Final Report - EEL7323

Bruno de Vargas Zimpel
Cláudio Vilas Boas
Gabriel Fernandes
Guiherme Fonseca

November 29, 2017

1 Product tree

A product tree, also called product breakdown structure, is a tool for analyzing, documenting and communicating the outcomes of a project, and forms part of the product based planning technique.

The diagrammatic representation of project outputs provides a clear and unambiguous statement of what the project is to deliver. The product tree precedes the WBS, feeding the information necessary to its creation.

With that in mind, the product tree developed for the Intercom project is divided in two main branches: the hardware and the software, where it's described all the requirements in each branch.

The product tree can be seen in figure 1, already revised for the final deadline.

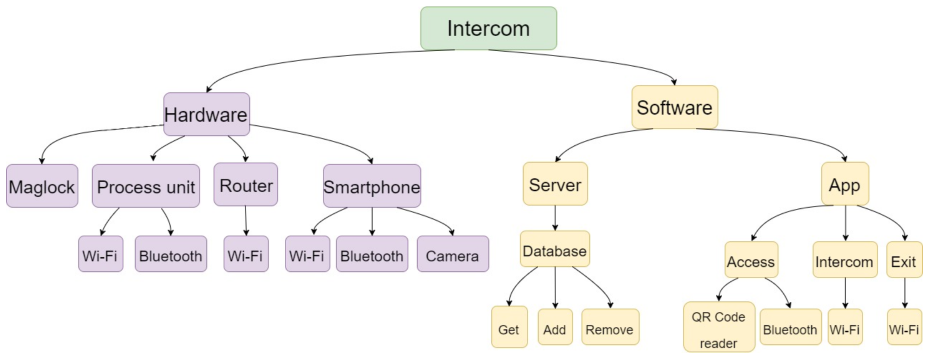


Figure 1: Product Tree

2 Work breakdown structure (WBS)

A work breakdown structure (WBS), also referred to as contract work breakdown structure or "CWBS", is a deliverable-oriented breakdown of a project into smaller components. A work breakdown structure is a key project deliverable that organizes the team's work into manageable sections.

Using the product tree, the WBS breaks down the Intercom project in deliverable tasks. One of this tasks is not only related with one of the two product tree branches, because it is the integration between both.

The WBS can be seen in figure 2, already revised for the final deadline.

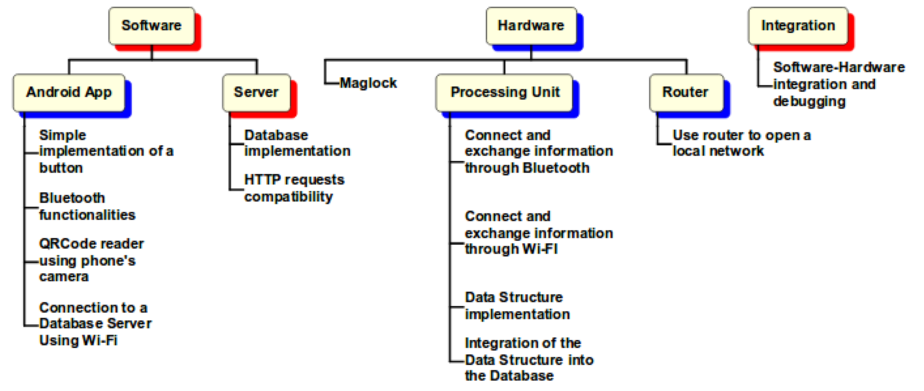

Software
Android App
Simple implementation of a button
Bluetooth functionalities
QRCode reader using phone's camera
Connection to a Database Server Using Wi-Fi
Server
Database implementation
HTTP requests compatibility
Hardware
Maglock
Processing Unit
Connect and exchange information through Bluetooth
Connect and exchange information through Wi-FI
Data Structure implementation
Integration of the Data Structure into the Database
Router
Use router to open a local network
Integration
Software-Hardware integration and debugging

Figure 2: Work Breakdown Structure

3 Requirements

For the final deadline, we still consider the following items:

- Process Unit:
  - Allow communication via Bluetooth and Wi-Fi with the user;
  - Control the opening of the magnetic door;
  - Control the database: inclusion and exclusion of new users and alteration of data;
- App:
  - QR code reader;
  - Activate the smartphone bluetooth;
  - Communicate via Wi-fi with the server to gain access to the list of user inside the room (Intercom functionality);

