Vanessa Sellart - Vicenç Parera - Sílvia Ruano - Óscar Parada - Iván Cedó

Memòria Final

1. Objectiu

- **Millora del CDR**
  Incorporar les millores suggerides en el feedback rebut, ajustant i optimitzant el document per complir amb els estàndards i requeriments establerts.
- **Desenvolupar l'Aplicació Mòbil**
  Crear una aplicació funcional i eficient que compleixi amb els objectius definits.
- **Desenvolupar la Web**
  Implementar també una plataforma web que compleixi les mateixes funcions.

2. Millora del CDR

Amb els suggeriments i el feedback rebut en l'entrega anterior del CDR, hem fet diverses millores en el servidor i el client. Les modificacions principals inclouen la millora del client amb un handler en el fetchdata per gestionar de manera més eficient les operacions asíncrones, el formateig de les dades per simplificar la lògica del servidor, la implementació de GLib::Idle.add per executar el lector en un fil auxiliar, i l'addició d'un mètode per ordenar les dades. Aquestes modificacions permeten una aplicació més robusta, escalable i fàcil de mantenir.

2.1. Servidor

Amb les modificacions implementades en el servidor, ara el codi és més genèric i adaptable a diferents taules de la base de dades, eliminant les condicions específiques que abans depenien de la taula amb la que treballavem. Hem pogut fer aquest canvi perquè hem reestructurat el tractament de les dades a la part del client, que ha permès una gestió més flexible de les consultes a la base de dades.

Una de les principals novetats és el mètode order, que ordena les dades de la taula timetables. Aquest mètode organitza les dades en funció de la prioritat dels dies de la setmana (basant-se en el dia actual) i l'hora, assegurant que les activitats es mostrin en un ordre lògic i útil per a l'usuari. En aquest mètode, els elements es comparen tenint en compte la distància relativa respecte al dia i hora actuals. Així, les dades es classifiquen primer per dia de la setmana i després per hora, mostrant preferència per les activitats més properes en el temps.

Les consultes a la base de dades MySQL ha quedat més senzilla i adaptada a qualsevol taula. Ara, en lloc de generar condicions específiques segons la taula, el codi utilitza una estructura més flexible que permet consultar qualsevol taula amb un conjunt de filtres. Només s'aplica el mètode order quan és necessari, és a dir, només per a les dades de la taula timetables.

Per la resta, el servidor funciona de manera similar a la versió anterior: el tractament de sessions, la validació de l'usuari i la gestió de les consultes es mantenen igual. Les dades es continuen processant de la mateixa manera.

```JavaScript
const http = require('http');
const mysql = require('mysql2');
const url = require('url'); //necessari per tractar els paràmetres de consulta

// Configuración de la conexión con la base de datos
const connection = mysql.createConnection({
  host: '192.168.1.46', // Direcció IP
  user: 'root',         // Usuari de MySQL
  password: '1234',     // Contrasenya de MySQL
```

```
  database: 'pbe',      // nom de la base de dades
  port: 3306            // Port de MySQL
});

// Verificar la conexión a la base de datos
connection.connect((err) => {
  if (err) {
    console.error('Error de conexión a la base de datos: ' + err.stack);
    return;
  }
  console.log('Conectado a la base de datos con el ID ' + connection.threadId);
});

//Gestionar sessions d'usuaris
const userSessions = {};
//temps d'inactivitat en mil·lisegons (2 minut)
const SESSION_TIMEOUT = 2*60*1000;

// Crear el servidor HTTP
const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'application/json');
  res.setHeader('Access-control-Allow-Origin', '*');

const parsedUrl = url.parse(req.url, true); //parsear la URL

if(req.method === 'GET' && parsedUrl.pathname === '/authenticate'){
        const {uid} = parsedUrl.query; //parsejar el cos de la solicitud per obtenir uid

        if(!uid){
            res.statusCode = 400;
            res.end(JSON.stringify({error: 'UID es necessari'}));
            return;
        }
        //consultar students per buscar uid
        connection.query( 'SELECT name FROM students WHERE student_id = ?',
            [uid], // Parametre: UID rebut
            (err, results) => {
                if(err){ //control d'errors de consulta
                    console.error('Error al realitzar consulta:', err);
                    res.statusCode = 500; //codi d'error del servidor
                    res.end(JSON.stringify({error:'Error al realitzar la consulta'})
                      );
                }else if(results.length===0){
                    //si no es troba a la taula
                    res.statusCode = 401; // Codi "no autoritzat"
                    res.end(JSON.stringify({error: 'UID no vàlid'}));
                }else{
                    //si uid vàlid, tornar el nom de l'estudiant
                    userSessions[uid]={
                        name: results[0].name,
                        lastActivity: Date.now() //guarda hora
                    };
                    res.statusCode = 200; // codi d'exit
                    res.end(JSON.stringify({name: results[0].name}));
                }
            }
        );
        //endpoint per realitzar consultes taules concretes
    }else if(req.method === 'GET' && parsedUrl.pathname === '/query'){
```

```
        const {table, limit, ...filters} = parsedUrl.query; //parsejar cos de la solicitud per
obtenir la taula

        //uid de la sessio
        const uid = Object.keys(userSessions).find((uid) =>
                    userSessions[uid]);

        if(!uid){
            res.statusCode = 600;
            res.end(JSON.stringify({error: 'Sessió no iniciada o caducada'}));
            return;
        }

        userSessions[uid].lastActivity=Date.now(); //actualitzar activitat

        //generar la consulta MySQL la taula solicitada filtrant per uid

        // --------- CONSTRAINTS ---------
        let query = `SELECT * FROM ${table} WHERE student_id = ?`;
        const params = [uid];

        if(Object.keys(filters).length > 0){
            const conditions = [];
            const opMap = {
                gte: '>=',
                gt: '>',
                lte: '<=',
                lt: '<',
                eq: '='
            };

            for( const [key, value] of Object.entries(filters)){
                if(key.includes('[') && key.includes(']')){
                    //treure el camp i l'operador
                    const field = key.split('[')[0];
                    const modifier = key.match(/\[(.+)\]/)[1];

                    const op = opMap[modifier];
                    if(op){
                        conditions.push(`${field} ${op} ?`);
                        params.push(value);
                    }else{
                        console.warn(`Modificador desconocido:${modifier}`);
                    }
                } else if (value.toLowerCase()==='now'){
                    if(table === 'timetables'){
                        if(key === 'day'){
                            const currentDay = new Date().toLocaleDateString('en-US', {weekday:
'short'});

                            conditions.push(`${key} = ?`);
                            params.push(currentDay);
                        } else if (key === 'hour'){
                       // Convertir "now" a hora actual, HH:00:00
                            const currentTime = new Date();
                            currentTime.setMinutes(0); //minuts a 0
                            currentTime.setSeconds(0); //segons a 0
                            currentTime.setMilliseconds(0); // ms a 0
                                    // Format HH:00:00
                          const roundedHour=currentTime.toTimeString().split(' ')[0];
                            conditions.push(`${key} = ?`);
```

```
                            params.push(roundedHour);
                        }
                    } else if (table === 'tasks'){
                        const currentDate = new Date().toISOString().split('T')[0];
                        conditions.push(`${key} = ?`);
                        params.push(currentDate);

                    } else {
                     console.warn(`Campo "now" no soportado para la tabla ${table}`);
                    }
                }else{
                    conditions.push(`${key} = ?`);
                    params.push(value);
                }
            }
            query += ' AND '+conditions.join(' AND ');
        }

        if(limit){
            query += ' LIMIT ?';
            params.push(parseInt(limit, 10));
        }

        //consulta a la taula
        connection.query(query,params, (err, results) => {
            if (err){ //control d'errors de consulta
                console.error('Error al realitzar la consulta: ', err);
                res.statusCode = 500; // error del servidor
                res.end(JSON.stringify({error: 'Error al realitzar la consulta'}));
            }else{
                //tornar les dades de la consulta
                res.statusCode = 200; //codigo de exito
                    if(table === 'timetables'){
                        results = order(results);
                        console.log(results);
                    }
                res.end(JSON.stringify(results));
            }
        });
    }else if(req.method === 'GET' && parsedUrl.pathname === '/logout'){
        const uid = Object.keys(userSessions).find((uid) => userSessions[uid]);

        if(!uid){
            res.statusCode = 400;
            res.end(JSON.stringify({ error: 'Sessió no iniciada' }));
            return;
        }
        //esborrar la sessio de l'usuari
            delete userSessions[uid];
            res.statusCode = 200;
            res.end(JSON.stringify({ message: 'Sessió tancada correctament' }));

        // Ruta para obtener los estudiantes
    } else {
        res.statusCode = 404;
        res.end(JSON.stringify({ error: 'Ruta no encontrada' }));
    }
    });

//Timer per gestionar les sessions
```

```
setInterval(() => {
    const now = Date.now();

    for(const uid in userSessions){
        if(now - userSessions[uid].lastActivity > SESSION_TIMEOUT){
            console.log(`Sessió de l'usuari ${uid} ha caducat`);
            delete userSessions[uid];
        }
    }
}, 60 * 1000);

// Escuchar en el puerto 3000
const PORT = 3000;
server.listen(PORT, '192.168.1.46', () => {
  console.log(`Servidor escuchando en http://192.168.1.46:${PORT}`);
});


function order(data) {
    const now = new Date();
    const currentDay = now.toLocaleDateString('en-US', { weekday: 'short' });
    const currentHour = now.getHours();

    const weekDays = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'];
    const todayIndex = weekDays.indexOf(currentDay);

    // Funció per obtenir només l'hora (`hh`) com a número a partir del string `hh:mm:ss`
    function getHourFromTime(hourString) {
        return parseInt(hourString.split(':')[0], 10);
    }

    // Primer, ordenem per dia i hora
    const sortedData = data.sort((a, b) => {
        const aDayIndex = weekDays.indexOf(a.day);
        const bDayIndex = weekDays.indexOf(b.day);

        // Calculem la prioritat cíclica del dia (a partir d'avui)
        const aDayPriority = (aDayIndex - todayIndex + weekDays.length) % weekDays.length;
        const bDayPriority = (bDayIndex - todayIndex + weekDays.length) % weekDays.length;

        // Si estem en dies diferents, ordenem pel dia més proper
        if (aDayPriority !== bDayPriority) {
            return aDayPriority - bDayPriority;
        }

        // Si estem en el mateix dia (avui), ordenem per hora
        const aHour = getHourFromTime(a.hour);
        const bHour = getHourFromTime(b.hour);

        return aHour - bHour;
    });

    // Després, movem els esdeveniments passats del dia actual al final
    const todayEvents = [];
    const otherEvents = [];
    const pastEvents = [];

    for (const event of sortedData) {
        if (weekDays.indexOf(event.day) === todayIndex) {
            const eventHour = getHourFromTime(event.hour);
```

```
            if (eventHour < currentHour) {
                // Afegim els esdeveniments passats al final del dia actual
                pastEvents.push(event);
                console.log(eventHour, currentHour);
            } else {
                // Afegim els esdeveniments futurs o actuals del dia actual
                todayEvents.push(event);
            }
        } else {
            // Afegim els esdeveniments d'altres dies
            otherEvents.push(event);
        }
    }

    return [...todayEvents, ...otherEvents, ...pastEvents];
}
```

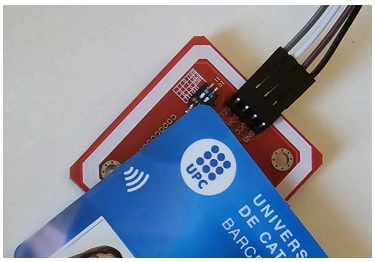
2.2. Client

Una de les principals modificacions ha sigut el mètode process_data. Abans, no teniem aquest mètode fent que haguessim d'especificar les columnes que voliem en el servidor. Ara, el mètode té un paràmetre, que conté les columnes específiques que volem mostrar en funció de la taula consultada. D'aquesta manera, podem personalitzar quines dades es mostren a l'usuari segons el tipus de taula que estem consultant (per exemple, en una taula de "timetables" només es mostren les columnes day, hour, subject, room). A més, s'ha optimitzat la forma en què es construeixen les dades per mostrar-les després a la interfície. S'agafen les dades i es converteixen en un diccionari on les claus son els noms de les columnes i els valors son strings amb totes les files d'aquella columna, en el cas de la columna date, es formatgeixen de manera correcta (usant Date.parse per convertir la data en un format estàndard). Això facilita la presentació de les dades de manera més ordenada i comprensible per l'usuari.

