M6 - Final user demo

PEI



**Rodrigo Rocha**, 71731

**Rodrigo Relvas**, 71860

**João Pedro**, 80098

**Fábio Nunes**, 80139

**Diana Silva**, 80239

**Susana Dias**, 80410

Mestrado Integrado em Engenharia de Computadores e Telemática

Index

[**1 Client area**](#_lj2pljjii4b9) **2**

[1.1 Final Result](#_72hmkl6cco0h) 2

[1.1.1 API:](#_ifuquuvu1sqi) 2

[1.1.2 BLE:](#_ajg9q6f8f3bv) 2

[1.1.3 Data Analysis:](#_dtwghbisru24) 2

[**2 Personas and goals**](#_87zk4ki3hrim) **4**

[**3 User Manual**](#_qnwzip4gtyjb) **5**

[**4 Specification area**](#_bbeio7m87got) **10**

[**Regardless of the convention used, our system does:**](#_mh3d8teoxso2) **10**

[**5 System Architecture**](#_6x8qpx3hgxt) **10**

[**6 Tests and validation**](#_5diykff622cr) **11**

[**7 Developer area**](#_t85lr6vfdrqe) **12**

[7.1 BLE](#_rsy10qyl5ypm) 12

[7.2 Mobile app](#_btmmbzsbomrt) 13

[7.3 Data Analysis](#_2qwlik1jk0p) 14

# 

# 

# 1 Client area

## 1.1 Final Result

### 1.1.1 API:

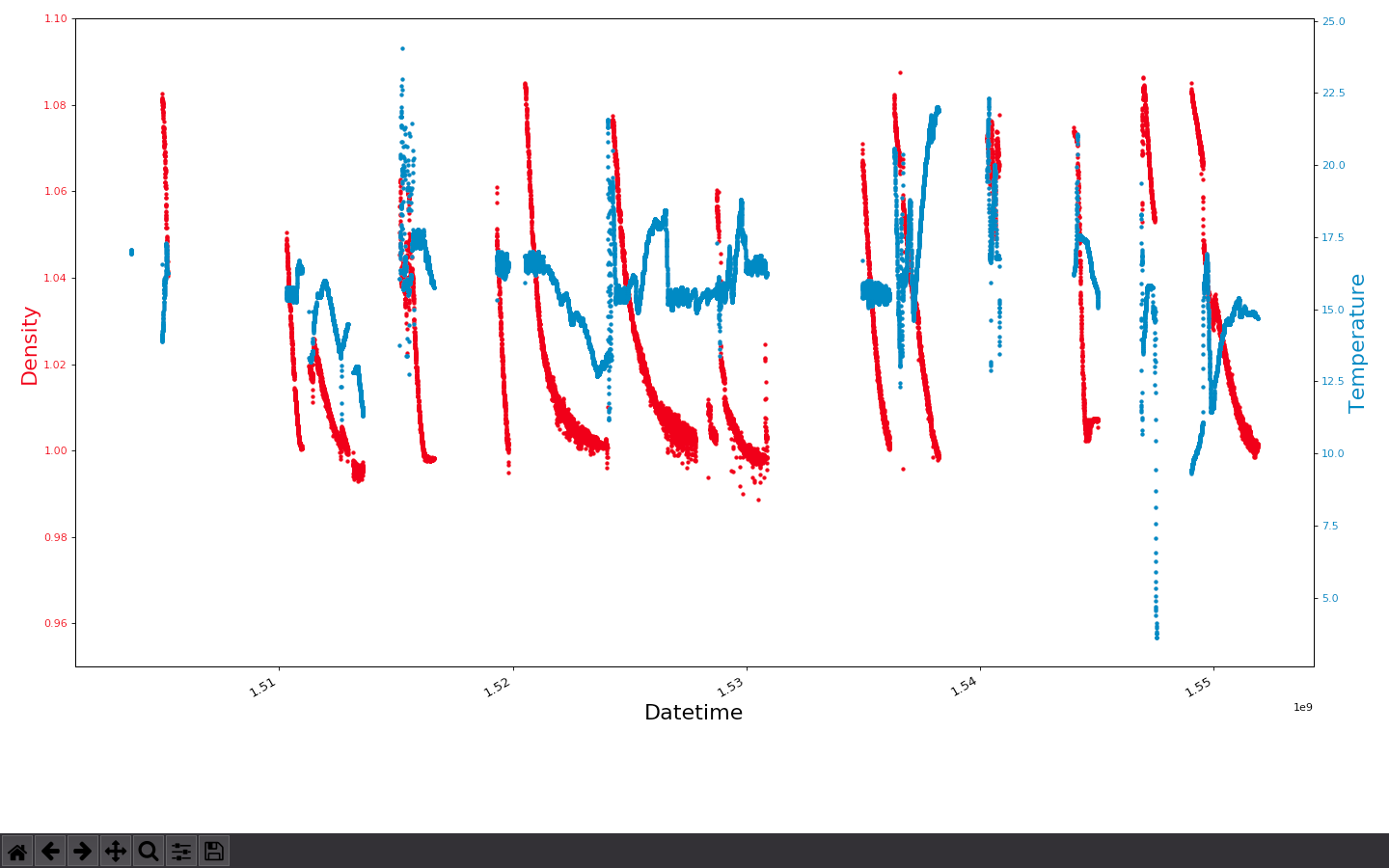
1. Login
2. Open note
3. Delete note
4. Show note
5. Create note
6. List notes
7. List process
8. Process detail
9. Stop Process

