Fast & Furious Report
Ubiquitous & Mobile Computing Systems 2023/24

Miguel Almeida
João Pedro Pereira
Gonçalo Rodrigues

Monday 10th June, 2024

Contents

1 Introduction

The goal of this project is to develop a ubiquitous and mobile system that can act upon client action or by automatic behavior triggered by some sensing information from the environment. The system must enable the communication between clients and ubiquitous components, and between the components themselves.

The application must present the values of the sensors in a friendly way to the user, and allow the user to interact with the system. The system must also be able to act upon the environment, by controlling actuators. The ubiquitous application must have actuators and sensors that can be controlled by the system, must be able to communicate with the system and be able to perform autonomous task triggered by sensing the environment or by command from the remote client.

Our objective is to assist users and managers of various parks, underground or outdoor, with a simple and accessible application.

With simple and common actuators, we can be able to tell drivers that a spot is occupied, empty or reserved. Each spot has a sensor node that also tells the driver when to stop, informing that car is well parked. Our mobile application enables users to reserve a spot, check the park availability, save favorite and most used parks to easy and fast travelling, and other features.

This system connects mobile applications and ubiquitous components, which allows users to interact with a variety parks, by getting information given by the ubiquitous devices or by giving instructions to them.

2 General Overview

The system is composed by 3 main components: the mobile application, the server and the ubiquitous components. There can be multiple mobile applications and multiple ubiquitous components, but only one server.

The user will mostly interact with the mobile application, which will be the main interface to the system. The mobile application will be able to communicate with the server. The server will also be responsible for managing the communication between the mobile application and the ubiquitous components. The server has a simple design, as it only functions as a database that stores and updates the information given by the ubiquitous components and the mobile applications. The ubiquitous components will be responsible for sensing and retrieving information from the environment and actuating upon it or upon the user's request.

The user interacts with a variety of parks, and the information given by each ubiquitous device is also shared with multiple users, guaranteeing an N to M relationship between users and parks.

Each park can have multiple ubiquitous components, one for each parking lot that the park has and an extra device for the entrance. Each device will be equipped with distance and movement sensors and LED actuators (*red*, *yellow*, *green* and *RGB*). The entrance device will only be equipped with a LED display.

The user will be able to interact with the system by reserving a spot, checking the park availability, saving favorite and most used parks, and other features. The ubiquitous devices will be able to change the LED color according to the spot status, and the entrance device will display the park availability. Every action performed by the user or the ubiquitous devices will be stored in the server, and updated in the ubiquitous device(s) or in the mobile application, respectively, afterwards.

3 System Architecture

As said before, our objective is to allow users to reserve a parking spot or check the availability of a certain parking lot, only by checking the mobile application and from everywhere, anytime. To allow this behavior, we have to design a system capable of maintaining this information updated and available to every user and ubiquitous devices, as well as allow communication between each other. We did not have into consideration possible security issues.

We used a centralized server system, where mobile applications and ubiquitous devices are connected with and by a centralized server that is capable of storing and updating the information given by the devices and the users.

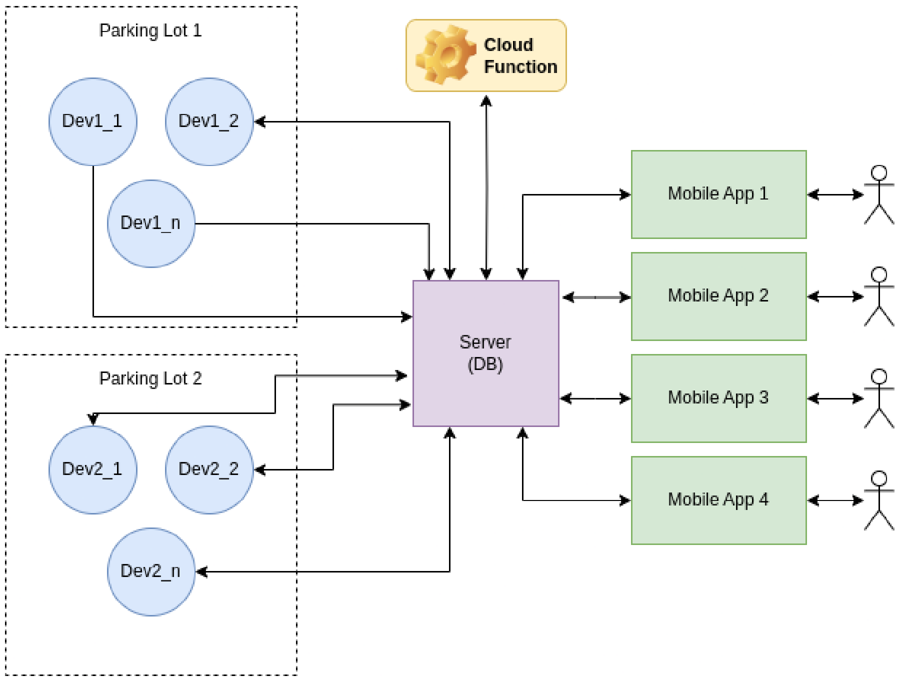


Figure 1: System Architecture

In *Figure 1*, we can see the system architecture.