– Communicate via Bluetooth with the server to gain access to the room (Maglock functionality);

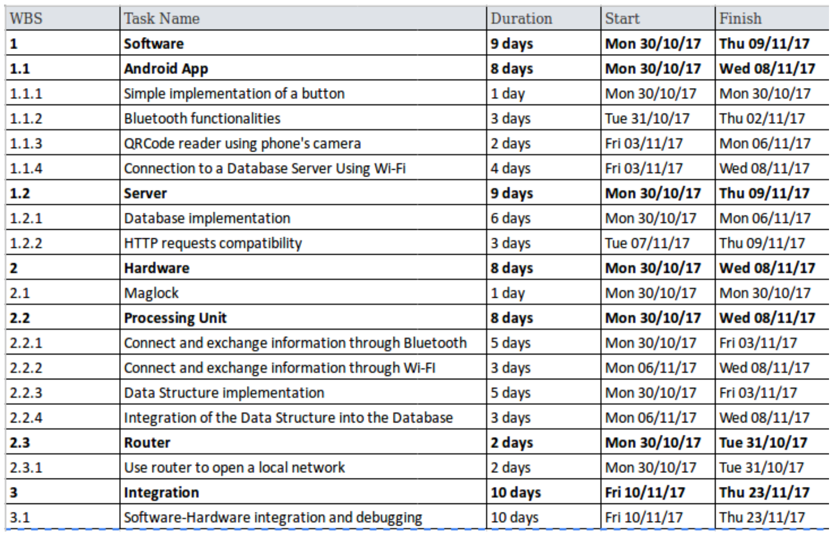
4 Detailed schedule

Based on the product tree and mainly on the WBS, the following scheduled was developed, aiming to have chronologic control over the activities of the project. In the first table the informations about duration of each task and its relationships are shown.

In the end of the project, there were a few activities that we couldn't complete. At the end of this report, there is a section exclusive for this matter. Here below, we can see a relation of the total hours each member of the team spend on the project.

Table 1: Hours spent on project

Member	Total hours spend
Bruno de Vargas Zimpel	30h
Cláudio Vilas Boas	30h
Gabriel Fernandes	30h
Guilherme Lacombe Oliva da Fonseca	30h
Total	120h



WBS	Task Name	Duration	Start	Finish
1	**Software**	**9 days**	**Mon 30/10/17**	**Thu 09/11/17**
1.1	**Android App**	**8 days**	**Mon 30/10/17**	**Wed 08/11/17**
1.1.1	Simple implementation of a button	1 day	Mon 30/10/17	Mon 30/10/17
1.1.2	Bluetooth functionalities	3 days	Tue 31/10/17	Thu 02/11/17
1.1.3	QRCode reader using phone's camera	2 days	Fri 03/11/17	Mon 06/11/17
1.1.4	Connection to a Database Server Using Wi-Fi	4 days	Fri 03/11/17	Wed 08/11/17
1.2	**Server**	**9 days**	**Mon 30/10/17**	**Thu 09/11/17**
1.2.1	Database implementation	6 days	Mon 30/10/17	Mon 06/11/17
1.2.2	HTTP requests compatibility	3 days	Tue 07/11/17	Thu 09/11/17
2	**Hardware**	**8 days**	**Mon 30/10/17**	**Wed 08/11/17**
2.1	Maglock	1 day	Mon 30/10/17	Mon 30/10/17
2.2	**Processing Unit**	**8 days**	**Mon 30/10/17**	**Wed 08/11/17**
2.2.1	Connect and exchange information through Bluetooth	5 days	Mon 30/10/17	Fri 03/11/17
2.2.2	Connect and exchange information through Wi-Fi	3 days	Mon 06/11/17	Wed 08/11/17
2.2.3	Data Structure implementation	5 days	Mon 30/10/17	Fri 03/11/17
2.2.4	Integration of the Data Structure into the Database	3 days	Mon 06/11/17	Wed 08/11/17
2.3	**Router**	**2 days**	**Mon 30/10/17**	**Tue 31/10/17**
2.3.1	Use router to open a local network	2 days	Mon 30/10/17	Tue 31/10/17
3	**Integration**	**10 days**	**Fri 10/11/17**	**Thu 23/11/17**
3.1	Software-Hardware integration and debugging	10 days	Fri 10/11/17	Thu 23/11/17

Figure 3: WBS reference to the schedule

5 Use case diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

In our case, the use case diagram is accompanied by the Product Tree, the WBS and the requirements of the project, and is shown in figure 4, already revised for the final deadline.

