
THE HOC PROJECT

GROUP ID - SO3

COMPUTER SCIENCE AND ENGINEERING

ADVANCED USER INTERFACES

Course 2021-2022



POLITECNICO

MILANO 1863

Authors

José Ignacio Daguerre Garrido
Francisco Javier Muñoz Ruiz
Kateryna Lapshyna
Noelia Carrasco Vilar

Abstract

This document describes the development process of “The HOC” device, whose main goal is to interact with gadgets facilitating specific tasks for the elderly.

The statement of the identified problem is based on a detailed analysis of the needs and of the stakeholders, as well as taking into account the requirements for the system, based on the goals set.

Within the framework of this document it is considered how feasible it is to create such a device as the proposed solution, as well as to evaluate the prospects of this system in the implementation of such technology in daily life.

The team

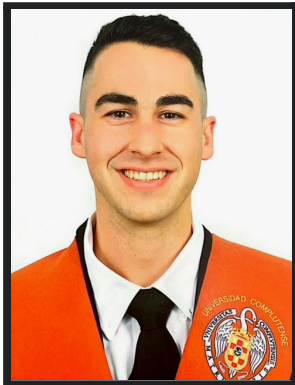


Francisco Javier Muñoz Ruiz

f.javier18.zafra@gmail.com

+34 673122568

Software engineering



José Ignacio Daguerre Garrido

joseignacioDG1999@gmail.com

+34 628191144

Computer engineering



Noelia Carrasco Vilar

noeliacarrascovilar@gmail.com

+34 622836974

Computer engineering



Kateryna Lapshyna

laktrynit@gmail.com

+39 351 7421620

Communication design

Index

1. Introduction	5
2. Requirements	6
3. Solution – UX Design	7
3.1 Scenarios	7
Elderly person uses HOC	7
Family member setting up the sticker	8
4. Solution – Implementation	9
4.1 HW architecture	9
4.1.1 Components	9
4.2 SW architecture	9
4.2.1 Tools	9
4.2.2 Mobile Application	11
Home Section	12
Add Stickers Section	14
Settings Section	14
Login Section	15
Register Section	16
Tasks Section	19
Calendar Section	20
Structure and Organisation of Java classes	20
4.2.3 Sticker Set	23
4.2.4 HOC Device	24
3D-model of HOC Device	25
4.2.5 Database	26
4.3 Usage	31
5. State of the art	32
6. Value Proposition	33
6.1 Main difficulties encountered	33
7. Future work	34
7.1 Improvements	34
7.2 Future directions	36
8. Bibliography	37

1. Introduction

The exponential growth in the usage of technologies in our everyday life has left the elderly confused. In combination with the problem of the Covid-19 pandemic, nowadays, the elderly are especially vulnerable. And, as quarantine times have shown, the younger generation may not always be able to be near and help the elderly.

As we live in a globalised world it is very difficult to be in the same place where the family is. Everything is based on remote interaction, it's incredibly important to keep in touch. Because if you're out of contact, it immediately brings a lot of anxiety to your loved ones. However, it is quite difficult for the older generation to master modern technology, even when it comes to cell phones.

That is, because of quarantine restrictions, many people are not only deprived of the opportunity to see their elderly parents and grandparents, they are also literally not always able to contact them by phone. Which brings a lot of problems and nerves in the already difficult circumstances of life.

So, based on this problem, it was decided to create a device that would facilitate the interaction of elderly people with their personal gadgets, which, in turn, will allow them to be in touch with their relatives, even if you are separated by cities or countries.

That is why this document describes the actions taken by our team in the process of creating a device called "The HOC".

2. Requirements

At this point, there are several problems that we would like to address with our project work. We are talking about the fact that elderly people often have problems with mastering modern technology (including cell phones) - they have difficulty using gadgets. And also the point that with the onset of the pandemic, had to stick to isolation, family members are not always able to be near.

After analysing the current situation, the main stakeholders in the use of the device created in the framework of the project are elderly people, as well as their family members who live separately. Stakeholders:

- An elderly man or woman (end-user)
- Family member (son, daughter or grandchildren)

Based on the existing problems, the needs of the user to be addressed by the device are:
Needs of the elderly person:

- To easily use the basic features of digital devices
- To be in touch with family members

Needs of family members:

- Remotely facilitate the elderly person's interaction with technology
- To stay in touch with the elderly person at all times

Regarding the context within which the use of the device is envisioned:

- When the elderly person is at home (alone)
- When relatives are far away and unable to come quickly to provide assistance or simply to visit
- Or when there are severe quarantine restrictions

Available restrictions.

- Time: The deadline for the study course.
- Human resources: 3 developers and 1 designer.
- Social set-up: server capacity.

Goals for the system, based on user needs:

- Provide a user-friendly, adapted interface
- Be remotely controllable
- Connect people in a quick and easy way
- Provide a stable communication

3. Solution – UX Design

We propose a solution in creating a system based on a physical portable smart object called "The HOC", which will have the basic functionality of modern digital devices. Management will be possible by scanning identifying tags. The device is equipped with direct voice functionality for the reproduction and creation of audio content. Also, the corresponding mobile application, which is connected to the smart object, will be managed by a family member to simplify the process of personalization of the device for the elderly person. Thus, the main idea is to provide a connection between the actions set via the mobile application and the actions to be performed in reality by the smart object to facilitate daily tasks.

That is, the functionality of the device "The HOC", which will be available to the main user:

- phone calls
- voicemail
- calendar events
- alarm clock
- weather information
- play music
- current time

All these functions are assigned to each sticker, so that the elderly person scans the sticker and can perform the function he/she wants.

3.1 Scenarios

Elderly person uses HOC

An elderly man with memory problems named Luca woke up in his Milano home and realized that he did not remember what he planned to do that day. So, he took a device in his hands and pointed to the sticker, which is glued to the calendar above the bed.

At the same moment, the device loudly announced that Luca was going to go to the store for bread to have sandwiches for breakfast. And that he also needs to call his son, who lives in London.

The elderly man got up and began to pack up to go to the grocery store. He took the device, went to the closet and scanned the special sticker that was attached to the door of the closet - the device immediately notified Luca about the weather forecast for today. Then, with the help of his device, the man scanned the sticker that was on the mirror. And then, the device began to play music - immediately it turned on Luca's favourite melody. So, it became more fun for him to get ready for the store's trip.

When the elderly man bought the bread, he returned home and turned on his favourite radio station using the device scanning the sticker that was on his cup. After finishing breakfast,

the man pointed the device at the clock, as he could hardly distinguish the hands of the clock. So, after pointing at the sticker, loudly announced the current time.

It was 11 am. The elderly man realised that he was ready to call his son, with whom he had not communicated for so long. His son, Andrea, lives in London, so they don't socialise very often.

Luca had problems with mastering a mobile phone, he did not remember all the possible actions and the sequence of keystrokes. He pointed with the device at the son's photograph, which was on the table. IWhere the sticker is located, which is responsible for speed dialling the number of this subscriber. After scanning, a call was displayed on the mobile phone. Thus, Luca was able to quickly and easily contact his son.

And after the conversation, the elderly man decided to check if he had missed any important message. To do this, he again took it in his hands and pointed it at the notebook sticker. And then the device loudly announced a new message from Ilaria, an old friend of Luca. She invited him to go to the theatre this week. Luca was glad to hear this and he was happy that he did not miss this message.

Family member setting up the sticker

Andrea, the son of the elderly man Luca, when he was drinking his morning coffee in London, received a call from his father. His father was worried because he didn't understand how to add the meeting with Ilaria to the calendar. Instead, he asked his son to do it. Andrea realised that he had to help his father with customising the device for his needs.

After receiving the call, Andrea opened the app for the device that his father was using. Firstly, he wanted to check if the meeting was there in case the father added it but does not remember. It was not added, so Andrea had to do it. He chose Saturday on the calendar, clicked the "Add Event" button, and wrote "Meeting Ilaria at the Theatre at 6 PM". Also, he added a reminder of this meeting at 11 am the same day.

Then Andrea called his father and said that he had added an appointment to his schedule. Luca remembered that he wanted to ask his son to set an alarm for every day at 8 am using a mobile application of device. Then Andrea went to the application again, on the main screen selected the "Add sticker" button, set the "Alarm" as an action, set 8 am and selected all days of the week, as his father had asked. Further, he gave the appropriate name to this action "Daily alarm at 8 am" and chose the image of the sticker that will be responsible for this action.

Then Andrea remembered that Luca had pasted the wrong sticker on the closet by mistake and therefore he could not find out the weather by scanning it. Then from the main screen of the application, the young man went to the "Existing stickers" section, found one called "Weather forecast on the closet", clicked "Edit". And in the screen that appeared, he chose another image of the sticker, which is now actually glued to the closet.

Thus, Andrey helped Luca a lot in customising the device to fit his needs.

4. Solution – Implementation

4.1 HW architecture

This chapter enumerates and describes the hardware components used for developing the HOC device, also presents the decisions made throughout the development of it.

4.1.1 Components

- Raspberry Pi
- Cable of Raspberry
- RFID Stickers
- RFID Controller/Reader
- ProtoBoard
- Cables
- Speaker
- Battery
- LEDs
- Confirmation button.
- Microphone

4.2 SW architecture

This chapter enumerates and describes the tools used for developing the application, as well as the databases from which the data samples needed for the different functionalities set to the HOC device. This chapter also presents the decisions made throughout the development of the app.