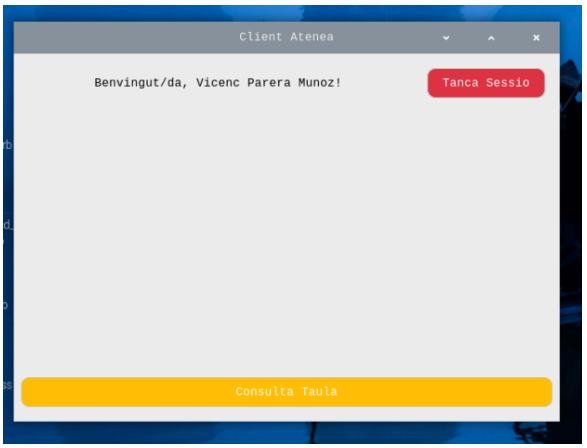
Una altra millora important ha estat la gestió del lector RFID, que abans podia bloquejar la interfície mentre es realitzava la lectura. Ara s'ha utilitzat un fil auxiliar per llegir el codi RFID de manera asincrònica, de manera que la interfície no es congela durant la lectura. Quan s'accedeix a la pantalla d'autenticació, el lector RFID es crida en un fil separat que utilitza el mètode read_uid per obtenir el UID de la targeta. En lloc de reimplementar el codi del lector RFID al client, hem aprofitat el codi del mòdul puzzle1 que ja implementava la lectura del lector RFID.

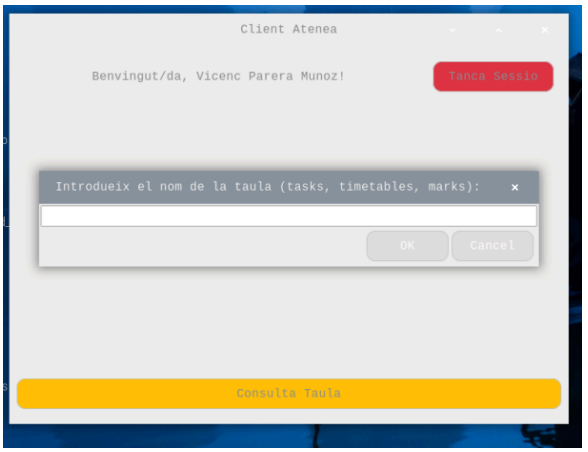
Pel que fa al mètode fetch_data, en lloc de fer servir directament les respostes dins del mateix mètode, hem afegit "handlers" per gestionar de manera diferent els casos d'èxit, error i excepció. Els paràmetres on_success, on_error i on_exception són opcions que permeten passar funcions específiques per tractar cadascun d'aquests casos. Quan es fa una petició HTTP a la URL definida, el mètode processa la resposta i, en cas d'èxit, crida al handler on_success amb les dades recuperades. Si hi ha un error, es crida al handler on_error amb el missatge d'error corresponent, i si es produeix una excepció (per exemple, un error de connexió), es crida al handler on_exception. Aquest enfocament facilita la gestió de respostes i errors, ja que permet gestionar la lògica de cada cas de manera independent i més clara.

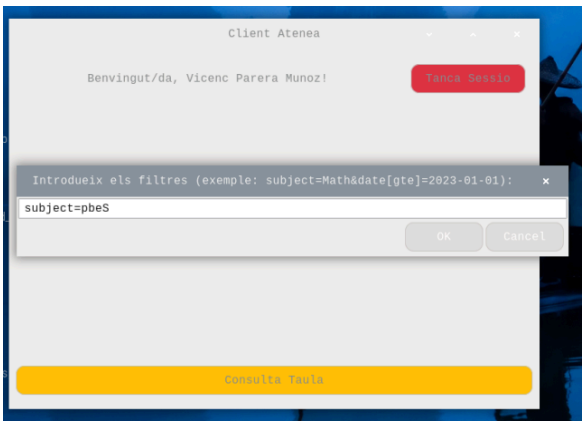

Client Atenea
Acerque su tarjeta NFC para autenticarse
Leer NFC

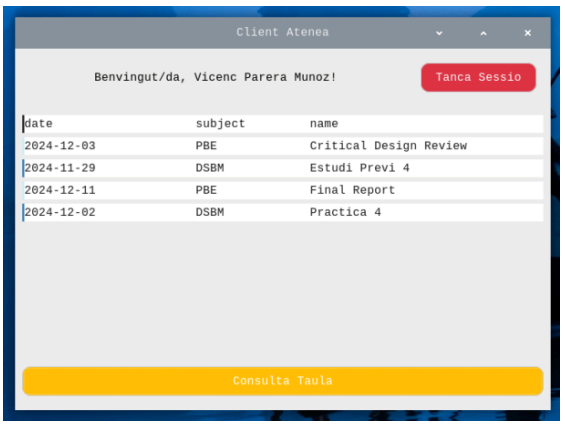



Welcome
Vicenc Parera Munoz!


Client Atenea
Benvingut/da, Vicenc Parera Munoz!
Tanca Sessio
Consulta Taula


Client Atenea
Benvingut/da, Vicenc Parera Munoz!
Tanca Sessio
Introdueix el nom de la taula (tasks, timetables, marks):
OK
Cancel
Consulta Taula


Client Atenea
Benvingut/da, Vicenc Parera Munoz!
Tanca Sessio
Introdueix els filtres (exemple: subject=Math&date[gte]=2023-01-01):
subject=pbeS
OK
Cancel
Consulta Taula


Client Atenea
Benvingut/da, Vicenc Parera Munoz!
Tanca Sessio
date subject name
2024-12-03 PBE Critical Design Review
2024-11-29 DSBM Estudi Previ 4
2024-12-11 PBE Final Report
2024-12-02 DSBM Practica 4
Consulta Taula