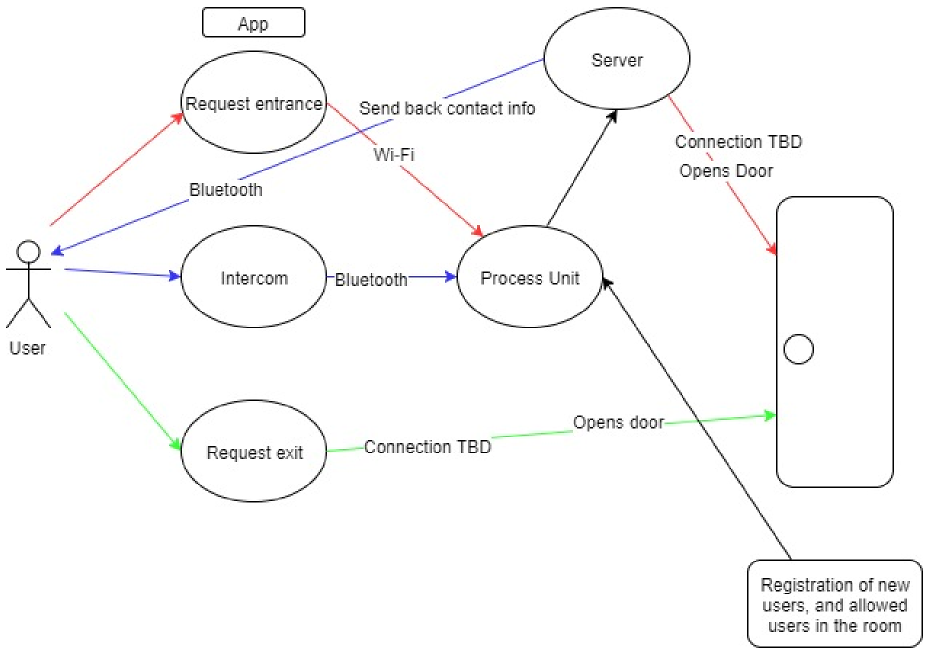


Figure 4: Use case diagram

6 Class diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.

The preliminary classes diagram developed for this report will serve as a guide to the definitive one, as we add new classes and methods for our main software written in C++, as well as remove unnecessary ones. The classes diagram can be seen in figure 5.

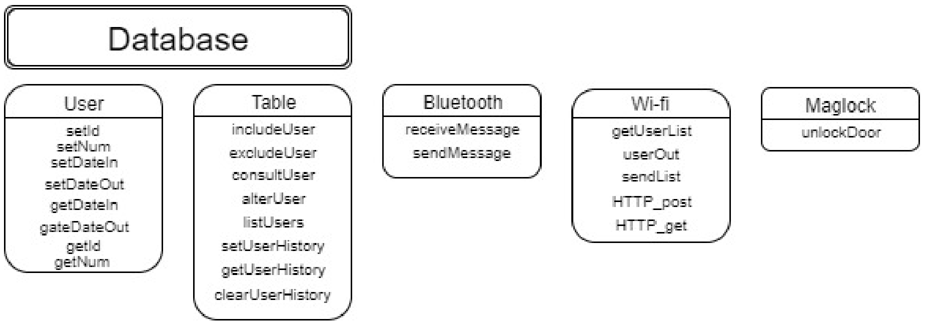


Figure 5: Classes Diagram

7 Sequence Diagram

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order.

Below we can see the sequence diagram of the final project, where ”data” implies update on the data of the users, like their status (if they are inside the room or not) or the list of users inside the room.