4.2.1 Tools

Android version

The Android operating system is constantly being updated. In each release, new features can be developed, security measures can be changed, etc. At this moment, the newest release is Android 12 Beta.

We decided to use Android 11 to develop our app as it was the latest release at the time, which ensured that our app was up to date with Android standards.

Version	SDK / API level	Version code	Codename	Cumulative usage ¹	Year
Android 13 ^{DEV}	Level 32	T	Tiramisu ²	No data	TBD
Android 12	Level 31	S	Snow Cone ²	No data	2021
Android 11	Level 30	R	Red Velvet Cake ²	33.3%	2020
▪ targetSdk must be 30+ for new apps by August 2021 and app updates by November 2021.					
Android 10	Level 29	Q	Quince Tart ²	61.9%	2019
▪ targetSdk must now be 29+ for all app updates.					
Android 9	Level 28	P	Pie	76.2%	2018
Android 8	Level 27 Android 8.1	O_MR1	Oreo	83.9%	2017
	Level 26 Android 8.0	O		87.2%	
Android 7	Level 25 Android 7.1	N_MR1	Nougat	89.3%	2016
	Level 24 Android 7.0	N		92.4%	
Android 6	Level 23	M	Marshmallow	96.0%	2015
Android 5	Level 22 Android 5.1	LOLLIPOP_MR1	Lollipop	98.2%	2015
	Level 21 Android 5.0	LOLLIPOP, L		98.6%	2014
▪ Jetpack Compose requires a minSdk of 21 or higher.					

Figure 1: Existing Android Versions

Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

This tool was used in this project to develop the entire application. It gives the developer the choice to code in Java or in Kotlin. Because of the previous knowledge of Java that the members of the team possess, this app was written in Java.

This IDE incorporates its own Android virtual machine to test and debug the application. However, in this project we decided to test it on a real Android device because of the efficiency and the faster execution time compared to the virtual machine. The IDE also has a GUI editor that helps the developer to edit the .xml view layouts.

Firebase Database

Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps at global scale. Using Cloud Functions (serverless compute product), can execute hosted backend code that responds to data changes in a database.

It can automatically synchronise your app data between different devices which facilitates the management of different existing users. Cloud Firestore provides powerful query functionality for specifying which documents you want to retrieve from a collection or collection group.

Github Desktop

Github Desktop is an application that enables users to interact with GitHub using a GUI instead of the command line or a web browser. We used this tool to manage the versions of the application and to seamlessly merge each contribution of all the participants of the team, allowing a parallel development.

Figma

Figma is a cloud-based design tool. It is a UI and UX design application that can be used in order to create websites, apps, or smaller user interface components that can be integrated into other projects. It's made so that users can collaborate on projects and work pretty much anywhere.

Autodesk Fusion 360

Fusion 360 is a cloud-based CAD/CAM tool for collaborative product development. Fusion 360 enables exploration and iteration on product ideas and collaboration within a distributed product development team. Fusion 360 combines organic shapes modelling, mechanical design and manufacturing in one comprehensive package.

Programming languages used

- **Python:** We use python to manage the actions of the HOC device and connect it to the database.
- **Java:** We use this language to program all the logic parts of the mobile application.
- **XML:** The whole graphical part of the application is designed and configured with XML.
- **Bash Linux:** This language is used to configure and manage all components of the HOC device.

4.2.2 Mobile Application

This chapter provides a detailed explanation of each layout of the application and the interaction of the user with it. An intuitive interface has been designed for the mobile application for easy interaction with it. The main function is the management of identifying stickers that ensure that "The HOC" device performs the functions for which it is intended. We have also designed a logo as shown in *Figure 73*.



Figure 73: Application Logo

The HOC application has been structured through the use of Activities with its own layout attached. Its main composition consists of 5 sections managed and implemented by ToolBarManagement java class, which can be accessed by clicking on the bottom five bar icon:

- **Home Section:** Entry point of the application. Displays the user's sticker list.
- **Add Stickers Section:** Displays a screen where the user inserts the remaining sticker data.
- **Settings Section:** Manage all app settings (change language, notifications, change role, logout,...).
- **Tasks Section:** Displays the user's tasks list.
- **Calendar Section:** Displays a calendar, and days that have a task will be marked.



Figure 8: Section bar organisation

In case the user is elderly, only the following will be displayed:



Figure 9: Section bar organisation

Home Section

This section is the visual entry point of the application, showing all the stickers a user may have, as shown in Figure 10. If the user has no stickers, an empty house is displayed (Figure 9). Once the user clicks on a specific sticker, the sticker information is displayed (Figure 11). We have also implemented Edit and Delete buttons in case the user wants to modify the preferences of a sticker or directly delete it (Figure 12).

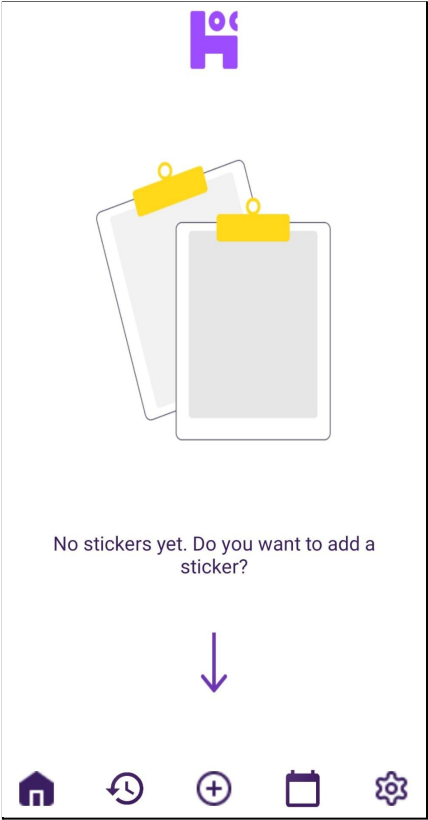


Figure 10: Stickers Section

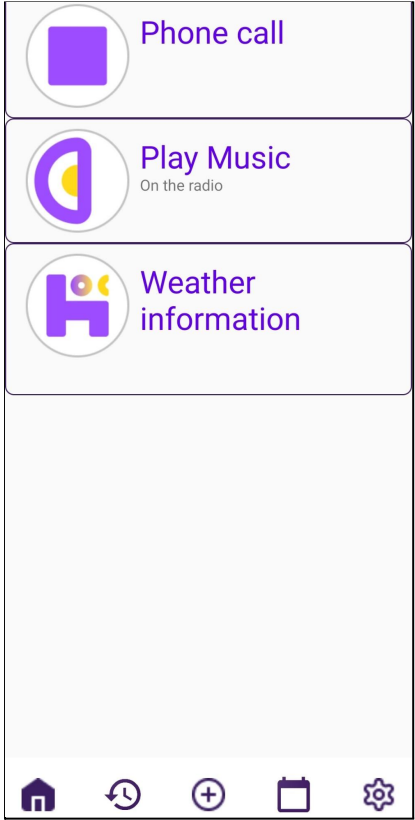


Figure 11: Stickers Section

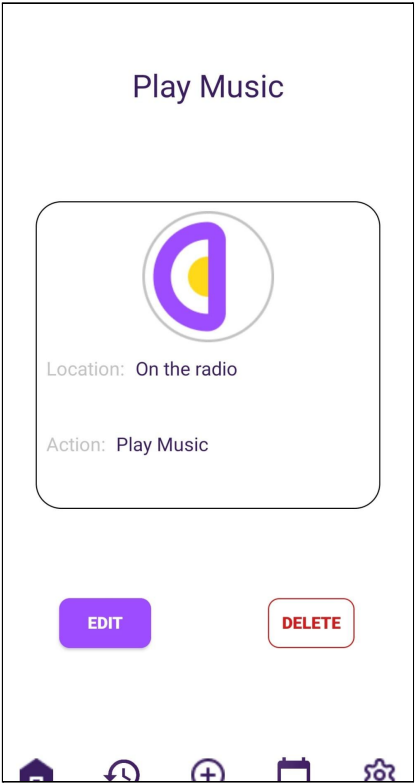


Figure 5: Stickers Information

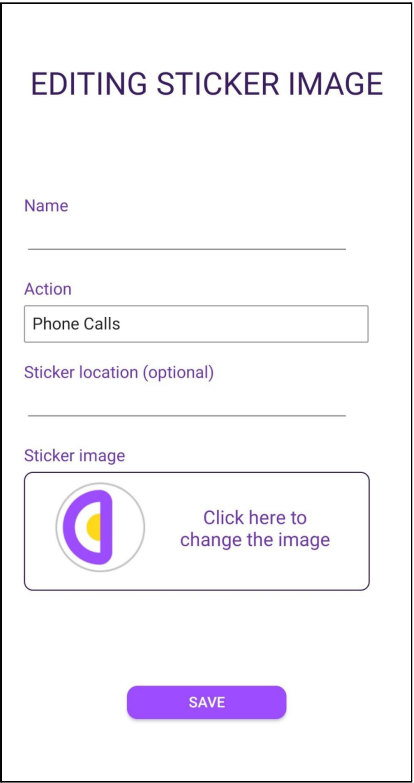


Figure 12: Edit Stickers

Add Stickers Section

This section is where the user adds the stickers (figure 13). Once all the sticker data has been entered, the user must choose which image the sticker has (figure 14).

