Projet Resto.fr – Itération 1

Affichage de la liste des restaurants

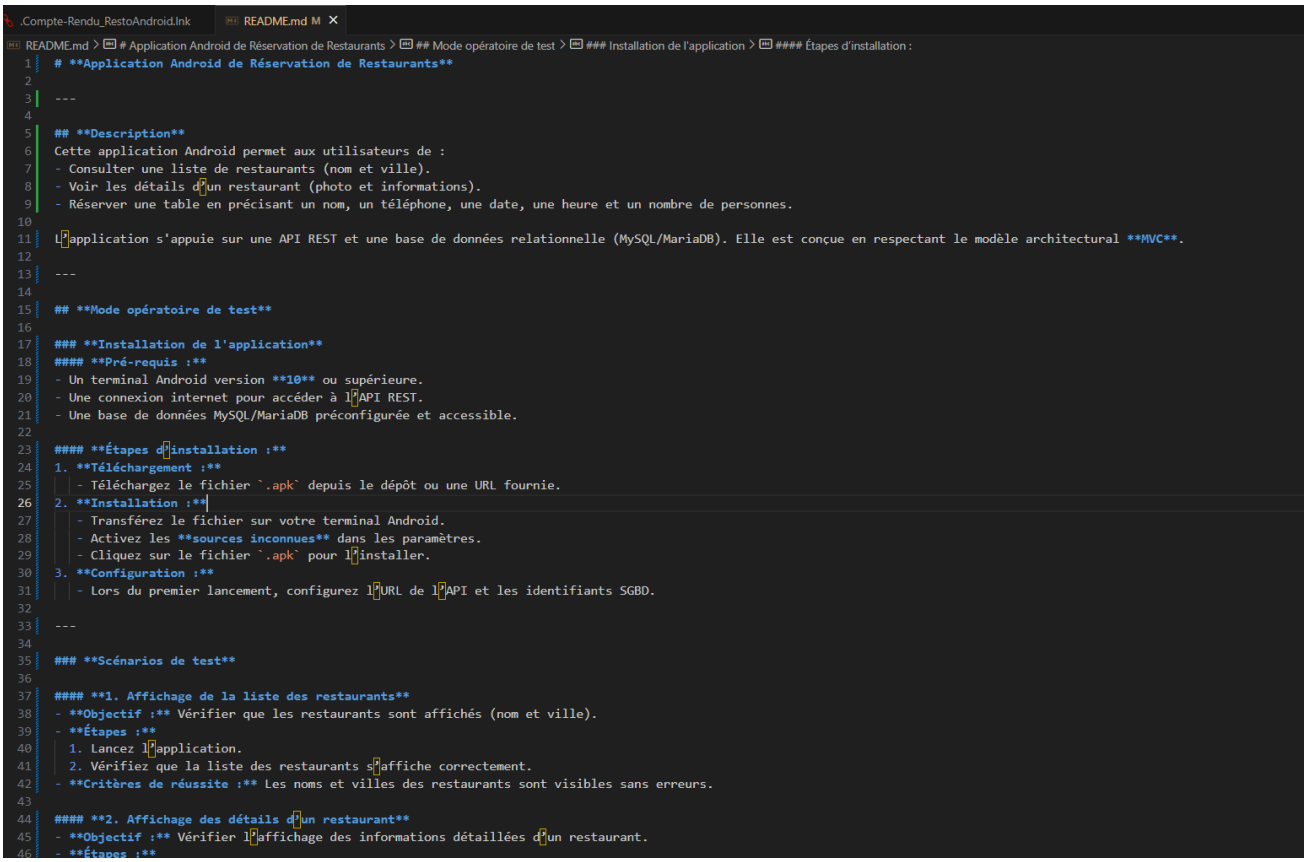
User story : En tant qu'utilisateur, je souhaite pouvoir consulter la liste des restaurants (nom et ville) afin de pouvoir en sélectionner un ensuite.

(Les tickets ne sont pas forcément dans un ordre logique)

TICKET 01 : REMPLISSAGE DU README (MODE OPERATOIRE DE TEST, ...)

Le ticket a pour objectif de rédiger un fichier README professionnel pour l'application Android de réservation de restaurants. Ce fichier décrit en détail le projet, en présentant ses fonctionnalités principales (consultation de restaurants, affichage des détails et réservation), le mode opératoire des tests et la documentation technique (architecture MVC, API REST, base de données).

