Cette fonctionnalité consiste à afficher les détails du restaurant sélectionné à partir de la liste. J'ai retravaillé à titre personnel sur cette étape afin d'obtenir un meilleur rendu : affichage de l'image du restaurant, application de CSS sur les horaires, ajout de CardView, d'une police d'écriture...

1 – Contrôleur

Comme pour toutes les fonctionnalités, il convenait de développer un contrôleur interagissant avec son layout associé, mais aussi en l'occurrence, les autres activités et la base de donnée au travers d'un fichier PHP.

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

Pour afficher les détails des restaurants, dans un premier temps, il est nécessaire d'ouvrir l'activité des détails et de transmettre les données du restaurant lorsque l'on clique sur un des noms de la liste.

Pour cela, il convient de modifier « MainActivity », il faut ajouter un listener associé à un intent afin de naviguer vers « DetailsRestoActivity » avec des données supplémentaires lors du clique de l'utilisateur.

```
// Configuration du listener pour les clics sur les éléments de la liste
Luke Rioux +1 *
listeRestos.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    Luke Rioux +1 *
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Resto selectedRestaurant = lesResto.get(position);
        // Intent pour naviguer vers DetailsRestoActivity avec des données supplémentaires
        Intent intent = new Intent( packageContext: MainActivity.this, DetailsRestoActivity.class);
        intent.putExtra( name: "idResto", selectedRestaurant.getIdR());
        intent.putExtra( name: "nomResto", selectedRestaurant.getNomR());
        intent.putExtra( name: "villeResto", selectedRestaurant.getVilleR());
        intent.putExtra( name: "numResto", selectedRestaurant.getNumAdrR());
        intent.putExtra( name: "voieResto", selectedRestaurant.getVoieAdrR());
        intent.putExtra( name: "cpResto", selectedRestaurant.getCpR());
        intent.putExtra( name: "descResto", selectedRestaurant.getDescR());
        intent.putExtra( name: "horaireResto", selectedRestaurant.getHoraireR());
        startActivity(intent);
    }
});
```

Ensuite, on récupère les données de l'intent dans le contrôleur correspondant à l'affichage des détails et on les affiche dans les View associés.

```
// Récupérer les données de l'intent
Bundle extras = getIntent().getExtras();

if (extras != null) {
    int idResto = extras.getInt( key: "idResto",  defaultValue: -1);
    String nomResto = extras.getString( key: "nomResto",  defaultValue: "Nom inconnu");
    String villeResto = extras.getString( key: "villeResto",  defaultValue: "Ville inconnue");
    String numResto = extras.getString( key: "numResto",  defaultValue: "Numéro inconnu");
    String voieResto = extras.getString( key: "voieResto",  defaultValue: "Voie inconnue");
    String cpResto = extras.getString( key: "cpResto",  defaultValue: "CP inconnu");
    String descResto = extras.getString( key: "descResto",  defaultValue: "Description non disponible");
    String horaireResto = extras.getString( key: "horaireResto",  defaultValue: "Horaires non disponibles");

    TextView tvNom = findViewById(R.id.TextViewNom);
    TextView tvAdresse = findViewById(R.id.TextViewAdresse);
    TextView tvDesc = findViewById(R.id.TextViewDesc);
    ImageView imageViewPhoto = findViewById(R.id.ImageViewPhoto);
    WebView webViewHoraires = findViewById(R.id.WebViewHoraires);

    tvNom.setText(nomResto);
    tvAdresse.setText(numResto + " " + voieResto + ", " + cpResto + " " + villeResto);
    tvDesc.setText(descResto);
    webViewHoraires.loadDataWithBaseURL( baseUrl: null, horaireResto,  mimeType: "text/html",  encoding: "UTF-8",  historyUrl: null);
```

Pour récupérer l'image du restaurant, cela est un peu plus complexe, en effet, cette action nécessite de passer par une méthode PHP afin de récupérer le nom de la photo à l'aide de l'id du restaurant, ce qui permettra par la suite d'aller chercher les images correspondantes dans le fichier drawable (/home/btssio/AndroidStudioProjects/projet-android/app/src/main/res) en générant le chemin qui pointe vers la photo associée.

```java
// Charger l'image à partir du serveur
OkHttpClient httpClient = new OkHttpClient();
Request requestPhoto = new Request.Builder().url("http://10.15.13.167/android/appResto/getPhotoById.php?idResto=" + idResto).build();
new *
httpClient.newCall(requestPhoto).enqueue(new Callback() {
    new *
    @Override
    public void onFailure(@NonNull Call call, @NonNull IOException e) {
        Log.e( tag: "DetailsRestoActivity",  msg: "Erreur de requête", e);
    }

    new *
    @Override
    public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {
        if (response.isSuccessful()) {
            final String jsonResponse = response.body().string();
            runOnUiThread(() -> {
                try {
                    JSONObject jsonObject = new JSONObject(jsonResponse);
                    String imageName = jsonObject.optString( name: "cheminP",  fallback: "default_image");
                    imageName = imageName.toLowerCase();
                    int imageResId = getResources().getIdentifier(imageName.substring(0, imageName.lastIndexOf( str: ".")),  defType: "drawable", getPackageName());
                    if (imageResId != 0) {
                        imageViewPhoto.setImageResource(imageResId);
                    } else {
                        Log.e( tag: "DetailsRestoActivity",  msg: "Image not found: " + imageName);
                    }
                } catch (JSONException e) {
                    Log.e( tag: "DetailsRestoActivity",  msg: "JSON parsing error", e);
                }
            });
        } else {
            Log.e( tag: "DetailsRestoActivity",  msg: "Server responded with: " + response.code());
        }
    }
});
```

Voici le fichier PHP.

