

Cette fonctionnalité fait partie des améliorations entreprises et développées au cours de mon travail personnel. Elle offre la possibilité aux utilisateurs de se connecter/déconnecter afin de réserver un restaurant et d'être en mesure d'obtenir une liste répertoriant leurs réservations sur une page dédiée à leur profil.

Pour accéder à la page de connexion, il suffit de cliquer sur l'icône dédiée présent dans la barre d'action. Une fois connecté, l'utilisateur est redirigé automatiquement vers la première page avec un message de redirection.

Si l'utilisateur souhaite par la suite accéder à ses réservations et/ou se déconnecter, il lui suffit de cliquer de nouveau sur l'icône de la barre d'action afin d'être redirigé vers la page de son profil.

1 – Les contrôleurs

Pour cette fonctionnalité, nous avons besoins de deux contrôleurs, ils interagissent avec leur layout associé, mais aussi en l'occurrence, avec les autres activités et la base de données au travers d'un fichier PHP.

Commençons par le contrôleur permettant la connexion de la part de l'utilisateur : « LoginActivity ».

1.a - « LoginActivity »

On réalise les importations et les déclarations.

```
package com.example.projet_android.controleur;

import ...

10 usages
public class LoginActivity extends AppCompatActivity {

    // Déclaration des composants de l'interface utilisateur
    2 usages
    EditText usernameField, passwordField;
    2 usages
    Button loginButton;
    2 usages
    TextView timerTextView, errorMessage;
    2 usages
    CardView timerCardView;
```

Ensuite, dans « onCreate », on initialise les composants par leurs identifiants et on script la gestion de l'évènement clic sur le bouton de connexion.

```
// Initialisation des composants par leurs identifiants
usernameField = findViewById(R.id.username);
passwordField = findViewById(R.id.password);
loginButton = findViewById(R.id.loginButton);
timerTextView = findViewById(R.id.timerTextView);
timerCardView = findViewById(R.id.timerCardView);
errorMessage = findViewById(R.id.errorMessage);

// Gestion de l'évènement clic sur le bouton de connexion
loginButton.setOnClickListener(v -> {
    String username = usernameField.getText().toString().trim();
    String password = passwordField.getText().toString().trim();
    loginUser(username, password);
});
```

On développe une méthode pour connecter l'utilisateur avec une récupération des données et une gestion des erreurs.

```
// Méthode pour connecter l'utilisateur
1 usage
private void loginUser(String username, String password) {
    OkHttpClient client = new OkHttpClient();
    RequestBody formBody = new FormBody.Builder()
        .add(name: "username", username)
        .add(name: "password", password)
        .build();

    Request request = new Request.Builder()
        .url("http://10.15.13.167/android/appResto/getUser.php")
        .post(formBody)
        .build();

    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if (response.isSuccessful()) {
                final String responseData = response.body().string();
                runOnUiThread() -> {
                    try {
                        JSONObject json = new JSONObject(responseData);
                        if (json.getBoolean(name: "success")) {
                            saveLoginStatus(isLoggedIn: true, username);
                            startCountdown();
                        } else {
                            showErrorMessage("Identifiant ou mot de passe erroné, veuillez réessayer.");
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }
        }

        @Override
        public void onFailure(Call call, IOException e) {
            runOnUiThread() -> {
                showErrorMessage("Erreur de connexion, veuillez réessayer.");
            }
        }
    });
}
```

On affiche un message d'erreur si la connexion échoue.

```
// Affiche un message d'erreur
2 usages
private void showErrorMessage(String message) {
    errorMessage.setText(message);
    errorMessage.setVisibility(View.VISIBLE);
}
```

À l'inverse, on lance un compte à rebours indiquant une redirection vers la liste des restaurants si la connexion fonctionne.

```
// Lance un compte à rebours après une connexion réussie
1 usage
private void startCountdown() {
    runOnUiThread() -> {
        findViewById(R.id.loginLayout).setVisibility(View.GONE); // Hide the login form
        timerCardView.setVisibility(View.VISIBLE); // Show the CardView for the countdown

        new CountdownTimer( millisInFuture: 3000, countDownInterval: 1000) {
            public void onTick(long millisUntilFinished) {
                timerTextView.setText("Vous êtes à présent connecté.\n\nRedirection dans : " + millisUntilFinished / 1000 + " seconde(s)");
            }

            public void onFinish() {
                Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        }.start();
    }
};
}
```

Enfin, on récupère l'état de connexion et le nom de l'utilisateur. Cette action a pour rôle :