Each *Parking Lot* can have multiple *Parking Spots*, each with a *Ubiquitous Device* (Dev).

Each user, represented by an *actor icon*, can interact with the system by using the *Mobile Application* (App). Each mobile application is connected to the server.

The *Server* (DB) is responsible for managing the communication between the *Mobile Application* and the *Ubiquitous Devices*, therefore is connected to all of them. We realize that this system is not scalable, as the server can become a bottleneck, but for the purpose of this project, we believe that this system is enough. To maintain the database updated, the group designed a cloud function that updates the information of the status of each parking spot according to the bookings, and also works as garbage collector, deleting old and prescribed bookings.

Both the *Mobile Applications* and the *Ubiquitous Devices* communicate with the server through a *REST API*, which is a simple and easy way to communicate between devices and servers.

3.1 Ubiquitous Device Architecture

The *Ubiquitous Devices* are responsible for reading the environment of the parking lot and its spots and giving this information to the server and to the users.

To sense the surroundings, each device is equipped with a *distance sensor* and a *movement sensor*. In order to output the information in the database to the "real world" or from the "real word" itself, each device is equipped with 4 LEDs. More details on this on the *Components* section.

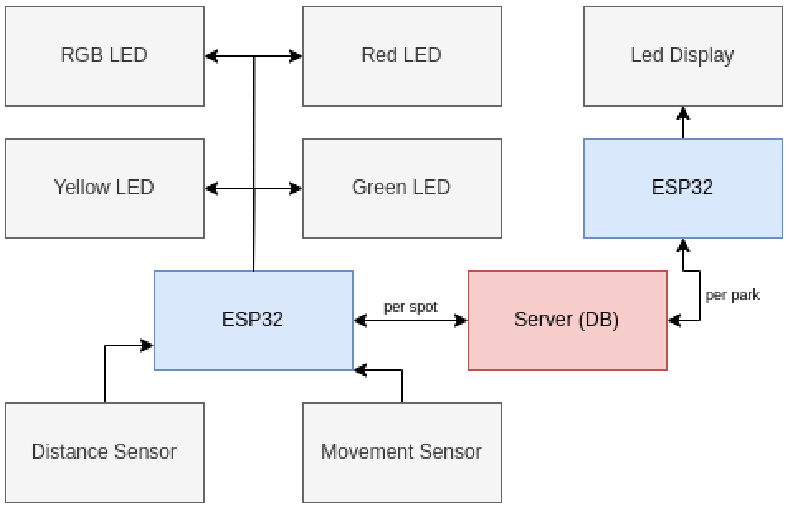


Figure 2: Ubiquitous Device Architecture

3.1.1 Components

I. **ESP 32**

ESP 32 are the microcontrollers used for each device needed for the system to work. We chose the ESP 32 over the Arduino due to its lower price and the already integrated Wi-Fi Development Board. This is the main element of each device because it connects all sensors and actuators, receives or executes information and communicates with the back-end.

II. **LEDs**

The RGB LEDs indicate which parking spots are being used, free and reserved. This way, when a certain user is stressed roaming the park looking for a spot, he can easily identify which spot to park. We used the RGB LEDs for each parking spot because the status of each spot can and will change.

The red, green and yellow LEDs are directly tied to the distance sensor. We used these 3 colors to express to the person in the car, whether he could safely continue to advance towards the wall (green), whether he should be careful (yellow) and whether he should stop the car (red). The microcontroller receives the distance from the sensor and according to the value, he turns one of the 3 lights. Those limits will be defined and need to be changed according to the scale.

III. **LED display**

The LED display is used to indicate the occupancy of the parking lot. This is calculated by checking if there are any spots available, set to free, or if all spots are reserved or occupied, set to occupied. This is done by the entrance device, which is the only device that has this display.

IV. **Movement Sensors**

Motion Sensor are used to allow a lower power consumption. These sensors read the environment from time to time and, when they sense movement, they inform the microcontroller and therefore the microcontroller turns on sensors like distance sensors and actuators like the 3 LEDs (red, yellow and green).

V. **Distance Sensors**

Distance sensors are used to get the distance between a car and a wall or obstacle. This information is directly connected with the 3 LEDs (red, yellow and green) through the microcontroller.

3.2 Technologies

It is also important to mention the technologies used to develop both the mobile application and the ubiquitous devices, in order to better understand the real function and workflow of the system.

3.2.1 Firebase

Firebase is a *Backend-as-a-service* from *Google* that works as a centralized server with Database and Authentication Services. With this backend, we don't need to worry about building a backend and as so, we can focus on our mobile application and building our parking devices.