C/C++

```
require 'gtk3'
require 'json'
require 'net/http'
require 'uri'
require 'i2c/drivers/lcd'
require_relative 'puzzle1'

BASE_URL = 'http://192.168.1.46:3000'

def fetch_data(path, params = {}, on_success: nil, on_error: nil, on_exception: nil)
  uri = URI("#{BASE_URL}#{path}")
  uri.query = URI.encode_www_form(params) unless params.empty?

  begin
    response = Net::HTTP.get_response(uri)
    if response.code.to_i == 200
      data = JSON.parse(response.body)
      on_success&.call(data) if on_success
    else
      if response.code.to_i == 600
        error_message = "Error: #{response.code} - #{response.body}"
      else
        error_message = "Error del servidor: #{response.code} - #{response.body}"
      end
      on_error&.call(error_message) if on_error
    end
  rescue StandardError => e
    exception_message = "Error al conectar con el servidor: #{e.message}"
    on_exception&.call(exception_message) if on_exception
  end
end

def process_data(data, selected_columns)
  resultado = {}
  selected_columns.each{ |columna| resultado[columna] = [] }
  data.each do |fila|
    selected_columns.each do |columna|
      if columna == 'date'
        date_ok = Date.parse(fila[columna]).strftime('%Y-%m-%d')
        resultado[columna] << date_ok
      else
        resultado[columna] << fila[columna] if fila.key?(columna)
      end
    end
  end
  resultado
end

class SimpleClientApp
  def initialize
    @window = Gtk::Window.new('Client Atenea')
    @window.set_size_request(800, 600)
    @window.signal_connect('destroy') { Gtk.main_quit }

    @vbox = Gtk::Box.new(:vertical, 10)
    @vbox.margin = 10
    @window.add(@vbox)

    # Crear la instÃ ncia de la pantalla LCD dins de la classe
    @lcd = I2C::Drivers::LCD::Display.new('/dev/i2c-1', 0x27, rows=4, cols=20)
```

```
    # Carregar els estils des d'un fitxer CSS extern
    load_css_from_file('styles.css')

    # Mostrar la pantalla d'autenticaciÃ³
    show_authentication_screen

    @window.show_all
  end

   def show_authentication_screen
  @vbox.children.each(&:destroy) # Limpia la interfaz

  lector = Rfid.new

  # Etiqueta de bienvenida inicial
  label = Gtk::Label.new('Acerque su tarjeta NFC para autenticarse')
  @vbox.pack_start(label, expand: false, fill: false, padding: 10)

  Thread.new do
    uid = lector.read_uid
    GLib::Idle.add do
      if uid
        authenticate_user(uid)
        false
      else
        show_error('No se pudo leer el UID. Intente nuevamente.')
      end
    end
  end

  @window.show_all
end

  def show_main_interface(name)
    @vbox.children.each(&:destroy)

    # Crear un contenidor horitzontal per al text de benvinguda i el botÃ³ de logout
    header_box = Gtk::Box.new(:horizontal, 5)

    # Mostrar un missatge de benvinguda a la pantalla LCD
    @lcd.clear
    @lcd.text('    Welcome', 1)
    @lcd.text("#{name}!", 2)

    welcome_label = Gtk::Label.new("Benvingut/da, #{name}!")
    header_box.pack_start(welcome_label, expand: true, fill: true, padding: 10)

    logout_button = Gtk::Button.new(label: 'Tanca Sessio')
    logout_button.set_name('logout_button')
    logout_button.signal_connect('clicked') { logout }
    header_box.pack_start(logout_button, expand: false, fill: false, padding: 10)

    @vbox.pack_start(header_box, expand: false, fill: false, padding: 10)

    @text_views_box = Gtk::Box.new(:vertical, 2)
    scrolled_window = Gtk::ScrolledWindow.new
    scrolled_window.add(@text_views_box)
    scrolled_window.set_policy(:automatic, :automatic)
    @vbox.pack_start(scrolled_window, expand: true, fill: true, padding: 10)
```

```
    query_button = Gtk::Button.new(label: 'Consulta Taula')
    query_button.set_name('query_button')
    query_button.signal_connect('clicked') { query_table }
    @vbox.pack_start(query_button, expand: false, fill: false, padding: 10)

    @window.show_all
  end

  def authenticate_user(uid)
    name = nil
    error = nil
    fetch_data('/authenticate', { uid: uid },
               on_success: -> (data) {name = data['name']},
               on_error: -> (error_message) {error = error_message},
               on_exception: -> (exception_message) {})
    if error != nil
      show_error_dialog(error)
    else
      if name != nil
        show_main_interface(name)
      else
        show_error_dialog('Error d\'autenticacio. Torna-ho a intentar.')
      end
    end
  end

  def query_table
    table = prompt('Introdueix el nom de la taula (tasks, timetables, marks):')
    return unless table

    filter_string = prompt('Introdueix els filtres (exemple:
subject=Math&date[gte]=2023-01-01):')
    filters = parse_filters(filter_string)
    result = nil
    error = nil
    params = { table: table }.merge(filters)
    fetch_data('/query', params,
               on_success: -> (data) {result = data},
               on_error: -> (error_message) {error = error_message},
               on_exception: -> (exception_message) {error = exception_message})
    puts error
    if error != nil
      show_error_dialog(error)

    else
      case table
      when 'timetables'    #Aqui les majuscules si que son importants
        data = process_data(result, ['day', 'hour', 'Subject', 'Room'])

      when 'tasks'
        data = process_data(result, ['date', 'subject', 'name'])

      when 'marks'
        data = process_data(result, ['Subject', 'Name', 'Marks'])

      else
        show_error_dialog(error)
      end
```

```
      if result.is_a?(Array) && !result.empty?
        populate_text_views(data)
      elsif result.is_a?(Array) && result.empty?
        show_error_dialog("No hi ha dades disponibles per a la taula #{table}")
      else
        show_error_dialog('Error: resposta inesperada')
      end
    end
  end

  def populate_text_views(data)
    @text_views_box.children.each(&:destroy)

    #Obtenir les columnes (les claus)
    columns = data.keys

    #Calcular amplada mxima de cada columna
    column_widths = columns.map do |col|
      #determinar la longitud maxima entre el nom de la columna i els seus valors
      [col.to_s.length, *data[col].map(&:to_s).map(&:length)].max
    end

    #afegir capÃ§aleres am el primer Text View
    headers = columns.each_with_index.map{ |col, i| col.to_s.ljust(column_widths[i])
}.join("\t\t")
    add_text_view(headers, 'header', 0)

    #determinar el nombre de files
    num_rows = data.values.first.size

    #iterar per cada fila segons l'index
    (0...num_rows).each do |row_index|
      #construir la fila accedint als valors per la seva posicio
      row_data = columns.each_with_index.map do |col, i|
        data[col][row_index].to_s.ljust(column_widths[i])
      end.join("\t\t")

      #alternem l'estil de cada fila
      style_class = row_index.even? ? 'row_even' : 'row_odd'
      add_text_view(row_data, style_class, row_index+1)
    end

    @window.show_all
  end

  def add_text_view(text, style_class, row_index)
    buffer = Gtk::TextBuffer.new
    buffer.text = text

    text_view = Gtk::TextView.new
    text_view.buffer = buffer
    text_view.editable = false
    text_view.cursor_visible = false
    text_view.set_name(style_class)
    @text_views_box.pack_start(text_view, expand: false, fill: false, padding: 2)
  end

  def parse_filters(filter_string)
    return {} if filter_string.nil? || filter_string.empty?
```

```
    filters = {}
    filter_string.split('&').each do |filter|
      key, value = filter.split('=')
      filters[key.strip] = value.strip if key && value
    end
    filters
  end

  def show_error_dialog(message)
    dialog = Gtk::MessageDialog.new(
      parent: @window,
      flags: :destroy_with_parent,
      type: :error,
      buttons: :close,
      message: message  )
    dialog.run
    dialog.destroy
  end

  def logout
    show_authentication_screen
  end

  def prompt(message)
    dialog = Gtk::Dialog.new(
      title: message,
      parent: @window,
      flags: :destroy_with_parent,
      buttons: [[Gtk::Stock::OK, :ok], [Gtk::Stock::CANCEL, :cancel]]
    )
    entry = Gtk::Entry.new
    dialog.child.add(entry)
    dialog.child.show_all

    response = dialog.run
    input = entry.text.strip
    dialog.destroy
    response == :ok && !input.empty? ? input : nil
  end

  def load_css_from_file(file_path)
    return unless File.exist?(file_path)
    provider = Gtk::CssProvider.new
    provider.load_from_path(file_path)
    Gtk::StyleContext.add_provider_for_screen(
      Gdk::Screen.default,
      provider,
      Gtk::StyleProvider::PRIORITY_USER
    )
  end
end

Gtk.init
SimpleClientApp.new
Gtk.main
```

styles.css

El fitxer de format segueix sent el mateix que ja teníem sense modificacions.

```C/C++
/* Fonts generals */
* {
  font-family: monospace;
}

/* Botons */
button {
  border-radius: 12px;
  padding: 10px 20px;
  color: white;
}

button:hover {
  opacity: 0.8;
}

button#auth_button {
  background-color: #28a745; /* Verd */
}

button#logout_button {
  background-color: #dc3545; /* Vermell */
}

button#query_button {
  background-color: #ffc107; /* Groc */
}

/* TextViews */
textview {
  border: 1px solid #ccc;
  padding: 2px;
  background-color: #f9f9f9;
}

textview#header {
  background-color: #2a2a2a; /* Fosc */
  color: white;
  font-weight: bold;
}

textview#row_even {
  background-color: #d3ecf8; /* Blau cel */
}

textview#row_odd {
  background-color: #4682b4; /* Blau fosc */
  color: white;
}
```

3. APP

Hem escollit Android 13 com a plataforma principal per desenvolupar la nostra aplicació per diversos motius tècnics i pràctics.

1. **Compatibilitat i estabilitat**: Android 13 ofereix un entorn estable i àmpliament compatible amb els dispositius mòbils que utilitzem. Els nostres dispositius són principalment Android 13 i 14, i Android 13 ens garanteix una compatibilitat òptima amb les funcionalitats clau necessàries per al desenvolupament i les proves.
2. **Accés a característiques avançades**: Aquesta versió d'Android inclou diverses millores pel que fa a la privadesa, la seguretat i el rendiment, així com noves APIs que ens permeten implementar funcionalitats modernes i atractives per a l'usuari.
3. **Eficiència en les proves**: Centrar-nos en Android 13 facilita el procés de proves, ja que sabem que el sistema operatiu està disponible a tots els nostres dispositius, i alhora podem validar que l'aplicació també funcioni correctament en dispositius amb Android 14.

Aquest enfocament ens permet oferir una experiència d'usuari coherent i assegurar que la nostra aplicació estigui optimitzada per a l'ecosistema dels dispositius que utilitzem i del mercat actual.

Per fer l'aplicació hem usat Android studio, amb java, i hem fet els codis següents:

MainActivity.java

```
Java
package lab.four.test;

import static androidx.core.content.ContentProviderCompat.requireContext;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.content.Intent;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    // Variables per als camps de text i botó
    private EditText editTextRFID, editTextUsername, editTextServerIP;
    private Button buttonLogin;
    private Connexio c;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Activa el mode Edge-to-Edge per millorar la integració amb la barra de sistema
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);

        // Configura el padding automàtic basant-se en els insets del sistema (barra de
navegació i d'estat)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        // Assigna els elements del layout a les variables de codi
        editTextRFID = findViewById(R.id.editTextTextPassword);
        editTextUsername = findViewById(R.id.editTextTextPassword2);
        editTextServerIP = findViewById(R.id.editTextTextPassword3);
        buttonLogin = findViewById(R.id.button2);

        // Configura el comportament quan es fa clic al botó de login
        buttonLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Recupera els valors introduïts en els camps de text
                String password = editTextRFID.getText().toString();
                String username = editTextUsername.getText().toString();
                String ruta = editTextServerIP.getText().toString();

                // Inicia la connexió amb el servidor
                c = new Connexio(ruta);

                // Valida les credencials introduïdes
                validateInputs(username, password, ruta,
                        isValid -> {
                            // Executa el codi en el fil principal per interactuar amb la UI
                            runOnUiThread(() -> {
                                if (isValid) {
                                    // Si la validació és correcta, mostra un missatge d'èxit
                                    Toast.makeText(MainActivity.this, "Login successful",
Toast.LENGTH_SHORT).show();

                                    // Configura la ruta global
                                    RutaManager.getInstance().setRuta(ruta);

                                    // Inicia la nova activitat (taula) després del login
                                    Intent intent = new Intent(MainActivity.this, taula.class);
                                    startActivity(intent);

                                    // Tanca l'activitat actual
                                    finish();
                                } else {
```

```
                                    // Mostra un missatge d'error si la validació falla
                                    Toast.makeText(MainActivity.this, "Incorrect username or
password", Toast.LENGTH_SHORT).show();
                                }
                            });
                        }
                );
            }
        });
    }

    // Retorna l'objecte de connexió actual
    public Connexio getConnexio() {
        return c;
    }

    // Funció per validar les credencials de l'usuari
    private void validateInputs(String username, String uid, String url, Callback<Boolean>
resultCallback) {
        // Crea una nova connexió amb el servidor
        Connexio connexio = new Connexio(url);

        // Autentica l'usuari amb el servidor
        connexio.authenticateUser(uid,
                nom -> {
                    // Comprova si el nom retornat coincideix amb l'usuari introduït
                    boolean isValid = nom.trim().equals(username.trim());
                    System.out.println(isValid);
                    resultCallback.call(isValid);
                },
                // Si hi ha un error, informa que la validació ha fallat
                error -> resultCallback.call(false)
        );
    }

    // Interfície funcional per passar callbacks
    @FunctionalInterface
    public interface Callback<T> {
        void call(T result);
    }
}
```