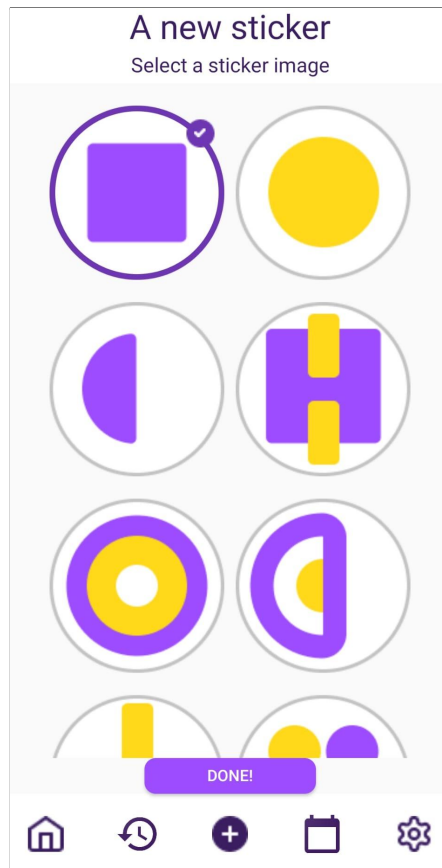


Figure 13: New Sticker Section

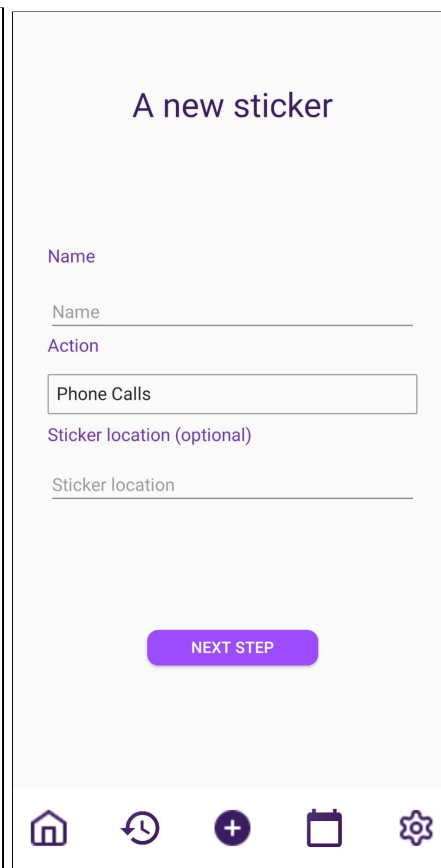


Figure 14: Picture from sticker

Settings Section

In this part of the interface we can see three language options [12] (Spanish, Italian and English), the user can choose the one he wants and it will be changed automatically.

In the second part we find the settings, first we can see a switch, which gives the user the option to enable or disable notifications; below this is the option for technical support, then to change the role and finally the option to log out. (Figure 15)

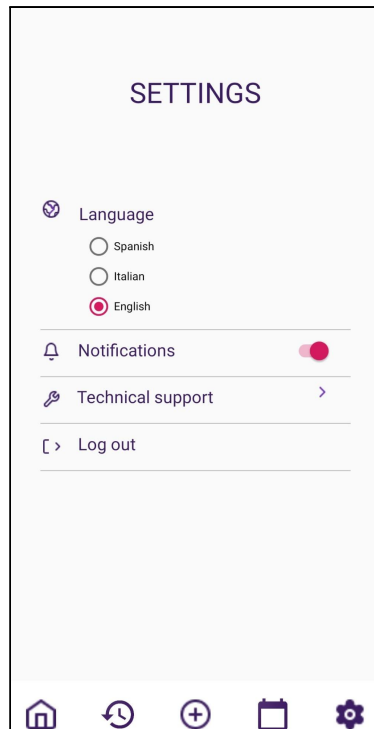


Figure 15: Settings Section

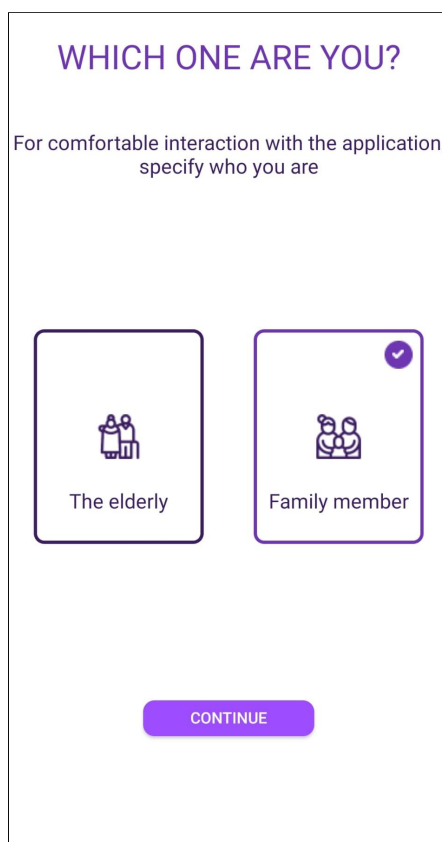
Login Section

This section displays a login form for users to access the application's functionalities.

Figure 16: Log Analyzer Fragment


Register Section


This section displays a registration form for new users in order to create a new account. First the user must choose which role he/she is (figure 17). Then he/she is redirected to the registration, here the user's name, email and password will be saved (figure 18). Once the user is done with the registration form the application will redirect the user to a section where depending on the role the user has chosen, instructions will appear (figures 19-22 for family members) (figures 23-26 for elderly).



WHICH ONE ARE YOU?

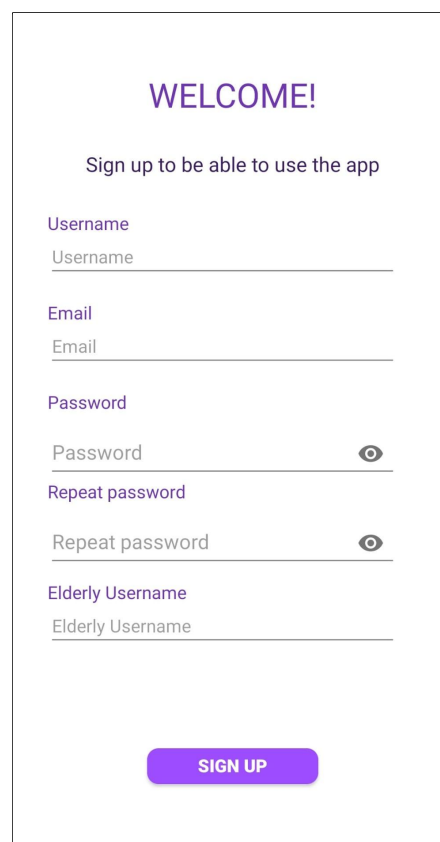
For comfortable interaction with the application, specify who you are


The elderly


Family member

CONTINUE

Figure 17: Select a role Fragment



WELCOME!

Sign up to be able to use the app

Username
Username

Email
Email

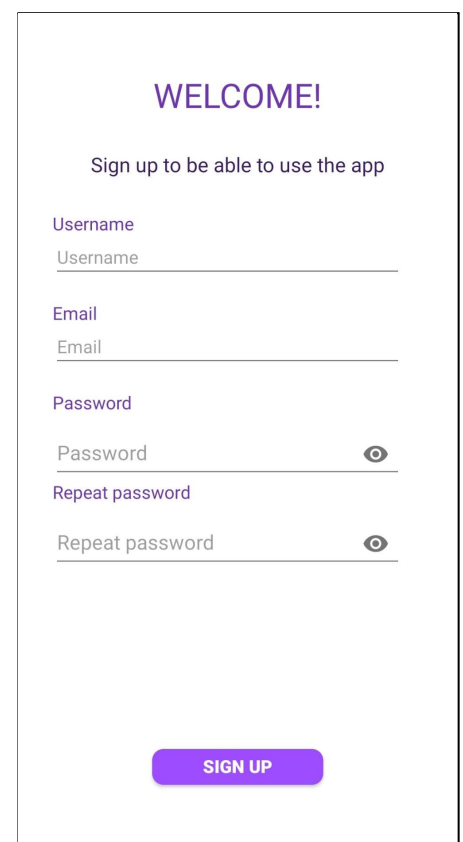
Password
Password

Repeat password
Repeat password

Elderly Username
Elderly Username

SIGN UP

Figure 18: Sign up family



WELCOME!