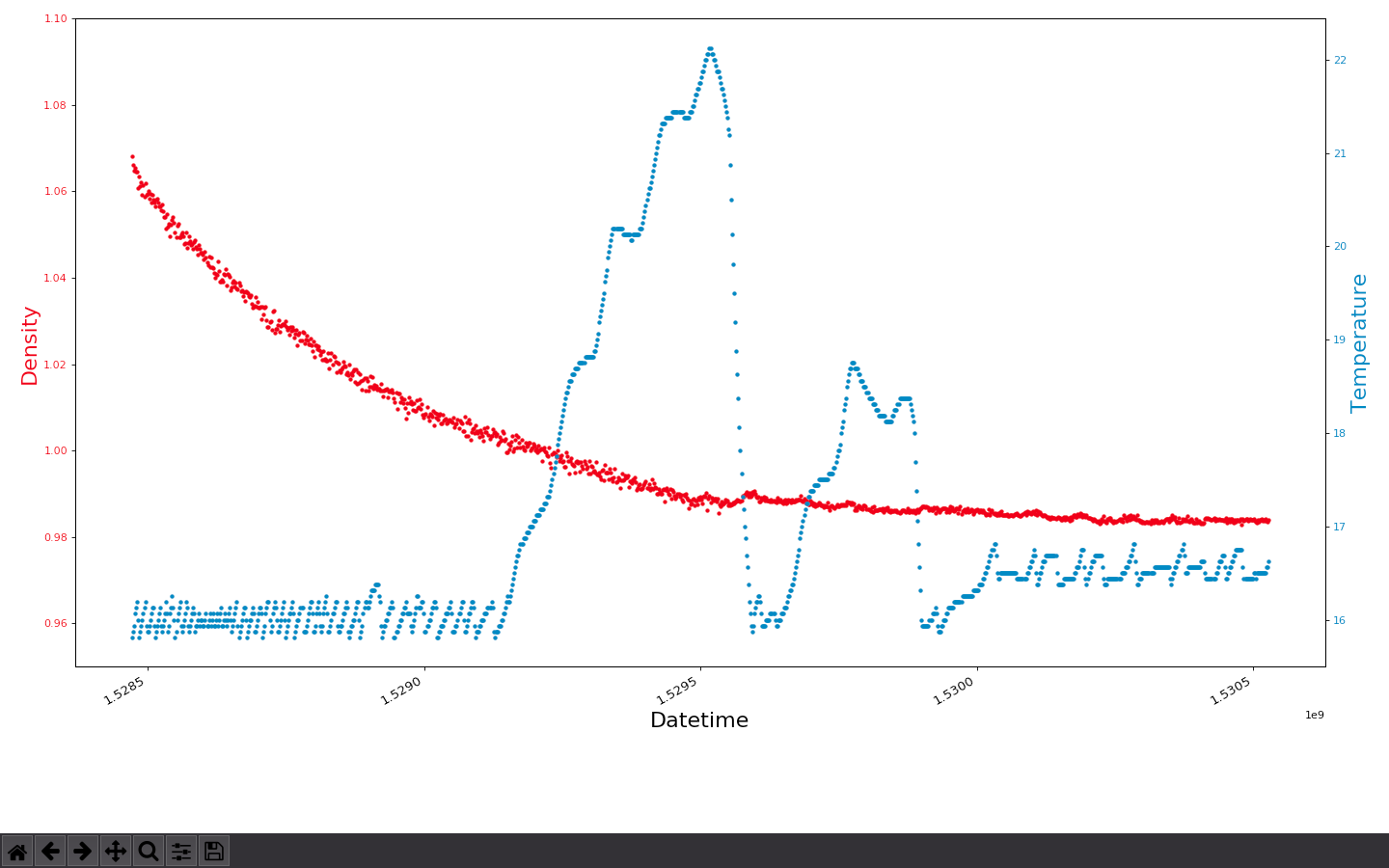
### 1.1.2 BLE:

1. Open serial COM to communicate with DevBoard (ESP32);
2. Dev Board reset (ESP32);
3. Open app;
4. Scan Barcode and QR code;
5. Dev Board connect;
6. Verify app and serial COM communication status;
7. Verify received data from Dev Board;
8. Improved security with diffie-hellman algorithm;

### 1.1.3 Data Analysis:

1. Process detection while analysing the curves;
2. Confidence Band;
3. Curve prediction;





Link to the final video: <https://www.youtube.com/watch?v=IxjM6iQPeZo>

## 

# 2 Personas and goals

Liliana Costa, operator of “Adega de Borba”



Liliana Costa, born on 4th December, 1980, in Benfica, Portugal, lives alone. She’s a very active person, she likes to hang out with her friends, go to parties, wine appreciator and she’s a volunteer at the municipal shelter.

Liliana has a problem in her hip since birth, that causes her uncomfortable pain.

She works at the *Adega de Borba* and her work consists in evaluate the vats, that is a time consumption work and painful for her hip.

Most of the time spent is due to the information she has to write on paper of the vats and then transcribe it to *Excel*.

She needs an application that allows her to save time and her hip.

