TELECOM PI

Final Report

REVISION HISTORY AND APPROVAL RECORD

Revision	**Date**	**Purpose**
0	23/11/2020	Document creation
1	29/12/2020	Document revision
2	30/12/2020	Approval of the document

DOCUMENT DISTRIBUTION LIST

Name	**E-mail**
Group members	
Pau Villaverde	pauvima15@gmail.com
Adrian Catalin	catalindiaconeasa@gmail.com
Andrea De Requesens	aerdna.4422@gmail.com
Client/specifier/professor	
Josep Cotrina	josep.cotrina@upc.edu

WRITTEN BY: Andrea De Requesens Martí		REVIEWED AND APPROVED BY: Pau Villaverde	
Date	22/12/2020	Date	31/12/2020
Name	Andrea De Requesens	Name	Pau Villaverde
Position	Docum. Resp.	Position	Project leader

0. CONTENTS

1. DOCUMENT SCOPE

The purpose of this document is to explain our project in detail. We explain all the parts of our project from the beginning.

First of all, this document will include a brief summary of all the steps we have followed and the difficulties we have encountered, so as not to repeat the same mistakes in the future.

Although we developed the first two parts individually, doing these parts individually has provided key things and necessary knowledge for the development of our project.

That was good because everyone has learned different things and how to do them differently when the purpose was the same for everyone. The third phase was the most complicated and due to the situation, we had to do it from home but we never lost contact as we had different meetings planned and a WhatsApp group where we kept ourselves constantly informed.

At the time of organizing, we used Tom's planner which is a tool that allows you to make a Gantt Diagram. We basically used it for having a reference of what work rhythm we should have in order to meet the deliveries. And to be clear about everything that needs to be done.

On the other hand, this document will also include the system design documentation, the implementation documentation and the characterization of the system.

Finally, once all the technical details have been explained, a cost analysis will be made as it was done in the business plan report and we will evaluate how much money will have to be invested and discuss whether or not the project is profitable

2. PROJECT SUMMARY

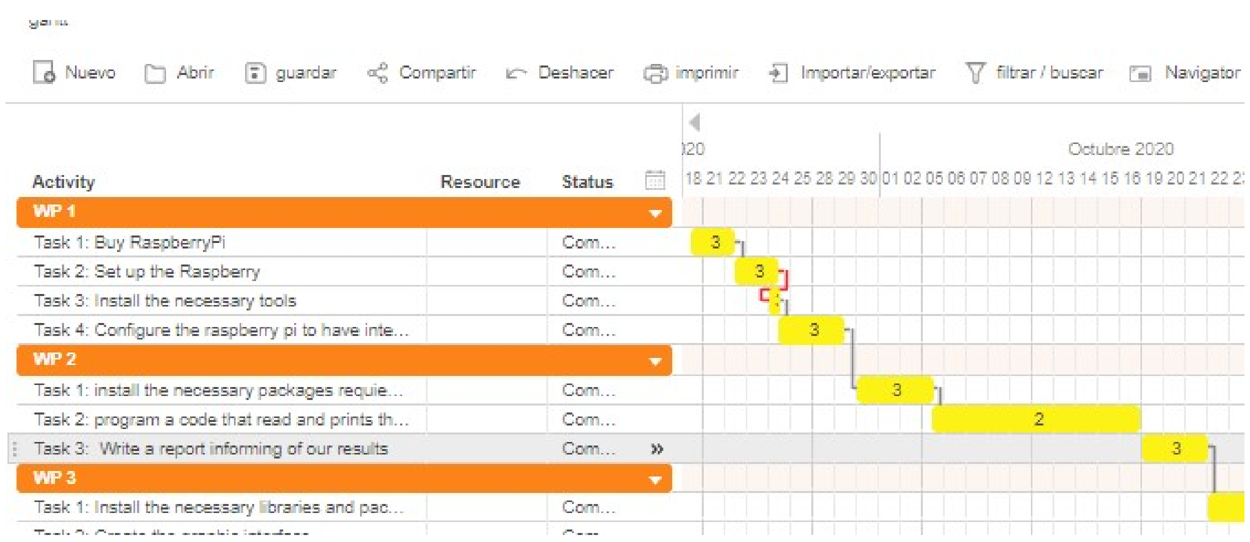
The ultimate goal of this project was to create a client / server application. To do this we divided the project into three parts Puzzle 1, Puzzle 2 and CDR.

First of all, we had to buy a Raspberry Pi and set it up. This will cost us quite a bit so we decided to spend some time researching and getting to know the RaspberryPi and how the Raspbian system works. Once up we start the first part by choosing a peripheral for each one of the group, we took the Rfid-MFCR522, NFC and LCD screen the three of us decided to program in python, the peripherals Rfid-MFCR55 and NFC are two readers which they had a uid and the LCD screen was used to print the desired screen. Therefore, the ultimate goal of this part is for the readers of a program that reads the uid of a rfid card and for the LCD screen a program that prints with the desired structure.
In the second part was implemented a graphical interface that asks in a window to pass the card to indicate to the user that the card can already be passed. Once the card is passed, we will see in the window inside a label the UID of the card. Furthermore, we added a button inside the window, to be able to delete the information and repeat the process as many times as necessary. On the other hand, the purpose of the LCD display was to print to the screen the message that the user writes on the computer.