- d'octroyer l'accès à certaines fonctionnalités autorisées uniquement aux utilisateurs connectés ;
- de modifier l'icône de la barre d'action en conséquence ;
- d'afficher un message personnalisé sur la page de profil avec le nom de l'utilisateur et d'assurer que la réservation du restaurant inclut le nom du client.

```
// Enregistre l'état de connexion de l'utilisateur
1 usage
private void saveLoginStatus(boolean isLoggedIn, String username) {
    SharedPreferences prefs = getSharedPreferences( name: "UserPrefs", MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();
    editor.putBoolean( s: "isLoggedIn", isLoggedIn);
    editor.putString( s: "username", username);
    editor.apply();
}

// Récupère le nom d'utilisateur actuel
2 usages
public static String getUsername(Context context) {
    SharedPreferences prefs = context.getSharedPreferences( s: "UserPrefs", MODE_PRIVATE);
    return prefs.getString( s: "username", s: "Guest"); // "Guest" est la valeur par défaut si rien n'est trouvé
}
}
```

La méthode « loginUser » fait appel au fichier « getUser.php » que vous trouverez ci-dessous :

```
<?php
try {
    // Connexion à la base de données MySQL
    $db = "resto_test";
    $dbhost = "localhost";
    $dbport = 3306;
    $dbuser = "root";
    $dbpasswd = "joliverie";

    // Récupérer les données envoyées par l'application Android
    $username = trim($_POST['username'] ?? '');
    $password = trim($_POST['password'] ?? '');

    // Configuration de la connexion PDO avec le jeu de caractères
    $dsn = "mysql:host=$dbhost;port=$dbport;dbname=$db;charset=utf8";
    $connexion = new PDO($dsn, $dbuser, $dbpasswd);
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Préparation et exécution de la requête
    $reponse = $connexion->prepare("SELECT mdpU FROM utilisateur WHERE pseudoU = :username");
    $reponse->execute(['username' => $username]);
    $user = $reponse->fetch(PDO::FETCH_ASSOC);

    if ($user) {
        $isPasswordCorrect = password_verify($password, $user['mdpU']);
        echo json_encode(['success' => $isPasswordCorrect, 'hash' => $user['mdpU']]);
    } else {
        echo json_encode(['success' => false, 'error' => 'User not found']);
    }
} catch (Exception $e) {
    echo json_encode(['success' => false, 'error' => $e->getMessage()]);
}
echo "Username received: " . $username;

?>
```

La méthode compare le mot de passe envoyé à l'aide du champ, avec celui contenu dans la base de donnée correspondant au pseudo de l'utilisateur, afin de s'assurer que le mot de passe est correct.

Pour s'assurer que l'application est sécurisée, on utilise une requête préparée et un mot de passe hashé.