Sign up to be able to use the app

Username
Username

Email
Email

Password
Password

Repeat password
Repeat password

SIGN UP

Figure 19: Sign up elderly

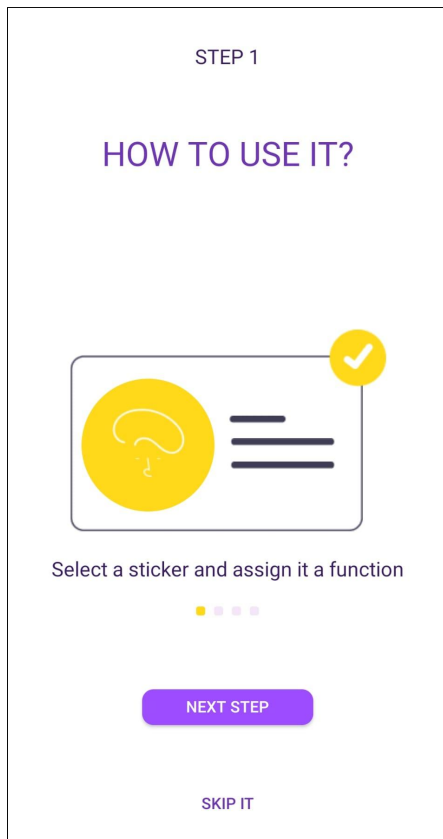


Figure 6: Step 1 for family member

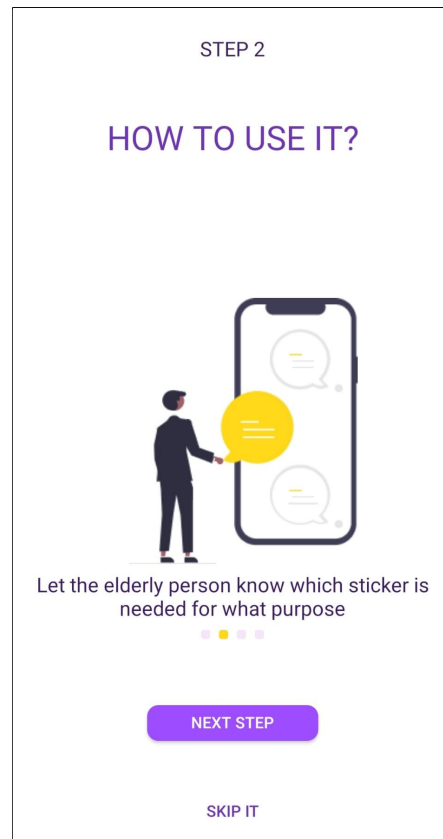


Figure 20: Step 2 for family member

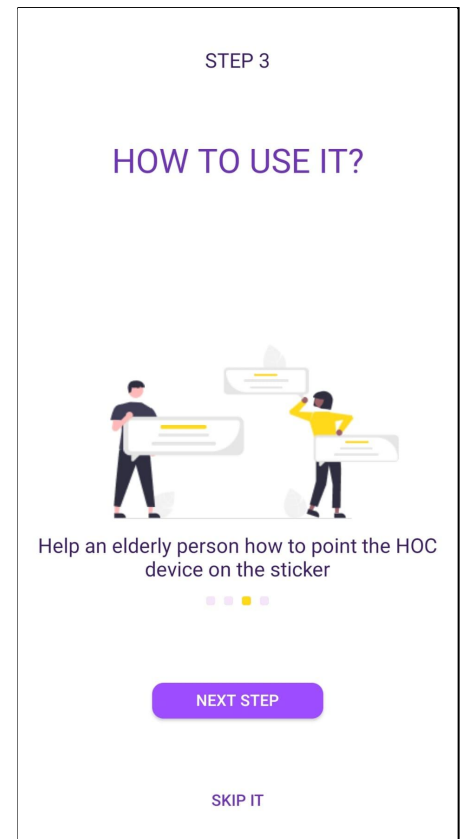


Figure 21: Step 3 for family member

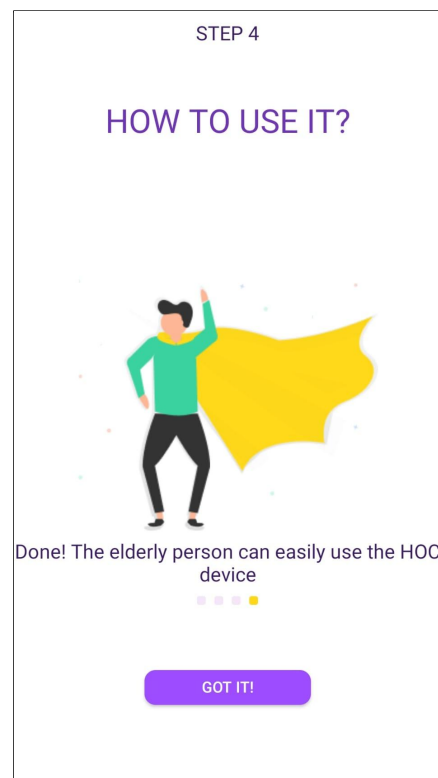


Figure 22: Step 4 for family member

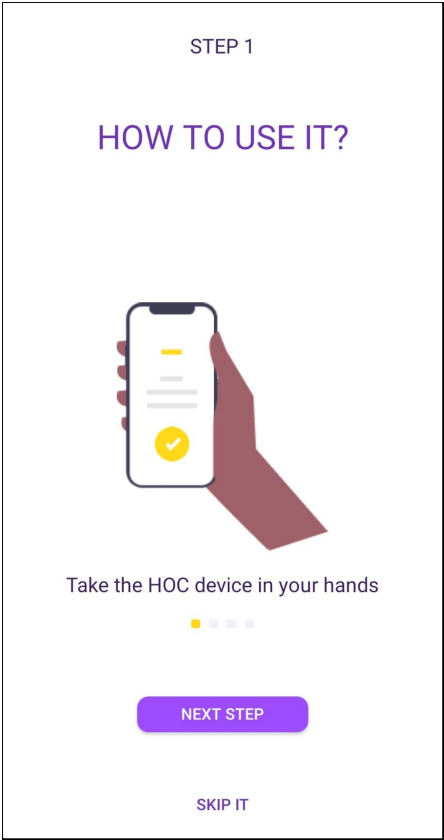


Figure 23: Step 1 for ederly

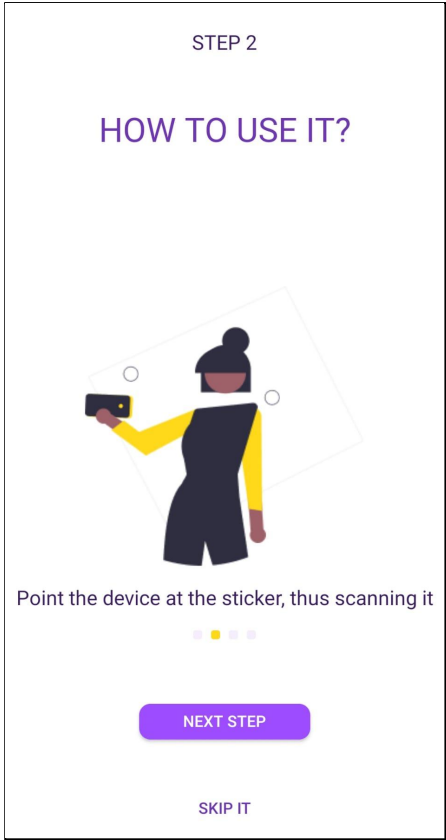


Figure 24: Step 2 for ederly

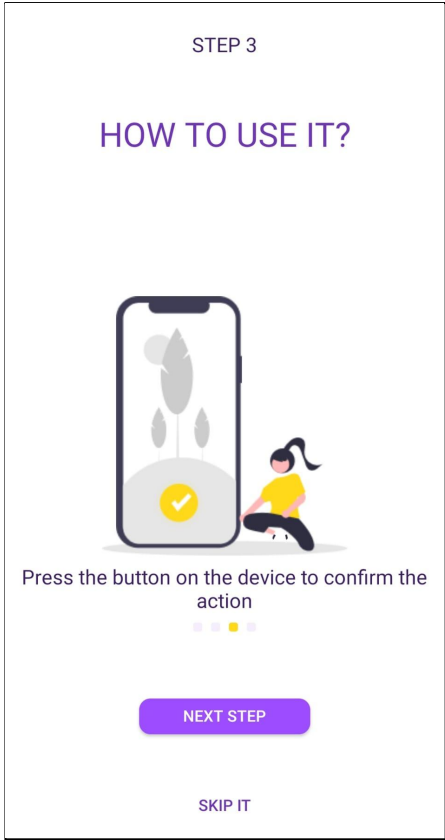


Figure 25: Step 3 for ederly

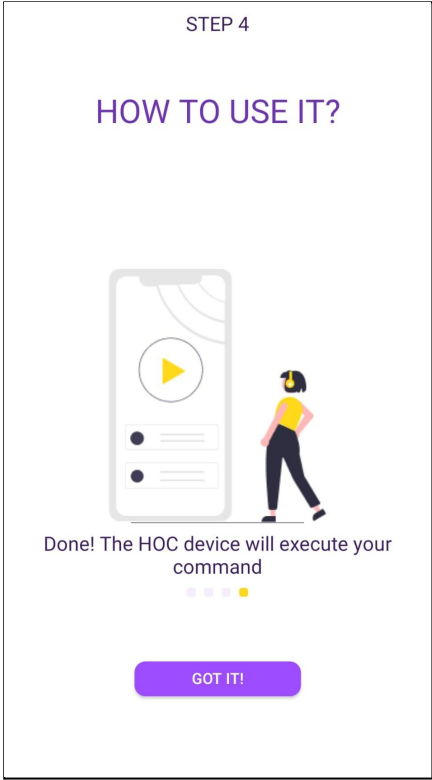


Figure 26: Step 4 for ederly

Tasks Section 🕒

In this section, the user can see the tasks he/she has scheduled, and can also add new tasks by choosing the frequency with which they will be displayed, the date, the time, etc.