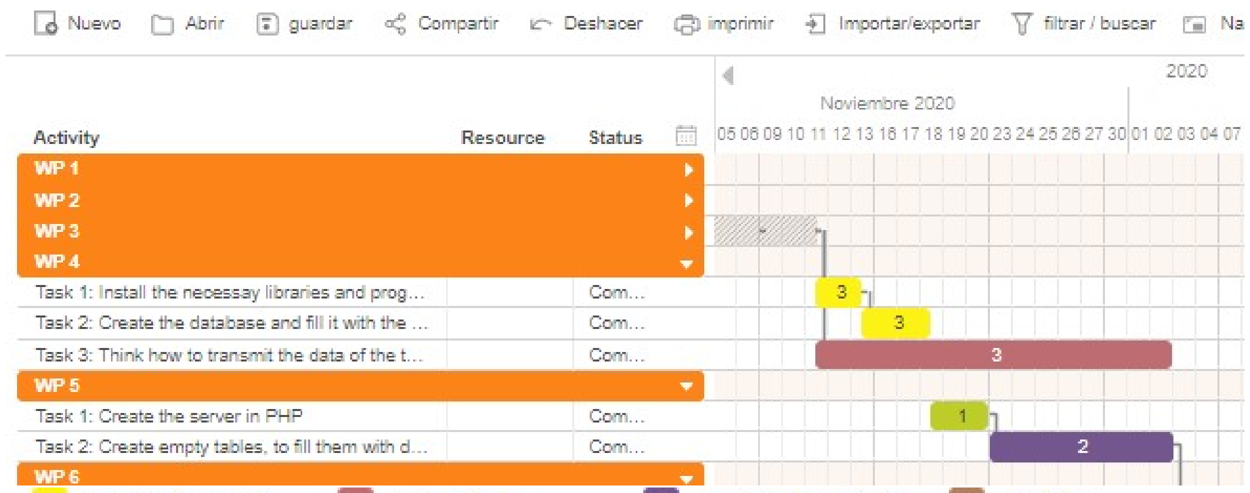
The last part of the project aims to allow users to consult three tables (tasks, schedules, notes) stored in a MySQL database. That is why the project was divided into two parts: Server and Client.
First of all, we create the databases and we fill it with the necessary tables,
then we wasted a lot of time thinking about how we can transmit the information from the MySQL database to the server. While we were thinking how to do the connections, we started programming the server in PHP where we create empty tables with the purpose of refill it with the dates of the database.
Secondly, we find out how to do the connections of the server with the database and also with the client. So, the next step was to generate the queries to show us the desired tables. At this point we could already say that we would have a basic functionality of what we would like our project to be.

Finally we must make sure that everything of our program is fitting well in the graphic interfaces and finishing perfecting the program design with CSS, to make it more comfortable and attractive for the users.

3. TIME PLAN UPDATED



Activity	Resource	Status
WP 1		
Task 1: Buy RaspberryPi		Com…
Task 2: Set up the Raspberry		Com…
Task 3: Install the necessary tools		Com…
Task 4: Configure the raspberry pi to have inte…		Com…
WP 2		
Task 1: install the necessary packages requie…		Com…
Task 2: program a code that read and prints th…		Com…
Task 3: Write a report informing of our results		Com…
WP 3		
Task 1: Install the necessary libraries and pac…		Com…



Activity	Resource	Status
WP 1		
WP 2		
WP 3		
WP 4		
Task 1: Install the necessay libraries and prog…		Com…
Task 2: Create the database and fill it with the …		Com…
Task 3: Think how to transmit the data of the t…		Com…
WP 5		
Task 1: Create the server in PHP		Com…
Task 2: Create empty tables, to fill them with d…		Com…
WP 6		

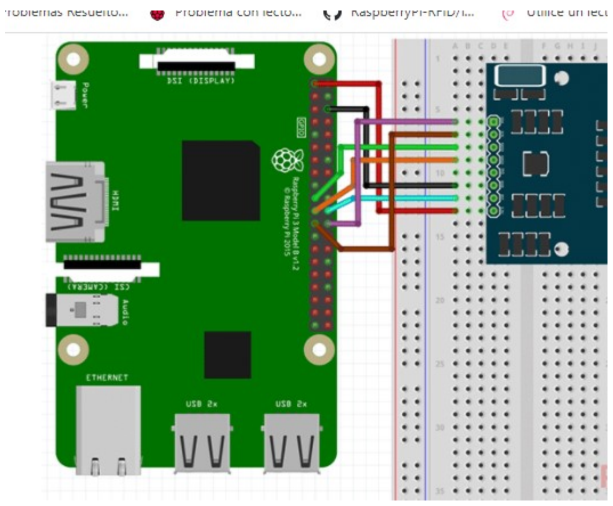
🟨	COMPLETE AND SUCCSESSFUL AC PV ADR
🟥	DELAY BUT COMPLETE
🟩	AC COMPLETE ON TIME
🟪	PV COMPLETE ON TIME
🟫	AC & PV COMPLETE ON TIME

We didn't find any problems during the project, only one that was how to do the different connections between the database and the server and between the server and the client.