This centralized server makes the communication between users and devices easier. When devices have new information – a car has left, for example – they will push these changes to the database. When a mobile application device logs in, they pull the newest information directly from the database and won't lose any information (not considering the delay of writes in database and travel times). This means that devices in the parking lot and mobile apps won't communicate directly with each other but will use the Firebase infrastructure as a "proxy".

We used 2 database services from Firebase: *Firestore* and *Real Time Database*. The Firestore Database is a relational SQL database used to store all data necessary for the mobile application. This database holds the information about the users, to allow authentication, and about the parking lots, like name, description, location and other information essential for the mobile application.

However, ESP 32 cannot connect to firestore, so we used Real Time Database, a non-relational NoSQL database that stores information in JSON format, and that hold the information of the status of all parking spots, of each park.

Firebase also offers an Authentication Service that allows users to login with their account that eases the authentication process of our app and the tracking of users' profiles.

3.2.2 React Native

React Native is the technology used to produce our mobile application for both Android OS and iOS. Although the teacher recommended using Flutter, we decided to use this framework because we had previous knowledge of it and thought it would be easier to implement the application with JavaScript instead with Dart, a language none of us are familiar with.

4 Mobile Application

Our mobile application is the main interface between the user and the system. It allows the user to interact with the system and reserve a spot, check a certain park's availability or save a park as a favorite.

4.1 Initial Page

The initial page is the first page that the every user sees when he opens the application. This page allows the user to sign in or register on the application. This page is composed of 4 simple components: a small welcoming text, the application's logo and 2 buttons, one that opens the *Sign-in* page (4a) and other that opens the *Register* page (4b). More details on the next section.

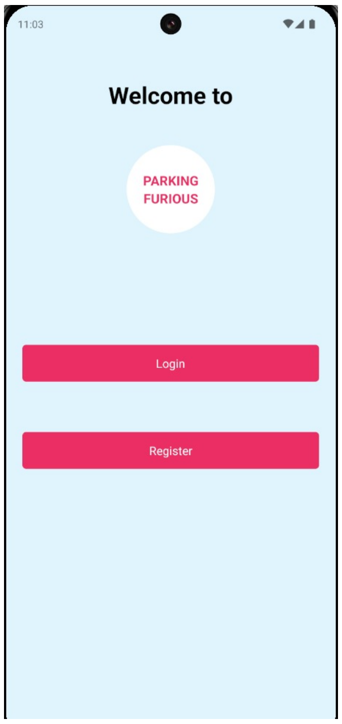


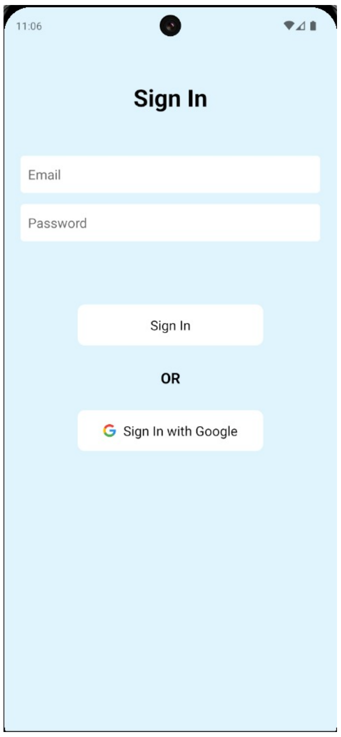
Figure 3: Initial Page

4.2 Sign in & Register Page

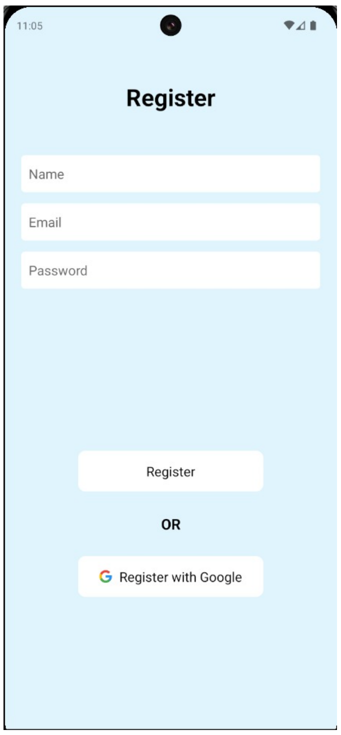
In order to authenticate each user, we have 2 pages to handle this process. The *Sign-in* page allows users to sign in with their account, while the *Register* page allows users to create a new account.

The *Sign-In* page is composed of 2 text input fields for the email and password of the user, and a button to sign in. The *Register* page is composed of 4 text input fields for the name, email a password and a button to register.

After this step, the user or the administrator will be redirected to the “Search Page”.