All the tasks added will be shown later in the calendar and the user will receive a notification when the date scheduled with the task arrives. (Figure 27)

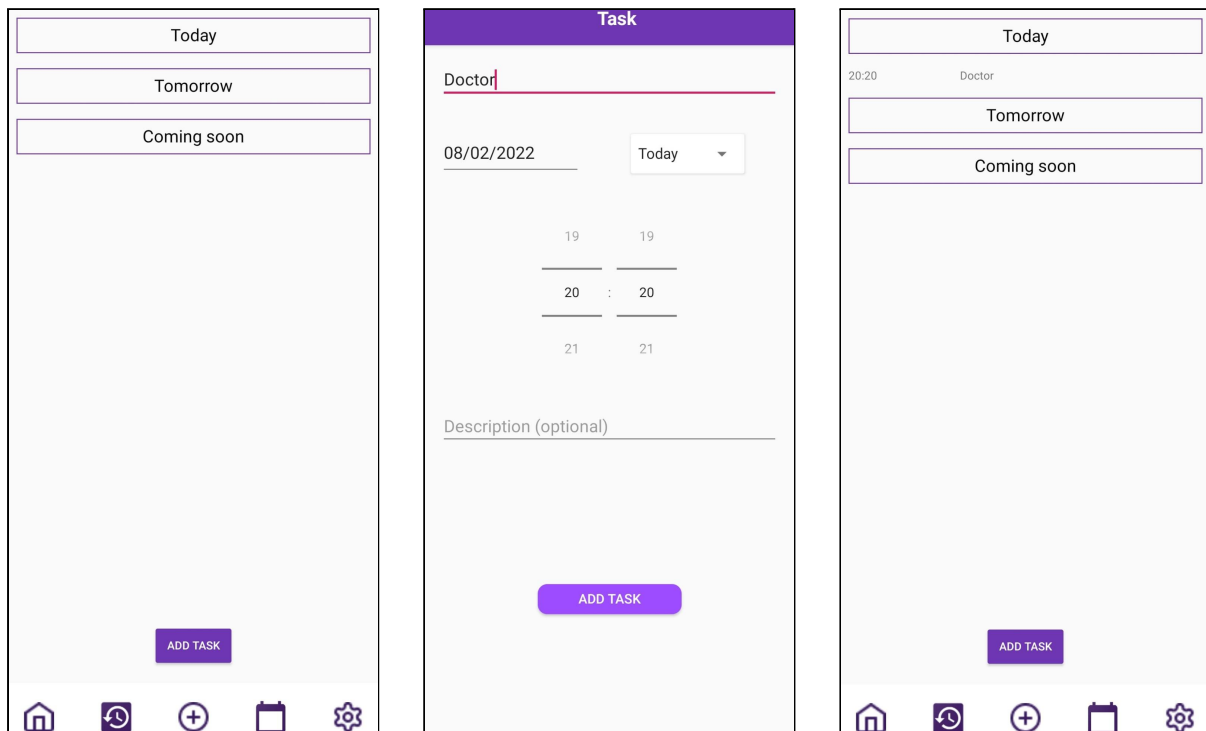


Figure 27: Task Section

Calendar Section

In this section, the user can see a calendar [11], in which he/she will be shown marked days, which have pending tasks. If the user clicks on that day, a list of all the tasks for that day will appear just below it. (Figures 28 and 29)

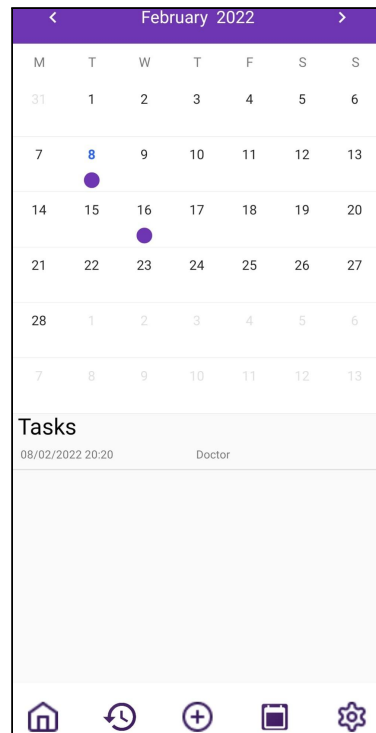


Figure 28: Family Calendar Section

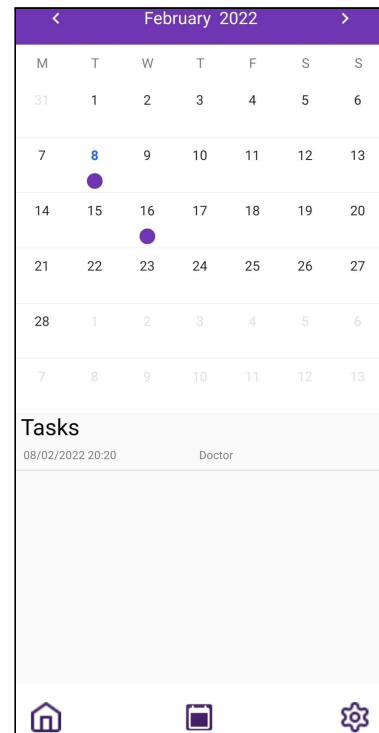


Figure 29: Elder Calendar Section

Structure and Organisation of Java classes

This chapter gives a highly detailed explanation of how we structure our application, the functionalities of each java class.

The application is composed of java classes together with XML components linked to its graphic design. Those java classes are grouped by packages to differentiate them by functionality from each other and to maintain a clear and solid structure. Moreover, we created icons and personalised png files for visual content.

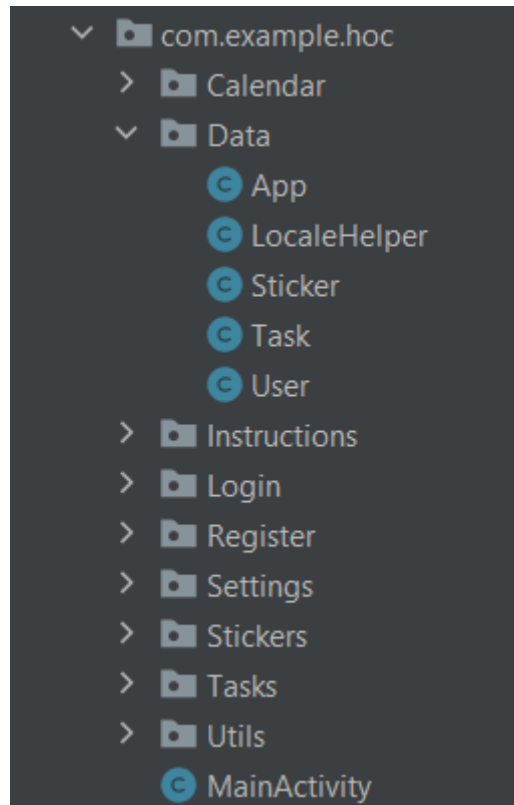


Figure 2: Structure and organisation of Java classes

We based on the MVP architecture for the application to be easily extensible and easy to maintain, we need to define well separated layers. Moreover, it provides a clear and simple structure so that future changes can be made more trivially and less complicated. There are many variations of MVP and everyone can adjust the pattern to their needs, which is why we have structured it as four components in this way [9]:

- **Activity Java class:** Responsible for rendering UI elements.
- **View Interface:** Responsible for loose coupling between view and model.
- **Presenter:** Responsible for view and model interaction.
- **Model:** Responsible for business behaviour and state management such as the provider of the data we want to display in the view from the database.

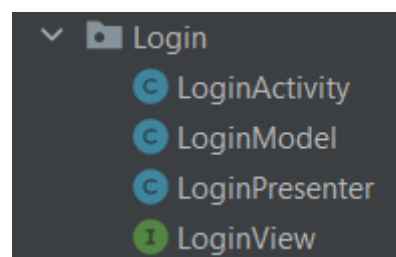


Figure 3: Structure and organisation of Java classes

This section will explain each package and its contents:

- **Data:** This package (see *Figure 4*) contains java classes used to store and retrieve values from database tables (Sticker, Task, User). Also to obtain application information (App) and for language or translation purposes (LocaleHelper).

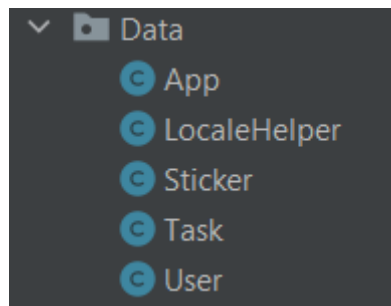


Figure 4: Structure of data

- **Calendar:** This package manages the user's calendar and pending tasks.

