[PBE intranet]

Final Report

REVISION HISTORY AND APPROVAL RECORD

Revision	**Date**	**Purpose**
0	14/05/24	Document creation
1	dd/mm/yyyy	Document revision

DOCUMENT DISTRIBUTION LIST

Name	**E-mail**
Alberto de Antonio	
Joaquin Mas	
Pau Climent	
Pau Burgos	
Lucas Mira	
Fransesc Oller	

WRITTEN BY:		REVIEWED AND APPROVED BY:	
Date	14/05/24	Date	dd/mm/yyyy
Name	Alberto de Antonio	Name	Zzzzzzz Wwwwwww
Position	e.g. Docum. Resp.	Position	e.g. Project leader

0. CONTENTS

1. DOCUMENT SCOPE

The goal of this document…. (Information about the document scope that allows the reader differentiating it from the other documents)

This document details the scope and objectives of the project, focusing on the development of a client application that communicates with a server connected to a database. Additionally, clients for Android and web will be implemented in the final stages of the project.

The project involves the development of a client application capable of communicating with a server connected to a database. The client application will communicate with the server to perform various tasks, such as data retrieval, storage, and processing. Additionally, clients for Android and web will be developed to provide user interfaces for accessing the system.

Specifically, our project focuses on an application similar to what we call a university virtual campus, that is, an intranet where the client can consult all the information they need about their progress in the degree program, class schedules, pending tasks, test results, ...

2. PROJECT SUMMARY

This section should provide an executive and concise summary of the project which was completed. It is important that this summary captures the scope of the project and contains enough detail to provide a full understanding of the project. Since this document will communicate what went right and wrong with the project, as well as lessons learned and recommendations for future projects, it is imperative that this section provide enough background information to base the details in the rest of the document on.

The project involved the development of an application that facilitates communication between a client and a server connected to a database. The main objective was to create a comprehensive solution that enables users to interact with devices remotely through intuitive user interfaces on Android and web devices.

The client application was designed to manage various functions, such as user authentication using radio frequency (RF) cards, reading unique identifiers (UID) from RFID cards, and displaying information on LCD screens. Additionally, a query system was implemented to allow users to access data stored in the database, such as schedules, tasks, and grades.

For the programming of the peripherals intended for the different client functions, we used a Raspberry Pi, which is a low-cost microcomputer board. Providing internet access to this device posed one of the most recurrent problems during development, as the Windows 11 operating system specifically encountered issues with the Internet Connection Sharing (ICS) function.

3. TIME PLAN UPDATED

- A posteriori, actual Time Plan
- Comment delays and changes (non performed parts) respect the previous Time Plan

SEMANAS	1	2	3	4	5	6	7	8	9	10	11	12	13
Adquisición de laRaspberry Pi													
Instalación del sistema Operativo													
Elección individual delperiférico													
Instalación de las librerías													
Elaboración del Puzzle 1													
Instalación librería GTK													
Diseño de la interfaz													
Incluir la clase Rfid													
Comprobacion													
Creación de Tablas													
Insertar valores													
Hostear localmente elservidor													
Parsing													
Funciones de Consulta													
Consultas													
Instalación de librerías													
Diseño de la interfaz													
Creación del css													

Diseño final de la GUI													
Correccion Server y GUI													
Cliente Android													

4. SYSTEM DESIGN DOCUMENTATION

- System behavioral modeling (Matlab scripts, simulation results) (if you already kept records of this part. If not, do not reconstruct it)
- System block diagram
- System blocks internal design
- Initial schematics / software blocks description
- Initial active components selection / user interface initial design
- Calculations and component values
-

The Raspberry Pi serves as the central node, orchestrating communication between various peripherals such as RFID devices, the database, and display units. This visual representation aids in understanding how these elements are interconnected to facilitate seamless operation and data flow throughout the system.

In the internal block design, we dissect the intricate functionalities of each system component and explore their interrelationships. The Raspberry Pi, acting as the core processing unit, executes tasks ranging from user authentication to data retrieval and display management. Concurrently, RFID readers interface with the Pi to extract unique identifiers, which are then processed and validated against the database.

The description of software blocks encapsulates the intricacies of our system's digital architecture, delineating the roles and responsibilities of each software module. At its core, the software architecture encompasses modules for user authentication, RFID data processing, database querying, and user interface management. These modules collaborate seamlessly to ensure smooth execution of system tasks, from user identification to data retrieval and presentation.