(a) Sign in Page



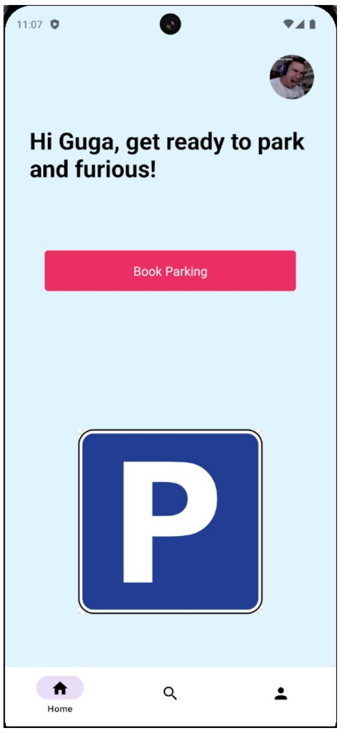
(b) Register Page

4.3 Landing Page

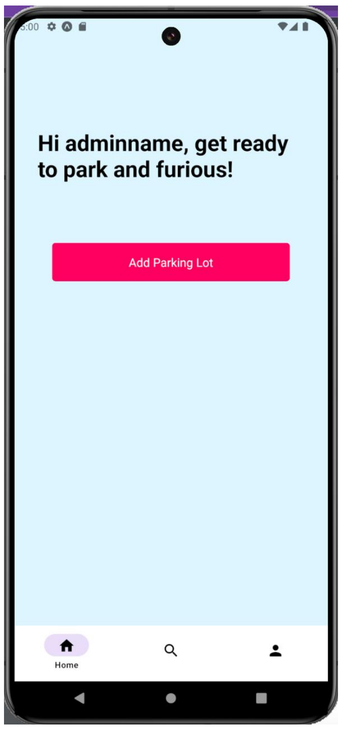
After the authentication process has ended and the user is logged in, he will be redirected to the "Landing Page". In this page the user is able to navigation between various functionalities. From this point on, the user will be able to access the full functionalities of the application, according to their rule (*administrator* or *user*)

The page is composed of 3 main components: the users' avatar, a simple text, a park logo, a footer navigation panel and a button. If the user is an admin, the button lets the user add a parking lot, which forwards him to the *Add Park Page*, otherwise the button forwards the user to the *Booking Menu* Page. More details on both pages in the next sections.

The navigation panel allows the user to go to the *Search Menu* Page or the *Profile* Page, or back to the Landing Page.



(a) Landing Page



(b) Admin Landing Page

4.4 Add Park Page

This page is exclusive to administrators and allows them to add a new park to the system. This page is composed with 5 text input fields for the name, location, description and link to a photo of the park. It also has 2 buttons: one to confirm and other to go back to the landing page.

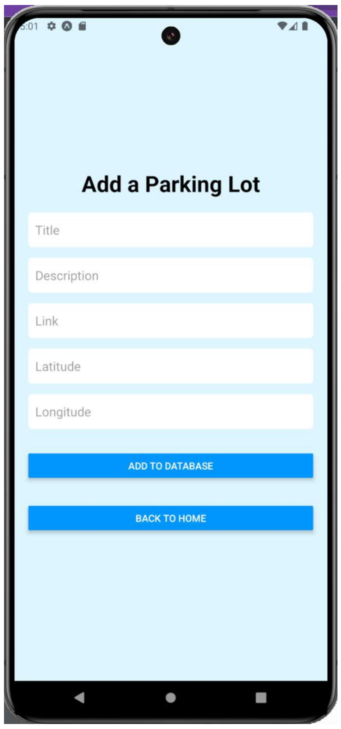


Figure 6: Add Park Page

4.5 Booking Menu

The *Booking Menu* is a simple page that allows user to book a parking spot in a certain park and hour.

The page has the list of parks that the user can book a spot, a button that opens a menu for the user to select the day and time of the booking.

This menu is composed of a list of button with the name of the available parks, a button to open the date menu, a button to confirm the booking and a ”back to home” button.

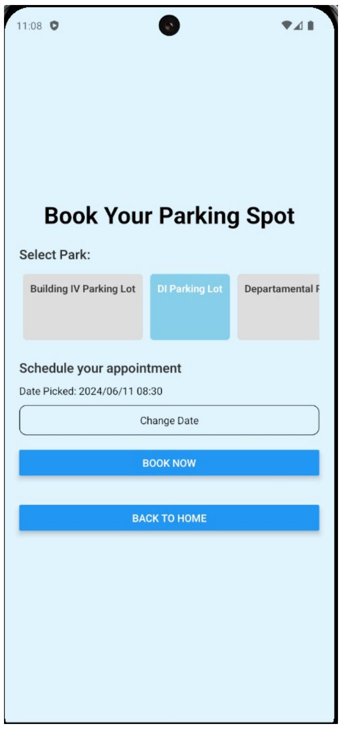


Figure 7: Booking Menu Page

4.6 Search Menu Page

In this menu, the user is able to search for parks that are registered in the application. The search can be done by location, name or favorites. Admins can also search for parks that they own. If the user wants to get the details of a park, he can click on the park in the search result and be redirected to the “Park Page”.