Voici une capture d'écran du README :

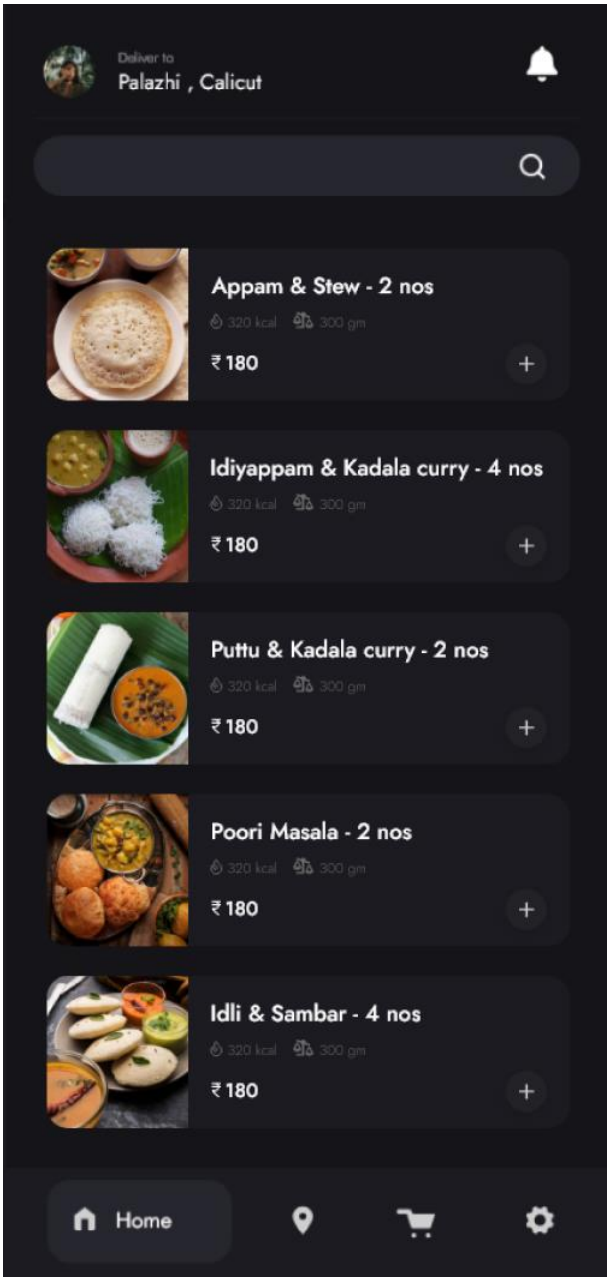


```
.Compte-Rendu_RestoAndroid.lnk   README.md M
README.md > # Application Android de Réservation de Restaurants > ## Mode opératoire de test > ### Installation de l'application > #### Étapes d'installation :
# **Application Android de Réservation de Restaurants**

---

## **Description**
Cette application Android permet aux utilisateurs de :
- Consulter une liste de restaurants (nom et ville).
- Voir les détails d'un restaurant (photo et informations).
- Réserver une table en précisant un nom, un téléphone, une date, une heure et un nombre de personnes.

L'application s'appuie sur une API REST et une base de données relationnelle (MySQL/MariaDB). Elle est conçue en respectant le modèle architectural **MVC**.

---

## **Mode opératoire de test**

### **Installation de l'application**
#### **Pré-requis :**
- Un terminal Android version **10** ou supérieure.
- Une connexion internet pour accéder à l'API REST.
- Une base de données MySQL/MariaDB préconfigurée et accessible.

#### **Étapes d'installation :**
1. **Téléchargement :**
   - Téléchargez le fichier `.apk` depuis le dépôt ou une URL fournie.
2. **Installation :**
   - Transférez le fichier sur votre terminal Android.
   - Activez les **sources inconnues** dans les paramètres.
   - Cliquez sur le fichier `.apk` pour l'installer.
3. **Configuration :**
   - Lors du premier lancement, configurez l'URL de l'API et les identifiants SGBD.

---

### **Scénarios de test**

#### **1. Affichage de la liste des restaurants**
- **Objectif :** Vérifier que les restaurants sont affichés (nom et ville).
- **Étapes :**
  1. Lancez l'application.
  2. Vérifiez que la liste des restaurants s'affiche correctement.
- **Critères de réussite :** Les noms et villes des restaurants sont visibles sans erreurs.

#### **2. Affichage des détails d'un restaurant**
- **Objectif :** Vérifier l'affichage des informations détaillées d'un restaurant.
- **Étapes :**
```

TICKET 03 : SCHEMATISATION DES INTERFACES

Dans ce ticket, je réalise une maquette de l'interface dédiée à la liste des restaurants, conformément aux exigences de l'itération 1. Cette maquette illustre la disposition des éléments clés tels que les menus, le profil, en respectant une certaine ergonomie et clarté visuelle. Elle sert de guide pour le développement de l'application.