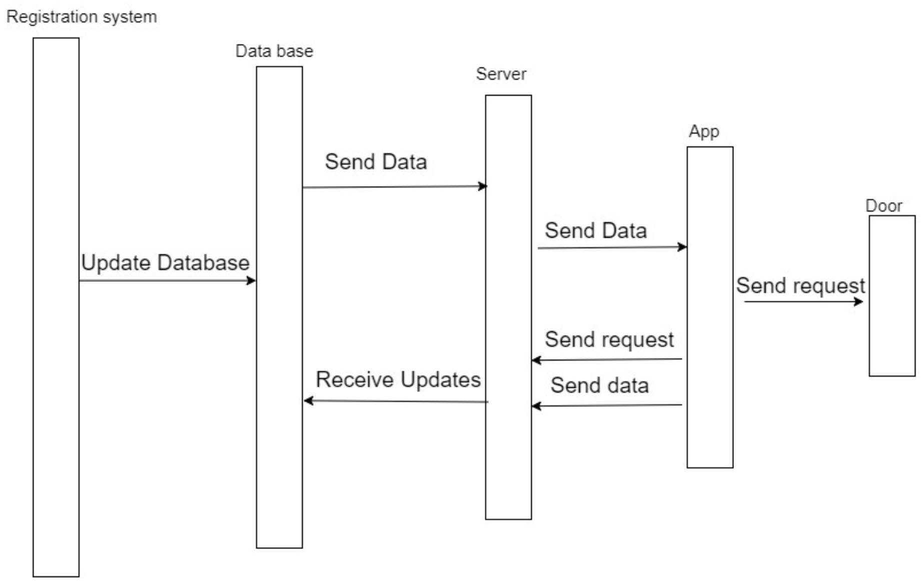


Figure 6: Sequence Diagram

8 Test Plan Execution

The validation of the project was made in a methodically way. Each component was tested isolated and then after that the smaller blocks were okay the whole implemented project was tested.

Starting with the communications. First we tested if the raspberry-pi is receiving correctly the bluetooth data, than afterwards if it sent a command correctly to the maglock (tested as an gpio).

The communication between the registration system and the database was tested including, excluding and altering data from the users.

The communication between server and database wasn’t implemented.

The app was tested separately. The QR Code reader and the communication with the Rasperry-pi.

9 User Manual

To use the system as a whole, you'll need, at least, a Raspberry Pi to work as a "Server" of the system and a smartphone to work as a "client" (a user of the system). First of all, you'll need to run the database and the CLI. This part is located on the Server folder, so the user can run it and then compile the file "main.cpp" with g++. After this step, you can run the executable.

This will open a homescreen, where we can read "Program for registration of users of the room - EEL7323 - USFC 2017". Here we have all the options, to include a new user, consult an specific user, list all the user, alter an specific user and exclude an specific user.

Initially, the database will be empty, so the user will include all the data from the users of the room, their ID's, their phone numbers, and the time they can enter and exit the room each day.

After all the users are registered, their data will be automatically uploaded to the Database.

With the database updated and running, comes the time for the other end of the system: the users of the room.

On the app, there are few options available. You can either get the .apk on our GitHub link and download it to your phone, you just need to ensure that on your phone settings you have the option download from unknown source enabled. Or you can clone the code to you computer and compile it using Android Studio. After doing it, is pretty simple to use the App: the names of the buttons are self explanatory.

Links repository

`https://github.com/PaulKrauseTeam/IntercomProject.git`

Hardware Implementation

9.1 Router

The router is used only for open the LAN network through the devices in order to connect all the elements of the system

9.2 Smartphone

In this project, the smartphone is used as way to interface the user with the system. Since the app that we've developed is a .apk, it can only run on Android devices.

9.2.1 Camera

To read the QRCode using the Smartphone, a digital camera is mandatory. Usually, this functionality is blocked by default on Android Smartphones, so the application must ask the user if he lets the software acquire control over the device. If the application is done right, you'll only need to do it one time, at the installation. If not, every time you try to use your QRCode scanner, the Android OS will ask the user for permission.

9.3 Processing Unit

In order to implement a access point for the Smartphones to the system itself (door, database, etc), we've used a Raspberry Pi 3 board connected to the Network Router. Since this powerful device has embedded Internet and Bluetooth resources, it is more than sufficient for what we aim in this project. We have used the RaspBian OS to develop this functionalities because of its support and user-friendly boot and installation.