Embarking on the initial user interface design journey, we envision an interface that seamlessly integrates functionality and aesthetics to deliver an intuitive user experience. For Android and web platforms alike, our design prioritizes clarity and simplicity, with streamlined authentication screens, visually engaging data displays, and intuitive navigation menus. By focusing on user-centric design principles, we aim to create an interface that empowers users to effortlessly interact with our system, enhancing overall usability and satisfaction.

5. SYSTEM IMPLEMENTATION DOCUMENTATION

- Final schematics / software blocks
- Circuit Layout / user interface screen captures
- Final components list with values
- Circuit / device pictures

Código Final de Server:

- Para llevar a cabo el código de Server hemos optado por las siguientes librerías:
  - Express
  - cookieParser: Ayuda a Express a parsear las cookies ya express que no tiene esta función implementada
  - unique String Generator: Generador aleatorio de Strings para crear las sesionesId.

```
const express = require('express');
const expressApp = express();
const PORT = 3000;
const mysql = require('mysql');
const {UniqueNumberId, UniqueStringId} = require('unique-string-generator');
const cookieParser = require("cookie-parser");
const path = require('path');

//Creamos un array de sesiones
const sessions = [];

//Conectar server con el archivo html
expressApp.use(express.static(__dirname));

//Parsear en Json i urlencoded
expressApp.use(express.json());
expressApp.use(express.urlencoded({ extended: true }));
expressApp.use(cookieParser());  //És una funcion para que express pueda parsear las
cookies


// Conexion a Base de datos
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
```

```
    database: 'bd_pbe',
    password: '*******',
    port: '3306'
})

connection.connect((err) => {
    if (err) {
        console.log(err);
        return;
    }

    console.log("Conexion a la base de datos exitosa");

})


// Función parsequery a partir de url
function parseQuery(url) {
    const querystring = url.split('?');
     if (querystring.length === 1) {
        const querytable = querystring[0];
        const query = {
            table: querytable,
        }
        return query;
    } else {
        const querytable = querystring[0];
        const queryconstraints = querystring[1];
        const constraints = queryconstraints.split('&').map(constraint => {
            return constraint.includes('=') ? constraint.split('=')[0] + "='" +
constraint.split('=')[1] + "'" : constraint;
        });
        const query = {
            table: querytable,
            constraints: constraints,
        }
        return query;
    }
}


//Funcion login del servidor
```

```
expressApp.post("/login", (req, res) => {
    const { user, password } = req.body; //Recibimos usuario y password del body que
nos envia cliente
    if (!user || !password) {
        return res.send("Datos no introducidos adecuadamente");
    }

    let sql = `SELECT * FROM students WHERE username = "${user}" AND password =
"${password}"`;//Buscamos en la base de datos un usuario que coincida con la password
    connection.query(sql, (error, results, fields) => {
        if (error) {
            console.error('Usuario o contraseña Incorrectos', error);
            return res.send('Usuario o contraseña Incorrectos');
        }
//En caso de que el usuario y la contraseña sean correctos, creamos un sessionid y lo
añadimos al array de sesiones. Despues le otorgamos la cookie con su sessionid al
cliente para así tenerlo identificado siempre.
        if (results.length > 0) {
            const usuario=results[0];
            const sessionId = UniqueStringId();
            sessions.push({sessionId,usuario});
            res.cookie('sessionId',sessionId,{
                httpOnly: true,
            });
            res.send(`Bienvenido ${usuario.name}`);
        } else {
            res.send('Usuario o contraseña Incorrectos');
        }
    });
});


// En este apartado manejaremos las queries una vez el usuario ya ha hecho el login
expressApp.post("/query", (req, res) => {
    //Primero identificamos de que usuario se trata utilizando su cookie
    const cookies =req.cookies;
    if(!cookies.sessionId){ return res.sendStatus(401);}
    const userSession = sessions.find(
        (session)=> session.sessionId === cookies.sessionId
    );
    if(!userSession){ return res.sendStatus(401);}
    const usuario = userSession.usuario;
```

```
    // Parseamos la query que nos pide cliente
    const{ query} = req.body;
    const parsedquery = parseQuery(query);
    //Construimos la query
    let sql = `SELECT * FROM ${parsedquery.table} WHERE uid = "${usuario.uid}"`;
    if (parsedquery.constraints) {
        parsedquery.constraints.forEach((constraint) => {
            sql += ` AND ${constraint}`;
        });
    }

    //Ejecutamos la query
    connection.query(sql,(error, results, fields) => {
        if (error) {
            console.error('Error al ejecutar la consulta:', error);
            res.status(500).send('Error del servidor');
            return;
        }
        res.json(results);

    });



});

expressApp.listen(PORT, () => {
    console.log("Servidor levantado");
});
```