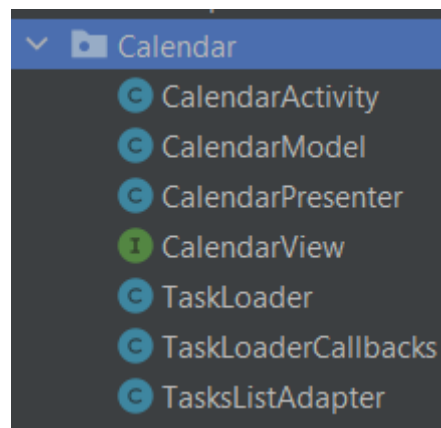


Figure 5: Structure of Calendar

- **Instructions:** This package contains the classes that handle the instructions to be followed by the user when registering in the app.

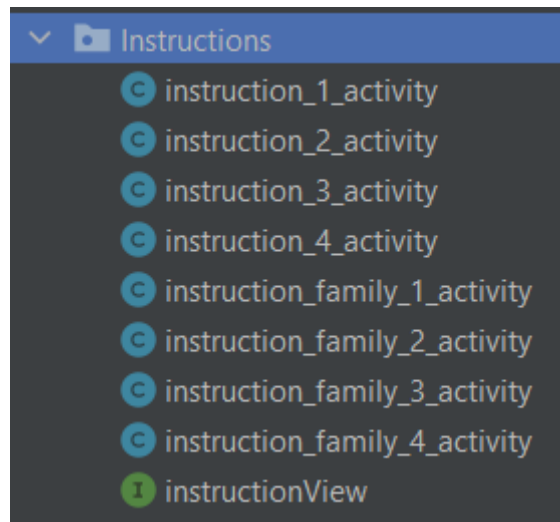


Figure 6: Structure of Instructions

- **Settings:** Saves user preferences, language, colour, notifications...
- **Task:** This package manages everything related to tasks.
- **Login:** It handles user login and manages the user's main data such as name and password.
- **Stickers:** This package manages everything related to stickers.
- **Register:** This package stores all user data, which will later be used by many of the app's functions.
- **Utils:** Here you manage all the functionalities of the bar below.

4.2.3 Sticker Set

In designing the sticker set (see *Figure 7*), the priority was to ensure that stakeholders could see them from a distance. Since the main users are elderly people, it was important to take into account that with age vision deteriorates and there are problems with distinguishing colours and outlines.

Therefore, complementary colours of bright hues (purple and yellow) were chosen for the stickers. These colours create a high contrast, making them easier to identify.

Since the shapes of the drawings had to be completely different and without small details, variations of the images of the letters H, O and C were used. From simple geometric figures to abstract patterns.

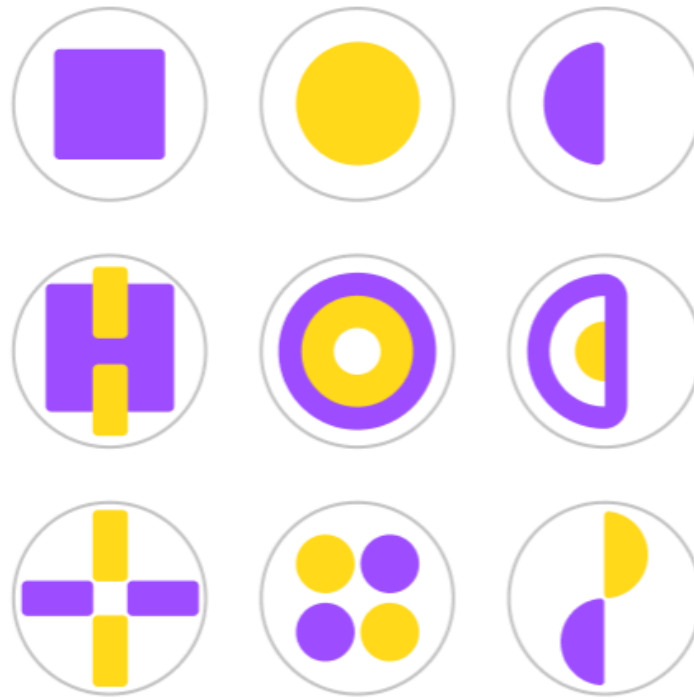


Figure 7: The sticker set

4.2.4 HOC Device

The HOC is a smart object designed for the elderly. It must be a helper and a facilitator in some tasks. Such as a voice call, a notification system, weather information, reproduction of music among others.

Regarding the hardware part, the different components used to build this project can be seen in the 4.1. We would also like to remark the following connection schema:

In contrast, regarding the software, the implementation consists of a python script that takes and executes the tasks that are scanned via the stickers by the elderly. The process would consist of different parts. First, as mentioned before, it would be needed to catch the HOC and scan one sticker. After scanning one sticker, the action must be confirmed via a button. If it is confirmed, it would proceed to do the action that is supposed. If not, the process would be finished. It is important to remark that the HOC is not the only technological solution, it has an important Firebase Database and Application. Obviously, to consult which action is needed to do, it would be obtained via the Database.

The Voice Call System (VoIP) has been a very complex problem to implementate due to the hardware specifications of the Raspberry Pi Zero. The conventional apps (Telegram, Zoom) have been a difficult to install. So, the best way to solve the problem was to use Jitsi Meet or similar (Google Meet, WorkShopX). Again, due to specification problems of the Raspberry (the lack of space at disk) it have been impossible to connect the HOC in that webpages via Selenium and ChromeDriver. The solution found have been done in the app. As, when it is need to call, it is send a link on the elderly phone and also on the receipt phone (it must be registred on the app).

The implementation of the Music Service has been split. The installation of Spotify have been achieved on the Raspberry. The main problem found is that the need to control the reproduction of the music with the phone we have found another solution. Music therapy is one of the most effective methods. So , right now is also implemented a music script that selects one song that have been selected and downloaded from a list.

The Weather Information given needs to know the coordinates of the elderly, obtained thanks to the app that save the information into the database. The information obtained are the actual temperature, the general day prediction, the maximum and minimum temperatures (how it feels), the humidity and if it is raining right now. All this have been done using the pyowm library of Python (Python Open Weather).

Another implementation made is the alarm, that catches the information and alarms from the database calendar. Another one, is the calendar that the HOC reads for the elderly.

3D-model of HOC Device

The 3D model (see *Figure 8*) was created using Autodesk Fusion 360 software. The device has a shape that allows it to hold comfortably in the hands.

There are holes:

- for the USB port
- for the LED
- for the speaker
- for the microphone
- for the button

Supporting elements have been created, which will contribute to the reliable fixation of the internal parts (such as the button and the speaker).

Also thought of the bottom removable cover, which will cover the RFID-controller when necessary.

The size of the device:

- 15 cm in height
- 10 cm in width



Figure 8: Elements of the 3D-model of HOC Device

4.2.5 Database

The structure and data of the created database consists of documents (which are basically key-value stores) and collections (which are collections of documents). Documents will also frequently point to subcollections, which contain other documents, which themselves can contain other documents, and so on. As you can see FigureX, we design a MySQL database design in order to be more comprehensive as a set of collections and documents could become complicated to understand their structure.

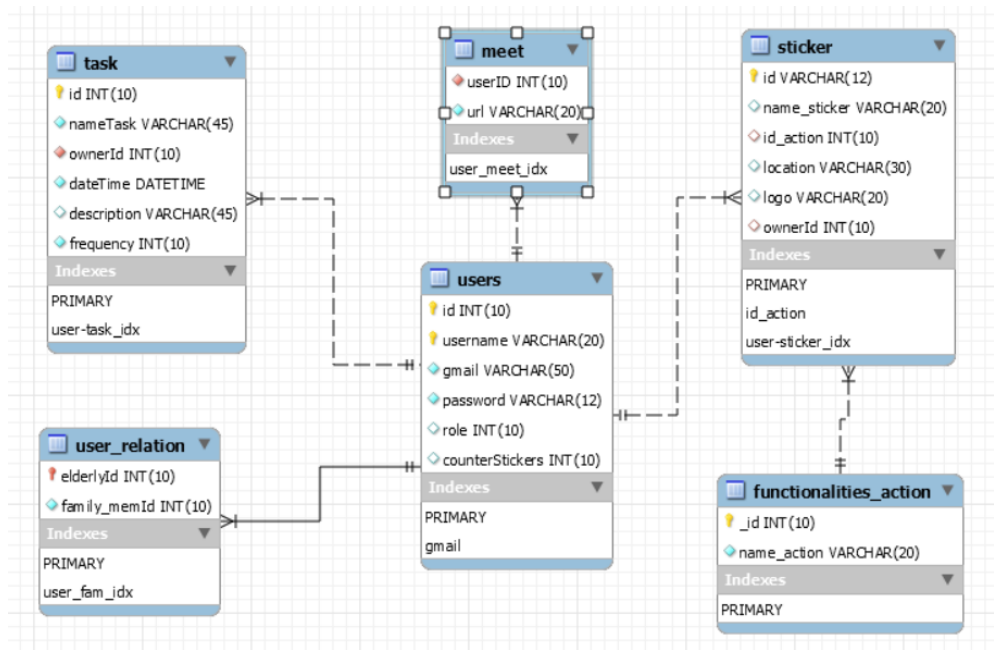


Figure X: Entity Relationship Diagram (ERD) of MySQL Database Design

hoc21-3f4e6	sticker	136,4,11,52
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
location	136,4,11,52	+ Agregar campo
meet	136,4,167,17	id: "136,4,11,52"
sticker	136,4,171,17	logo: "sticker2"
task	136,4,175,17	
user	136,4,178,16	
	136,4,182,16	
	136,4,195,17	
	136,4,2,52	
	136,4,213,17	
	136,4,7,52	

Figure 9: Firebase Database Structure

User Table stores all the users registered and it is made up for:

- **id**: Assigns an unique numerical identifier to each user.
- **username**: Attribute which sets the username of the user.
- **password**: Attribute which sets the password of the user.
- **gmail**: Attribute which sets the gmail of the user.
- **role**: Attribute which sets the role of the user. (0 = elderly, 1 = family member)
- **counterStickers**: Attribute which sets the number of the sticker a user can have.