This menu is composed of a search bar and a list of the registered parking lots.

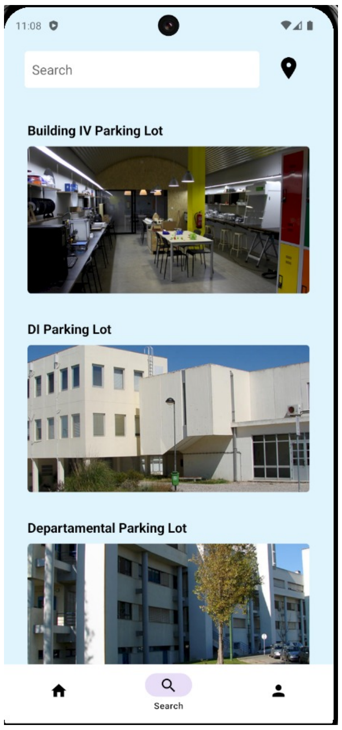
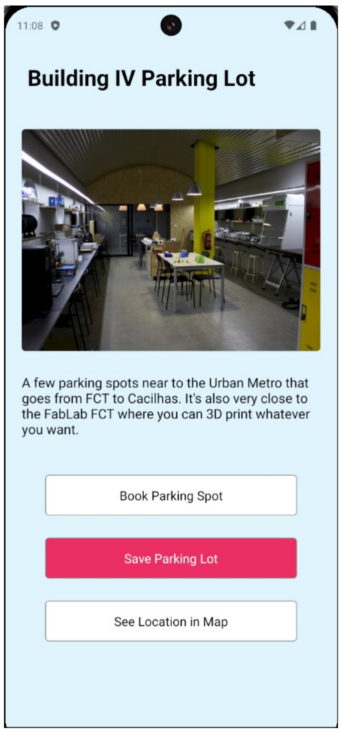


Figure 8: Search Menu Page

4.7 Park Page

Each park registered in our application will have a page dedicated to it. In it, there is general information and the user is able to make a reservation for a specific time and spot, changing the color of the LED actuator to blue, to indicate that the spot is reserved. When spots are reserved or when a car is absent, the devices notify the backend to show availability in real-time. The administrator can change the details about his park like the name, or the number of total parking spaces.

These pages are composed with a photo and description of the parking lot and 3 buttons: one to book a spot, one to save the park in the favorites and a button that forwards the user to the *Map Page*.



A few parking spots near to the Urban Metro that goes from FCT to Cacilhas. It's also very close to the FabLab FCT where you can 3D print whatever you want.

Figure 9: Park Page

4.8 Map Page

Each park is associated with a location to help users find the park. This page shows the location of the park in a map.

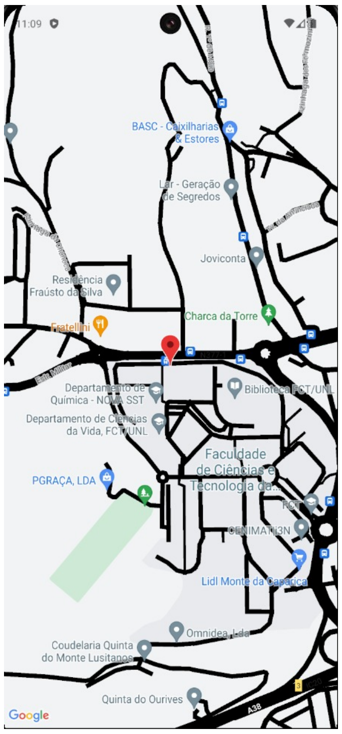

BASC - Caixilharias & Estores
Lar - Geração de Segredos
Joviconta
Residência Fraústo da Silva
Charca da Torre
Departamento de Química - NOVA SST
Departamento de Ciências da Vida, FCT/UNL
Biblioteca FCT/UNL
Faculdade de Ciências e Tecnologia da
PGRAÇA, LDA
Lidl Monte da Caparica
Omnidea, Lda
Coudelaria Quinta do Monte Lusitano
Quinta do Ourives
Google

Figure 10: Map Page

4.9 Profile Menu Page

This page allows users to customize their appearance and their user experience. They can change their nickname, their car license plate and settings like notifications. For admins, they can access their “Parks Page” that redirects them to the search page and allows them to search their registered parks.