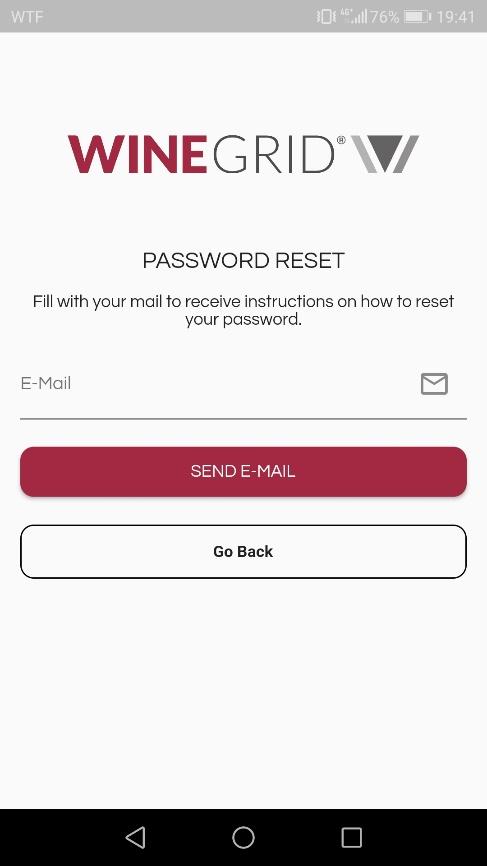
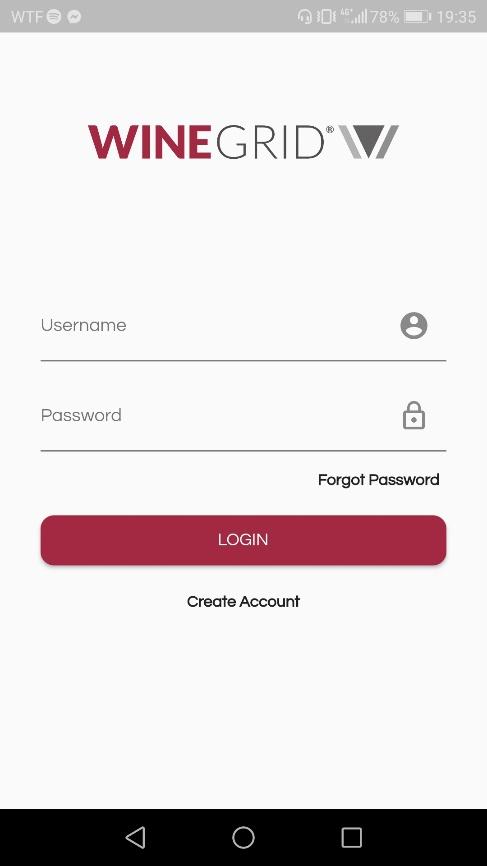
Scenarios

* Liliana needs to walk around the building to see the changes of the vats with the help of the application she can open the dashboard and see them.
* Liliana needs to control the time to make changes at some vats, she can open the application and set an alarm to know when she needs to go to the vat and make a change at the wine;
* Liliana has to take paper notes of changes watched she can open the application and take notes of the changes she has observed.

# 

# 3 User Manual

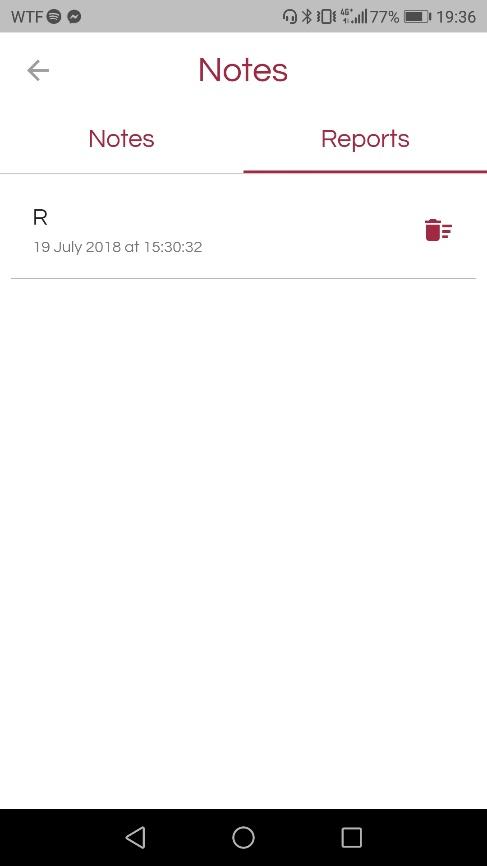
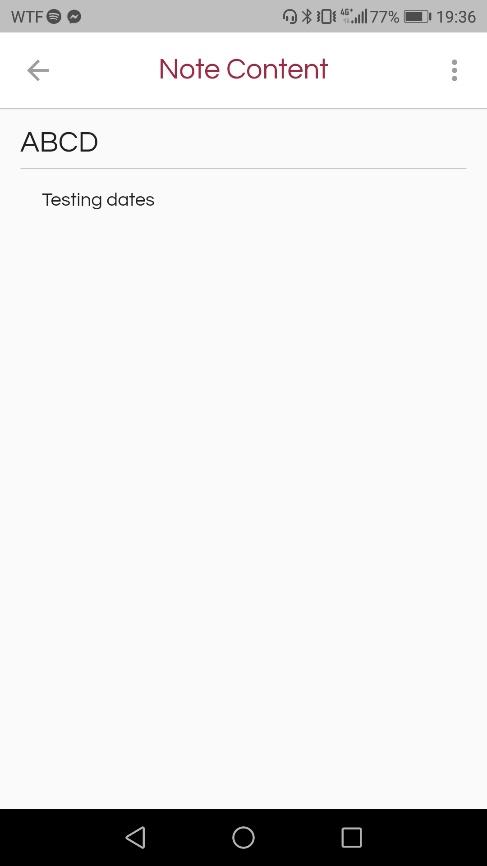
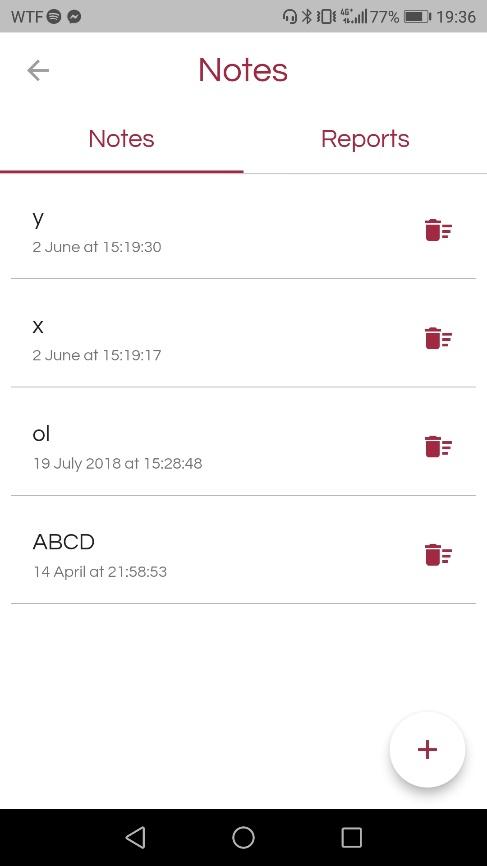
As soon as the user opens the app, is prompted with a option to login. The user need to be logged in to order to be able to use the application.