Parte grafica del cliente web:

- Elaborada con html.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PBE</title>
    <style>
        body {
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            background-color: #f5f5f5;
        }
        .container {
            width: 300px;
            padding: 20px;
            background-color: darkgray;
            border: 1px solid black;
            border-radius: 10px;
            text-align: center;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            margin-bottom: 20px;
        }
        .container-large {
            width: 80vw;
            padding: 20px;
            background-color: darkgray;
            border: 1px solid black;
            border-radius: 10px;
            text-align: center;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            margin-bottom: 20px;
        }
        table {
            width: 100%;
            border-collapse: collapse;
```

```
        margin-top: 20px;
    }
    th, td {
        border: 1px solid #ddd;
        padding: 8px;
        text-align: left;
    }
    th {
        background-color: #f2f2f2;
    }
    .logout-button {
        position: absolute;
        top: 10px;
        right: 10px;
        background-color: red;
        color: white;
        border: none;
        padding: 10px;
        border-radius: 5px;
        cursor: pointer;
    }
    #queryResult {
        margin-top: 20px;
        background-color: #fff;
        border: 1px solid #ddd;
        border-radius: 10px;
        padding: 10px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        width: 100%;
        max-height: 50vh;
        overflow-y: auto;
    }
```

```
        </style>
    </head>
    <body>
        <div class="container-large" style="background-color: rgb(2, 119, 187); border: 1px solid black; text-align: center;
            <h2>Atenea</h2>
        </div>
        <br>
        <div id="loginForm" class="container">
            <form action="/login" method="POST">
                <label for="username">Nombre de usuario:</label>
                <input type="text" id="username" name="user" required><br><br>
                <label for="password">Contraseña:</label>
                <input type="password" id="password" name="password" required><br><br>
                <button type="submit">Iniciar sesión</button>
            </form>
        </div>

        <div id="queryForm" class="container" style="display: none;">
            <button class="logout-button" id="logoutButton">Log out</button>
            <h2>Realizar consulta</h2>
            <form id="querySubmission" action="/query" method="POST">
                <label for="queryInput">Consulta:</label>
                <input type="text" id="queryInput" name="query" required><br><br>
                <button type="submit">Enviar consulta</button>
            </form>
        </div>

        <div id="queryResultContainer" class="container-large" style="display: none;">
            <div id="queryResult"></div>
        </div>

        <script>
            document.getElementById('loginForm').addEventListener('submit', function(event) {
                event.preventDefault();
                const user = document.getElementById('username').value;
                const password = document.getElementById('password').value;
```

```
        fetch('/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ user, password })
        })
        .then(response => response.text())
        .then(data => {
            if (data.includes('Bienvenido')) {
                document.getElementById('loginForm').style.display = 'none';
                document.getElementById('queryForm').style.display = 'block';
            } else {
                alert('Usuario o contraseña incorrectos');
            }
        })
        .catch(error => console.error('Error:', error));
    });

    document.getElementById('querySubmission').addEventListener('submit', function(event) {
        event.preventDefault();
        const query = document.getElementById('queryInput').value;
        fetch('/query', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ query })
        })
        .then(response => response.json())
        .then(data => {
            const queryResultDiv = document.getElementById('queryResult');
            const queryResultContainer = document.getElementById('queryResultContainer');
            queryResultContainer.style.display = 'block';
            queryResultDiv.innerHTML = ''; // Limpiar el contenido anterior
```