connexio.java

Java

```
package lab.four.test;

import org.json.JSONArray;
import org.json.JSONException;
```

```
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.*;
import java.util.stream.Collectors;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Connexio {

    // Base URL per connectar amb el servidor
    private static String BASE_URL = ""; // Es configura quan es crea una instància de Connexio

    private static String nom_text; // Emmagatzema el nom autenticat d'un usuari

    // Constructor que configura la URL base
    public Connexio(String url) {
        BASE_URL = url;
    }

    /**
     * Realitza una petició HTTP GET al servidor.
     *
     * @param path         Ruta al servidor (exemple: "/authenticate")
     * @param params       Paràmetres de la petició (filtratge, identificadors, etc.)
     * @param onSuccess    Callback executat quan la petició és exitosa
     * @param onError      Callback executat quan hi ha un error a la resposta del servidor
     * @param onException  Callback executat quan hi ha una excepció durant la petició
     */
    public static void fetchData(String path, Map<String, String> params, Callback<Object>
onSuccess,
                                 Callback<String> onError, Callback<String> onException) {

        ExecutorService executorService = Executors.newCachedThreadPool(); // Executor per a
tasques asíncrones
        executorService.execute(() -> {
            try {
                // Construcció de la URL amb els paràmetres
                StringBuilder urlBuilder = new StringBuilder(BASE_URL).append(path);
                if (params != null && !params.isEmpty()) {
                    urlBuilder.append("?");
                    for (Map.Entry<String, String> entry : params.entrySet()) {
                        urlBuilder.append(URLEncoder.encode(entry.getKey(), "UTF-8"))
                                .append("=")
                                .append(URLEncoder.encode(entry.getValue(), "UTF-8"))
                                .append("&");
                    }
                    urlBuilder.setLength(urlBuilder.length() - 1); // Elimina el darrer "&"
```

```
                }

                URL url = new URL(urlBuilder.toString());

                // Estableix la connexió HTTP
                HttpURLConnection connection = (HttpURLConnection) url.openConnection();
                connection.setRequestMethod("GET"); // Configura la petició com GET

                int responseCode = connection.getResponseCode(); // Obté el codi de resposta

                // Llegeix la resposta del servidor
                try (BufferedReader reader = new BufferedReader(new InputStreamReader(
                        responseCode == 200 ? connection.getInputStream() :
connection.getErrorStream()))) {

                    String response = reader.lines().collect(Collectors.joining("\n")); //
Llegeix tota la resposta

                    if (responseCode == 200) {
                        // Processa la resposta com JSON (JSONObject o JSONArray)
                        try {
                            Object jsonResponse = response.trim().startsWith("{")
                                    ? new JSONObject(response)
                                    : new JSONArray(response);
                            onSuccess.call(jsonResponse); // Retorna el resultat al callback
                        } catch (JSONException e) {
                            onError.call("Error en el format de la resposta del servidor.");
                        }
                    } else {
                        onError.call("Error: " + responseCode + " - " + response);
                    }
                }
            } catch (Exception e) {
                // Gestiona excepcions durant la connexió
                if (onException != null) {
                    onException.call("Error al connectar amb el servidor: " + e.getMessage());
                }
                e.printStackTrace();
            }
        });
    }

    /**
     * Processa dades en format JSON i les retorna com un mapa de columnes a llistes de valors.
     *
     * @param data            JSONArray amb les dades a processar
     * @param selectedColumns Llista de les columnes que s'han d'incloure
     * @return Mapa amb les columnes seleccionades i els seus valors
     */
    public static Map<String, List<Object>> processData(JSONArray data, List<String>
selectedColumns) {
        Map<String, List<Object>> result = selectedColumns.stream()
```

```
                .collect(Collectors.toMap(column -> column, column -> new ArrayList<>())); //
Inicialitza el mapa

        // Itera sobre cada fila del JSONArray
        for (int i = 0; i < data.length(); i++) {
            try {
                JSONObject row = data.getJSONObject(i); // Obté un objecte JSON
                for (String column : selectedColumns) {
                    if (row.has(column)) { // Comprova si la columna existeix
                        Object value = "date".equals(column) ?
formatDate(row.getString(column)) : row.get(column);
                        result.get(column).add(value); // Afegeix el valor al resultat
                    }
                }
            } catch (JSONException e) {
                throw new RuntimeException("Error al processar dades: " + e.getMessage(), e);
            }
        }
        return result;
    }

    /**
     * Autentica un usuari mitjançant el seu UID.
     *
     * @param uid    Identificador de l'usuari
     * @param nom    Callback per gestionar el nom de l'usuari si l'autenticació és exitosa
     * @param error  Callback per gestionar errors d'autenticació
     */
    public void authenticateUser(String uid, Callback<String> nom, Callback<String> error) {
        fetchData(
                "/authenticate",
                Map.of("uid", uid), // Afegeix el UID com a paràmetre
                data -> {
                    if (data instanceof JSONObject) { // Verifica que la resposta és un
JSONObject
                        JSONObject jsonObject = (JSONObject) data;
                        String name = jsonObject.optString("name", null); // Obté el nom de
l'usuari
                        nom_text = name;
                        if (name != null) {
                            nom.call(name); // Retorna el nom al callback
                        } else {
                            error.call("Error d'autenticació. Torna-ho a intentar.");
                        }
                    } else {
                        error.call("Resposta inesperada del servidor.");
                    }
                },
                error::call,
                exception -> error.call("Error al connectar amb el servidor: " + exception)
        );
    }
```

```
    /**
     * Consulta dades d'una taula especificada.
     *
     * @param nomTaula Nom de la taula a consultar
     * @param taula    Callback per retornar les dades de la taula
     * @param error    Callback per gestionar errors
     */
    public void queryTable(String nomTaula, Callback<Map<String, List<Object>>> taula,
Callback<String> error) {
        Map<String, String> params = new HashMap<>();
        String table = null;

        // Divideix el nom de la taula i els filtres (si n'hi ha)
        if (!nomTaula.contains("?")) {
            table = nomTaula;
        } else {
            String[] parts = nomTaula.split("\\?", 2);
            table = parts[0];
            params.putAll(parseFilters(parts[1])); // Afegeix els filtres al mapa de paràmetres
        }

        if (table == null || table.isEmpty()) {
            error.call("Error: taula desconeguda.");
            return;
        }

        params.put("table", table); // Afegeix la taula als paràmetres

        fetchData(
                "/query",
                params,
                result -> {
                    if (result instanceof JSONArray) {
                        JSONArray jsonArray = (JSONArray) result;
                        Map<String, List<Object>> data;
                        // Selecciona les columnes segons la taula
                        switch (table) {
                            case "timetables":
                                data = processData(jsonArray, List.of("day", "hour", "Subject",
"Room"));
                                break;
                            case "tasks":
                                data = processData(jsonArray, List.of("date", "subject",
"name"));
                                break;
                            case "marks":
                                data = processData(jsonArray, List.of("Subject", "Name",
"Marks"));
                                break;
                            default:
                                error.call("Error: taula desconeguda.");
```

```
                                return;
                            }
                            if (!data.isEmpty()) {
                                taula.call(data); // Retorna les dades al callback
                            } else {
                                error.call("No hi ha dades disponibles per a la taula " + table);
                            }
                        } else {
                            error.call("Resposta inesperada del servidor.");
                        }
                    },
                    error::call,
                    exception -> error.call(exception)
            );
        }

        /**
         * Format de les dates (retalla a AAAA-MM-DD).
         */
        private static String formatDate(String date) {
            return date.length() >= 10 ? date.substring(0, 10) : date;
        }

        /**
         * Analitza els filtres a partir d'una cadena en format "clau=valor".
         */
        private Map<String, String> parseFilters(String filterString) {
            Map<String, String> filters = new HashMap<>();
            if (filterString != null && !filterString.isEmpty()) {
                Arrays.stream(filterString.split("&"))
                        .map(pair -> pair.split("=", 2))
                        .filter(keyValue -> keyValue.length == 2)
                        .forEach(keyValue -> filters.put(keyValue[0], keyValue[1])); // Afegeix els
    filtres al mapa
            }
            return filters;
        }

        // Interfície genèrica per callbacks
        public interface Callback<T> {
            void call(T result);
        }

        // Retorna el nom autenticat
        public static String getNom() {
            return nom_text;
        }
    }
```

RutaManager.java

Java

```java
package lab.four.test;

/**
 * Classe singleton per gestionar i emmagatzemar la ruta base de la URL que es fa servir.
 * Aquesta classe garanteix que només hi hagi una única instància compartida de `RutaManager`.
 */
public class RutaManager {
    private static RutaManager instance; // Instància única de la classe
    private String ruta; // Variable per emmagatzemar la ruta

    // Constructor privat per evitar la creació d'instàncies externes
    private RutaManager() {}

    /**
     * Mètode per obtenir la instància única de `RutaManager` (patró Singleton).
     *
     * @return Instància única de `RutaManager`.
     */
    public static synchronized RutaManager getInstance() {
        if (instance == null) { // Crea la instància només si encara no existeix
            instance = new RutaManager();
        }
        return instance;
    }

    /**
     * Retorna la ruta emmagatzemada.
     *
     * @return Ruta base actual.
     */
    public String getRuta() {
        return ruta;
    }

    /**
     * Assigna una nova ruta a la variable interna.
     *
     * @param ruta Nova ruta a emmagatzemar.
     */
    public void setRuta(String ruta) {
        this.ruta = ruta;
    }
}
```