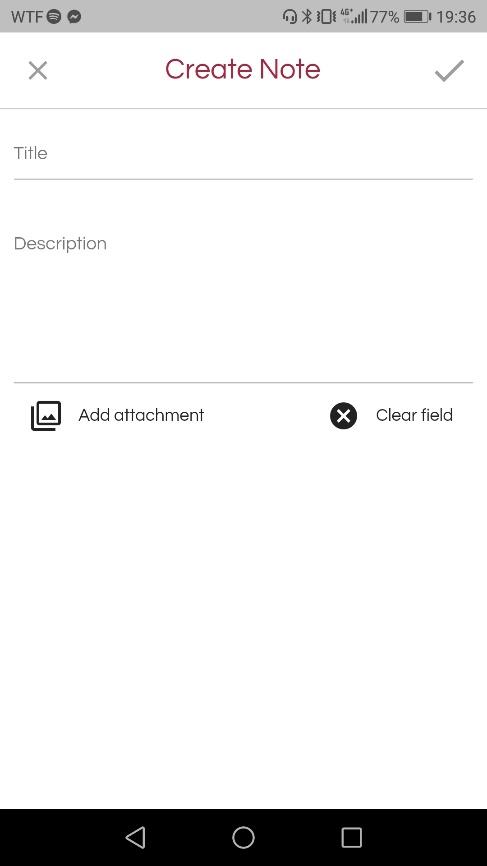


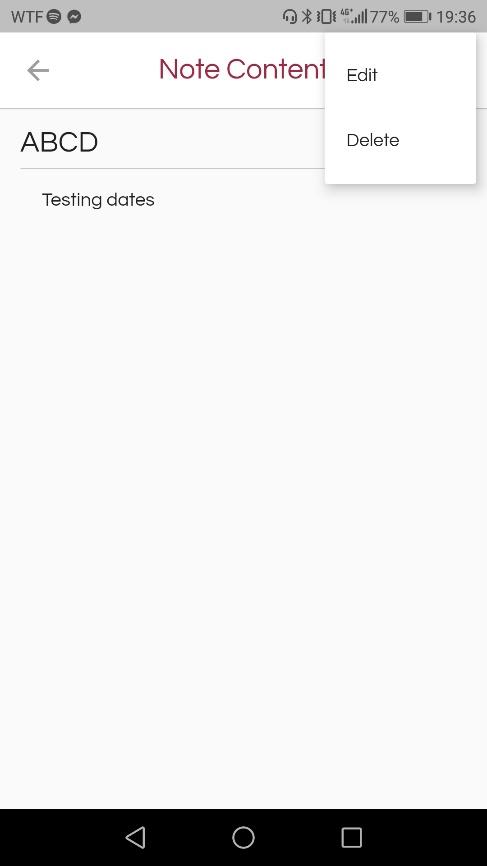
The next page is the main menu page. Contains QR Code Scanner, Search Available Devices, Notes, Processes and Data Analysis pages.