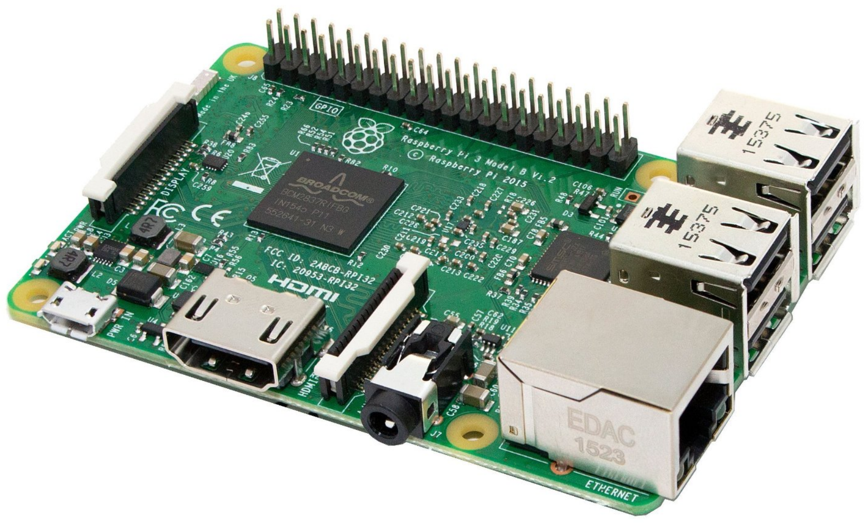


Figure 7: Raspberry Pi 3

9.3.1 Wi-Fi

To get Raspberry Pi 3 connected to the Network is simple as it gets: just connect the device to an internet router through a twisted-pair cable and it's done. The router will connect the board to the network and give an IP address to it automatically.

9.3.2 Bluetooth

Despite of having the neccessary hardware to use the Bluetooth procol, it doesn't work immediately. A few steps are needed in order to make the board and the devices work correctly:

1. Open a Raspberry Pi terminal and type: *sudo bluetoothctl*. If you're using an older version of the board, maybe Raspbian will not find this application. To overcome this, you should download the bluetoothctl application through *sudo apt-get install pi-bluetooth*;

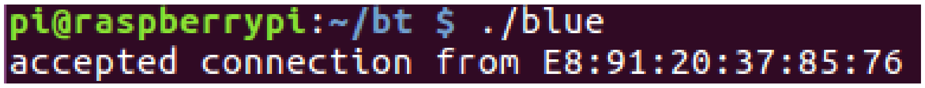
2. Enter *agent on* and then *default-agent* to start the resource;

3. Now it's time to start the scanning for available devices. After turning on the Bluetooth adapter of the devices that you want communicate via bluetooth with the Raspberry Pi on (in this case, the cellphone), enter in the terminal *scan on*. After a couple of seconds, you should be able to see the unique addresses of the Bluetooth devices around your Pi as an alphanumeric string like **XX:XX:XX:XX:XX:XX**. If the cellphone

are in "Discoverable Mode", the nickname of the device will appear right after this string;

4. To pair to this device, enter *pair <uniqueaddress>* ;
5. In general, Bluetooth devices confirm the will to pair though a numerical confirmation (usually 6 digits). If your cellphone are in this group, confirm this code through a pop-window that will appear, comparing it with the number shown on the terminal after the pair command;
6. At this point, your Raspberry Pi should be paired to the cellphone. But it isn't enough in order to exchange information. To do this, you have to connect the two devices, typing *connect <uniqueaddress>* on Pi's terminal. If the connection goes well, you'll receive a confirmation and you're good to go;

After doing this, you can open directly a RFComm socket between the Raspberry Pi and the user phones, in order to make the connection between this elements faster, since it don't lay on excessive checkings and validatings. It was through this method that we've had developed this part of our project: the Raspberry Pi opens a socket waiting for the bluetooth devices to connect and then start to exchange data, without the need of pairing.



```
pi@raspberrypi:~/bt $ ./blue
accepted connection from E8:91:20:37:85:76
```

Figure 8: Bluetooth Connection through RFComm

Software Implementation

9.4 Server

9.4.1 Registration System

For the Registration System, we choose to use Doubly Linked List as the Data Structure, which is a variation of Linked List in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List.

Each class and method was created thinking about the communication with the database, the new classes are shown in the figure 5. The Registration System will communicate with the Database, passing all the parameters of each user.