4. SYSTEM DESIGN DOCUMENTATION

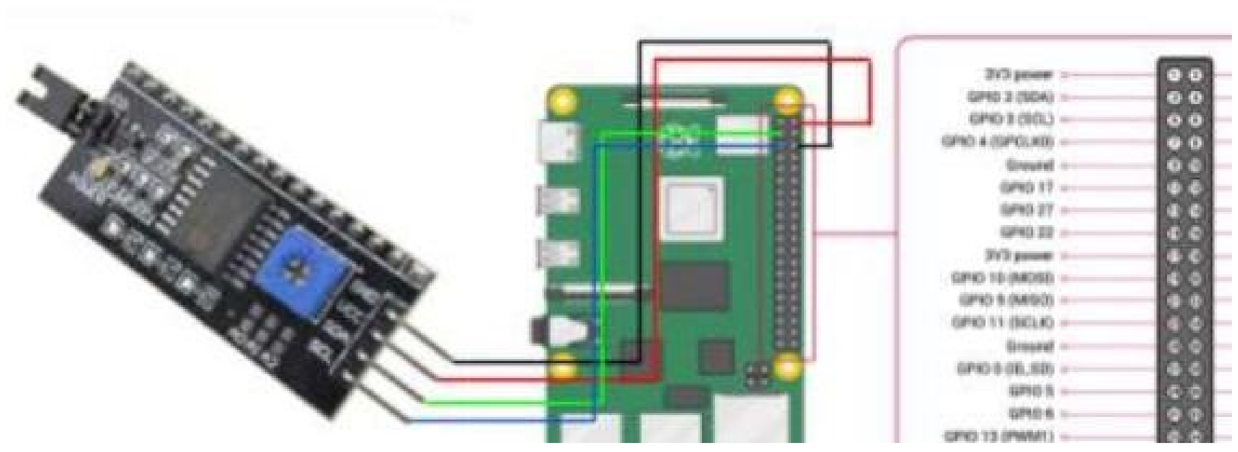
In this part you will be able to see the diferents connections between the elements that we have used for the project:

- This is a schematical of how we connect the RFID MFCR-522 to our Raspberry Pi:



- **SDA** connects to **Pin 24**.
- **SCK** connects to **Pin 23**.
- **MOSI** connects to **Pin 19**.
- **MISO** connects to **Pin 21**.
- **GND** connects to **Pin 6**.
- **RST** connects to **Pin 22**.
- **3.3v** connects to **Pin 1**.

This is a schematical of how we connect the LCD display to our Raspberry Pi:



5. SYSTEM IMPLEMENTATION DOCUMENTATION

USED CODE

- **Puzzle 1 Rfid MFCR-522**

```
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

class Rfid:
  def __init__(self):
    self.reader = SimpleMFRC522()

  def read_uid(self):
    try:
      uid = self.reader.read_id()
      return hex(uid).upper().strip("0X")

    finally:
      GPIO.cleanup()
```

- **Puzzle 1 LCD**

```
import I2C_LCD_driver
import sys
from time import *

class my_lcd():

  def __init__(self):

    self.mylcd = I2C_LCD_driver.lcd()

  def display(self, msg, y, x):
      self.mylcd.lcd_display_string(msg.replace("\n", ""), y, x)

  def clear(self):
    self.mylcd.lcd_clear()
```

- **SERVER PHP**

```
<?php
session_start();
// Variables globals
$host = "localhost";
$user = "root";
$password = "agrfs910.";
$db = "PBE";

// Determinem quina funció s'ha de fer segons la taula
switch(parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH)){
        case "/students":
                login();
                break;
        case "/tasks":
                dbsearch();
                break;
        case "/timetables":
                dbsearch();
                break;
        case "/marks":
                dbsearch();
                break;
        case "/logout":
                logout();
                break;
```

```
				default:
					echo "Invalid petition";
		}
		// Funció que inicia sessió (Troba el nom de l'usuari)
		function login(){
				global $host, $user, $password, $db;

				// Connexió amb l base de dades
				$conn = mysqli_connect($host, $user, $password, $db);

				// Retorna error si no es pot realitzar la connexió amb la base de dades
				if (mysqli_connect_errno()) {
					printf("Failed to connect %s\n", mysqli_connect_error());
					exit();
				}

				// Obtenim la variable QUERY_STRING actual i separem per variables
				$str = $_SERVER['QUERY_STRING'];
				parse_str($str);

				// Enviem la query a la base de dades
				$result = mysqli_query($conn, "SELECT name FROM students WHERE student_id = '$student_id'");

				// Emmagatzemem el resultat de la consulta en un array
				$rows = array();
				while($r = mysqli_fetch_assoc($result)) {
					$rows[] = $r;
				}

				// Retornem l'array en format json
				echo json_encode($rows);

				mysqli_close($conn);

				//Comprovem que l'usuari està a la base de dades i iniciem un comptador
				if(count($rows) == 1){
					$_SESSION['LAST_ACTIVITY'] = time(); // update last activity time stamp
				}

				//En cas de no ser-hi destruim la sessió actual
				else{
					sleep(1);
					session_unset();
					sleep(1);
					session_destroy();
					return;
				}
		}

		// Funció que fa la consulta de la query
		function dbsearch(){
				global $host, $user, $password, $db;

				if (isset($_SESSION['LAST_ACTIVITY']) && (time() - $_SESSION['LAST_ACTIVITY'] > 20)) {
	// last request was more than 20 seconds ago
	echo "Not logged in";
	session_unset();     // unset $_SESSION variable for the run-time
	session_destroy();   // destroy session data in storage
	return;
				}
				$_SESSION['LAST_ACTIVITY'] = time(); // update last activity time stamp

				$conn = mysqli_connect($host, $user, $password, $db);

				// Obtenim la taula on farem la consulta
				$table = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
```

```
        $table = preg_replace("/[^a-zA-Z0-9\s]/", "", $table);

        // Retorna error si no es pot realitzar la connexió amb la base de dades
        if (mysqli_connect_errno()) {
            printf("Failed to connect %s\n", mysqli_connect_error());
            exit();
        }

        // Iniciem la sentència de la query
        $query = "SELECT * FROM $table ";

        // Creem un array on emmagatzemem parelles de dades (clau, valor)
        $carray = array();
        $str = $_SERVER['QUERY_STRING'];
        parse_str($str, $carray);

        // Recorrem l'array i afegim les constraints a la query
        $i = 0;
        foreach($carray as $constr=>$cvalue){
            switch($constr){
                case "limit":
                    $query .= "$constr $cvalue ";
                    break;
            default:
                if($i!=0){
                    $query .= "and ";
                } else {
                    $query .= "WHERE ";
                }
                $i++;
                $query .= "$constr = '$cvalue' ";
            }
        }

        // Enviem la query a la base de dades
        $result = mysqli_query($conn, $query);

        // Emmagatzemem el resultat de la consulta en un array
        $rows = array();
        while($r = mysqli_fetch_assoc($result)) {
            $rows[] = $r;
        }

        // Retornem l'array en format json
        echo json_encode($rows);

        mysqli_close($conn);
    }

    function logout(){
        session_unset();
        session_destroy();
    }

    ?>
```