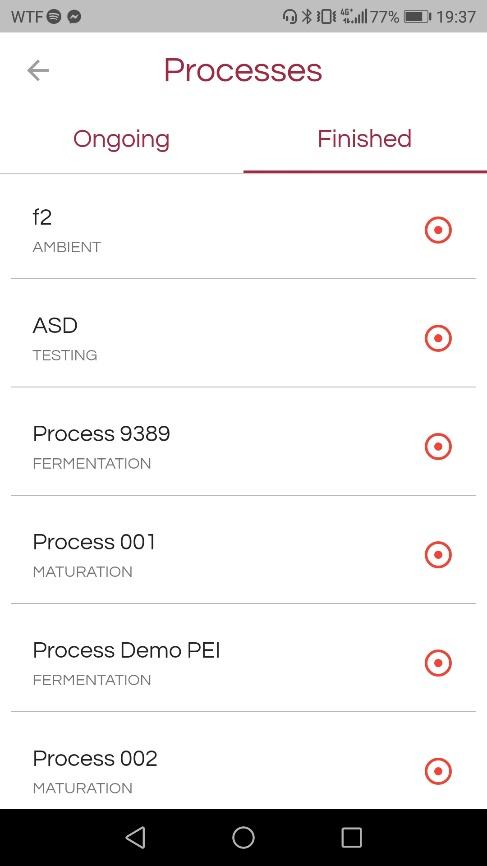


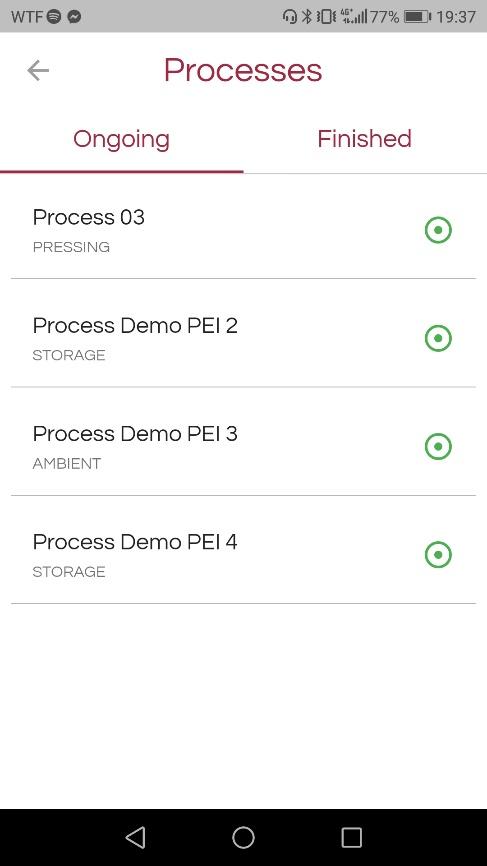
* Notes

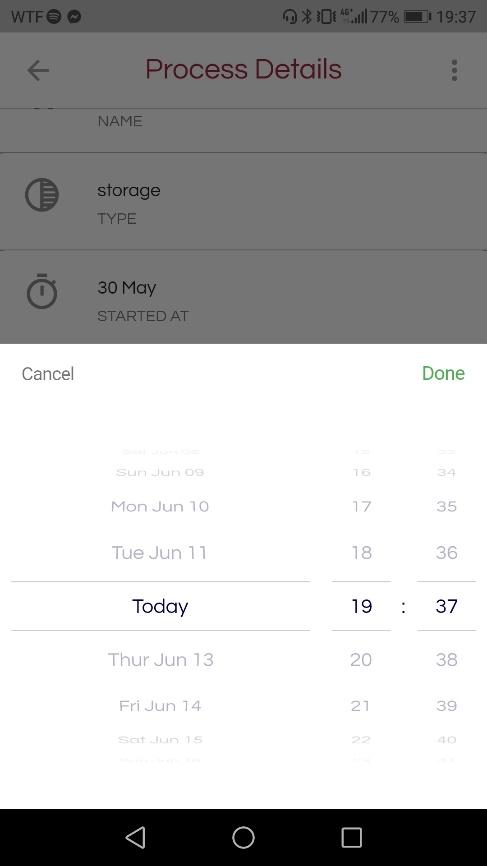
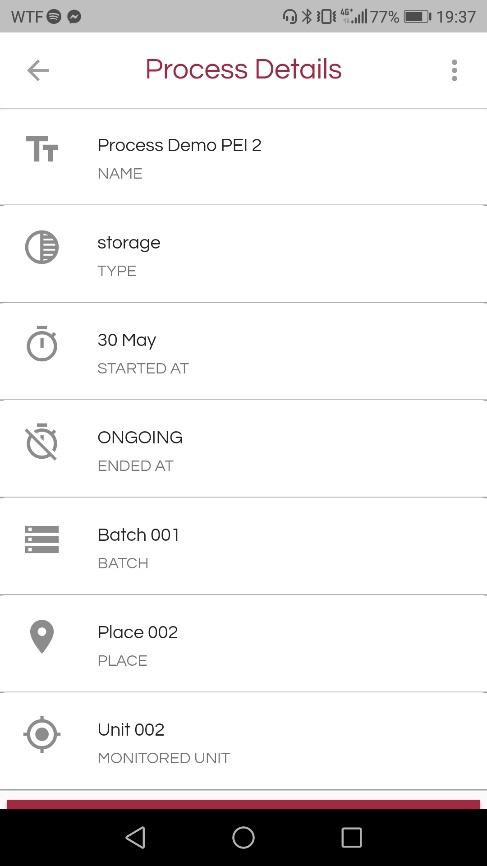






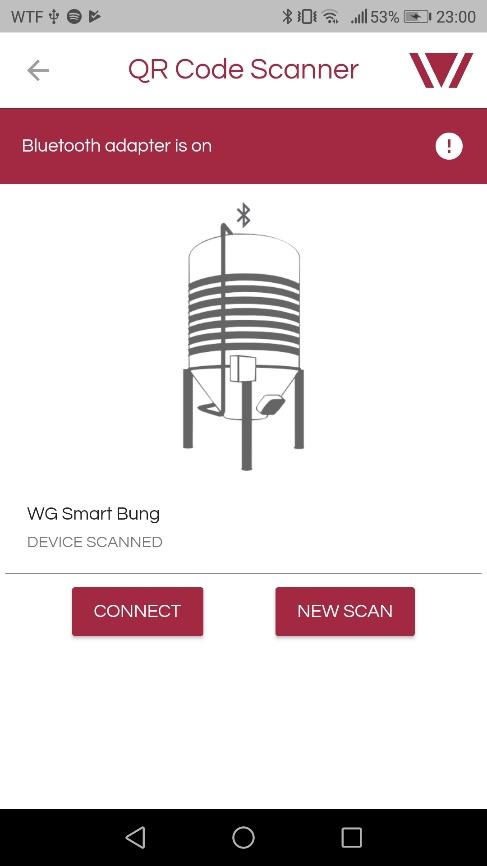
* Processes





It is divided into current and finished processes. The used can stop a process at any time.