```
                queryResultDiv.innerHTML = ''; // Limpiar el contenido anterior
                if (data && data.length > 0) {
                    const table = document.createElement('table');
                    const headers = Object.keys(data[0]);
                    const headerRow = document.createElement('tr');
                    headers.forEach(header => {
                        const th = document.createElement('th');
                        th.textContent = header;
                        headerRow.appendChild(th);
                    });
                    table.appendChild(headerRow);
                    data.forEach(rowData => {
                        const row = document.createElement('tr');
                        headers.forEach(header => {
                            const td = document.createElement('td');
                            td.textContent = rowData[header];
                            row.appendChild(td);
                        });
                        table.appendChild(row);
                    });
                    queryResultDiv.appendChild(table);
                } else {
                    queryResultDiv.textContent = 'No se encontraron resultados.';
                }
            })
            .catch(error => console.error('Error:', error));
        });

        document.getElementById('logoutButton').addEventListener('click', function() {
```

```
            document.getElementById('queryForm').style.display = 'none';
            document.getElementById('queryResultContainer').style.display = 'none';
            document.getElementById('loginForm').style.display = 'block';
            document.getElementById('loginForm').reset();
            document.getElementById('queryResult').innerHTML = '';
        });
    </script>
</body>
</html>
```

Código usado en el cliente Android:

Base de datos:

```
-- Database creation
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Verano2024$';
flush privileges;
CREATE DATABASE if not exists app_db3;

USE app_db3;

-- Table users creation
CREATE TABLE if not exists USUARIO(
   cedula VARCHAR(100),
   contrasena VARCHAR(100),
   nombre VARCHAR(100),
   apellido VARCHAR(100),
   telefono VARCHAR(20)
);
INSERT INTO USUARIO VALUES('Paucli', 'pbe1', 'Pau', 'Climent', '3101234446');
INSERT INTO USUARIO VALUES('Lukas', 'pbe2', 'Lukas', 'Mira', '3111245556');
INSERT INTO USUARIO VALUES('Burgos', 'pbe3', 'Pau', 'Burgos', '3101234344');
INSERT INTO USUARIO VALUES('Joaquin', 'pbe4', 'Joaquin', 'Mas', '3109234446');
INSERT INTO USUARIO VALUES('Alberto', 'pbe5', 'Alberto', 'de Antonio', '3101238446');
```

Servidor:

```
const express = require('express');
const mysql = require('mysql');
const app = express();
const port = 3001;

app.use(express.json());
app.use(express.urlencoded({ extended: false }));

// MySQL connection
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'Verano2024$',
    database: 'app_db3'
});

connection.connect((err) => {
```

```
    if (err) throw err;
    console.log('Connected to MySQL!');

});

// Endpoint for login
app.post('/login', (req, res) => {
    const { username, password } = req.body;

    // Validate the username and password
    const query = 'SELECT * FROM users WHERE username = ? AND password = ?';
    connection.query(query, [username, password], (err, results) => {
        if (err) {
            res.status(500).json({ error: 'Database error' });
        } else if (results.length > 0) {
            res.status(200).json({ message: 'Login successful' });
        } else {
            res.status(401).json({ message: 'Invalid credentials' });
        }
    });
});

app.route('/')

.get((req, res) => {
    let cedula = req.query.cedula;
    let contrasena = req.query.contrasena;

    let query_iniciar = "SELECT * FROM `USUARIO` WHERE `cedula` = ? AND `contrasena`
= ?";

    connection.query(query_iniciar, [cedula, contrasena], (err, results, fields) => {
        if (err) {
            console.log("There was an error");
            console.log(err);
            res.json({
                    'code': 500,
                    'message': "There was an server error."
            });
        } else {

            if (results.length > 0) {
                res.json({
                    'code': 200,
```

```
                    'message': 'Get values with success.',
                    'data': results[0]
                });
            } else {
                res.json({
                    'code': 300,
                    'message': 'There are no users in the database with that
values.',
                });
            }
        }
    })
})



// Start server
app.listen(port, () => {
    console.log(`Server running on port ${port}`);
});
```

Para hacer el cliente Android, he usado Android Studio. Los códigos que he usado son estos:

Código que contiene el main del programa:

```
package com.example.clienteandroid




import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import org.json.JSONObject
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL
import java.net.URLEncoder
```