- **CLIENT PYTHON**

```
import gi
import threading
import requests
import math

gi.require_version("Gtk", "3.0")

from gi.repository import Gtk, Gdk, GLib
from puzzle1 import Rfid    #import de lector de targetes

#si fa falta s'ha de canviar a la direcció on es troba el puzzle de l'LCD
import site
site.addsitedir('/home/pi/Desktop/Puzzle 1 ')
import puzzle_lcd #import de l'LCD

# Variable global d'usuari
user = {
   "uid" : "",
   "name" : ""
}

# Variable global de query
query = ""

# Variable global de la sessió
s = requests.Session()


# Classe de la finestra
class Window(Gtk.Window):

   # Inicialitza i configura Window
   def __init__(self):

      # Es crea un objecte LCD
      self.lcd = puzzle_lcd.my_lcd()

      # Fem que l'LCD mostri per pantalla el missatge de login
      self.lcd.clear()

      self.lcd.display("Please, login with",2,1)
      self.lcd.display("your university card",3,0)

      # Es crea una variable ACK per controlar quan es pot utilitzar el botó
      self.ACK = 0;

      # Inicia la finestra amb un títol i un tamany
      Gtk.Window.__init__(self, title="Uid Scanner")
      self.set_size_request(800, 500)

      # Quan es tanca la finestra, s'acaba el programa
      self.connect("destroy", Gtk.main_quit)

      # Es crea un widget tipus Fixed per a mostrar els widgets
      # a la posició desitjada i s'afegeix a la finestra
      self.fixed = Gtk.Fixed()
      self.add(self.fixed)

      # Es crea un botó invisible
      self.button = Gtk.Button(label="Logout")
      self.button.connect("clicked", self.on_button_clicked)
      self.button.set_opacity(0)

      # Es crea un widget tipus Label per a mostrar text
      self.label = Gtk.Label()
      self.label.set_use_markup(True)
      self.label.set_label("Please, login with your university card")
      self.label.set_name("login")

      self.nameLabel = Gtk.Label()
      self.nameLabel.set_use_markup(True)
      self.nameLabel.set_label("")
      self.nameLabel.set_opacity(0)
# Es creen 2 models de llistes i un widget TreeView per a mostrar les taules
```

```
        self.listmodel = Gtk.ListStore(str, str, str)
        self.listmodel2 = Gtk.ListStore(str, str, str, str)
        self.result = Gtk.TreeView()
        self.result.set_name("table")


        # Es crea un widget tipus Entry per a introduir la query
        self.entry = Gtk.Entry()
        self.entry.set_size_request(780,40)
        self.entry.connect("key_press_event", self.on_key_press_event)
        self.entry.set_opacity(0)

        # Es posen els widgets a les posicions desitjades
        self.fixed.put(self.label, 230,235)
        self.fixed.put(self.button, 690, 10)
        self.fixed.put(self.entry, 10, 75)
        self.fixed.put(self.result, 10, 150)
        self.fixed.put(self.nameLabel, 10,10)

        # Es mostra la finestra amb el seu contingut
        self.show_all()

        # S'inicia un thread on s'executarà read_uid
        thread = threading.Thread(target=self.read_uid)
        thread.daemon = True
        thread.start()

    # Determina que es fa al fer click al botó
    def on_button_clicked(self, widget):
        # El botó només funciona quan s'ha llegit una targeta
        if self.ACK is 1:
            # Reiniciem el valor d'ACK
            self.ACK = 0

            # Canviem el text del Label
            GLib.idle_add(self.label.set_name, "login")
            self.lcd.clear()
            self.lcd.display("Please, login with",2,1)
            self.lcd.display("your university card",3,0)
            self.label.set_label('Please, login with your university card')
            self.label.set_opacity(1)
            self.nameLabel.set_opacity(0)
            self.entry.set_opacity(0)

            # En cas d'estar visualitzant una taula l'esborrem
            self.listmodel.clear()
            self.listmodel2.clear()

            # Fem el botó invisible
            self.button.set_opacity(0)

            threading.Thread(target=self.http_com("logout", "logout"), daemon = True).start()

            # Es torna a iniciar un thread on s'executarà read_uid
            thread = threading.Thread(target=self.read_uid)
            thread.daemon = True
            thread.start()

    # Determina que es fa al polsar una tecla
    def on_key_press_event(self, widget, event):
        # Tecla Enter
        if Gdk.keyval_name(event.keyval) == 'Return':
            # Obtenim el text i executem un thread on s'executarà http_com
            query = self.entry.get_text()
            self.entry.set_text("")
            threading.Thread(target=self.http_com("search", query), daemon = True).start()
```

```
# Implementació de Rfid.read_uid()
    def read_uid(self):
        # Obtenim l'uid de la targeta i actualitzem la variable global usuari
        rf = Rfid()
        uid = rf.read_uid()

        # Actualitzem el valor d'ACK
        self.ACK = 1

        #global user
        user["uid"] = uid
        threading.Thread(target=self.http_com("login", "students?student_id="+uid), daemon = True).start()

        # Fem el botó visible
        GLib.idle_add(self.button.set_opacity, 1)

    # Envia la query al servidor i retorna la taula
    def http_com(self, request, query):
        # Esborrem les columnes i les files prèvies
        for col in self.result.get_columns():
            self.result.remove_column(col)
        self.listmodel.clear()
        self.listmodel2.clear()

        # Enviem la query al servidor i la transformem a format json
        global s

        r = s.get("http://127.0.0.1:81/" + query)

        if(r.text == "Not logged in"):
            self.lcd.clear()
            self.lcd.display("Session expired,",2,2)
            self.lcd.display("please login again",3,1)
            GLib.idle_add(self.label.set_label,"Session expired, please login again") #'Please, login with your university card'
            GLib.idle_add(self.label.set_opacity, 1)
            GLib.idle_add(self.nameLabel.set_opacity, 0)
            GLib.idle_add(self.entry.set_opacity, 0)

            # En cas d'estar visualitzant una taula l'esborrem
            self.listmodel.clear()
            self.listmodel2.clear()

            # Fem el botó invisible
            GLib.idle_add(self.button.set_opacity, 0)

            thread = threading.Thread(target=self.read_uid)
            thread.daemon = True
            thread.start()
            return

        if(request == "logout"):
            return

        jsn = r.json()
        #global user
        # En cas de ser un inici de sessió
        if request is "login":
            # Actualitzem la variable global usuari i el text del Label
            if(jsn):
                user["name"] = jsn[0]["name"]
                self.lcd.clear()
                self.lcd.display("Welcome",2,7)
                n = math.trunc((20-len(user["name"]))/2)
                self.lcd.display(user["name"], 3, n)
                GLib.idle_add(self.button.set_label, "Logout")
                GLib.idle_add(self.nameLabel.set_label,"Welcome " + user["name"])
                GLib.idle_add(self.nameLabel.set_opacity, 1)
                GLib.idle_add(self.label.set_opacity, 0)
                GLib.idle_add(self.entry.set_opacity, 1)
            else:
                self.lcd.clear()
                self.lcd.display("Invalid user card,", 2, 1)
                self.lcd.display("please try again", 3, 2)
                GLib.idle_add(self.label.set_name, "invalid")
                GLib.idle_add(self.label.set_label, "Invalid user card, please try again")
                GLib.idle_add(self.label.set_opacity, 1)
                GLib.idle_add(self.button.set_label, "Try again")


        # En cas de ser una búsqueda
        elif request is "search":
            # Si és la taula timetables, es necessiten 4 columnes
            if "timetables" in query:
                lm = self.listmodel2

                # Amplada a aplicar a les columnes
                sz = 195

            # Si és qualsevol altra taula, es necessiten 3 columnes
            else:
                lm = self.listmodel

                # Amplada a aplicar a les columnes
                sz = 260

            # Apliquem el model de llista al TreeView
            self.result.set_model(model=lm)

            # Afegim els valors al model
            for i in range(len(jsn)):
                lm.append(jsn[i].values())

            # Per cada columna
            for i, column in enumerate(jsn[0].keys()):
                # Renderitzem el text
                cell = Gtk.CellRendererText()

                # Creem la columna i determinem la seva amplada
                col = Gtk.TreeViewColumn(column, cell, text=i)
                col.set_fixed_width(sz)

                # Afegim la columna al TreeView
                self.result.append_column(col)
def app_main():
    win = Window()
    win.style_provider = Gtk.CssProvider()
    win.style_provider.load_from_path('custom.css')
    Gtk.StyleContext.add_provider_for_screen(
        Gdk.Screen.get_default(), win.style_provider,
        Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION
        )


# Programa principal
if __name__ == "__main__":
    app_main()
    Gtk.main()
```