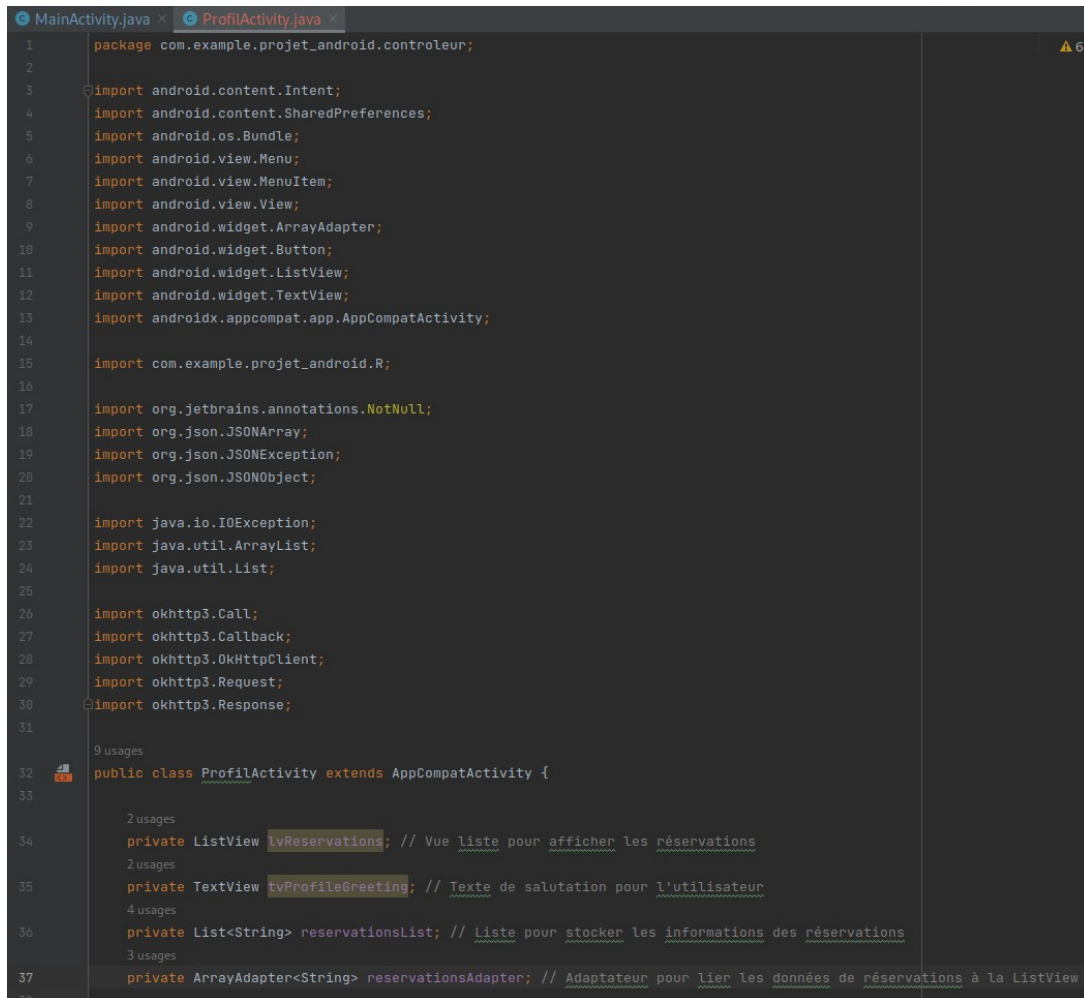
Pour le respect des convenances, on réalise une gestion des erreurs.

« success » est utilisé comme une clé dont la valeur est un booléen afin de valider ou non la connexion.

2.a - « Profil Activity »

Le contrôleur associé aux détails des restaurants est « ProfilActivity ».

On réalise les importations et les déclarations.



```
1 package com.example.projet_android.controleur;
2
3 import android.content.Intent;
4 import android.content.SharedPreferences;
5 import android.os.Bundle;
6 import android.view.Menu;
7 import android.view.MenuItem;
8 import android.view.View;
9 import android.widget.AdapterView;
10 import android.widget.Button;
11 import android.widget.ListView;
12 import android.widget.TextView;
13 import androidx.appcompat.app.AppCompatActivity;
14
15 import com.example.projet_android.R;
16
17 import org.jetbrains.annotations.NotNull;
18 import org.json.JSONArray;
19 import org.json.JSONException;
20 import org.json.JSONObject;
21
22 import java.io.IOException;
23 import java.util.ArrayList;
24 import java.util.List;
25
26 import okhttp3.Call;
27 import okhttp3.Callback;
28 import okhttp3.OkHttpClient;
29 import okhttp3.Request;
30 import okhttp3.Response;
31
32 9 usages
33 public class ProfilActivity extends AppCompatActivity {
34
35     2 usages
36     private ListView lvReservations; // Vue liste pour afficher les réservations
37     2 usages
38     private TextView tvProfileGreeting; // Texte de salutation pour l'utilisateur
39     4 usages
40     private List<String> reservationsList; // Liste pour stocker les informations des réservations
41     3 usages
42     private ArrayAdapter<String> reservationsAdapter; // Adaptateur pour lier les données de réservations à la ListView
43 }
```

À la suite des importations et des déclarations des différents éléments du layout, on développe « onCreate ». N'hésitez pas à vous référer aux commentaires du programme pour une lecture claire de celui-ci.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_profil);

    // Modifier le titre de la barre d'action
    if (getSupportActionBar() != null) {
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("R3STO.FR");
    }

    // Initialiser les composants de la vue
    lvReservations = findViewById(R.id.lvReservations);
    tvProfileGreeting = findViewById(R.id.tvProfileGreeting);

    // Récupérer le nom d'utilisateur depuis LoginActivity et afficher le message de bienvenue
    String pseudoU = LoginActivity.getUsername(context: this);
    tvProfileGreeting.setText("Bonjour " + pseudoU + " !");

    // Initialiser la liste et l'adaptateur pour les réservations
    reservationsList = new ArrayList<>();
    reservationsAdapter = new ArrayAdapter<>(context: this, android.R.layout.simple_list_item_1, reservationsList);
    lvReservations.setAdapter(reservationsAdapter);

    // Charger les réservations de l'utilisateur
    fetchReservations(pseudoU);

    // Générer le clic sur le bouton de déconnexion
    Button btnDeconnexion = findViewById(R.id.btnDeconnexion);
    btnDeconnexion.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { logoutUser(); }
    });
}
```


Ensuite, il convient de récupérer les réservations de l'utilisateur depuis le serveur et d'assurer la mise en place de toutes les informations qui seront affichées sur la page.