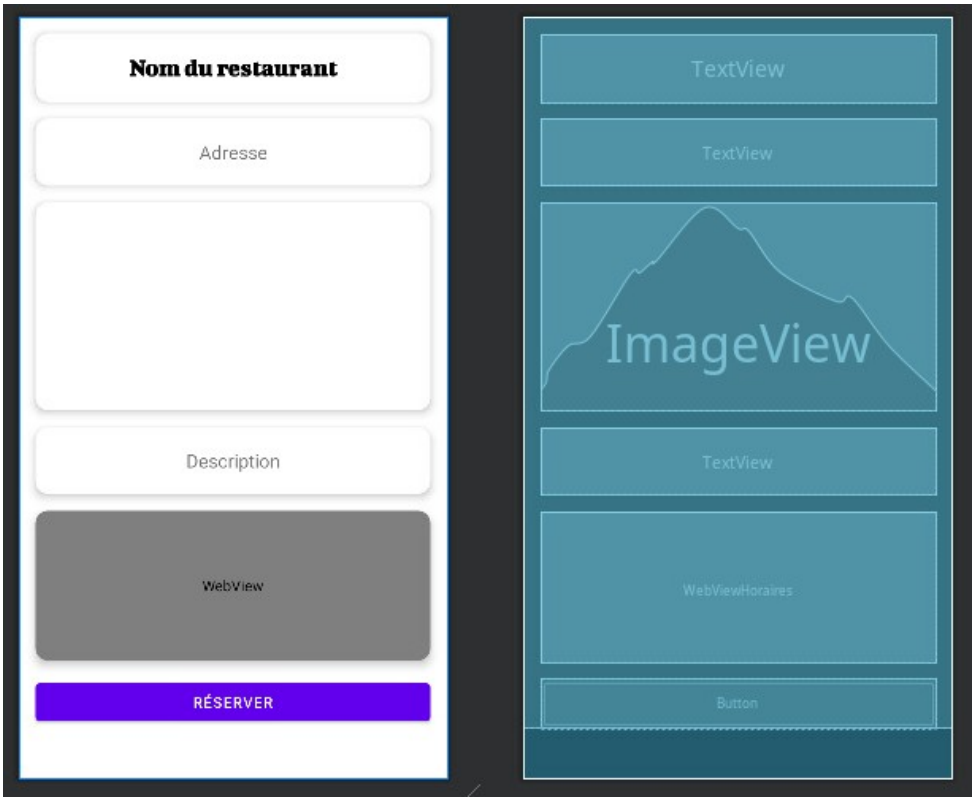
```php
<?php
try {
    // Configuration de la connexion à la base de données
    $db = "resto_test";
    $dbhost = "localhost";
    $dbport = 3306;
    $dbuser = "root";
    $dbpasswd = "joliverie";

    // Création de l'objet PDO pour la connexion à la base de données
    $connexion = new PDO("mysql:host=$dbhost;port=$dbport;dbname=$db;charset=utf8", $dbuser, $dbpasswd);

    // Vérification de la présence de l'ID du restaurant dans les paramètres GET et que c'est un nombre
    if (isset($_GET['idResto']) && is_numeric($_GET['idResto'])) {
        $idResto = (int) $_GET['idResto'];

        // Préparation de la requête SQL pour récupérer le chemin de la photo associée à l'ID du restaurant
        $reponse = $connexion->prepare("SELECT cheminP FROM photo WHERE idR = :idResto LIMIT 1");
        $reponse->bindParam(':idResto', $idResto, PDO::PARAM_INT);
        $reponse->execute();   // Exécution de la requête

        // Récupération du résultat de la requête
        $res = $reponse->fetch(PDO::FETCH_ASSOC);

        // Vérification si une photo est trouvée et renvoi du chemin en format JSON
        if ($res) {
            echo json_encode($res);
        } else {
            // Renvoi d'une erreur si aucune photo n'est trouvée pour l'ID donné
            echo json_encode(['error' => 'No photo found for this restaurant']);
        }
    } else {
        // Renvoi d'une erreur si l'ID du restaurant n'est pas valide ou absent
        echo json_encode(['error' => 'Invalid or missing restaurant ID']);
    }
} catch (Exception $e) {
    // Capture et renvoi de l'erreur si une exception est levée
    die('Erreur : ' . $e->getMessage());
}
?>
```

Référez-vous aux commentaires pour une bonne compréhension du programme.

Afin de respecter les bonnes pratiques et d'assurer une meilleure sécurité et efficacité du programme, on utilise une requête préparée et un système de gestion d'erreurs.

2 – Vue

En toute logique, il faut un layout associé au contrôleur.



Celui-ci contient des champs pour : le titre, l'adresse, l'image, la description, les horaires et le bouton de réservation.

Il faut évidemment des id correspondant à ceux utilisés dans le contrôleur pour que cela fonctionne.

En outre, j'ai ajouté la possibilité de scroll dans le contexte où la description serait longue et empêcherait l'affichage du bouton de réservation par manque de place.

```
<ScrollView 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"
    xmlns:android="http://schemas.android.com/apk/res/android">
```

Et des cardView pour un rendu plus agréable.

```
<androidx.cardview.widget.CardView
    android:id="@+id/cardViewNom"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    app:cardElevation="6dp"
    app:cardCornerRadius="12dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent">
```

3 – Jeu d’essai

Le jeu d’essai est simple, il s’agit de cliquer sur un des restaurants de la liste afin de s’assurer que les détails du restaurant en question s’affiche comme attendu.

Nous obtenons le résultat escompté.