- **CUSTOM IN CSS**

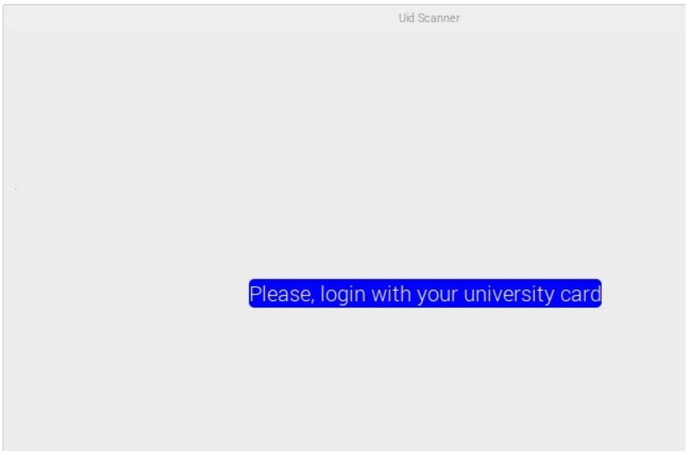
```
#login{
        color:white;
        border-radius:5px;
```

```
            background-color:blue;
            font-size:20px;
    }
    #invalid{
            color:white;
            border-radius:5px;
            background-color:red;
            font-size:20px;
    }
    #table{
        background-color:cyan;
    }
```

6. SYSTEM CHARACTERIZATION

HOW TELECOM PI LOOKS?