User		
Name	Type	Schema
id	INTEGER	PRIMARY KEY AUTOINCREMENT
username	TEXT	PRIMARY KEY
gmail	NUMERIC	NOT NULL
password	NUMERIC	NOT NULL
role	NUMERIC	NOT NULL
counterStickers	NUMERIC	NOT NULL

Table 1: Users Table structure

Sticker Table contains a list of the existing stickers and it is composed of:

- **id**: Assigns an unique numerical identifier to each sticker. (Identifier already defined in each sticker RFID)
- **name_sticker**: Attribute which sets the name of the sticker.
- **id_action**: Differentiates the type of action a sticker implements. (Figure V)
- **logo**: Attribute which sets the logo or image of the sticker.
- **location**: Attribute which sets the location of the sticker.
- **ownerId**: Contains the id of the user who created or owns the pet.

Sticker		
Name	Type	Schema
id	TEXT	PRIMARY KEY
name_sticker	TEXT	NOT NULL
id_action	INTEGER	FOREIGN KEY REFERENCES Functionalities_Action(_id)
logo	TEXT	NOT NULL
location	TEXT	
ownerId	INTEGER	FOREIGN KEY REFERENCES User(id)

Table 2: Sticker Table structure

Functionalities_Action Table stores all functionalities a sticker can have and it is made up for:

- **_id**: Assigns an unique numerical identifier to each functionality.
- **name_action**: Attribute which sets the functionality name of the action.

Functionalities_Action		
Name	Type	Schema
_id	INTEGER	PRIMARY KEY
name_action	TEXT	NOT NULL

Table 3: Functionalities_Action Table structure

_id	name_action
0	Phone Calls
1	Play Music
2	Weather Information
3	Calendar Events
4	Alarm Clock
5	Current Time

Figure V: Existing functionalities for stickers

Task Table has the task created for an specific user (family member) and it is made up for:

- **id**: Assigns an unique numerical identifier to each task.
- **nameTask**: Attribute used for setting the name of each task.
- **dateTime**: Attribute for storing the date and time of the scheduled task.
- **description**: Contains a description of a task.
- **frequency**: Sets a frequency for a task (Daily, Weekly, Monthly and Yearly or Today)
- **ownerId**: Contains the id of the user to which a task has been assigned.

Task		
Name	Type	Schema
_id	INTEGER	PRIMARY KEY
nameTask	TEXT	NOT NULL
dateTime	DATETIME	NOT NULL
description	TEXT	
frequency	INTEGER	NOT NULL
ownerId	INTEGER	FOREIGN KEY REFERENCES User(id)

Table 4: Task Table structure

Meet Table stores all the users registered and it is made up for:

- **userId**: Assigns an unique numerical identifier to each user.
- **url**: Attribute which sets an url for a meeting call for a user.

Meet		
Name	Type	Schema
userId	INTEGER	PRIMARY KEY
url	TEXT	NOT NULL

Table 5: Meet Table structure

4.3 Usage

In actuality, the usage of HOC is conditioned to be connected to a computer due that the execution of each action must be done by the execution of a program, and if we want to do another action we need to execute the same code.

To explain it better, inside the HOC there is a script that contains the name exam, must be something `script_exam.py` or `main_exam.py`. This script must be executed when the server is active. To have it active we must run the server code on an external computer (`server_2.py`, that can be found in the Code folder). Then, when it is running, the HOC device must run the script `main_exam.py` so when a sticker is connected the action can be performed. If it is needed to scan another sticker, then it must run again the script `main_exam.py` without the need to restart the server.

The steps can be resume in:

1. The script `server_2.py` must run on a computer that is not the HOC
2. Connection of the HOC to a computer via ssh (`ssh pi@ip.address`). The computer and the HOC must be connected to the same network.
 - a. To connect the HOC to Wi-Fi, it needs an ssh (*empty file with name ssh*) and `wpa_supplicant.conf` file. (vid. Code folder)
3. On a computer (with no relation with the HOC) the script `server_2.py` is run.
 - a. While the script it is running, on the HOC we execute the `main_exam.py` script.
 - i. Then we proceed to scan the sticker desired.
 - ii. The action is done.
 - iii. The `main_exam.py` finishes the execution. (Note: If the music action is performed, it must listen to the whole song before finishing the execution).
 - b. The point 3. a it is repeated as many actions are desired to do.

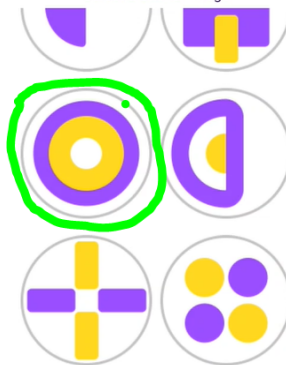
Regarding the mobile Application the HOC, we created two test users:

- Elderly
 - Username: elder
 - Password: 1
- Family member, which owns the elderly user
 - Username: family
 - Password: 1

You can create, delete or edit stickers and check thier functionality by scanning with the HOC device. Some of the will give some configuration problems such as the one selected in green, maybe it would be an issue of the RFID sticker.

EDITING STICKER IMAGE

Select a sticker image



At the moment of creating or modifying an existing sticker with phone call action, we added a new field so that the user, in this case a family member, could set an specific phone number.

A new sticker

Name

Action:

Phone Calls

Sticker location (optional)

Phone number

NEXT STEP

If a family member user wants to create a new phone call sticker or edit an existing sticker with a different functionality to change it to phone call action, a notification will appear warning the user that the application allows just one phone call sticker per user.

Once created the phone number will be stored in the firebase database, it will always store the last value so if you change the phone number of an existing one, the new one will be set as a new entry in the database.

5. State of the art

These products make similar sense and are competitive works, but do not achieve the goals that our project achieves. Related works/projects/products in the research or market arena:

- **Alexa**

It is a virtual assistant that responds to voice commands and performs user tasks. With the help of the Echo speaker, equipped with the Amazon Alexa virtual assistant, you can: make to-do lists and activities, give the device smart home management tasks, find out the weather, news, sports results, interesting facts, call a cab, listen to music and create playlists on Spotify, make cocktails using The Bartender extension and so on. Alexa's capabilities are initially limited to a standard set of features, but any user can develop their assistant using Skills extensions.

- **Google Nest Hub (2nd gen)**

It's a smart voice assistant with a 7-inch touchscreen display that acts as a speaker for music and radio, a photo frame, a clock, and a hub for controlling smart home devices that support Google Assistant. It also uses low-power Soli radar technology to track your sleep and detect non-camera movements (such as certain gestures). Setup is done through the Google Home Hub app on your phone or tablet.

- **Siri**

It is a personal voice assistant built into Apple devices. You can use it to answer calls and messages, create notes and reminders, set alarms, route, listen to music, and perform minor tasks such as translating phrases, doing calculations, and so on. Also, supports Apple's smart home system. Siri is a useful assistant that minimises the time spent on everyday tasks. Machine learning allows the algorithm to improve and adjust to the needs of the owner.

- **Xiao Ai**

This is an artificial intelligence-based voice interaction system that belongs to Xiaomi and is also its virtual AI, which is also compatible with its various smart IoT products. Just say a single phrase to send a message, find out all the details about traffic jams, get information about the weather outside the window, listen to the latest important news, record a reminder, set an alarm for a certain time, call a cab, and so on. In addition, the Xiao Ai voice assistant is great at managing a smart home.

6. Value Proposition

Initially, we wondered to what extent the elderly, who often do not react very positively to the implementation of various technologies in their lives, would be ready to accept the new device - "The HOC"? So we were challenged to create a system that the main user would really be interested in and want to use in his or her daily life.

"The HOC" is a good solution because it satisfies the needs and closes all kinds of fears of the end user (the elderly person), which these needs can not be found in the current market so that it could be focused on a more global market. After all, many people in this category are constantly haunted by fears such as being confused by the buttons, not understanding the technology, or breaking something with their wrong action. But now older people will not have any questions about the sequence of buttons that need to be pressed to achieve results. All it takes is pointing the device at a particular sticker. The device is like a guide for the older generation to the world of technology and new possibilities.

"The HOC" is a system that allows you to use the main functions of your smartphone without any problems thanks to a simple and clear control based on scanning RFID stickers.