taula.java

Java

```java
package lab.four.test;

import static java.security.AccessController.getContext;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class taula extends AppCompatActivity {

    private EditText editTextFilter; // Camp d'entrada per filtrar dades
    private TextView text; // TextView per mostrar missatges o informació
    private Button buttonSend, logoutButton; // Botons per enviar la consulta i tancar sessió

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new); // Assigna el disseny de l'activitat

        buttonSend = findViewById(R.id.buttonSend); // Inicialitza el botó d'enviament
        TableLayout tableLayout = findViewById(R.id.tableLayout); // Referència al TableLayout
per mostrar dades
        editTextFilter = findViewById(R.id.editTextText); // Camp d'entrada del filtre
        text = findViewById(R.id.text_gallery); // Missatge de benvinguda o estat
        logoutButton = findViewById(R.id.logoutButton); // Botó per tancar sessió

        text.setText("Welcome " + Connexio.getNom().split(" ")[0] + "!"); // Mostra un missatge
de benvinguda

        // Comprova si el botó d'enviament està inicialitzat correctament
        if (buttonSend == null)
            throw new NullPointerException("El botó no es troba en el Layout");

        // Listener per gestionar l'enviament de la consulta
```

```
        buttonSend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String filterText = editTextFilter.getText().toString().trim(); // Obté el text
del filtre
                System.out.println(filterText);

                String serverIP = RutaManager.getInstance().getRuta(); // Obté la ruta del
servidor
                Connexio c = new Connexio(serverIP); // Crea una connexió amb la URL base

                // Fa la consulta a la taula
                c.queryTable(filterText,
                        data -> {
                            // Les dades rebudes es mostren en un fil UI
                            runOnUiThread(() -> displayDataInTable(data, tableLayout));
                        },
                        error -> {
                            // Maneja errors i mostra missatges
                            runOnUiThread(() -> {
                                if (error != null) {
                                    if (error.equals("Error: 600 - {\"error\":\"Sessió no
iniciada o caducada\"}")) {
                                        Toast.makeText(taula.this, "Sessió caducada",
Toast.LENGTH_SHORT).show();
                                        Intent intent = new Intent(taula.this,
MainActivity.class); // Redirigeix a MainActivity
                                        startActivity(intent);
                                        finish(); // Finalitza l'activitat actual
                                    } else {
                                        Toast.makeText(taula.this, "Error al realitzar la
consulta, torna-ho a provar", Toast.LENGTH_SHORT).show();
                                    }
                                }
                            });
                        });
            }
        });

        // Listener per gestionar el tancament de sessió
        logoutButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(taula.this, MainActivity.class); // Torna a
l'activitat principal
                startActivity(intent);
                finish(); // Finalitza l'activitat actual
            }
        });
    }

    // Mètode per mostrar les dades en un TableLayout
```

```
    private void displayDataInTable(Map<String, List<Object>> data, TableLayout tableLayout) {
        // Esborra les files existents al TableLayout
        tableLayout.removeAllViews();

        // Crea una llista de claus i determina el màxim nombre de files
        List<String> keys = new ArrayList<>(data.keySet());
        int maxRows = 0;
        for (List<Object> values : data.values()) {
            maxRows = Math.max(maxRows, values.size()); // Determina el nombre màxim de valors
        }

        // Afegeix una fila d'encapçalament amb els noms de les columnes
        TableRow headerRow = new TableRow(this);
        for (String key : keys) {
            TextView textView = new TextView(this);
            textView.setText(key); // Text del nom de la columna
            textView.setPadding(16, 16, 16, 16); // Defineix els marges del text
            textView.setTextSize(16); // Ajusta la mida del text
            textView.setBackgroundColor(0xFFEFEFEF); // Color de fons clar
            textView.setTextColor(0xFF000000); // Color de text negre
            headerRow.addView(textView); // Afegeix la cel·la a la fila
        }
        tableLayout.addView(headerRow); // Afegeix la fila d'encapçalament a la taula

        // Afegeix les files amb els valors corresponents
        for (int i = 0; i < maxRows; i++) {
            TableRow tableRow = new TableRow(this);
            for (String key : keys) {
                TextView textView = new TextView(this);
                List<Object> columnData = data.get(key);
                String cellValue = (i < columnData.size()) ? columnData.get(i).toString() : "";
// Evita valors nulls
                textView.setText(cellValue); // Text del valor de la cel·la
                textView.setPadding(16, 16, 16, 16); // Ajusta els marges de les cel·les
                textView.setTextSize(16); // Ajusta la mida del text
                textView.setBackgroundColor(0xFFF5F5F5); // Fons alternatiu
                textView.setTextColor(0xFF000000); // Color del text negre
                tableRow.addView(textView); // Afegeix la cel·la a la fila
            }
            tableLayout.addView(tableRow); // Afegeix la fila a la taula
        }
    }
}
```

AndroidManifest.xml

JavaScript

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

```
    xmlns:tools="http://schemas.android.com/tools" >

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:networkSecurityConfig="@xml/network_security_config"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Test"
        tools:targetApi="31" >

        <!-- Activitat principal -->
        <activity
            android:name=".MainActivity"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- Nova activitat -->
        <activity
            android:name=".taula"
            android:exported="false" />

    </application>

</manifest>
```

activity_main.xml

JavaScript

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#B4CFEC"
    tools:context=".MainActivity">

<TextView
```

```
android:id="@+id/text_home"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginStart="8dp"
android:layout_marginEnd="8dp"
android:textAlignment="center"
android:textSize="20sp"
app:layout_constraintBottom_toTopOf="@+id/textView3"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="1.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.039" />

<TextView
android:id="@+id/textView3"
android:layout_width="228dp"
android:layout_height="38dp"
android:layout_marginTop="104dp"
android:text="Credencials UPC"
android:textAlignment="center"
android:textAppearance="@style/TextAppearance.AppCompat.Large"
android:textStyle="bold"
app:layout_constraintBottom_toTopOf="@+id/editTextTextPassword3"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.497"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.837" />

<EditText
android:id="@+id/editTextTextPassword"
android:layout_width="295dp"
android:layout_height="58dp"
android:ems="10"
android:hint="Contrasenya"
android:inputType="textPassword"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.523" />

<EditText
android:id="@+id/editTextTextPassword2"
android:layout_width="299dp"
android:layout_height="48dp"
android:layout_marginTop="252dp"
android:ems="10"
android:hint="Nom d'usuari"
app:layout_constraintBottom_toTopOf="@+id/editTextTextPassword"
app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintHorizontal_bias="0.497"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="1.0" />

<EditText
android:id="@+id/editTextTextPassword3"
android:layout_width="296dp"
android:layout_height="56dp"
android:layout_marginTop="184dp"
android:ems="10"
android:hint="http://&amp;quot;IP&amp;quot;:&amp;quot;PORT&amp;quot;"
android:text="http://"
app:layout_constraintBottom_toTopOf="@+id/editTextTextPassword2"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.497"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="1.0" />

<Button
android:id="@+id/button2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Login"
android:textColorLink="#9C27B0"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.498"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editTextTextPassword"
app:layout_constraintVertical_bias="0.054" />

    </androidx.constraintlayout.widget.ConstraintLayout>
```

activity_new.xml

JavaScript

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#B4CFEC">

    <TextView
        android:id="@+id/text_gallery"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:gravity="center"
        android:text="Benvingut!"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="369dp"
        android:layout_height="84dp"
        android:layout_marginTop="16dp"
        android:orientation="horizontal"
        android:padding="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/text_gallery">

        <EditText
            android:id="@+id/editTextText"
            android:layout_width="0dp"
            android:layout_height="55dp"
            android:layout_weight="1"
            android:hint="Filters ie: marks?mark=9"
            android:inputType="text" />

        <Button
            android:id="@+id/buttonSend"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:text="Send" />
    </LinearLayout>

    <ScrollView
        android:id="@+id/scrollView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@id/linearLayout"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toTopOf="@id/logoutButton"
        android:layout_margin="16dp">

        <TableLayout
            android:id="@+id/tableLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:contentDescription="Taula"
            android:scrollbarStyle="outsideInset"
            android:scrollbars="vertical"
            android:stretchColumns="*" />
    </ScrollView>

    <Button
        android:id="@+id/logoutButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Logout"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_margin="16dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

network_security_config.xml

```
JavaScript
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

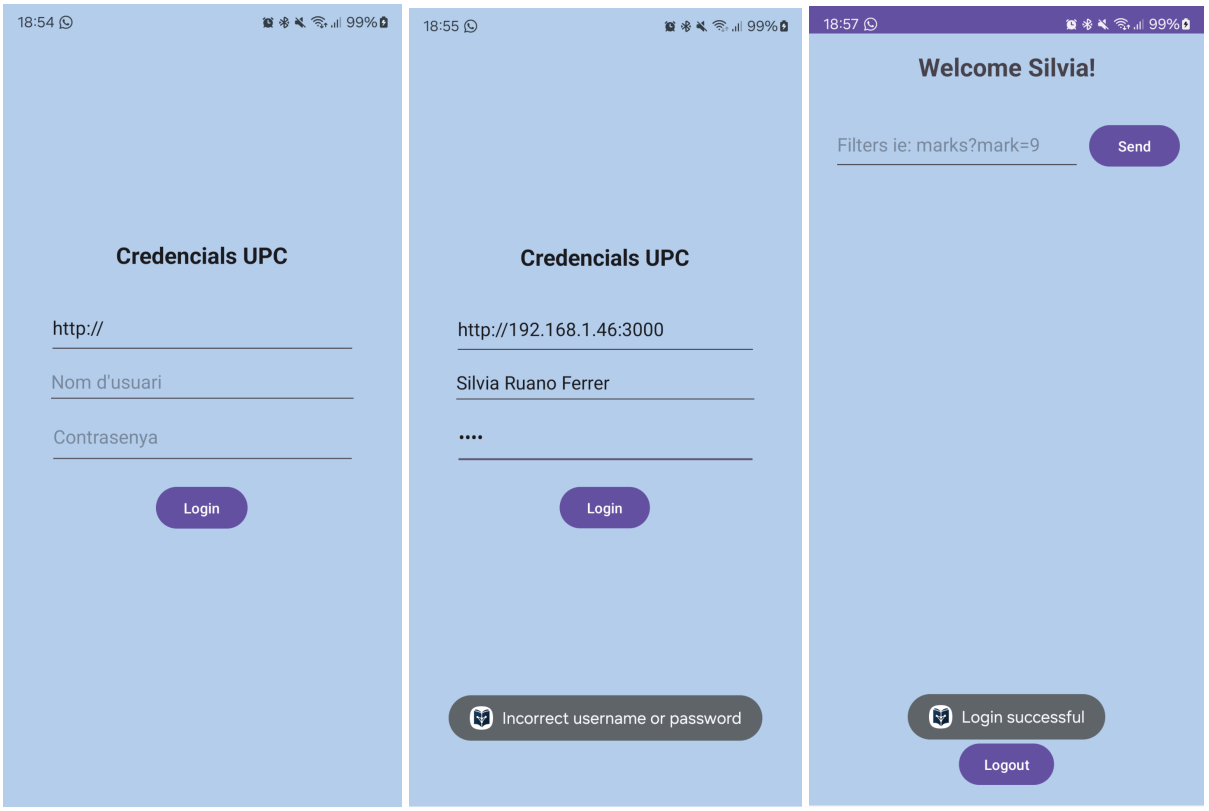
Un dels problemes que vam tenir durant el desenvolupament de l'app va ser la connexió amb el servidor. Tot i utilitzar el mateix codi (adaptat a un altre idioma) que el del CDR, vam detectar dificultats. Després d'investigar, vam descobrir que, a partir de la versió d'Android 9, per motius de seguretat, no es permeten connexions no segures (com les realitzades a través d'adreces IP sense HTTPS) de manera directa. Per solucionar aquest problema, va ser necessari crear el fitxer network_security_config.xml, que permet establir connexions en text clar amb el servidor.

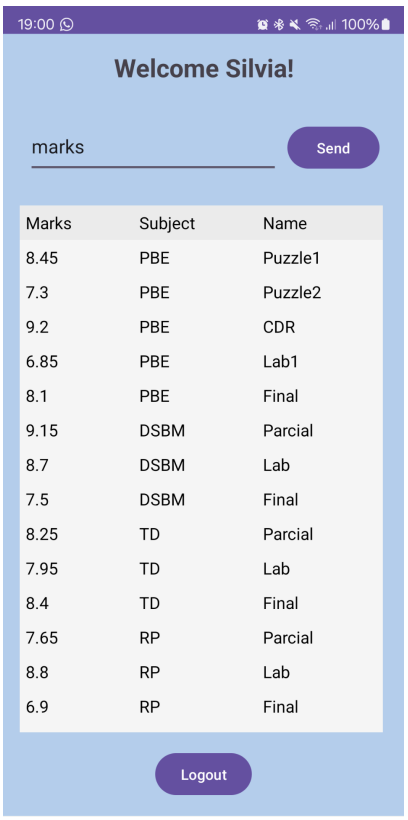
En la pantalla d'inici de sessió, es demana l'URL del servidor al qual ens volem connectar, juntament amb el nom d'usuari i la contrasenya, que correspon al uid de l'estudiant. Mitjançant aquest identificador, es realitza una consulta al servidor per obtenir el nom associat. Si el nom retornat no coincideix amb el que l'usuari ha introduït o si el uid no existeix, es mostra un missatge d'error indicant que l'usuari o la contrasenya són incorrectes. En canvi, si la informació és correcta, es mostra un missatge de confirmació d'accés i l'usuari passa a la pantalla de cerca de taules.