First of all, you will see clearly the login and then you only have to follow orders.


Uid Scanner

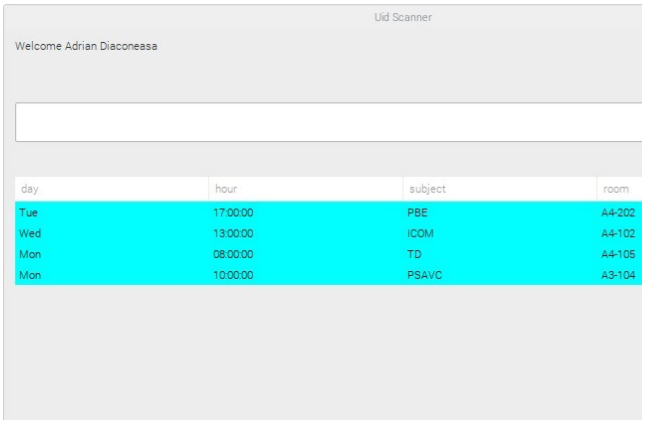
Please, login with your university card

Then you can process to write what you want to consult in the query bar.

On the search bar you can make as many queries as you want

For example:

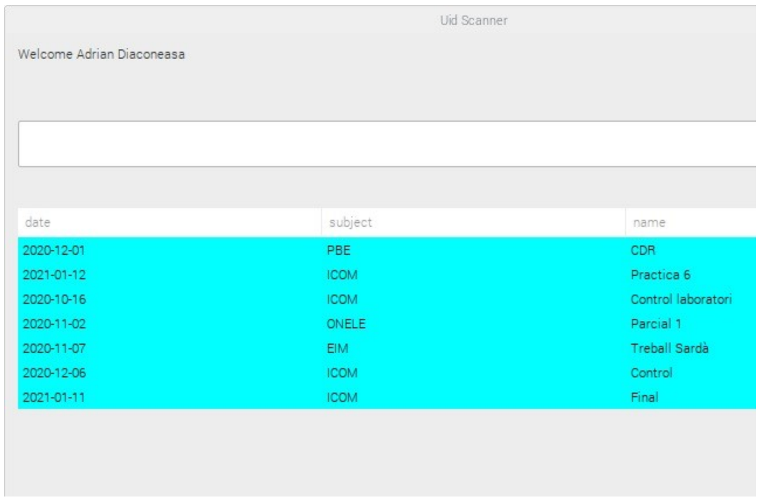
1. Timetables:


Uid Scanner

Welcome Adrian Diaconeasa

day	hour	subject	room
Tue	17:00:00	PBE	A4-202
Wed	13:00:00	ICOM	A4-102
Mon	08:00:00	TD	A4-105
Mon	10:00:00	PSAVC	A3-104

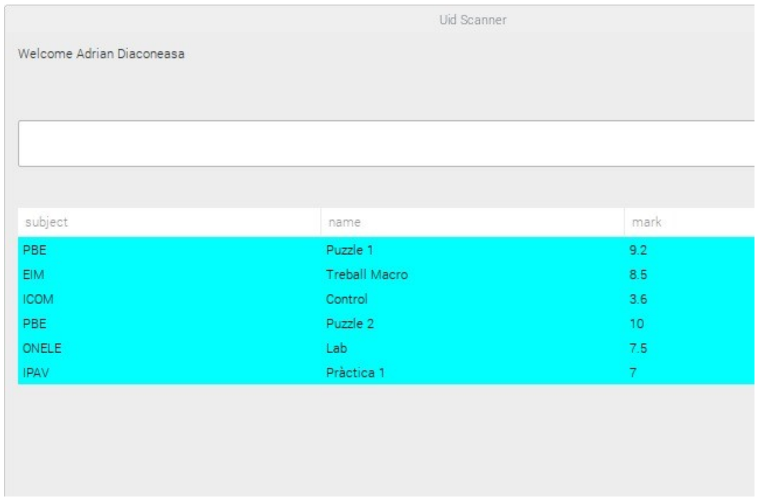
2. Tasks



Welcome Adrian Diaconeasa

date	subject	name
2020-12-01	PBE	CDR
2021-01-12	ICOM	Practica 6
2020-10-16	ICOM	Control laboratori
2020-11-02	ONELE	Parcial 1
2020-11-07	EIM	Treball Sardà
2020-12-06	ICOM	Control
2021-01-11	ICOM	Final

