Cette fonctionnalité consiste à afficher une liste des restaurants récupérés dans notre base de données au sein de l'application. J'ai retravaillé à titre personnel sur cette étape afin d'obtenir un meilleur rendu.

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 à la liste des restaurants est « MainActivity » étant donné qu'on peut considérer cette page comme la page d'accueil de l'application.

Décortiquons le code pour mieux entrevoir la logique de développement :

Premièrement, il convient d'importer toutes les ressources nécessaires.

MainActivity.java

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

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.example.projet_android.R;
import com.example.projet_android.modele.metier.Resto;

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

import java.io.IOException;
import java.util.ArrayList;

import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
```

Ensuite, de déclarer des variables pour la vue.

```java
// Déclaration des variables pour la vue
3 usages
ListView listeRestos;
5 usages
ArrayList<Resto> lesResto;
2 usages
ArrayAdapter<Resto> dataAdapter;
```

La procédure onCreate est composée de divers éléments, les commentaires permettent de comprendre avec intelligibilité le déroulement du programme.

```
@Override
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Définir le layout à utiliser pour l'activité
    setContentView(R.layout.activity_main);

    // Configuration de la barre d'action avec un titre
    if (getSupportActionBar() != null) {
        getSupportActionBar().setDisplayShowTitleEnabled(true);
        getSupportActionBar().setTitle("R3STO.FR");
    }

    // Initialisation des composants de la vue
    listeRestos = findViewById(R.id.ListViewResto);
    lesResto = new ArrayList<>();

    // Préparation de la requête HTTP pour récupérer les informations des restaurants
    Request requestResto = new Request.Builder().url("http://10.15.13.167/android/appResto/getAllResto.php").build();
    OkHttpClient httpClient = new OkHttpClient();
```

Un pdf est dédié à la barre d'action de l'application dans le GitHub.

Penchons-nous sur la préparation de la requête : pour que le fichier PHP puisse communiquer avec la bdd, il était nécessaire de le déposer à l'emplacement du serveur web (/var/www/html/android/appResto).
La méthode qui récupère les restaurants afin de les afficher dans une liste est « getAllResto ».

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

$connexion = new PDO('mysql:host='.$dbhost.';port='.$dbport.';dbname='.$db.'', $dbuser, $dbpasswd);
$connexion->exec("SET CHARACTER SET utf8");

        $reponse=$connexion->prepare("SELECT * FROM resto;");
        $reponse->execute();
        $datas = array();

        while($res=$reponse->fetch(PDO::FETCH_ASSOC)) {
            $datas['resto'][]=$res;

        }

      echo json_encode($datas);

}
catch (Exception $e) {
die('Erreur : ' . $e->getMessage());
}

?>
```

Le fichier est assez simple, on se connecte à la bdd, on récupère les données de tous les restaurants avec un SELECT, puis on les stocke dans une array qu'on retournera en une chaîne de caractères formatée en JSON, le tout avec une gestion des erreurs.

Vous trouverez un fichier sur le GitHub contenant la structure de la base de données pour de plus ample informations.

Par la suite, il convient d'exécuter la requête avec une gestion d'erreurs.

```
// Exécution de la requête asynchrone
Simon Michaud +2 *
httpClient.newCall(requestResto).enqueue(new Callback() {
    hriou +1 *
    @Override
    public void onFailure(@NonNull Call call, @NonNull IOException e) {
        // Gestion des erreurs de la requête
        Log.e( tag: "MainActivity",  msg: "Erreur de requête", e);
    }
```