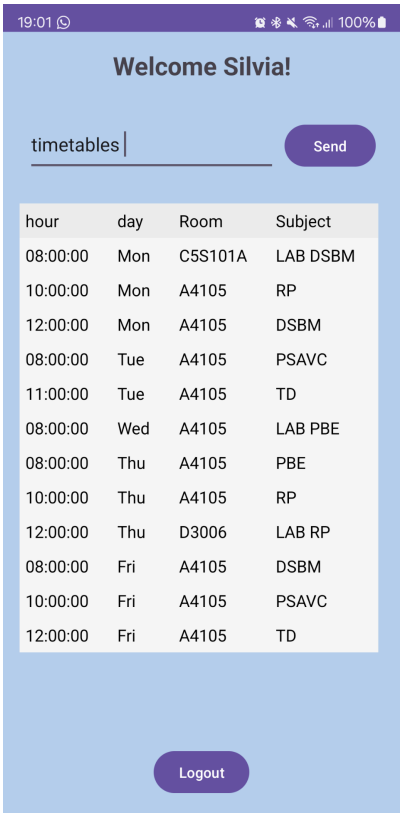
En aquesta nova pantalla, es pot introduir el nom de la taula a la qual es vol accedir, podent posar filtres si es vol. Quan es prem el botó de cerca, es mostra la taula demanada. En cas que el nom de la taula o els filtres introduïts no siguin vàlids, es genera un missatge d'error. Aquesta pantalla també inclou un botó per tancar la sessió i tenint en compte el sistema que tanca automàticament la sessió després de dos minuts d'inactivitat del servidor, en intentar realitzar una consulta, es mostra un avís de "sessió caducada" i es redirigeix l'usuari a la pantalla d'inici.

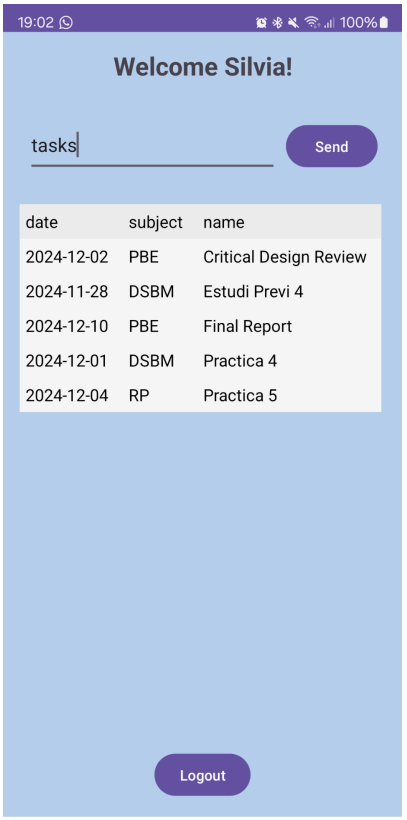
Les taules implementades permeten fer scroll, amb un cursor visible al lateral per facilitar el desplaçament i assegurar que totes les dades es puguin visualitzar còmodament. Com que es tracta d'una taula dinàmica, adaptar els colors a les dades específiques resultava tècnicament complicat. Tot i això, s'ha treballat per oferir un disseny net i estètic que facilita la lectura i la comprensió de la informació mostrada.

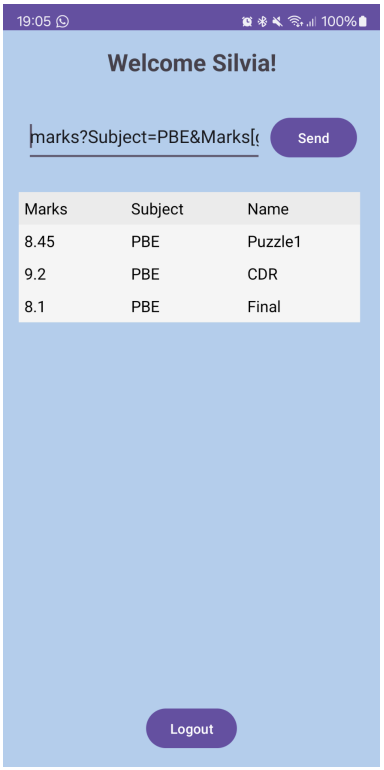
En l'exemple de taula amb filtres, s'ha utilitzat el següent format: Marks?Subject=PBE&Marks[gte]=8. Aquest filtre permet mostrar les notes de l'assignatura PBE amb valors iguals o superiors a 8.

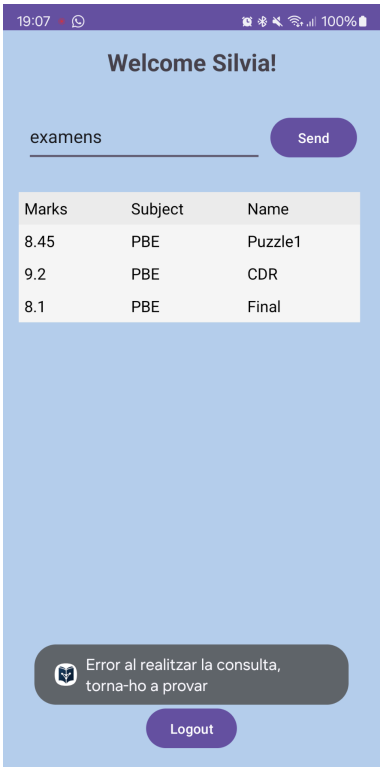


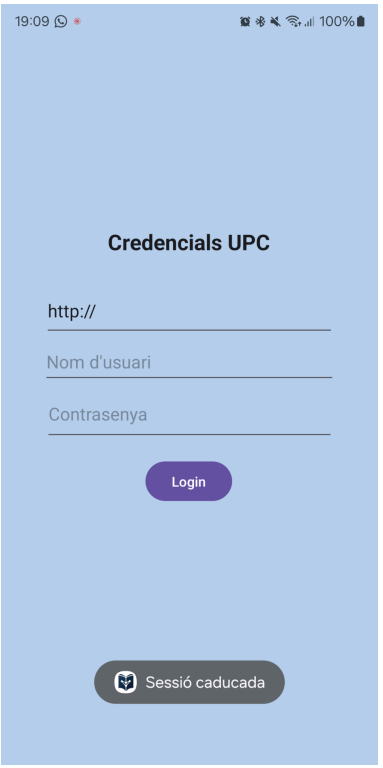

19:00
100%
Welcome Silvia!
marks
Send
Marks Subject Name
8.45 PBE Puzzle1
7.3 PBE Puzzle2
9.2 PBE CDR
6.85 PBE Lab1
8.1 PBE Final
9.15 DSBM Parcial
8.7 DSBM Lab
7.5 DSBM Final
8.25 TD Parcial
7.95 TD Lab
8.4 TD Final
7.65 RP Parcial
8.8 RP Lab
6.9 RP Final
Logout


19:01
100%
Welcome Silvia!
timetables
Send
hour day Room Subject
08:00:00 Mon C5S101A LAB DSBM
10:00:00 Mon A4105 RP
12:00:00 Mon A4105 DSBM
08:00:00 Tue A4105 PSAVC
11:00:00 Tue A4105 TD
08:00:00 Wed A4105 LAB PBE
08:00:00 Thu A4105 PBE
10:00:00 Thu A4105 RP
12:00:00 Thu D3006 LAB RP
08:00:00 Fri A4105 DSBM
10:00:00 Fri A4105 PSAVC
12:00:00 Fri A4105 TD
Logout


19:02
100%
Welcome Silvia!
tasks
Send
date subject name
2024-12-02 PBE Critical Design Review
2024-11-28 DSBM Estudi Previ 4
2024-12-10 PBE Final Report
2024-12-01 DSBM Practica 4
2024-12-04 RP Practica 5
Logout


19:05
100%
Welcome Silvia!
marks?Subject=PBE&Marks[(
Send
Marks Subject Name
8.45 PBE Puzzle1
9.2 PBE CDR
8.1 PBE Final
Logout


19:07
100%
Welcome Silvia!
examens
Send
Marks Subject Name
8.45 PBE Puzzle1
9.2 PBE CDR
8.1 PBE Final
Error al realitzar la consulta, torna-ho a provar
Logout


19:09
100%
Credencials UPC
http://
Nom d'usuari
Contrasenya
Login
Sessió caducada

4. WEB

index.html

```JavaScript
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Redirigint a Login</title>
    <script>
        // Redirigeix immediatament a login.html
        window.location.href = "login.html";
    </script>
</head>
<body>
    <p>Si no es redirigeix automàticament, <a href="login.html">fes clic aquí</a>.</p>
</body>
</html>
```

Aquest codi HTML crea una pàgina que redirigeix automàticament l'usuari a la pàgina `login.html` mitjançant JavaScript. Si el navegador no permet executar JavaScript, es proporciona un enllaç alternatiu perquè l'usuari pugui accedir manualment a la pàgina de destinació. La pàgina informa que l'usuari està sent redirigit i ofereix una solució en cas que la redirecció automàtica no funcioni.