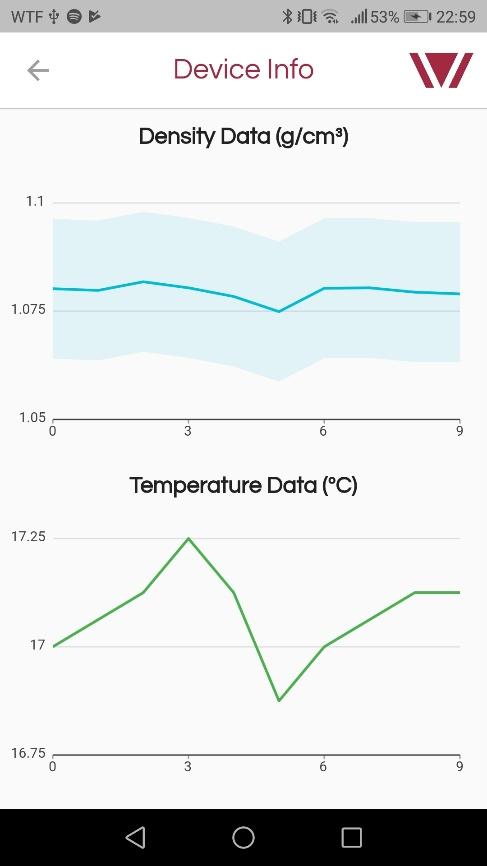
* QR Code Scanner and Search BLE Devices pages

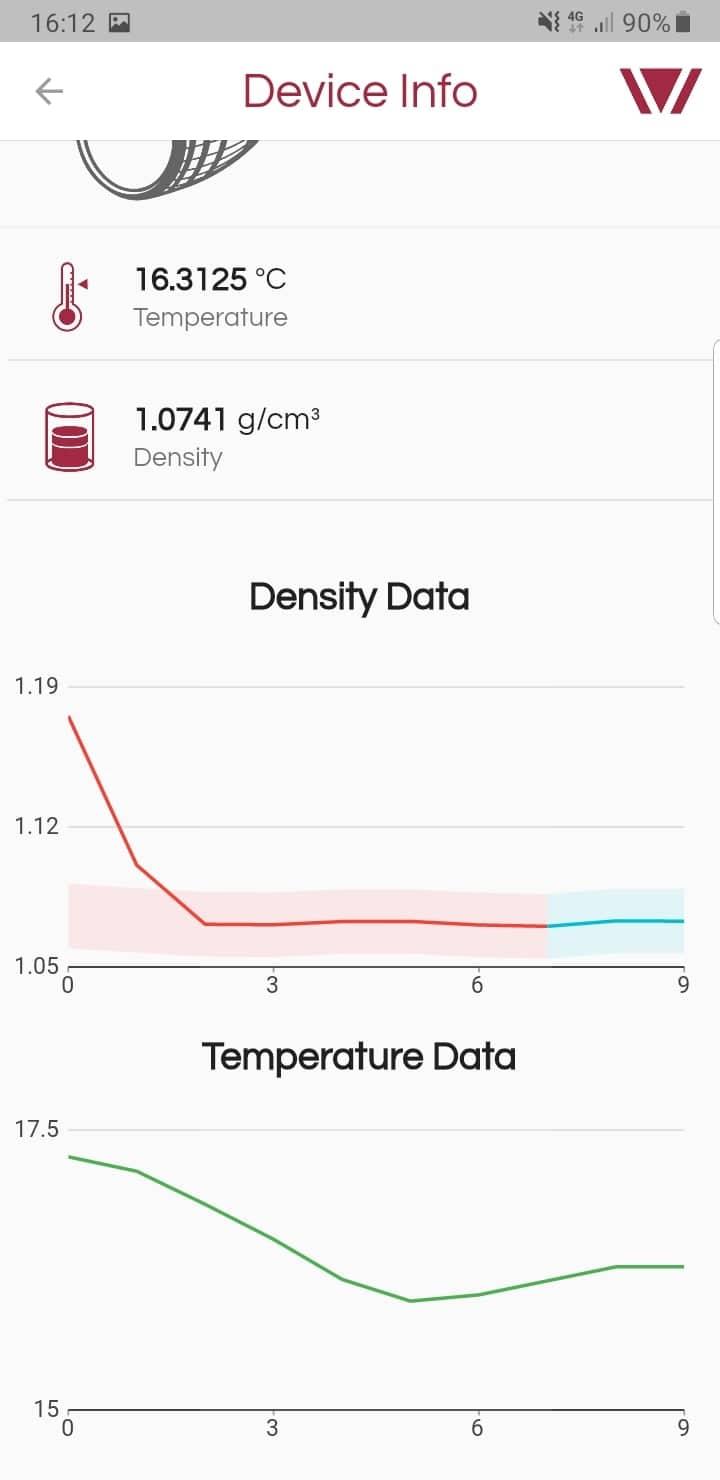




The user can Scan BLE Devices wither by QR Code or search the list of available devices. Once connected, it is redirected to a page containing real time data of the sensor.