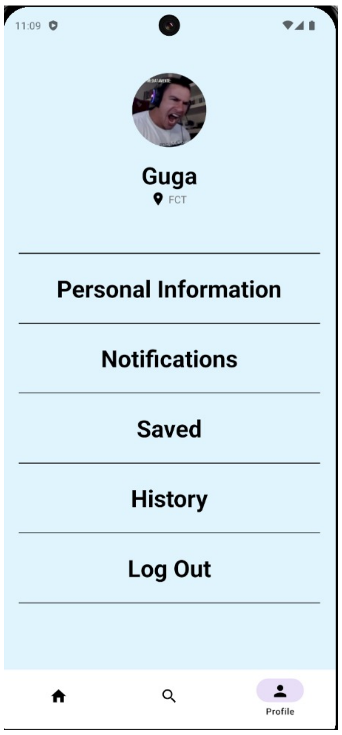
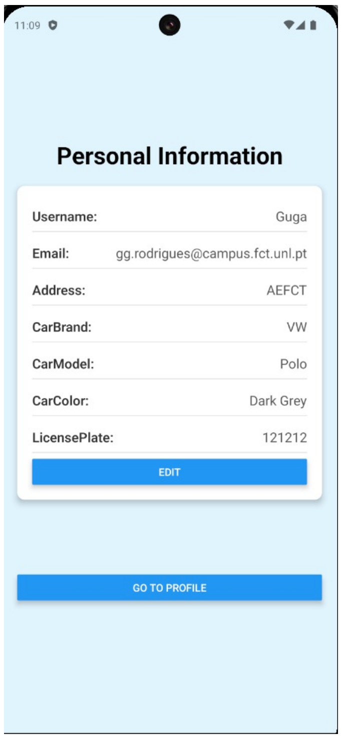


Figure 11: Profile Menu

4.10 Personal Information Page

This page allows users to see and update their personal information, like their name, email, address, and car's brand, model, color and license plate.



Username:	Guga
Email:	gg.rodrigues@campus.fct.unl.pt
Address:	AEFCT
CarBrand:	VW
CarModel:	Polo
CarColor:	Dark Grey
LicensePlate:	121212

Figure 12: Personal Information Page

5 Sensing & Reacting

As said previously in section 3.1, our system uses 2 types of ubiquitous devices per park: a single device, equipped with a LED display, and multiple devices equipped with LEDs and sensors, one for each parking spot.

The single device is used at the entrance of the park and is used to display the availability of the park. The device makes a request to the servers *Firebase API* to get the number of spots available in the park and displays it in the LED display.

The other devices are more complex and are equipped with sensors and actuators. These devices are the main devices of the system and the functionality of the system is based on them. Both movement and distance sensors are used to check if a car is present and the spot is occupied or not. When the sensors detect the presence of a car, the device updates the data of the park in the database to make the spot occupied. This is the only request made by the device to the database to update something. However, in order to the actuators to work, the device must request the server to get the updated information about the park and the user. When the user reserves a spot, the data about the spot, in the database, states that the spot is reserved. After this, when the device requests the updated information, it will know that the spot is reserved and will change the color of the LED to blue, indicating that the spot is reserved.

The devices also perform some tasks intra-device. The distance sensor is also used to check the distance of the car being parked and help the driver know the distance, by the use of the 3 LEDs. The red LED indicates that the car is too close to the wall, the yellow LED indicates that the car should stop, and the green LED indicates that the car can continue to advance. Is to be noted that the distance sensor is only used when the movement sensor detects a car, to save power.

5.1 Sensors Handling

The project uses 2 sensors per device, each produces different signals and these must be handled differently.

The movement sensor produces a signal when it detects movement.

The distance sensor produces a signal linearly proportional to the distance between the sensor and, in our case, the car.

5.2 Actuators Handling

The project uses, in total, 3 types of different actuators.

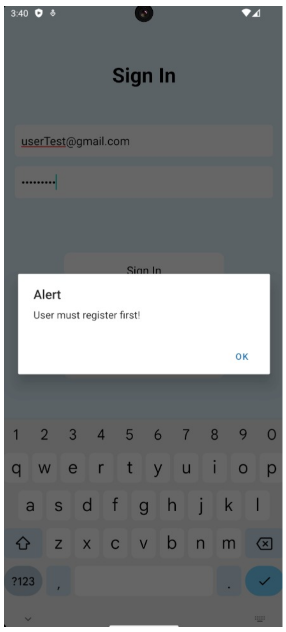
6 Experiments

In order to test the functionalities of our system, we conducted some experiments:

User Authentication

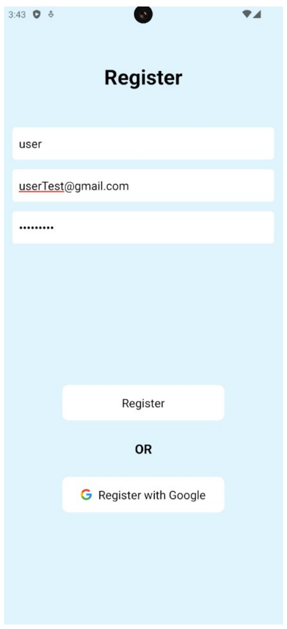
This experiment was conducted to verify that users can register successfully and login afterwards. Firstly we assumed that the user was not already registered.