The program supports the addition of new users, exclusion of users, alterations of parameters of the users (such as the period in which the user can be inside the room or their contact information), consult the data of an specific user and list of all users.

When a new user is included, the program creates three strings with all the data concatenated, these strings will be sent for the database, where the data will be distributed correctly.

When a user is deleted or its data altered, new strings are created, and theses strings are sent to the database as well.

All the codes are saved on Github, in the repository created for this end.

9.4.2 Database

A Database is an organized collection of data. This project uses a relational database, which is a collection of schema, tables and other elements. A database-management system (DBMS) is a computer-software application to capture and analyze data from the database. There are a bunch of DBMS available, but for this project, the team choose the PostgreSQL 9.5.9 for its wide usage and stability.

The database, called *intercomdb.db*, has 4 (four) tables. The first table, *users*, contains information about all users, like *id*, *phone_number*, *status* (a boolean that is set if the user is inside the room or not). The *entry_permission* table has the time in each day of the week a user can enter the room. And the *exit_permission* has the final time that a user can stay in the room. The last table, *users_history*, stores every time a user enters or leaves the room.

The communication between the Data Structure and the Database is done by methods called that receive a string and from this change the database. These methods utilize library for C++, the libpqxx-4.0, specialized to run SQL functions in a .cpp file.

When the Data Structure create a new user, it concatenate three strings and call the methods *new_user(string str)*, *new_entry(string str)* and *new_exit(string str)*. The new_user method receives a string containing the id and the phone_number, and from this execute a 'INSERT' SQL Query in the users table, setting the status to false as default. The new_enter and the new_exit methods also receives a string, but now containing the id and all the times in each day of that user permissions.

For delete a user the Data Structure calls also three methods, *delete_user(string str)*, *delete_entry(string str)*, *delete_exit(string str)*. All three receives only the id and runs a 'DELETE' SQL Query in the corresponding table.

The communication between the Database and the Android App, are made by two methods, *getUserTime(string str)* and *insideUsers()*. The first one receives from the App, via Bluetooth connection, the user id, and return, also via Bluetooth, the entry and exit permissions for this user for the actual week day, running a 'SELECT' SQL Query, and from this, if the user is allowed to enter, update the user_history table. Now, the insideUsers() is called from the App via a Wi-Fi requisition, and return the id and phone_number from users that are inside the room, also via Wi-Fi.

All the codes are saved on Github, in the repository created for this end.

9.5 App

At this last part of the project, the App has been fully developed.

9.5.1 Bluetooth

The Bluetooth communication on the App works as the client side. Were it scans for Devices to pair, once the device is found it displays at a list were the user can choose, as shown over figure 11, one device then it connects over the rfcomm channel through a bluetooth socket. If the server side has the same UUID the connection is well-established.Once the communication is made, the app changes to a login layout, over there the user writes over an EditText its id, than the app waits for an answer with the time interval that the user can enter the facility, if the id exists over the database.If it's everything in order the App then send a command to the door be unlocked.

9.5.2 Wi-Fi

The Wi-Fi communication is done using Volley library through an HTTP get method.Once the raspberry-pi initializes the server, it can communicate with the App through local network.Once the HTTP get is sent to the answer, the routine waits for an JSONObject as answer, than it parses the JSONObject in two JSONArrays and displays it in a ListView so the user can see the ID and phone number of people that are currently at the facility. But we couldn't finish writing the server code to have this data available.

9.5.3 QR reader

We used the open source library ZXingScannerView to enable the Smartphone camera to work as a QR code scanner.Then we generated a QR code that was scanned successfully and got data from an URL. An important matter over our implementation was that besides enabling the camera permission over the manifest file in our application, we needed to access the configuration off our application and enable the camera permission manually what can vary from phone to phone.

9.5.4 Working example of App

The Android App working on a phone is showed in the figure 9, figure 18, , figure 12 and figure 13below.