El codil utilitza la codificació UTF-8. Això permet que es mostrin correctament tots els caràcters, inclosos accents i símbols especials. També, hem configurat la visualització per fer que la pàgina sigui responsive. En dispositius mòbils, aquesta línia ajusta la mida de la pàgina a l'amplada de la pantalla del dispositiu.

connexio.js

```JavaScript
class Connexio {
    constructor(url) {
        this.BASE_URL = url; // Especifica la URL del servidor
    }

    // Petición HTTP GET
    fetchData(path, params = {}, onSuccess, onError, onException) {
        const url = new URL(this.BASE_URL + path);

        // Asegurarse que todos los parámetros sean cadenas y sean agregados correctamente a la
URL
        Object.keys(params).forEach(key => {
            // Convierte el valor del parámetro en cadena si no lo es
            let value = params[key];
            if (typeof value !== 'string') {
```

```
            value = String(value); // Convertir a cadena
        }
        url.searchParams.append(key, value);
    });

    // Verifica que la URL final es la correcta
    console.log("URL construida:", url.toString());  // Agregado para depurar

    // Realiza la petición HTTP
    fetch(url)
        .then(response => {
            if (response.ok) {
                return response.json(); // Retorna el JSON si la respuesta es correcta
            } else {
                throw new Error(`Error: ${response.status} - ${response.statusText}`);
            }
        })
        .then(data => {
            onSuccess(data);  // Si la respuesta es correcta, llama a onSuccess
        })
        .catch(error => {
            if (onException) {
                onException("Error al conectar con el servidor: " + error.message);
            } else {
                onError(error.message);
            }
        });
}

// Processar las datos y retornarlas en un formato adecuado
processData(data, selectedColumns) {
    const result = selectedColumns.reduce((acc, column) => {
        acc[column] = [];
        return acc;
    }, {});

    data.forEach(row => {
        selectedColumns.forEach(column => {
            if (row.hasOwnProperty(column)) {
                let value = row[column];
                if (column === 'date') {
                    value = this.formatDate(value); // Formatear la fecha
                }
                result[column].push(value);
            }
        });
    });

    return result;
}

// Formato de la fecha (YY-MM-DD)
```

```
formatDate(date) {
    return date.length >= 10 ? date.substring(0, 10) : date;
}

// Autentificar al usuario
authenticateUser(uid, onSuccess, onError) {
    this.fetchData(
        "/authenticate",
        { uid: String(uid) }, // Asegúrate de convertir el uid en cadena
        data => {
            if (data && data.name) {
                onSuccess(data.name); // Retorna el nombre del usuario
            } else {
                onError("Error de autenticación. Vuelve a intentarlo.");
            }
        },
        onError,
        (exception) => {
            onError("Error al conectar con el servidor: " + exception);
        }
    );
}

// Consultar la tabla
queryTable(nomTaula, onSuccess, onError) {
    let params = {};
    let table = null;

    if (!nomTaula.includes("?")) {
        table = nomTaula;
    } else {
        const parts = nomTaula.split("?", 2);
        table = parts[0];
        params = this.parseFilters(parts[1]);
    }

    if (!table) {
        onError("Error: tabla desconocida.");
        return;
    }

    params.table = table;

    this.fetchData(
        "/query",
        params,
        data => {
            if (Array.isArray(data)) {
                let result;
                switch (table) {
                    case "timetables":
```

```
                            result = this.processData(data, ["day", "hour", "Subject",
"Room"]);
                            break;
                        case "tasks":
                            result = this.processData(data, ["date", "subject", "name"]);
                            break;
                        case "marks":
                            result = this.processData(data, ["Subject", "Name", "Marks"]);
                            break;
                        default:
                            onError("Error: tabla desconocida.");
                            return;
                    }

                    if (Object.keys(result).length > 0) {
                        onSuccess(result);
                    } else {
                        onError("No hay datos disponibles para la tabla " + table);
                    }
                } else {
                    onError("Respuesta inesperada del servidor.");
                }
            },
            onError
        );
    }

    // Parsear filtros
    parseFilters(filterString) {
        const filters = {};
        if (filterString) {
            const pairs = filterString.split("&");
            pairs.forEach(pair => {
                const [key, value] = pair.split("=");
                if (key && value) {
                    filters[key] = value;
                }
            });
        }
        return filters;
    }
}

// Ejemplos de funciones de callback (pueden ser usadas en la aplicación web)

function onSuccess(result) {
    console.log("Datos obtenidos:", result);
}

function onError(error) {
    console.log("Error:", error);
}
```

```
function onException(exception) {
    console.log("Excepción:", exception);
}

// Ejemplo de uso
const connexio = new Connexio("http://localhost:3000"); // URL de la API

// Autentificar usuario
connexio.authenticateUser("1234",
    name => console.log("Usuario autenticado:", name),
    error => console.log(error)
);

// Consultar tabla
connexio.queryTable("timetables",
    data => console.log("Tabla de horarios:", data),
    error => console.log(error)
);
```

Aquest codi implementa una classe anomenada *Connexio*, dissenyada per gestionar la comunicació amb un servidor a través de peticions HTTP. Aquesta classe facilita l'obtenció i el processament de dades des del servidor, així com la gestió de diferents operacions relacionades amb l'autenticació d'usuaris i la consulta de taules de dades.

El constructor de la classe inicialitza una URL base que es farà servir com a punt de partida per construir les peticions al servidor. Aquesta URL s'utilitza com a referència per a totes les operacions que necessiten interactuar amb el servidor.

Un dels mètodes principals és fetchData, que realitza peticions HTTP de tipus GET. Aquest mètode permet enviar paràmetres al servidor com a part de la cadena de consulta (query string) i gestiona les respostes. Si la resposta del servidor és correcta, es retorna el contingut en format JSON. En cas d'error, el mètode llança una excepció.

El codi inclou funcionalitats per organitzar i filtrar dades. El mètode *processData* permet extreure només les columnes rellevants d'un conjunt de dades, mentre que el mètode *formatDate* assegura que les dates segueixin un format estàndard. Aquestes funcions són útils per adaptar les dades obtingudes del servidor a un format adequat per al seu ús en l'aplicació

Mitjançant el mètode *authenticateUser*, el sistema comprova si un identificador d'usuari és vàlid i retorna el nom de l'usuari autenticat si l'operació té èxit.

El mètode *queryTable* permet accedir a taules del servidor, com ara horaris, tasques o notes. També inclou funcionalitats per aplicar filtres a les consultes i processar els resultats retornats.

El codi està dissenyat per gestionar errors i excepcions de manera clara i eficient. Els errors relacionats amb la connexió o amb la resposta del servidor es notifiquen mitjançant callbacks, permetent a l'aplicació reaccionar adequadament davant de situacions imprevistes.

RutaManager.js

JavaScript

```
class RutaManager {
    constructor() {
        this.ipPort = localStorage.getItem("ipPort");
        this.username = localStorage.getItem("username");
        this.pasword = localStorage.getItem("pasword");
        this.connexio = this.ipPort ? new Connexio(this.ipPort) : null;
    }

    // Comprova si l'usuari està autenticat
    isAuthenticated() {
        return this.username !== null && this.pasword !== null;
    }

    // Recupera la connexió
    getConnexio() {
        if (!this.connexio) {
            throw new Error("Connexió no inicialitzada.");
        }
        return this.connexio;
    }

    // Autenticació de l'usuari
    authenticateUser(uid, onSuccess, onError) {
        const connexio = this.getConnexio();
        connexio.authenticateUser(uid, onSuccess, onError);
    }

    // Recuperar les dades de la taula
    queryTable(nomTaula, onSuccess, onError) {
        const connexio = this.getConnexio();
        connexio.queryTable(nomTaula, onSuccess, onError);
    }

    // Comprova si tenim la connexió disponible
    checkConnexio() {
        if (!this.isAuthenticated()) {
            throw new Error("Usuari no autenticat. Redirigint a la pàgina de login.");
        }
    }

    // Redirecció a la pàgina de login si no està autenticat
    redirectToLogin() {
        if (!this.isAuthenticated()) {
            window.location.href = "login.html"; // Redirigeix si no està autenticat
        }
    }
}
```

Aquest codi defineix una classe anomenada *RutaManager*, que serveix com un gestor central per manejar connexions, autenticacions, i accés a dades en una aplicació web. Utilitza l'objecte *localStorage* per emmagatzemar i recuperar informació de l'usuari i els detalls de connexió, i facilita la interacció amb una altra classe *(Connexio)* que realitza operacions amb el servidor.

Es crea una instància de *RutaManager* i s'inicialitzen els següents atributs:

- *ipPort:* Direcció IP i port del servidor, recuperat de *localStorage.*
- *username* **i** *pasword:* Nom d'usuari i contrasenya, també recuperats de *localStorage*.
- *connexio:* Objecte de la classe *Connexio* que s'inicialitza si *ipPort* està disponible.

Si *ipPort* no està definit, l'atribut *connexio* es deixa com a *null*.

El mètode *isAuthenticated()* comprova si l'usuari està autenticat revisant si tant username com *pasword* existeixen en *localStorage*. Si l'usuari no està autenticat el mètode *redirectToLogin()* redirigeix automàticament a la pàgina *login.html*.

El mètode *getConnexio()* retorna l'objecte *Connexio* inicialitzat. Si no hi ha cap connexió disponible, es llança un error.

El mètode *authenticateUser(uid, onSuccess, onError)* delega la tasca d'autenticació a l'objecte *Connexio*, que comprova si un usuari determinat està autoritzat.

El mètode *queryTable(nomTaula, onSuccess, onError)* facilita la recuperació de dades d'una taula específica, passant el nom de la taula al mètode corresponent de *Connexio*. Això permet accedir a diferents conjunts de dades del servidor.

El codi està preparat per gestionar situacions en què no es pugui establir una connexió o quan l'usuari no estigui autenticat:

login.html

```JavaScript
<!DOCTYPE html>
<html lang="ca">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="login-container">
        <div class="header">
            <h1>Credencials UPC</h1>
        </div>

        <form class="login-form" id="loginForm">
```

```
            <div class="input-group">
                <label for="ip-port">IP i Port:</label>
                <input type="text" id="ip-port" name="ip-port" placeholder="http://IP:PORT"
required>
            </div>

            <div class="input-group">
                <label for="username">Nom d'usuari:</label>
                <input type="text" id="username" name="username" placeholder="Nom d'usuari"
required>
            </div>

            <div class="input-group">
                <label for="password">Contrasenya:</label>
                <input type="password" id="password" name="password" placeholder="Contrasenya"
required>
            </div>

            <button type="submit" class="login-button">Login</button>
        </form>
    </div>

    <!-- Vincula els arxius JavaScript -->
    <script src="Connexio.js"></script>
    <script src="login.js"></script>
    <script src="RutaManager.js"></script>
</body>
</html>
```

Aquest codi HTML crea la pàgina de login per autenticar-se en la nostra aplicació. Conté un formulari amb camps per introduir l'IP i port del servidor, el nom d'usuari i la contrasenya, tots obligatoris. Quan l'usuari fa clic a "Login", el formulari envia les dades a través de JavaScript (login.js) per validar les credencials amb el servidor (Connexio.js i RutaManager.js).

login.js

JavaScript

```
document.addEventListener("DOMContentLoaded", function () {
    const loginForm = document.getElementById("loginForm");

    loginForm.addEventListener("submit", function (event) {
        event.preventDefault(); // Evita el comportament per defecte (canvi de URL i recàrrega
de la pàgina)

        // Obtenim les dades del formulari (recorda que els camps estan invertits)
        const ipPort = document.getElementById("ip-port").value.trim();
```

```
        const password = document.getElementById("password").value.trim(); // El "password"
conté el rfid
        const username = document.getElementById("username").value.trim(); // El "username"
conté el user

        if (!ipPort || !username || !password) {
            alert("Tots els camps són obligatoris!");
            return;
        }

        // Inicialitzem el RutaManager
        const rutaManager = new RutaManager();

        // Intentem autenticar l'usuari
        rutaManager.authenticateUser(
            password,
            // Callback d'èxit
            (serverUsername) => {
                console.log("Nom d'usuari retornat pel servidor:", serverUsername);

                // Comprova si el nom d'usuari retornat pel servidor coincideix amb el del
formulari
                if (serverUsername === username) {
                    alert("Usuari autenticat amb èxit!");

                    // Desa informació necessària al localStorage
                    localStorage.setItem("ipPort", ipPort);
                    localStorage.setItem("password", password);
                    localStorage.setItem("username", username);

                    // Redirigeix a la pàgina de taula
                    window.location.href = "taula.html";
                } else {
                    alert("Error: el nom d'usuari retornat pel servidor no coincideix amb el
nom introduït.");
                }
            },
            // Callback d'error
            (error) => {
                console.error("Error d'autenticació:", error);
                alert("Error en l'autenticació. Torna-ho a provar.");
            }
        );
    });
});
```

Aquest codi és el script que gestiona l'autenticació d'usuaris de l'aplicació web. Quan la pàgina es carrega completament, s'activa un event listener associat al formulari d'inici de sessió (loginForm). Quan l'usuari

intenta enviar el formulari, el codi evita el comportament predeterminat (recarregar la pàgina) i valida que tots els camps requerits (ip-port, username i password) estiguin omplerts.