Below is the *authenticate procedure flow*.

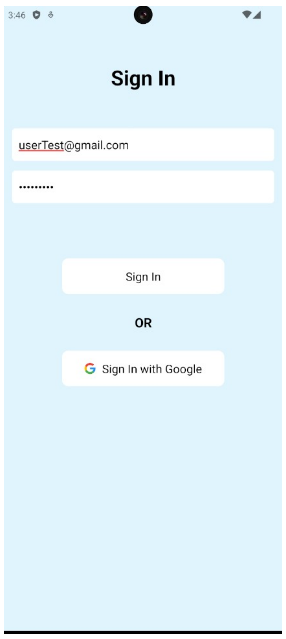


User must register first!

(a) Not Registered Try



(b) Registration



(c) Sign In Successful

In the first image 13a, the unregistered user that tries to sign-in fails and receives an alert, with the message *"User must register first!"*.

In the second image 13b, the user registers successfully and is sent to the landing page [4.3]. After this, the user is able to sign in successfully, as shown in the third image 13c.

With this test, we were able to verify that the user authentication system is working correctly, considering that users may disconnect and reconnect to the application at any time, with or without the same mobile device.

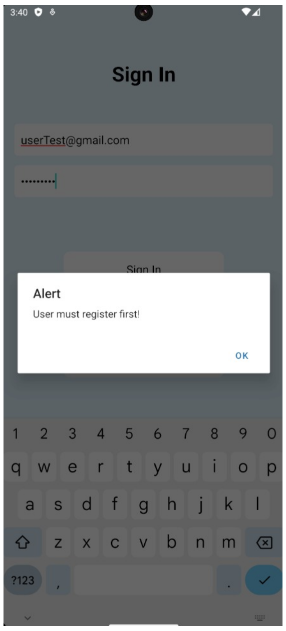
Parking Spot Booking

This experiment was conducted to verify that users can book a parking spot in the app, that the spot is reserved and that the LED color changes to blue. To conduct the experiment, we assumed that the user was already registered and logged in, and that the given parking spot is free.

Firstly, the user selects the park and the time and day of the booking. After this, the user selects the spot and confirms the booking, as shown in image 14a.

When the user confirms the booking, the spot is reserved, the data in the database is updated (image 14b) and the LED color changes to blue, as shown in image 14c.

With this experiment, we were able to verify that the booking system is working correctly, and that the LED color changes according to the spot status, therefore testing as well the communication between the mobile application and the ubiquitous device.


3:40

Sign In

userTest@gmail.com

•••••••••

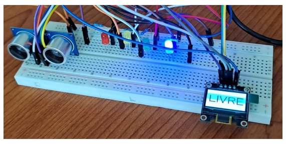
Sign In

Alert

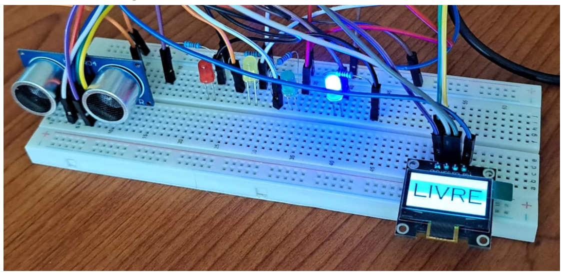
User must register first!

OK

(a) Booking



(b) Database Update

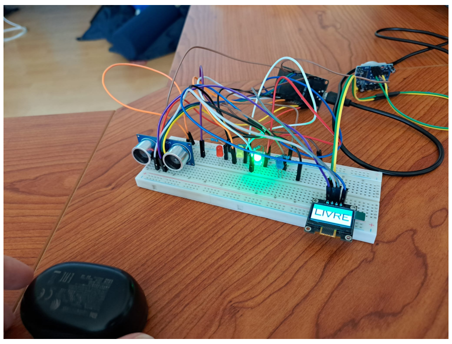


(c) LED Color Change

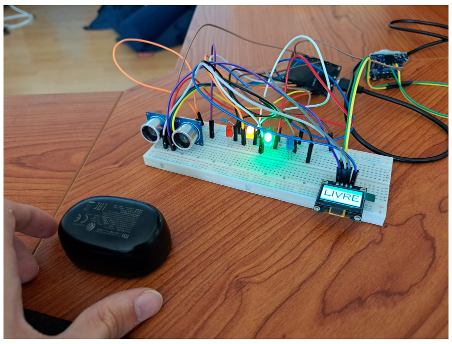
Parking Assistance

This experiment was conducted to verify that the LEDs light up according to the distance between the car and the wall, and that the spot state changes, from reserved or free to occupied. The only assumption made was that the car was parking in a free and not reserved spot.