```
// Récupérer les réservations de l'utilisateur depuis le serveur
1 usage
private void fetchReservations(String pseudoU) {
    OkHttpClient client = new OkHttpClient();
    Request request = new Request.Builder()
        .url("http://10.15.13.167/android/appResto/getReservations.php?pseudoU=" + pseudoU)
        .build();

    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onResponse(@NotNull Call call, @NotNull Response response) throws IOException {
            if (response.isSuccessful()) {
                try {
                    String responseData = response.body().string();
                    JSONObject jsonResponse = new JSONObject(responseData);
                    JSONArray reservationsArray = jsonResponse.getJSONArray( name: "reservations");

                    reservationsList.clear();
                    for (int i = 0; i < reservationsArray.length(); i++) {
                        JSONObject reservation = reservationsArray.getJSONObject(i);
                        String resto = reservation.getString( name: "resto");
                        String date = reservation.getString( name: "date");
                        String heure = reservation.getString( name: "heure");

                        String reservationInfo = resto + " - Date : " + date + ", Heure : " + heure;
                        reservationsList.add(reservationInfo);
                    }

                    runOnUiThread() -> reservationsAdapter.notifyDataSetChanged();
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }

        @Override
        public void onFailure(@NotNull Call call, @NotNull IOException e) {
            e.printStackTrace();
        }
    });
}
```

Pour cela, comme à notre habitude, on se sert d'un fichier PHP « getReservations.php », qui permet à l'aide du pseudo de retourner les réservations au format JSON prêtes à être employées. Encore une fois, pour de plus amples détails référez-vous aux commentaires.

```
<?php
header('Content-Type: application/json');

// Connexion à la base de données MySQL
$db = "resto_test";
$dbhost = "localhost";
$dbport = 3306;
$dbuser = "root";
$dbpasswd = "joliverie";

try {
    $connexion = new PDO("mysql:host=$dbhost;port=$dbport;dbname=$db;charset=utf8", $dbuser, $dbpasswd);
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Récupérer le nom d'utilisateur depuis la requête
    $pseudoU = $_GET['pseudoU'] ?? '';

    if ($pseudoU) {
        // Requête SQL pour récupérer les réservations de l'utilisateur
        $reponse = $connexion->prepare("SELECT resto, date, heure FROM reservation WHERE pseudoU = :pseudoU");
        $reponse->execute(['pseudoU' => $pseudoU]);
        $reservations = $reponse->fetchAll(PDO::FETCH_ASSOC);

        // Retourner les réservations en format JSON
        echo json_encode(['reservations' => $reservations]);
    } else {
        echo json_encode(['reservations' => []]);
    }
} catch (Exception $e) {
    echo json_encode(['error' => $e->getMessage()]);
}
```

Pour finir, il reste à configurer le bouton de déconnexion.

```
Button btnDeconnexion = findViewById(R.id.btnDeconnexion);

btnDeconnexion.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { logoutUser(); }
});

// Usage
private void logoutUser() {
    // Mettre à jour les SharedPreferences pour indiquer que l'utilisateur est déconnecté
    SharedPreferences prefs = getSharedPreferences("UserPrefs", MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();
    editor.putBoolean("isLoggedIn", false);
    editor.apply();

    // Rafraîchir le menu pour afficher l'icône de connexion (optionnel ici si on redirige)
    invalidateOptionsMenu();