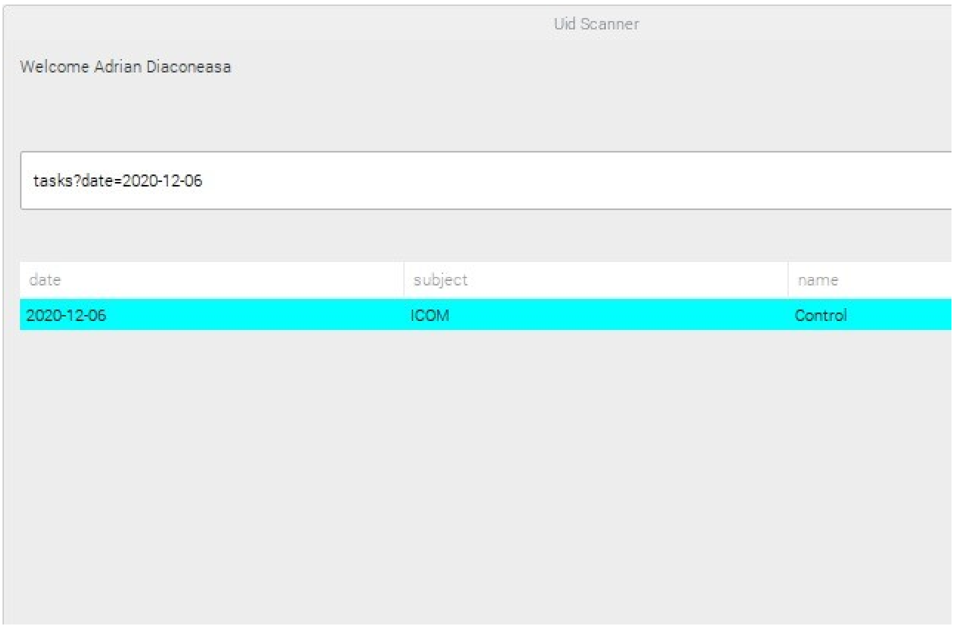
3. Marks



Welcome Adrian Diaconeasa

subject	name	mark
PBE	Puzzle 1	9.2
EIM	Treball Macro	8.5
ICOM	Control	3.6
PBE	Puzzle 2	10
ONELE	Lab	7.5
IPAV	Pràctica 1	7

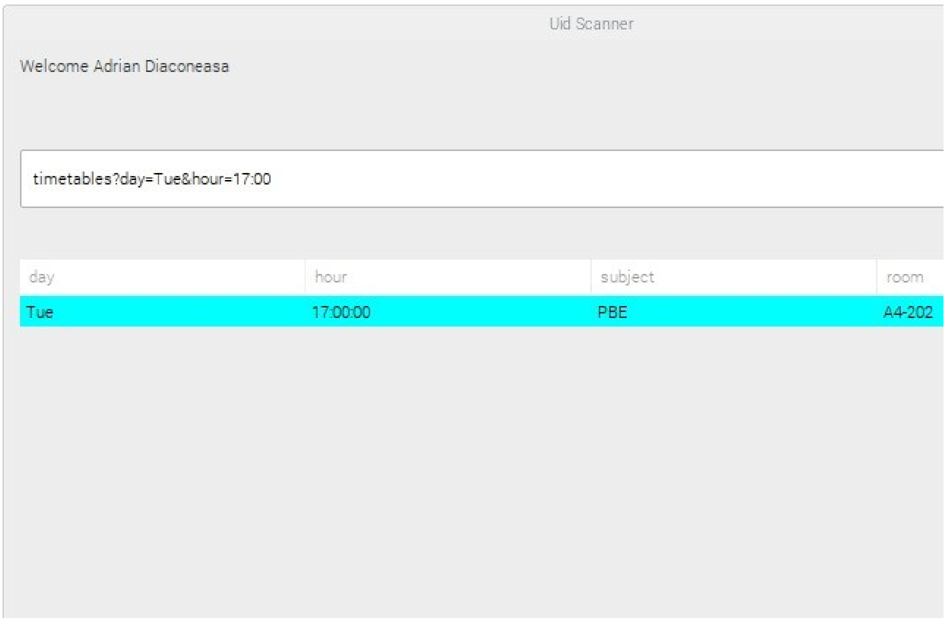
Also, you can make more specifical queries such as for example “tasks?date=2020-12-06” the program will only return the tasks for the assigned date. And the same type of queries can be made in the different tables.


Uid Scanner

Welcome Adrian Diaconeasa

tasks?date=2020-12-06

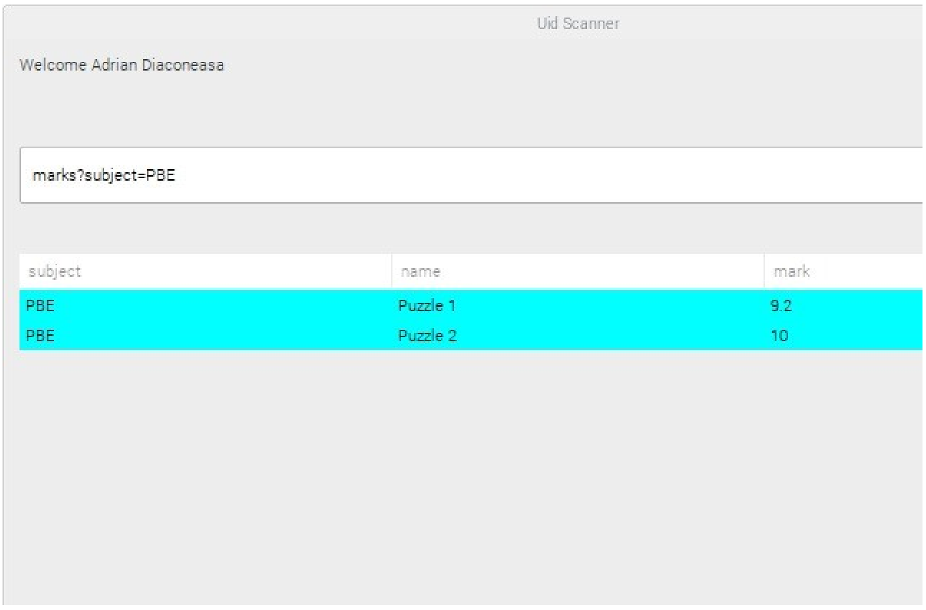
date	subject	name
2020-12-06	ICOM	Control