Després, es crea una instància de RutaManager, un objecte que gestiona la connexió amb el servidor. A través del mètode authenticateUser, es comprova si el password introduït és correcte. Si el servidor retorna un nom d'usuari que coincideix amb el que s'ha introduït al formulari, el codi desa informació al localStorage i redirigeix l'usuari a la pàgina taula.html. Si hi ha una discrepància o un error en el procés, es mostren missatges d'avís a l'usuari.

En resum, aquest codi s'encarrega de validar les credencials de manera robusta i de controlar l'accés a funcionalitats posteriors de l'aplicació en funció de l'èxit de l'autenticació.

taula.html

```JavaScript
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Consulta de Taula</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div id="container">
        <h1 id="welcomeMessage">Welcome!</h1>
        <div class="input-section">
            <input id="filterInput" type="text" placeholder="Filters ie: marks?mark=9" />
            <button id="sendButton">Send</button>
        </div>
        <div id="scrollView">
            <table id="dataTable">
                <!-- Dinàmicament afegirem contingut aquí -->
            </table>
        </div>
        <button id="logoutButton">Logout</button>
    </div>
    <script src="taula.js"></script>
    <script src="Connexio.js"></script>
    <script src="RutaManager.js"></script>

</body>
</html>
```

Aquest codi HTML crea una pàgina per consultar i filtrar dades en una taula. Inclou un camp per introduir filtres (filterInput) i un botó per enviar la consulta (sendButton). Les dades s'insereixen dinàmicament en una taula (dataTable). També hi ha un botó per tancar la sessió (logoutButton). El codi carrega els pertinents arxius CSS i scripts (taula.js, Connexio.js, i RutaManager.js) per gestionar la funcionalitat i les connexions amb el servidor.

taula.js

JavaScript

```
document.addEventListener("DOMContentLoaded", function () {
    const welcomeMessage = document.getElementById("welcomeMessage");
    const filterInput = document.getElementById("filterInput");
    const sendButton = document.getElementById("sendButton");
    const dataTable = document.getElementById("dataTable");
    const logoutButton = document.getElementById("logoutButton");

    // Inicialitzem el RutaManager
    const rutaManager = new RutaManager();

    // Verifiquem si l'usuari està autenticat
    rutaManager.redirectToLogin();  // Si no està autenticat, redirigeix a login

    const userName = rutaManager.username;

    // Comprova si tenim el nom d'usuari a localStorage
    if (userName) {
        welcomeMessage.textContent = `Welcome ${userName.split(" ")[0]}!`;
    } else {
        welcomeMessage.textContent = "Welcome, guest!";
    }

    // Funció per crear una fila a la taula
    function createTableRow(cells, isHeader = false) {
        const row = document.createElement("tr");
        cells.forEach(cellText => {
            const cell = isHeader ? document.createElement("th") :
document.createElement("td");
            cell.textContent = cellText;
            row.appendChild(cell);
        });
        return row;
    }

    // Funció per mostrar dades a la taula
    function displayDataInTable(data) {
        // Netejar la taula
        dataTable.innerHTML = "";

        // Obtenim les claus i el número màxim de files
        const keys = Object.keys(data);
        if (keys.length === 0) {
            dataTable.innerHTML = "<tr><td>No data available</td></tr>";
            return;
        }
        const maxRows = Math.max(...keys.map(key => data[key].length));

        // Afegim la capçalera
        const headerRow = createTableRow(keys, true);
        dataTable.appendChild(headerRow);
```

```
        // Afegim les files
        for (let i = 0; i < maxRows; i++) {
            const rowCells = keys.map(key => data[key][i] || ""); // Evitar valors nulls
            const row = createTableRow(rowCells);
            dataTable.appendChild(row);
        }
    }

    // Funció per fer una consulta a la taula
    async function queryTable(filter) {
        try {

            // Realitzem la consulta passant el nom de la taula i el filtre
            rutaManager.queryTable(filter, displayDataInTable, (error) => {
                console.error("Error querying table:", error);
                alert("Error al realitzar la consulta. Torna-ho a provar.");
            });
        } catch (error) {
            console.error("Error en la connexió:", error.message);
            alert("Error en la connexió. Torna-ho a provar.");
        }
    }

    // Assignem l'esdeveniment al botó de "Send"
    sendButton.addEventListener("click", () => {
        const filterText = filterInput.value.trim();
        if (filterText) {
            queryTable(filterText); // Passant el filtre correcte
        } else {
            alert("Please enter a valid filter.");
        }
    });

    // Assignem l'esdeveniment al botó de "Logout"
    logoutButton.addEventListener("click", () => {
        alert("You have logged out.");
        window.location.href = "login.html"; // Exemple de redirecció
    });
});
```

Aquest script JavaScript s'encarrega de gestionar la funcionalitat que permet consultar dades i mostrar-les en una taula dinàmica.

Primer, inicialitza un objecte *RutaManager* per gestionar connexions amb el servidor i verifica si l'usuari està autenticat amb *redirectToLogin()*. Si no ho està, es redirigeix automàticament a la pàgina de login. Després, personalitza el missatge de benvinguda segons el nom de l'usuari desat a localStorage.

El codi inclou funcions per gestionar la taula. La funció *createTableRow* crea una fila, ja sigui de dades o de capçalera. La funció *displayDataInTable* rep les dades del servidor, neteja el contingut anterior i omple la taula amb els nous registres, evitant valors buits.

Quan l'usuari escriu un filtre i fa clic a "Send", la funció queryTable envia aquest filtre al servidor i mostra els resultats a la taula. Si hi ha errors, es mostren missatges d'alerta. Finalment, el botó "Logout" permet tancar la sessió i redirigeix l'usuari a la pàgina de login.

styles.css

JavaScript

```
/* ------------------------------ */
/* ESTIL PER A LOGIN */
/* ------------------------------ */

/* Fons blau clar */
body {
    background-color: #B4CFEC;
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

/* Contenidor central */
.login-container {
    background-color: white;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 100%;
    max-width: 400px;
    text-align: center;
}

/* Títol de la pantalla */
.header h1 {
    font-size: 24px;
    margin-bottom: 20px;
    color: #333;
}
```

```
/* Formulari d'entrada */
.login-form {
    display: flex;
    flex-direction: column;
}

/* Grup de camps d'entrada */
.input-group {
    margin-bottom: 15px;
}

.input-group label {
    display: block;
    text-align: left;
    font-size: 14px;
    margin-bottom: 5px;
    color: #333;
}

.input-group input {
    width: 100%;
    padding: 10px;
    font-size: 14px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box;
}

.input-group input:focus {
    border-color: #9C27B0; /* Color de focus */
    outline: none;
}

/* Botó de login */
.login-button {
    padding: 10px;
    background-color: #9C27B0;
    color: white;
    font-size: 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.login-button:hover {
    background-color: #7B1FA2; /* Color més fosc en hover */
}

/* ------------------------------ */
/* ESTIL PER A TAULA.HTML */
```

```
/* ------------------------------ */

/* Fons de la pàgina */
#container {
    background-color: #B4CFEC;
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px;
    height: 100vh;
    box-sizing: border-box;
}

/* Títol de benvinguda */
#welcomeMessage {
    font-size: 24px;
    font-weight: bold;
    margin-bottom: 20px;
    color: #333;
}

/* Secció d'entrada de text */
.input-section {
    display: flex;
    gap: 10px;
    margin-bottom: 20px;
}

.input-section input {
    padding: 10px;
    font-size: 14px;
    border: 1px solid #ccc;
    border-radius: 5px;
    flex: 1;
}

.input-section input:focus {
    border-color: #007BFF; /* Color de focus */
    outline: none;
}

.input-section button {
    padding: 10px 20px;
    background-color: #007BFF;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 14px;
    transition: background-color 0.3s;
}
```

```
.input-section button:hover {
    background-color: #0056b3;
}

/* Estil de la taula */
#scrollView {
    width: 100%;
    max-height: calc(100vh - 200px);
    overflow-y: auto;
    margin-bottom: 20px;
}

#dataTable {
    width: 100%;
    border-collapse: collapse;
    text-align: left;
}

#dataTable th, #dataTable td {
    padding: 10px;
    border: 1px solid #ccc;
}

#dataTable th {
    background-color: #f5f5f5;
    font-weight: bold;
}

#dataTable tr:nth-child(even) {
    background-color: #f9f9f9;
}

/* Botó de logout */
#logoutButton {
    padding: 10px 20px;
    background-color: #dc3545;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 14px;
    transition: background-color 0.3s;
}

#logoutButton:hover {
    background-color: #a71d2a;
}
/* Estil per a la taula amb colors de blau més sòlids */
#dataTable {
    width: 100%;
    border-collapse: collapse;
    text-align: left;
```

```
}

/* Colors de fons per a les cel·les de la taula */
#dataTable th, #dataTable td {
    padding: 10px;
    border: 1px solid #ccc;
    color: #FFFFFF; /* Text blanc per garantir un bon contrast */
}

/* Color de fons per a la capçalera amb un blau intens */
#dataTable th {
    background-color: #2C5A9D; /* Blau més fosc per a la capçalera */
    font-weight: bold;
    color: white; /* Text blanc per contrastar amb el fons */
}

/* Fons alternant entre un blau suau més fosc i un blau fosc per les files */
#dataTable tr:nth-child(even) {
    background-color: #7FA9E2; /* Blau més fosc per a files parelles */
}

#dataTable tr:nth-child(odd) {
    background-color: #2C5A9D; /* Blau més fosc per a files imparells */
}

/* Eliminat efecte hover */
#dataTable tr:hover {
    background-color: transparent; /* Sense canvi en el hover */
}

/* Fonts i estil de les cel·les */
#dataTable th, #dataTable td {
    color: white; /* Text blanc per a la capçalera i les cel·les */
}
```

Aquest arxiu CSS és el que controla l'estil de tots els arxius HTML que hem necessitat. La seva funcionalitat es simplement estètica.

Imatges

A continuació mostrem unes imatges que reflecteixen el resultat de la nostra aplicació web.