    // Redirection vers l'activité d'accueil
    Intent intent = new Intent(packageContext, ProfilActivity.this, MainActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK); // Important pour nettoyer la pile d'activités
    startActivity(intent);
    finish(); // Termine l'activité de profil pour que l'utilisateur ne puisse pas revenir en arrière
}
```

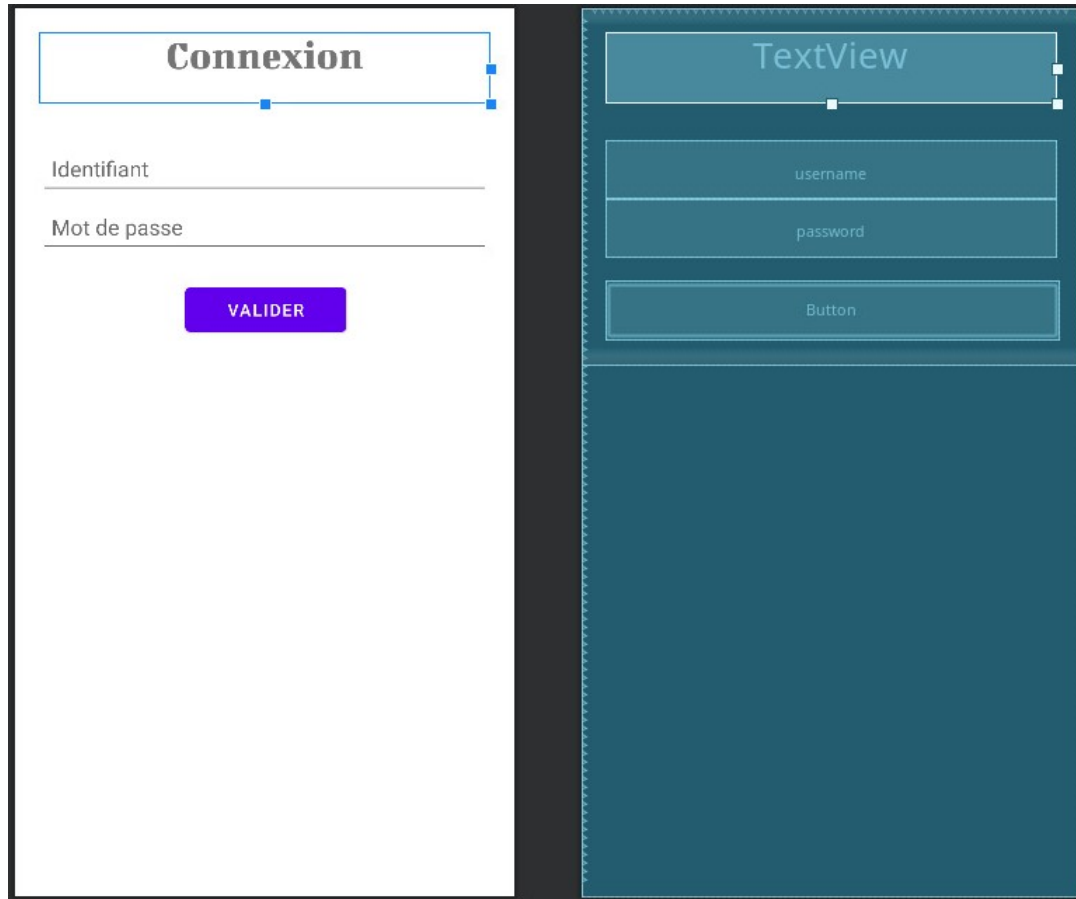
Ce bouton déconnecte l'utilisateur, met à jour l'icône de la barre d'action, et redirige l'utilisateur vers l'accueil. Durant cette étape, il est important de nettoyer la pile d'activités, en effet, dans le cas contraire, l'utilisateur pourrait retourner en arrière après s'être déconnecté et accéder à des zones de l'application réservées aux utilisateurs connectés.

2 – Les vues

Pour cette étape on dispose de deux vues (une par contrôleur).

Commençons par la page de connexion :

2.a - « activity_login »



Les spécificités de cette vue qui n'apparaissent pas sur le screen ci-dessus sont :

1 – Le TextView permettant l'affichage du message d'erreur ;