Competitors for this system are various voice assistants (both embedded in smartphones and separate devices). And the advantage of the system "The HOC", created by us, is the possibility to help with the personalization remotely. It is easy and intuitive thanks to the user-friendly interface. Synchronising the mobile app with your device will help you make changes quickly, no matter where you are. And just that contributes to the needs of users in the category of family members who live far away but want to be able to help an elderly person.

6.1 Main difficulties encountered

- The first database was implemented in MySQL, we managed to establish a connection between the application and the MySQL server that contained the database but the problem was the version of the connector we were using (mysql-connector-5.1.4 this version was the only one functional and compatible with Android Studio) that allowed just one connection, leaving the other requests to the database blocked. Finally we decided to replace and create it in Firebase, which allowed us a much more efficient and faster response time, in addition to its simple structuring in documents and collections. Not only did it extend our connection limitations to a public IP, but it also offered trivial connections using python, the language in which the HOC device is implemented.

7. Future work

This chapter reviews the work carried out by the team and the result of the project. It also discusses some improvements the team didn't have the time to carry out for the three types of analysis and brainstorms of future work that could be done to give a suitable closure to the application.

7.1 Improvements

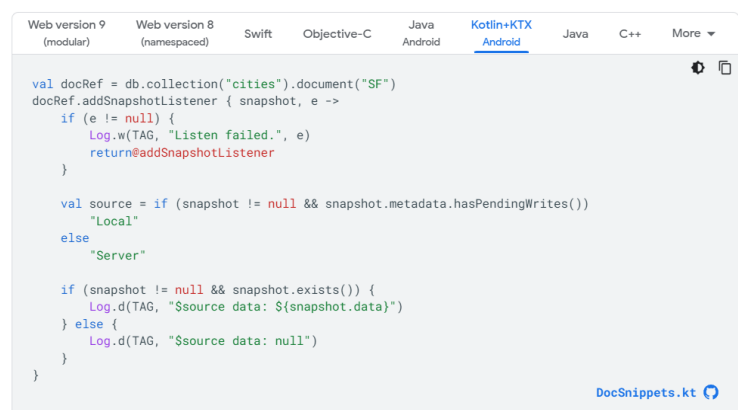
This section reflects various improvements of the different analyzers that we have not had enough time to carry out despite having the knowledge to be able to implement them.

- Implement a method in the register section in order to force the user to create a more secure password such as a minimum number of characters, requiring a symbol and even capital letters.
- Implement the cross-platform application as well as make an iOS compatible version.
- We have configured only 10 RFID stickers for the use of the HOC device as a test, in addition we have created a per-user limitation of 10 stickers for the bd and the mobile application. It would be very trivial to add more stickers by scanning with an RFID reader and get their id, then add a new entry to the database and design new logos or images.
- Create in each activity or window of the application, an auto-refresh for the correct obtaining of the data retrieved from the database by implementing the code seen in Figure 33. Because sometimes there is a small milliseconds delay due to the database connection, so that the user has to refresh the page by clicking again on the section or or by making a section change so that it then returns to the desired updated section.

Events for local changes

Local writes in your app will invoke snapshot listeners immediately. This is because of an important feature called "latency compensation." When you perform a write, your listeners will be notified with the new data *before* the data is sent to the backend.

Retrieved documents have a `metadata.hasPendingWrites` property that indicates whether the document has local changes that haven't been written to the backend yet. You can use this property to determine the source of events received by your snapshot listener:



```
Web version 9 (modular) Web version 8 (namespaced) Swift Objective-C Java Android Kotlin+KTX Android Java C++ More ▼

val docRef = db.collection("cities").document("SF")
docRef.addSnapshotListener { snapshot, e ->
    if (e != null) {
        Log.w(TAG, "Listen failed.", e)
        return@addSnapshotListener
    }

    val source = if (snapshot != null && snapshot.metadata.hasPendingWrites())
        "Local"
    else
        "Server"

    if (snapshot != null && snapshot.exists()) {
        Log.d(TAG, "$source data: ${snapshot.data}")
    } else {
        Log.d(TAG, "$source data: null")
    }
}
```

DocSnippets.kt

Figure 33: Existing functionalities for stickers

- We have designed that the created tasks remain in the database in case the elderly user forgets what he/she did in previous days. However, only the family member user is in charge of deleting and adding tasks. So in case we want to optimize the database, we could delete the tasks by monitoring them with a 48 hour timer for example.
- We have designed a 3D model for the HOC device, but due to the unavailability of laboratories to print on a 3D printer, we have been forced to create an "arts and crafts" model.
- To achieve a direct connection between the HOC and the firebase, that has been very difficult to achieve due to the space limitations.
- Implement the voice call system completely on the HOC for not using the elderly phone. Right now, we consider the elderly phone as the voice call systems achieved have been very complex and slow, so to increase the portability and usage of the HOC has been decided to go with this solution instead.
- Get a better alarm system that the elderly can control via voice, rather than using the calendar of the app.
- We just implemented the app in such a way that a user can create just one phone call sticker. It would be trivial to add more than one but we did not have enough time to implement it.

7.2 Future directions

This work has been carried out over approximately six months, which is the expected time for the development of this project. However, the work done here can be improved by extending its utilities as indicated below.

- When a user creates a task, and at the moment the "Pending Task" pop-up notification appears, knowing that the application is closed, the application can be accessed through the notification by simply clicking on the pop-up, showing the task information of the notification.
- As for the call system, we would have liked to improve its call processing performance and efficiency:
 - Implementing a new functionality/action called VideoCall to separate the phone call from the video call, which are now together as the same functionality.
 - Produce call or call pop-up notification (with the online meeting link) to other users who do not own or have installed The HOC application.
 - The call notification can be received while the app is closed.

- Implement the call system directly on the HOC with a microphone, better speaker module and increasing the memory to be able to perform all the calls without the use of the app.
- The HOC device could connect to the wireless WiFi network of the elderly user in their residence or home, so that he/she could configure their wifi connection to their phone by downloading the mobile application and then registering as can be seen in Alexa or Google Home.
- Solve the connection problems between the server and the HOC, having the program execute for more than one action, to scan more than one sticker a time.
- The implementation of a confirmation button to not perform an action if the elderly has scanned it by error.
- The correct implementation of the alarm system either via consulting the alarm on the app calendar or directly saving it on the raspberry.
- The humanization of the voice that is reproduced by the HOC when the Weather sticker is scanned.

8. Bibliography

- [1] Amazon на миллион, 2022. Amazon Alexa — что это за устройство и как его можно использовать.. [online] Амазон на миллион. Retrieved from <https://amzmln.com/stati/pokupki-na-amazon/amazon-alexa-cto-eto/>
- [2] *Android*. Retrieved from <https://www.android.com/>
- [3] CCN-CERT IA-04/19. (2019, January). *Informe Anual 2018 Dispositivos y comunicaciones móviles*. Retrieved from <https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/3464-ccn-cert-ia-04-19-informe-anual-2018-dispositivos-moviles/file.html>
- [4] Google Nest Hub (2nd gen) review: Sleep sells. (2021, April 5). [online] Retrieved from <https://websetnet.net/google-nest-hub-2nd-gen-review-sleep-sells/>
- [5] Google Nest Hub (2-е поколение) против Nest Hub - стоит ли обновляться? (2021, March 30). [online] Retrieved from <https://www.tremplin-numerique.org/ru/google-nest-hub-2e-generation-vs-nest-hub-devez-vous-mettre-a-niveau>
- [6] Introduction to Autodesk Fusion 360. Retrieved from https://platform.europeanmoocs.eu/course_introduction_to_autodesk_fusion#:~:text=Fusion%20360%20is%20a%20cloud,manufacturing%20in%20one%20comprehensive%20package.
- [7] It was really helpful consulting the blog for any kind of debug error. Retrieved from <https://es.stackoverflow.com/>
- [8] The MVP (Model View Presenter) architecture. Retrieved from <https://antonioleiva.com/mvp-android/>
- [9] We followed the steps of this website to develop the translation of the app. Retrieved from <https://www.geeksforgeeks.org/how-to-change-the-whole-app-language-in-android-programmatically/#:~:text=Go%20to%20app%20%E2%03res%20%E2%03values,from%20the%20drop%2Ddown%20list.>
- [10] We have taken as a reference and guide the official site specialised in the design of applications for the Android market. Retrieved from <https://developer.android.com/>
- [11] We took the idea of the implementation of the calendar of this github user. Retrieved from <https://github.com/Applandeo/Material-Calendar-View>
- [12] What Is Figma? a 101 Intro. Retrieved from <https://designshack.net/articles/software/what-is-figma-intro/>

[13] Xiao AI: Голосовой помощник от Xiaomi (2021, February 26). [online] Retrieved from <https://iptv-russia.ru/download/xiao-ai/>

[14] Xiaomi Xiao AI регистрирует более 100 миллионов активных пользователей ежемесячно (2021, August 17). [online] Retrieved from <https://astera.ru/news/xiaomi-xiao-ai-registriruet-bolee-100-millionov-aktivnyh-polzovatelej-ezhemesyachno>

[15] Кто такая Сири и что она умеет: обзор возможностей голосового помощника (2021, February 4). [online] Retrieved from <https://blog.calltouch.ru/kto-takaya-siri-i-cto-ona-umeet-obzor-vozmozhnostej-golosovogo-pomoschnika/>