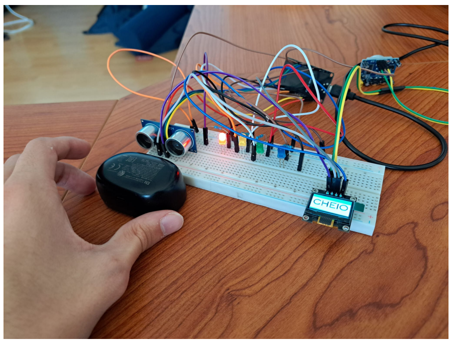
To conduct the experiment, we used a box to simulate the car and moved the box towards the distance sensor. The LEDs changed color according to the distance between the box and the sensor, as shown in the images below.



(a) Go Forward



(b) Be Cautious



(c) Stop

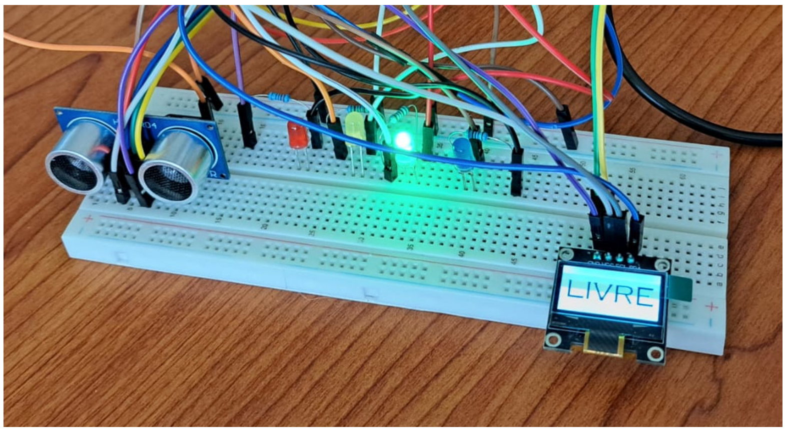
Finally, when the box was close enough to the sensor, the red LED lit up, and the spot state changed to occupied, in the real time database.

With this experiment, we were able to test the communication between the sensors and the actuators, and verify that the LEDs change color according to the distance between the car and the wall.

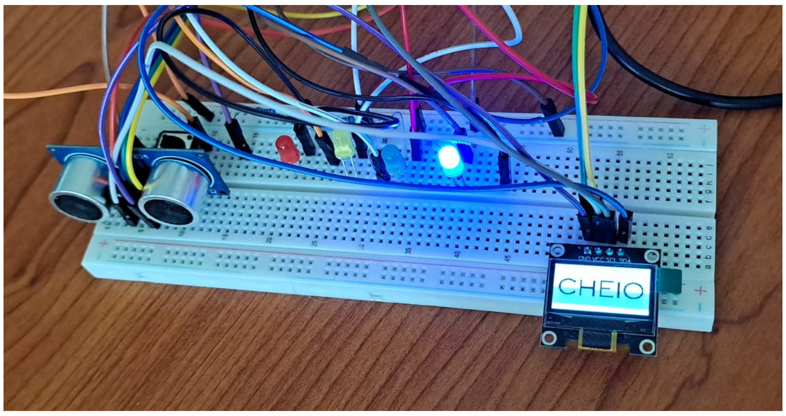
Parking Lot Occupancy

Finally, we conducted an experiment to verify that the LED display at the entrance of the park displays the number of cars inside the park. We assumed that the park was empty, and the park had only 1 spot available. This meant that, if a car parked in the parking lot, the availability of the park would change from "*livre*" to "*cheio*". It was also used the same ESP 32 device and not different devices because of the lack of resources.

As shown in the images below, the LED display shows the availability of the park, "*livre*" if the park was a free parking spots, and "*cheio*" if the park was no available spots.



(a) Free Park



(b) Full Park

With this experiment, we were able to verify that the LED display at the entrance of the park displays the availability of the park, and that the availability changes according to the number of cars parked in the park. In this case, the park was free because it had a free spot, and then it was occupied because the spot was reserved. The same behavior could be observed if the spot was occupied, instead of occupied.

7 Conclusion

In this project, we developed a system that allows users to interact with a variety of parks, by getting information given by the ubiquitous devices or by giving instructions to them. The system is composed by 3 main components: the mobile application, the server and the ubiquitous components. The mobile application is the main interface to the system, and allows the user to interact with the system. The server is responsible for managing the communication between the mobile application and the ubiquitous components. The ubiquitous components are responsible for sensing and retrieving information from the environment and actuating upon it or upon the user's request.

The system allows users to reserve a parking spot, check information of a certain parking lot, like its availability, save favorite parks and other features. The ubiquitous devices are able to change the LED color according to the spot status, and the entrance device is able to display the park availability.