```
class MainActivity : AppCompatActivity() {

   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
       setContentView(R.layout.activity_main)

       val usernameEditText = findViewById<EditText>(R.id.username_edit_text)
       val passwordEditText = findViewById<EditText>(R.id.password_edit_text)
       val loginButton = findViewById<Button>(R.id.login_button)

       loginButton.setOnClickListener {
           iniciarSesion()
       }
   }


   fun iniciarSesion() {
       val thread = Thread {
           try {
               val charset = "UTF-8"
               val ip = "192.168.1.199"
               val edCedula = findViewById<EditText>(R.id.username_edit_text)
               val edContrasena =
findViewById<EditText>(R.id.password_edit_text)

               val cedula = edCedula.text.toString()
               val contrasena = edContrasena.text.toString()

               val query = String.format("?cedula=%s&contrasena=%s",
                   URLEncoder.encode(cedula, charset),
                   URLEncoder.encode(contrasena, charset))

               val context = applicationContext

               val url = URL(String.format("http://%s:3001/%s", ip, query))
               val urlConnection = url.openConnection() as HttpURLConnection
               urlConnection.requestMethod = "GET"
               urlConnection.setRequestProperty("Accept-Charset", charset)
               urlConnection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded;charset=$charset")
               val rd = BufferedReader(InputStreamReader(
                   urlConnection.inputStream))

               val jsonResponse = rd.readLine()

               val jsonValue = JSONObject(jsonResponse)
               val code = jsonValue.getInt("code")

               if (code == 200) {
```

```
                    val data = jsonValue.getJSONObject("data")
                    val nombre = data.getString("nombre")
                    val apellido = data.getString("apellido")
                    val telefono = data.getString("telefono")

                    runOnUiThread {
                        Toast.makeText(context, "Welcome!",
Toast.LENGTH_SHORT).show()
                    }

                    // Passing data to another Activity
                    val i = Intent(this@MainActivity,
LoginSuccessActivity::class.java)
                    i.putExtra("cedula", cedula)
                    i.putExtra("contrasena", contrasena)
                    i.putExtra("nombre", nombre)
                    i.putExtra("apellido", apellido)
                    i.putExtra("telefono", telefono)
                    startActivity(i)
                } else {
                    runOnUiThread {
                        Toast.makeText(context, "Wrong credentials!",
Toast.LENGTH_SHORT).show()
                    }
                }
            } catch (e: Exception) {
                Log.d("Error on sign up", "Ocurrió un error al intentar iniciar
sesión.")
                Log.d("Error on sign up", e.toString())
            }
        }

        if (thread.isAlive) {
            // Ending thread after there was a successful login
            thread.interrupt()
        }

        thread.start()
    }
}
```

Código que contiene lo que hace el programa una vez se ha iniciado sesión:

```
package com.example.clienteandroid

import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class LoginSuccessActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login_success)
        val nombreTextView = findViewById<TextView>(R.id.nombreTextView)
        val nombre = intent.getStringExtra("nombre")
        nombreTextView.text = "Bienvenido, $nombre!" //
    }
}
```

XML de activity_main (interfaz gráfica del inicio de sesión):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/linearLayout"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="match_parent">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:layout_width="384dp"
            android:layout_height="wrap_content"
            android:text="Usuario"
            android:textSize="30sp" />

        <EditText
            android:id="@+id/username_edit_text"
            android:layout_width="384dp"
            android:layout_height="wrap_content"
            android:hint="usuario"
            android:padding="20sp" />

    </LinearLayout>
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
    <TextView
        android:layout_width="392dp"
        android:layout_height="wrap_content"
        android:text="Contraseña"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/password_edit_text"
        android:layout_width="388dp"
        android:layout_height="wrap_content"
        android:hint="contraseña"
        android:padding="20sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">


    </LinearLayout>

    <Button
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Entrar"
        android:onClick="iniciarSesion"/>

</LinearLayout>
```

Código que contiene el XML de login_activity_success.xml (Interfaz gráfica después de iniciar sesión):