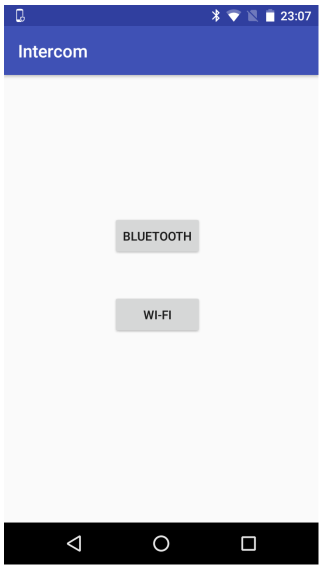


Figure 9: App Main Screen

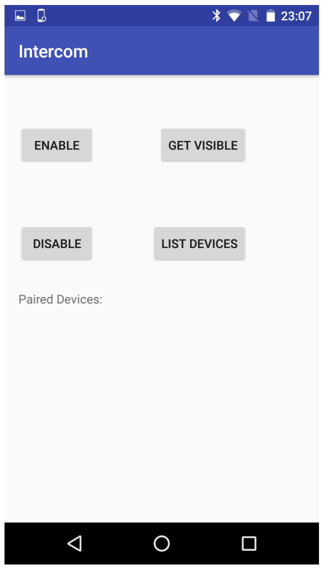


Figure 10: Bluetooth Screen

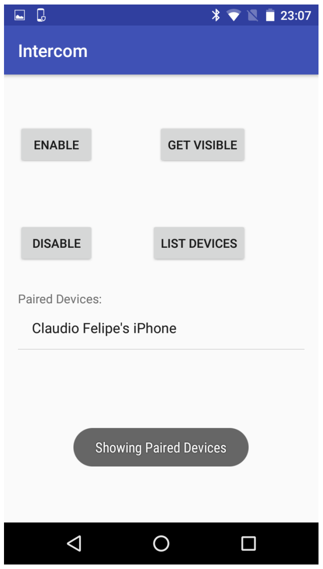


Figure 11: Bluetooth showing list of devices

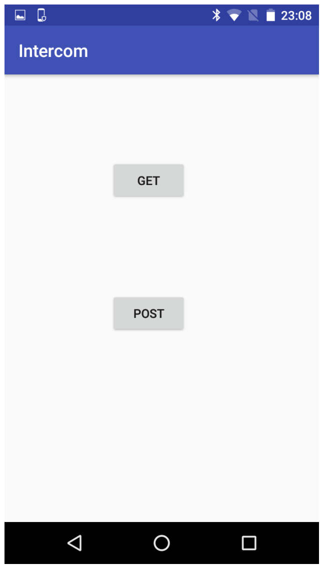
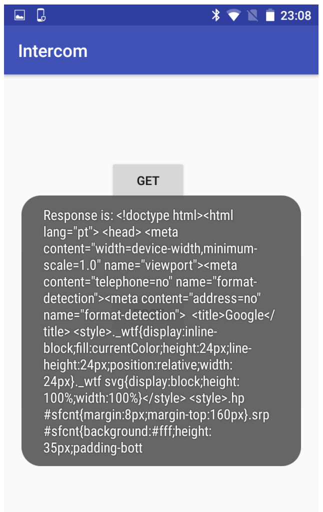
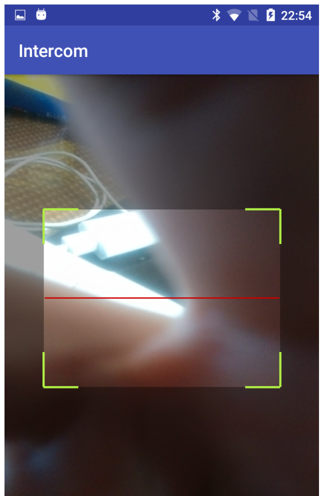


Figure 12: WiFi Screen

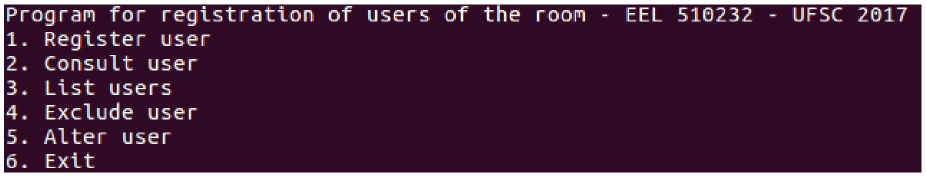


```
Response is: <!doctype html><html
lang="pt"> <head> <meta
content="width=device-width,minimum-
scale=1.0" name="viewport"><meta
content="telephone=no" name="format-
detection"><meta content="address=no"
name="format-detection">  <title>Google</
title> <style>._wtf{display:inline-
block;fill:currentColor;height:24px;line-
height:24px;position:relative;width:
24px}._wtf svg{display:block;height:
100%;width:100%}</style> <style>.hp
#sfcnt{margin:8px;margin-top:160px}.srp
#sfcnt{background:#fff;height:
35px;padding-bott
```