```
<!-- TextView pour les messages d'erreur -->
<TextView
    android:id="@+id/errorMessage"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@android:color/holo_red_dark"
    android:text="identifiant ou mot de passe erroné, veuillez réessayer."
    android:visibility="gone"
    android:textAlignment="center"
    android:paddingBottom="10dp"/>
```

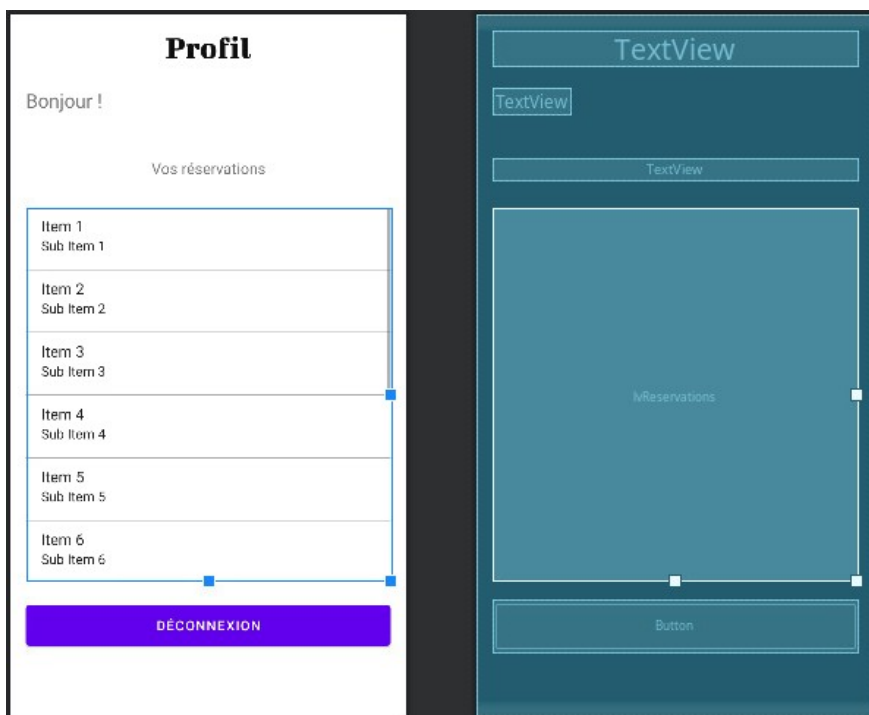
2 – Le TextView imbriqué dans un CardView qui affiche le message de redirection lors de la connexion.

```
<androidx.cardview.widget.CardView
    android:id="@+id/timerCardView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:layout_margin="32dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="4dp"
    android:visibility="gone">

    <TextView
        android:id="@+id/timerTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:padding="16dp"
        android:textSize="24sp"
        android:text="Vous êtes à présent connecté. Redirection vers la liste des restaurants dans ." />

</androidx.cardview.widget.CardView>
```

2.b - « activity profil »



Le TextView correspondant au « Bonjour ! » est en réalité un message personnalisé comportant le pseudonyme de l'utilisateur.

3 – Jeu d’essai

Dans ce jeu d’essai, il conviendra de réaliser une connexion, puis une déconnexion en passant par la page de profil.

L'utilisateur est déconnecté, comme on peut l'observer avec l'icône.



En cliquant dessus, on arrive sur cette page.

A screenshot of the login page. It features a blue header bar with "R3STO.FR". Below it, the word "Connexion" is centered. There are two input fields: "Identifiant" and "Mot de passe". A blue button labeled "VALIDER" is positioned below the password field.

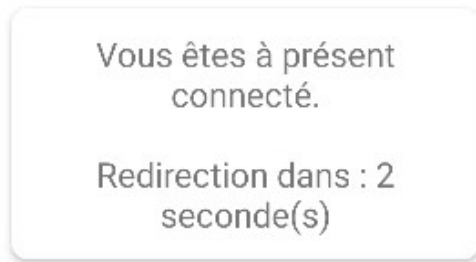
Dans un premier temps, on entre des informations incorrects pour afficher le message d'erreur.

A screenshot of the login page showing an error state. The blue header bar with "R3STO.FR" is at the top. Below it, "Connexion" is centered. A red error message "Identifiant ou mot de passe erroné, veuillez réessayer." is displayed. The "Identifiant" field contains the text "erreur". The "Mot de passe" field contains six dots. A blue button labeled "VALIDER" is at the bottom.

On réitère l'opération avec un identifiant et un mot de passe enregistré dans la bdd (ID : Testeur sio et MDP : sio).

A screenshot of the login page with the correct credentials entered. The blue header bar with "R3STO.FR" is at the top. Below it, "Connexion" is centered. The "Identifiant" field contains the text "Testeur sio". The "Mot de passe" field contains three dots. A blue button labeled "VALIDER" is at the bottom.

Le message indiquant une connexion réussie s'affiche.



Puis, nous sommes redirigés automatiquement vers la liste des restaurants, avec l'icône indiquant que l'on est connecté.



En cliquant de nouveau sur l'icône de connexion, on arrive cette fois sur la page dédiée à notre profil.



On voit bien le message personnalisé suivi de nos réservations.

Pour finir, on se déconnecte en cliquant sur le bouton de déconnexion.



Liste des restaurants

Bordeaux l'entrepote

La déconnexion et la redirection s'est déroulé avec succès.