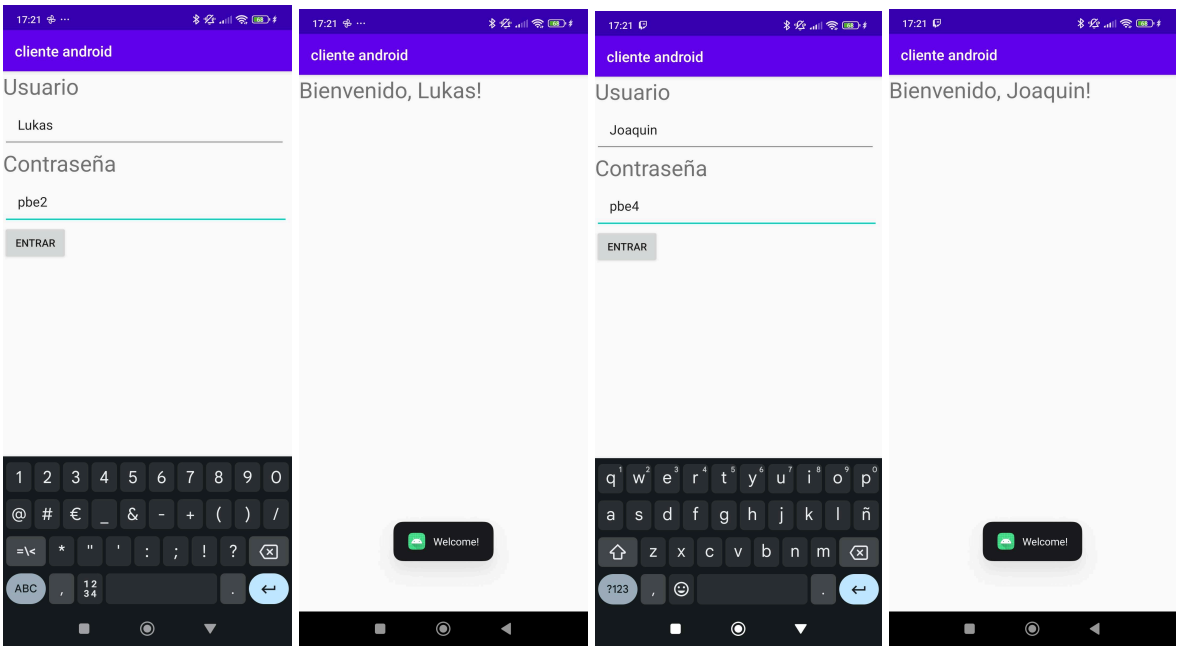
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">



    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

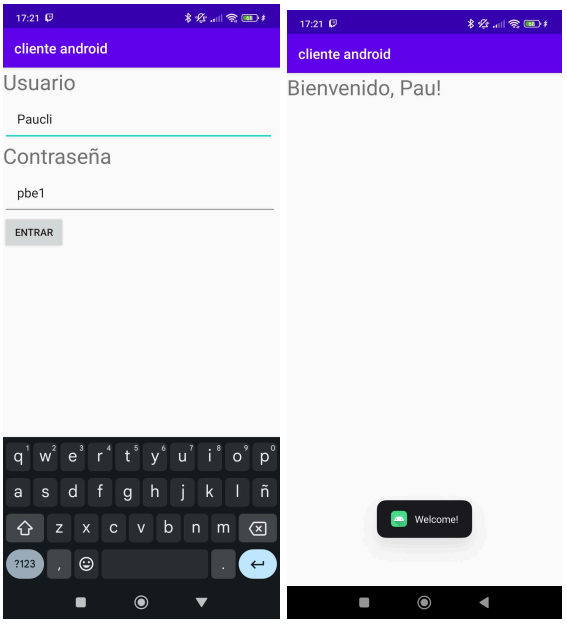
```
        android:orientation="vertical">

        <TextView
            android:id="@+id/nombreTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Inicio de sesion correcto"
            android:textSize="30sp" />

    </LinearLayout>

</LinearLayout>
```