10 System working so far

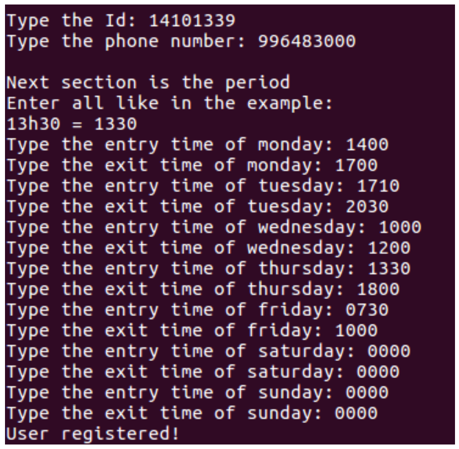
The first part of the project is the Registration System, with the home screen as shown below.



```
Program for registration of users of the room - EEL 510232 - UFSC 2017
1. Register user
2. Consult user
3. List users
4. Exclude user
5. Alter user
6. Exit
```

Figure 19: Homescreen

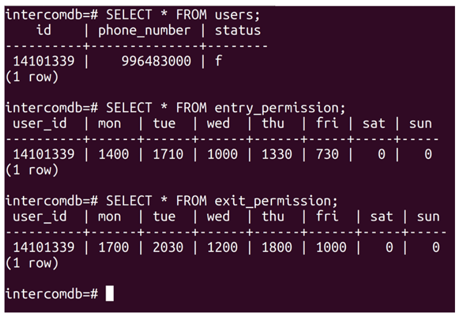
When we add a new user, as shown in figure 20a, the Database is updated, as we can see in figure 20b.



```
Type the Id: 14101339
Type the phone number: 996483000

Next section is the period
Enter all like in the example:
13h30 = 1330
Type the entry time of monday: 1400
Type the exit time of monday: 1700
Type the entry time of tuesday: 1710
Type the exit time of tuesday: 2030
Type the entry time of wednesday: 1000
Type the exit time of wednesday: 1200
Type the entry time of thursday: 1330
Type the exit time of thursday: 1800
Type the entry time of friday: 0730
Type the exit time of friday: 1000
Type the entry time of saturday: 0000
Type the exit time of saturday: 0000
Type the entry time of sunday: 0000
Type the exit time of sunday: 0000
User registered!
```

(a) User registration screen



```
intercomdb=# SELECT * FROM users;
    id    | phone_number | status
----------+--------------+--------
 14101339 |    996483000 | f
(1 row)

intercomdb=# SELECT * FROM entry_permission;
 user_id  | mon  | tue  | wed  | thu  | fri | sat | sun
----------+------+------+------+------+-----+-----+-----
 14101339 | 1400 | 1710 | 1000 | 1330 | 730 |   0 |   0
(1 row)

intercomdb=# SELECT * FROM exit_permission;
 user_id  | mon  | tue  | wed  | thu  | fri  | sat | sun
----------+------+------+------+------+------+-----+-----
 14101339 | 1700 | 2030 | 1200 | 1800 | 1000 |   0 |   0
(1 row)

intercomdb=# █
```

(b) Database updated

Figure 20: User registered, and the result on the database

When we alter or exclude a user already registered on the Database, the results are similar.

On the App, the user can choose between Bluetooth or Intercon options, if the user chooses bluetooth than he can alter his phone bluetooth connection and try to connect with some previous paired device. If the device he tries to pair has any routine that implements serial bluetooth rfcomm protocol, the communication its going to be established, than he can send values to it that would be responsible to send the user id and data to unlock the door. If he chooses the Wi-Fi facility he can then open a QRCode reader that is suppose to get an URL, if the URL has JSON data, with an ”id” and ”number” tag, then a ListView with an ID and number column is displayed.

11 Tasks left incomplete

- HTTP Connection between the Database and the Server running on the Raspberry Pi 3;
- Stable Bluetooth connection using RFcomm between the phone and the Raspberry Pi;
- Script to populate the database with all students from EEL 7223;