* Data Analysis





The user is able to see the data from the sensor and the respective confidence band. In case the lines are blue, the values are inside the confidence band and red if they aren’t.

A detail explanation of the application is explained in Developer Area.

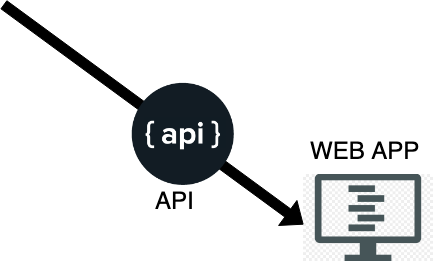
# 

# 4 Specification area

# Regardless of the convention used, our system does:

* Create/edit/delete a note through our app to the the web app;
* Delete/edit a process through our app to the the web app;
* Have a BLE server running on a micro controller and connect it to our app;
* Show some graphics where we can detect processes while analysing the curves.
* Give the user a confidence band in real-time;
  + Predict curve of density;
* Login of our app with the data collected from the API;
* Read QR/barcode;
* Search available bluetooth devices in case of QR/barcode damage;
* Improved security with Diffie-Hellman algorithm;
* Receive data from microcontroller and show it in our app through BLE;

# 5 System Architecture



We have a smartphone running our app. Our app will communicate to the webapp using the API. We also have a microcontroller running a BLE server. The app will connect to the microcontroller via QR/Barcode, read data and show it in the mobile app. Our app will also analyse and give the users confidence bands related to datasets collected from sensors.

# 6 Tests and validation

**We have implemented**:

* A BLE Server running on the **ESP32** microcontroller:
  + Using **ESP32** Arduino framework;
  + 1 Service with 4 characteristics used to implement a real time data transfer protocol[**Client** writes to “*characteristic 4*” a KEY, **Server** reads the sensor data and notifies the **Client**, **Client** reads the data];
  + CallBacks checking device connection status, updating data accordingly;
  + Some additional security introduced with Diffie-Hellman algorithm;
* A QRCode/BarCode Decoder on the web App to decode the BLE **Server** name:
  + The connection will be made through that name collected;
* A BLE **Client** running on the web App that connects to the BLE **Server**:
  + FlutterBlue framework used;
  + **Client** tries to connect to the board that has the localName calculated with the QRCode/BarCode Decoder;
  + *If connected* the **Client** implements the real time data transfer protocol mentioned above and updates the Home Page with the given values;
  + End user can reload the values tapping a refresh button;
  + We give the end user the option to see value in seconds intervals if they want;
* A Mobile App
  + An API driven Login:
    - HTTP Request which uses a POST method where a payload is given with the username and password;
    - HTTP Response that returns a JSON Web Token and an API Key that allows to authenticate the user;
  + A method to add/delete notes and processes through the given API in the web App:
    - HTTP Requests;
    - POST to create;
    - GET to list;
    - PATCH to edit;
    - DELETE to remove;

# 7 Developer area

We have implemented a BLE server running in a microcontroller, a QRCode/Barcode decoder, a Mobile App and Data Analysis.

## 7.1 BLE

* Server side
  + Used platformio to compile and push updates to the esp32 board(<https://docs.platformio.org/en/latest/platforms/espressif32.html>);
  + Mainly used the *espressif arduino-esp32* framework developed to create an abstraction layer between the backend of the esp32 and the developer(<https://github.com/espressif/arduino-esp32>);
  + Set the device name as one of the real sensors;
  + Created 1 BLE Server to host all data exchange;
  + Created 2 BLE Service one to send the read values from the sensors and the other to implement Diffie-Hellman algorithm ;
  + On the Sensor Service created 4 BLE Characteristics, one for each sensor(density and temperature), one to send the timestamp of the machine and one to write the pin code that unlocks the board;
  + On the Security Service added 4 characteristics, 2 for the prime numbers and 2 for the public keys from the client and the server;
  + The sensor data is converted using IEEE 754 protocol with Little Endian to 4 bytes which are sent to the client;
  + The read protocol works as follows, the client connects to the board, exchanges the primes numbers and public keys calculated through the Diffie-Hellman algorithm. After that they both encrypt the pin code concatenated with the secret and the client sends it the server. If the hash is correct the client can now activate notifications on the sensor characteristics, the server now starts reading the values from the sensors and notifies the client each time a new value is read. On disconnect the server resets and keeps waiting for another client to connect;
* Client Side
  + Used flutter to make the app design([https://flutter.dev/](https://flutter.dev/?gclid=Cj0KCQjwxYLoBRCxARIsAEf16-v-wldNqEYo1O7kQ_uG3bbKNpKVslb_eew8unP5lzAgew2l_N3MJGkaAn9OEALw_wcB));
  + Used a flutterblue plugin made by Paul de Marco to implement all client side BLE(<https://github.com/pauldemarco/flutter_blue>);
  + The app reads the QrCode or scans nearby devices;
  + After the read or scan is done, the client tries to connect to the boar;
  + After that the same protocol from the server is implemented so reads can only occur after a correct pin is sent to the esp32;
  + The Clients reads indefinitely until a disconnect is requested by the user;
  + The data comes in bytes so it must be converted back to floating point;
  + After that a linear regression must be done to convert the incoming data to user understandable data;
  + In the meantime everything is put on real time graphics so the user can easily evaluate the progress of the readings;