6. SYSTEM CHARACTERIZATION

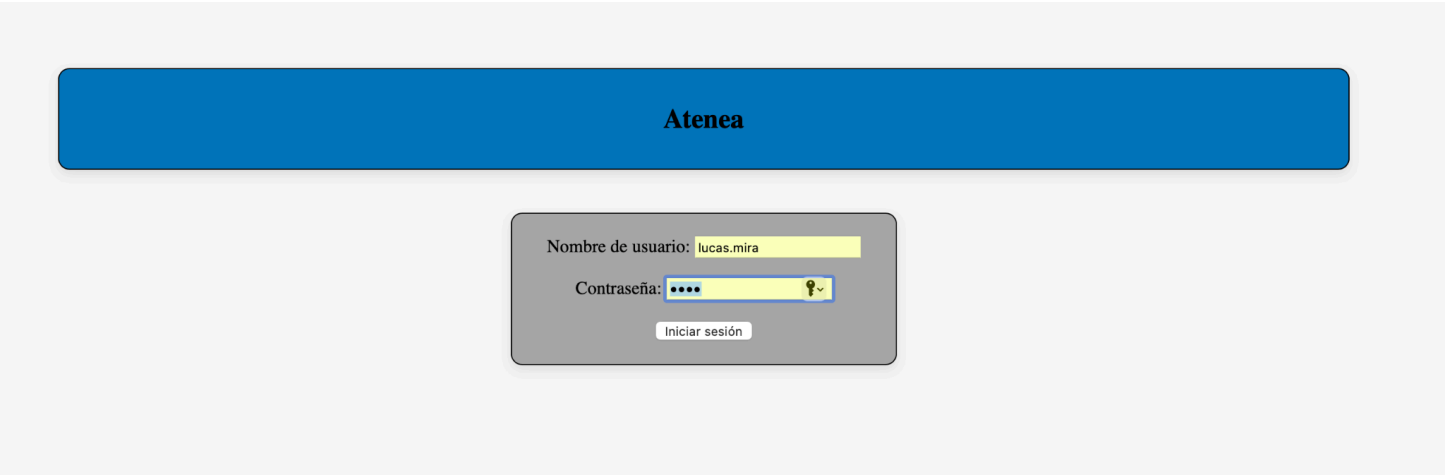
- Measurement / demonstration set-up
- Measurement graphs and results. Final performance.
- Short discussion

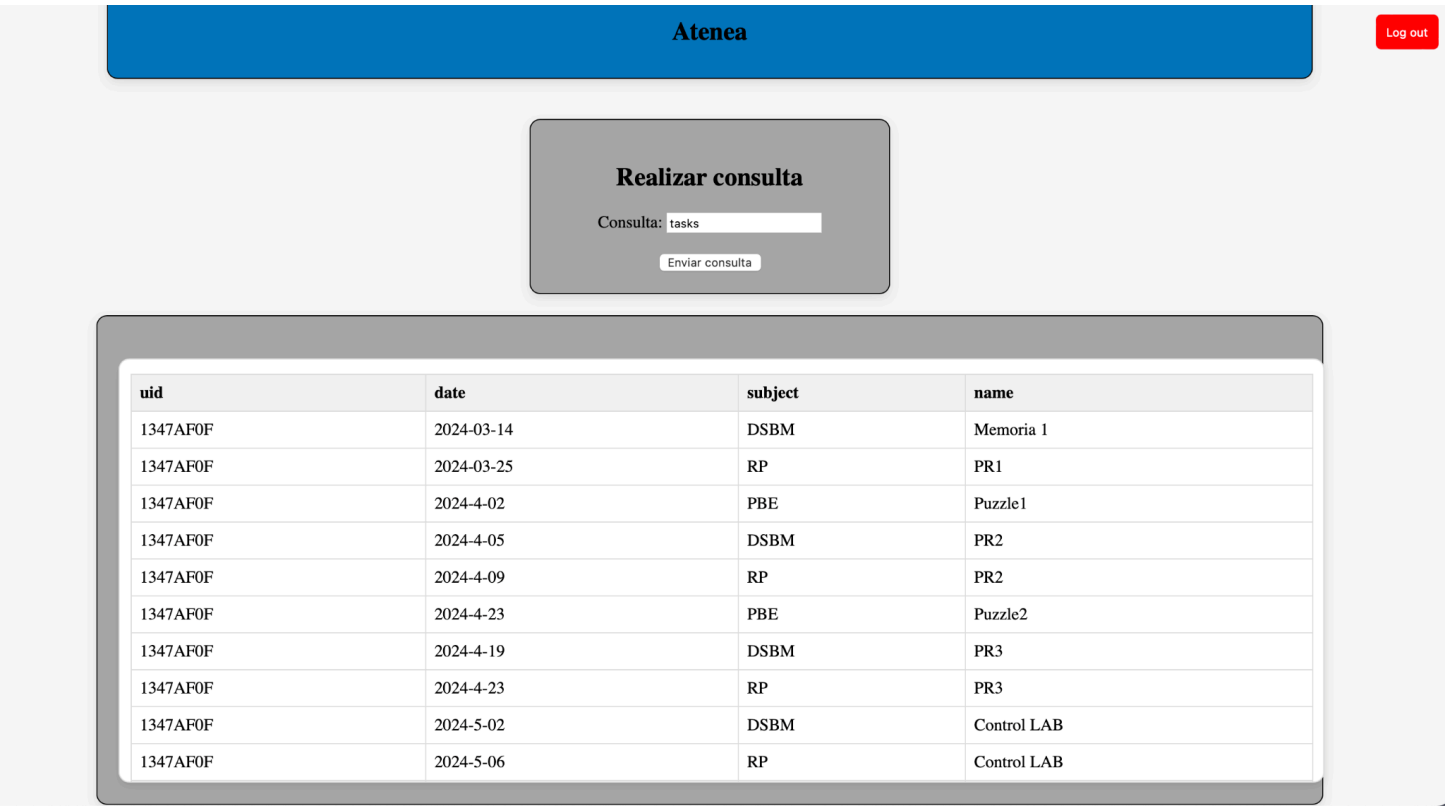
Capturas funcionamiento cliente Android:





Capturas funcionamiento cliente WEB:





uid	date	subject	name
1347AF0F	2024-03-14	DSBM	Memoria 1
1347AF0F	2024-03-25	RP	PR1
1347AF0F	2024-4-02	PBE	Puzzle1
1347AF0F	2024-4-05	DSBM	PR2
1347AF0F	2024-4-09	RP	PR2
1347AF0F	2024-4-23	PBE	Puzzle2
1347AF0F	2024-4-19	DSBM	PR3
1347AF0F	2024-4-23	RP	PR3
1347AF0F	2024-5-02	DSBM	Control LAB
1347AF0F	2024-5-06	RP	Control LAB

7. COSTS

- Components list with approximate costs (prototype)
- Design and prototyping costs separate by main tasks (hours·person x cost)

Taking into account that the minimum wage for a software engineer in Spain is 18 euros per hour, and that in this project we have worked with 5 people, each dedicating a minimum of 3 hours per week for 13 weeks, the base budget would be 3510 euros. This calculation does not include the hours outside of regular academic/work hours that each of our engineers dedicated to finishing, improving, and fixing various parts of the project, as well as the more individualized tasks.

Productos	Coste (€)
RaspberryPi	90
Elechouse	7.4
ITEAD	10.3
RC522	1.5
LCD	9
Otros periféricos	40
Total	158.2

8. CONCLUSIONS

The project has been a testament to the hard work and challenges that a programmer faces in developing a large-scale application. From acquiring hardware to implementing specific functionalities, a series of technical obstacles have arisen that have required solid skills and the ability to solve problems effectively.

In this context, the importance of continuous self-learning for programmers is highlighted. In a field that is constantly evolving, staying updated with the latest tools, technologies, and best practices is crucial. The ability to learn new skills and adapt quickly to changes is essential for success in projects of this nature.

Additionally, teamwork has been fundamental. Effective collaboration among multiple team members, each contributing their individual skills and knowledge, has been crucial to overcoming challenges and advancing towards project goals. Open communication, task coordination, and joint problem-solving have been key elements for progress.

Furthermore, the project has highlighted the resilience and persistence required to face setbacks and obstacles. Staying focused on goals despite technical challenges, implementation setbacks, and resource limitations has been essential for achieving success.

In summary, the project has been an enriching experience that has underscored the importance of continuous learning, effective collaboration, and resilience for programmers working on complex projects. These skills are essential for addressing the challenges of software development and achieving successful outcomes in a dynamic and challenging environment.

9. REFLECTION DOCUMENT

- Things that could have been done better by the organizers/lecturers
- Things that could have been done better by the team
- Performance as a work team
- Self assessment (remember the team mark goal in the team constitution document)

The project provided valuable insights into areas where improvement could have been made both by the organizers/lecturers and the team. Clearer communication and structured support from the organizers/lecturers would have helped the team better understand expectations and navigate project challenges. On the other hand, the team could have improved planning and proactive problem-solving to ensure smoother progress and mitigate setbacks.

Despite these areas for improvement, the team demonstrated strong collaboration and adaptability throughout the project. Conducting a comprehensive self assessment against the initial team mark goal outlined in the team constitution document would provide valuable insights for future projects.