

Economic budget

This appendix details an adequate budget to bring about the project.

B.1 Physical resources

For the realization of this work has been necessary to purchase the following equipment:

- A Mac Book Pro computer, where it is installed the Android Studio IDE. Here is where the application has been developed.
- Samsung LG K9. With this smartphone we have made the app instalations in order to test the app in a real device and not only in the AVD.

In the work they have invested four months of duration, with a useful life of each equipment of 4 years, (it takes into account the high reliability of the equipment, but also the speed with which they become obsolete), what makes the amortization of them:

Amortization = 4 months / 48 months = 8.3%

The cost of the equipment is as follows.

Equipment	Price	Amortization	Cost
MacBook Pro	1,755.59€	8.33%	146.3€
Samsung K9	110€	8.33%	9.16€

Table B.1: Equipment Costs

B.2 Staff resources

This project has been made by a graduate in telecommunications engineer. The inverted time in the project has been around 3 hours per day during february and 4 hours per day from march to june. We estimate a base salary of 1000€ per month.

Phase	Employee Time	Cost
Analysis and annotation	100 hours	625€
Design and Development	250 hours	1562.5€
Validation	30 hours	187.5€

Table B.2: Staff Costs

B.3 Project Cost

The total cost of the project would be the following:

Resource	Cost
Equipment	155.46€
Staff	2375€
Total	2,530.46€

Table B.3: Project Costs

Bibliography

[1] Javier Aranceta-Bartrina and Carmen Pérez-Rodrigo. La obesidad infantil: una asignatura pendiente. *Revista Española de Cardiología*, pages 888–891, 2018.

[2] Martin Mitrevski. Speech, synthesizers, and dialogflow. In *Developing Conversational Interfaces for iOS*, pages 91–142. Springer, 2018.

[3] Xataka. José García Nieto. Así es como android se ha comido el mercado en diez años. `https://www.xatakamovil.com/sistemas-operativos/asi-como-android-se-ha-comido-mercado-diez-anos`, January 2019. (Accessed on 06/04/2019).

[4] TIOBE Index. Tiobe-the software quality company. *TIOBE Index— TIOBE–The Software Quality Company [Electronic resource]. Mode of access: https://www. tiobe. com/tiobe-index/-Date of access*, 1, 2018.

[5] Erich Lück and Autorra Pérez Torromé. *Conservación química de los alimentos*. Acribia, 1981.

[6] Ana Zugasti Murillo. Intolerancia alimentaria. *Endocrinología y Nutrición*, 56(5):241–250, 2009.

[7] Lantern. The green revolution. `http://www.lantern.es/papers/the-green-revolution-entendiendo-el-auge-del-mundo-veggie`, 2019. (Accessed on 03/20/2019).

[8] WEF Wold Economic Forum. The future of jobs report 2018. `https://www.weforum.org/reports/the-future-of-jobs-report-2018`, 2018. (Accessed on 03/22/2019).

[9] El Confidencial. Dieta: Cómo elegir la mejor app para hacer una dieta y comer bien. `https://www.alimente.elconfidencial.com/nutricion/2018-06-11/apps-adelgazar-imc_1569919/`, June 2018. (Accessed on 03/25/2019).

[10] MyFitnessPal. `https://www.myfitnesspal.com/es/`. (Accessed on 03/25/2019).

[11] Appsnutricion. Plixi: - apps nutrición top 50. `http://appsnutricion.blogspot.es/1520967876/plixi-/`, March 2018. (Accessed on 04/15/2019).

[12] The fabulous. `https://thefabulous.co/`. (Accessed on 04/18/2019).

[13] Vegaffinity. Vegaffinity dieta y recetas vegetarianas. `https://www.vegaffinity.com/`. (Accessed on 05/02/2019).

[14] HealthifyMe. Healthifyme weight loss diet, exercise and yoga coach. `https://www.healthifyme.com/`. (Accessed on 05/05/2019).

[15] Manzanaroja. https://www.manzanaroja.eu/. (Accessed on 05/09/2019).

[16] Divyanshu Bhargava. Building chatbot with firebase and api.ai – androidpub. https://android.jlelse.eu/building-chatbot-with-firebase-and-api-ai-5b787fe693a9, September 2017. (Accessed on 05/12/2019).

[17] Botsifuel. Botsify - create automated chatbots online for facebook messenger or website. https://botsify.com/. (Accessed on 05/23/2019).

[18] Juanan Pereira and Oscar Díaz. A quality analysis of facebook messenger's most popular chatbots. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 2144–2150. ACM, 2018.

[19] Beep & boop. http://archive.parentschoice.org/product.cfm?product_id=32156&StepNum=1&award=aw. (Accessed on 05/27/2019).

[20] P Domashnev, V Alexeev, T Lavrukhina, and O Nazarkin. Usage of telegram bots for message exchange in distributed computing. *International Journal of Open Information Technologies*, 7(6):67–72, 2019.

[21] Laurence Moroney. *The Definitive guide to firebase: build Android apps on Google's mobile platform*. Apress, 2017.