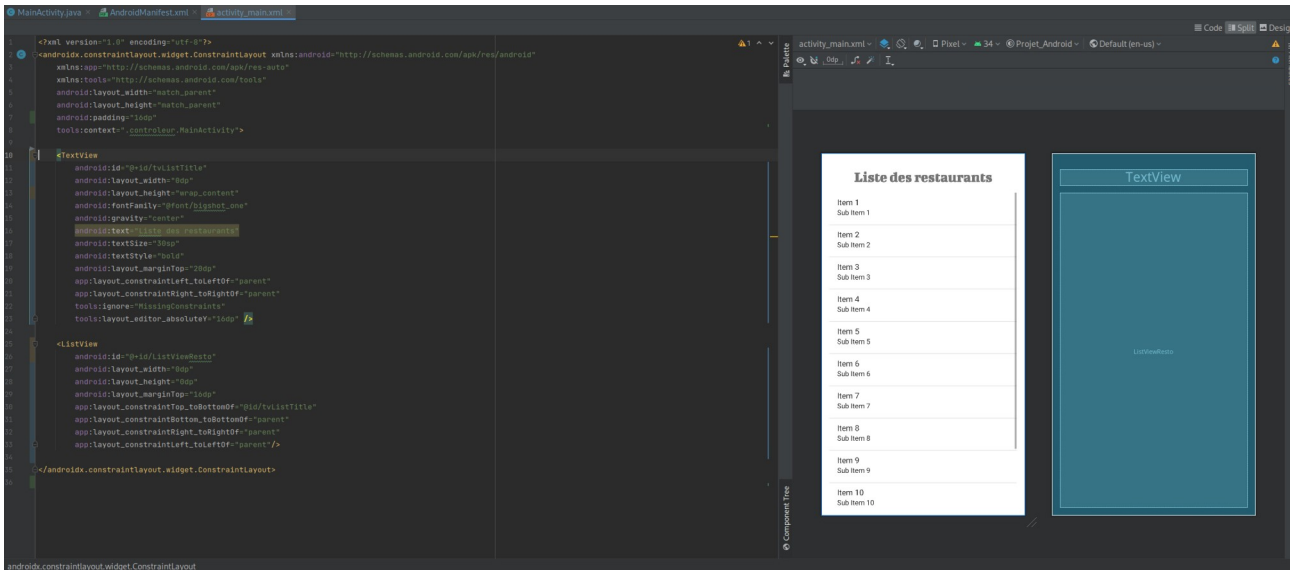
Et en cas de réussite, on traite la réponse en créant un objet restaurant pour chaque restaurant inscrit dans la table, puis on les affiche dans la liste de l'application.
Vous noterez le CSS qui permet un affichage plus esthétique et lisible des horaires dans les détails des restaurants.

```
@Override
public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {
    // Traitement de la réponse
    if (response.isSuccessful()) {
        final String myResponse = response.body().string();
        MainActivity.this.runOnUiThread(() -> {
            try {
                JSONObject jsonObject = new JSONObject(myResponse);
                JSONArray jsonArray = jsonObject.optJSONArray( name: "resto");
                lesResto.clear();
                // Parse et création des objets Resto
                for (int i = 0; i < jsonArray.length(); i++) {
                    JSONObject jsonObjectResto = jsonArray.getJSONObject(i);
                    int idResto = jsonObjectResto.optInt( name: "idR");
                    String nomResto = jsonObjectResto.optString( name: "nomR",  fallback: "N/A");
                    String villeResto = jsonObjectResto.optString( name: "villeR",  fallback: "N/A");
                    String numResto = jsonObjectResto.optString( name: "numAdrR",  fallback: "N/A");
                    String voieResto = jsonObjectResto.optString( name: "voieAdrR",  fallback: "N/A");
                    String cpResto = jsonObjectResto.optString( name: "cpR",  fallback: "N/A");
                    String descResto = jsonObjectResto.optString( name: "descR",  fallback: "N/A");
                    String horaireResto = jsonObjectResto.optString( name: "horairesR",  fallback: "N/A");
                    horaireResto = horaireResto + "<style>\n" +
                            "        table {\n" +
                            "            width: 100%;\n" +
                            "            border-collapse: collapse;\n" +
                            "        }\n" +
                            "        td {\n" +
                            "            padding: 8px;\n" +
                            "            text-align: left;\n" +
                            "            border-bottom: 1px solid #ddd;\n" +
                            "            font-size: 12px;\n" +
                            "        }\n" +
                            "        th {\n" +
                            "            padding: 8px;\n" +
                            "            text-align: left;\n" +
                            "            border-bottom: 1px solid #ddd;\n" +
                            "            background-color: #f2f2f2;\n" +
                            "        }\n" +
                            "    </style>";

                    Resto unResto = new Resto(idResto, nomResto, villeResto, numResto, voieResto, cpResto, descResto, horaireResto);
                    lesResto.add(unResto);
                }
                // Mise à jour de l'adaptateur de la liste
                dataAdapter = new ArrayAdapter<>( context: MainActivity.this, android.R.layout.simple_list_item_1, lesResto);
                listeRestos.setAdapter(dataAdapter);
            } catch (JSONException e) {
                Log.e( tag: "MainActivity",  msg: "Erreur de parsing JSON", e);
            }
        });
    } else {
        Log.e( tag: "MainActivity",  msg: "Réponse non réussie : " + response);
    }
}
```

2 - Vue

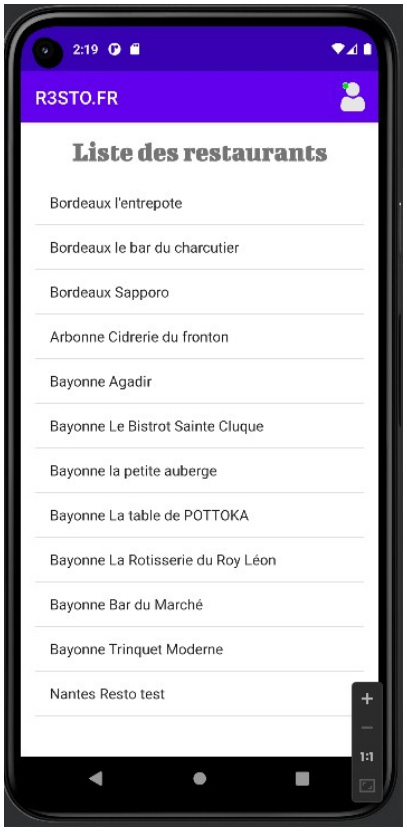
En toute logique, il faut un layout associé au contrôleur.



Celui-ci est basique, il contient un titre, en gras et muni d'une police, et une listView permettant d'afficher les restaurants, avec un scroll intégré. Il faut évidemment des id correspondant à ceux utilisés dans le contrôleur pour que cela fonctionne.

3 – Jeu d'essai

Assurons-nous du bon fonctionnement de notre programme, pour cela rien de plus simple, il suffira de lancer l'application dans l'attente d'obtenir l'affichage de la liste des restaurants.



Comme on peut le voir, tout fonctionne parfaitement. Cette fonctionnalité est maintenant opérationnelle.

Vous trouverez le reste des étapes à la suite de celle-ci au sein du GitHub.