Uid Scanner

Welcome Adrian Diaconeasa

timetables?day=Tue&hour=17:00

day	hour	subject	room
Tue	17:00:00	PBE	A4-202

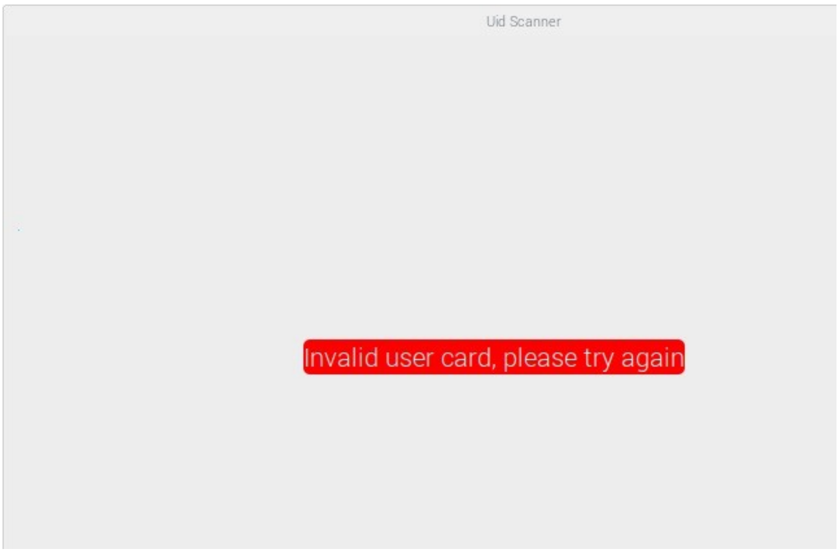

Uid Scanner

Welcome Adrian Diaconeasa

marks?subject=PBE

subject	name	mark
PBE	Puzzle 1	9.2
PBE	Puzzle 2	10

And finally, here some error message that we used to get advice and protect our customers, first of all if you don't login with the correct card or someone is trying to get access to your information the program will show you this message:


Uid Scanner

Invalid user card, please try again

Also, if you have already login and you spend some time without making any queries, the program would logout automatically for security.

7. COSTS

This kind of project did not require a large amount of material; however it requires a lot of personal work and time. Basically, what is needed
is a RaspberryPi 4 4GB Kit, an Rfid MFCR-522 sensor or an NFC and an LCD screen.
But in order to be able to develop the project and work more efficiently and individually, we needed a little more material. Therefore, the total cost of the whole project would be:

DIRECT COST
Physical material

Equipment	Unit.	Price/Unit	Price/Total	Years amortization	Final Price/Total	Amortization/year
Ordinadors	3.0	700	2.000	5	750	250
RaspberryPi 4 4G Kit	3.0	98,59	295,77	5	120	33,154
Rfid MFRC-522	1.0	6,99	6,99	3	1	1,99
NFC	1.0	8,99	8,99	3	1,5	2,49
LCD	1.0	9,99	9,99	3	3	2,33
Total			2.321,74			

Salaries:

	Price/Hour	Hours/Month	Price/Month	Hours/Year	Total price
Class work	8.0	12	96	144	1.152
Home work	17.0	24	408	288	4.896
Total work	25.0	36	504	432	6.048

Number of Workers	Salary/worker/Year	Total Salary	Salary without Social security status tax 30%	Total Salary
3	6.048	18.144€	4.233,6€	12.700.8€

INDIRECT COSTS
Fixed cost

	total/month	total/year
rental work office	800€	2400€
Internet	85€	1200€
Product delivery (company)	300€	3600€
Patent	-	4000€
TOTAL		11200€

Variable cost

	total/month	total/year
water	120€	1440€
electricity	200€	2400€
Total	320€	3840€

TOTAL COST: 30.062,54€
We think that for this kind of project the money needed to carry it out is affordable. Because it's a project that has a lot of possible customers and in the future will give us many benefits so we have no doubt that big businesses will invest in our company.

8. CONCLUSIONS

Our team is satisfied with the result of the project as the main goal has been achieved. The results have been very successful and we will soon be able to bring the project to market. We are now beginning to develop the web client part, in order to make our excess even easier for our clients.

In this project we learned a lot because we had never programmed in python, nor had we done any web development so we had to learn CSS and PHP.

On the other hand, it has been a bit complicated because from the beginning we had already thought about how to work in a group at the university and due to the situation in which we found ourselves and the world we had to reorganize quickly to work separately so as not to waste time and work efficiently. Nowadays we have a lot of available tools that made our job easier even though we believe that if we had developed the project physically together, we would have gotten better results.

Finally, it should be noted that the web model is already being worked on and that we hope that it will be up and running by mid-January.

9. REFLECTION DOCUMENT

We think that first of all we should have devoted more time to study and learn how to program in the different languages, because at the time of developing the project it would have been easier. But that was not possible because we have a limited time to deliver our project.

As a team I think that we should have made more team because we started the project without knowing each other in weird situations as we find ourselves now. So, we have very few situations where we have seen each other. That make us feel more uncomfortable when we had to communicate some problem, or when we got stock and we don’t know how to keep going or when basically you don’t know how to do something but as professionals, we’re we have been able to deal with it.

The performance of the team has been correct, despite what has been explained above, we have been able to distribute the work and each has ended up doing what was easier for him and felt more able to do.

Name	**Mark**
Pau Villaverde	10
Adrián Catalin	10
Andrea De Requesens	10