## 7.2 Mobile app

Mobile application designed in Flutter Framework (<https://flutter.dev/>), to support the data already available in the dashboard. It is available for both Android and iOS. All the data in the system is obtained from the dashboard, using HTTP Methods. The system was designed to support five main points:

* Authentication and Authorization
  + To be able to use the system, the user needs to connect with valid credentials (username and password). After successful connection, it produces a JSON web token (JWT), which is an open standard ([RFC 7519](https://tools.ietf.org/html/rfc7519)) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token.
  + To incorporate the HTTP Methods in Flutter, was used the http package (<https://pub.dev/packages/http>).
* QR Code Scanner
  + Used the package barcode\_scan (<https://flatteredwithflutter.com/barcode-scanner-in-flutter/>) to be able to read QR Code and Barcode from the phone’s camera.
  + This produces a value, that is then used to connect to the BLE device.
* Scan BLE Devices
  + The user is able to search for all available devices and connect to the one he desires. This option was introduced in form of error prevention, in case the phone cannot read the QR Code or the QR Code itself has problems.
  + It was used flutter\_blue package (<https://github.com/pauldemarco/flutter_blue>) to interact with the phone’s bluetooth module.
* Notes
  + The user can view, create and modify notes. Notes are in text form and can include images.
* Processes
  + The user can control certain data values of a sensor, unit or batch during a determined time span. It has a beginning and an end date.
  + The user can schedule to finish a process in the desired date.
* Data Analysis
  + To incorporate the Data Analysis part of the project, it was created a page in the application dedicated to it.
  + This page has years of data from a density sensor. It shows the graphs and presents the confidence band (described in the Data Analysis section in more detail).
  + To present the graphs, the package charts\_flutter (<https://google.github.io/charts/flutter/gallery.html>) and to read the CSV files, was used the package csv.

This application has full connectivity with the dashboard, which means, changes made in the application will also be on the dashboard. All the data read from the sensors are in real time using the BLE server implemented.

## 7.3 Data Analysis

The data analysis module receive the sensor data (temperature, density and timestamp) that constantly analyses the values read and checks if the data is inside the predicted interval (sensors can have invalid reads).

To calculate the interval of the predict band we use

{

dens: int,

timestamp: float,   
 previous\_value: int[ ][ ]

}

Inside this function the values are processed by a method fit and predict, that determine the returned value from the function, a tuple [lower\_value, upper\_value], those values define the expectable interval to the next value. Therefore the sensor can miss the reading and we normalize and interpolate missed values.

The user is visually notified with a graph if the value read is out of the predicted band.

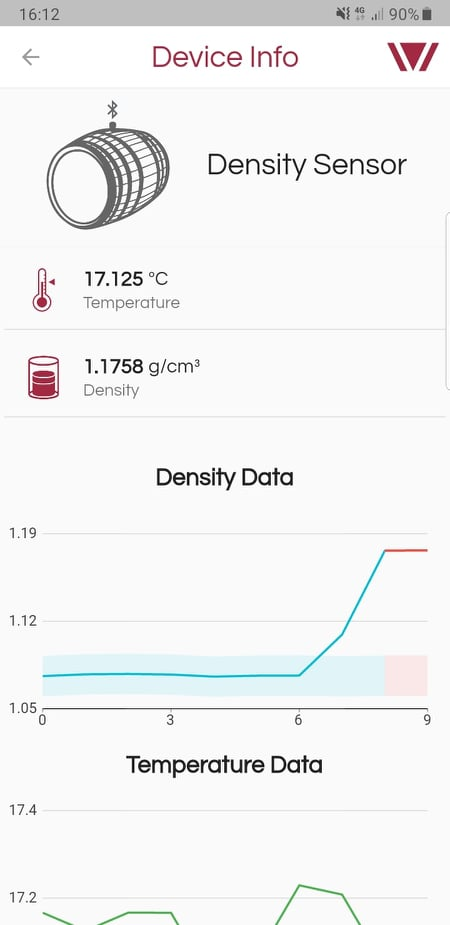
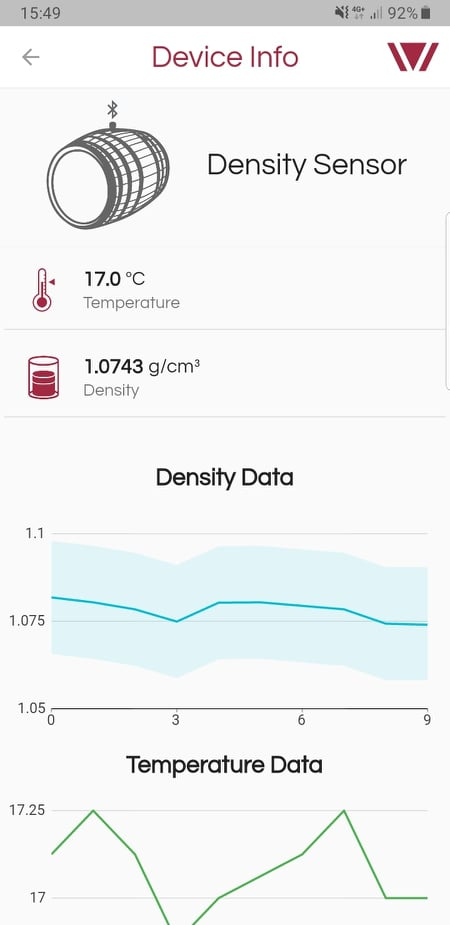


Fig 1. Values are inside the predicted interval Fig 2. Values are out the predicted interval