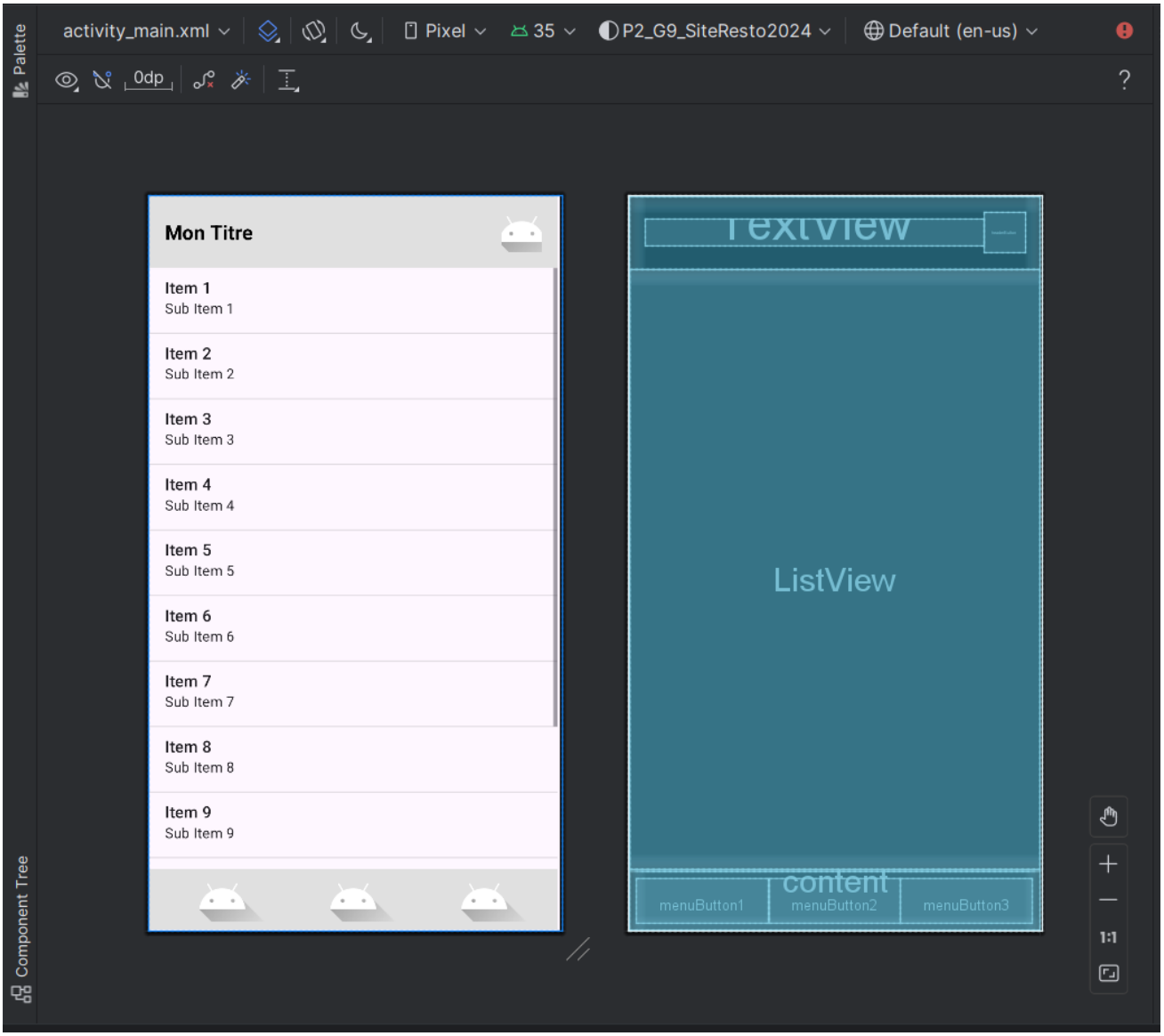
Voici la maquette de l'interface dédiée à la liste des restaurants :



TICKET 04 : CREATION DE L'INTERFACE --> LISTE DES RESTAURANTS

Dans ce ticket, je réalise une maquette de l'interface dédiée à la liste des restaurants, conformément aux exigences de l'itération 1 et de la maquette (ticket 03). Pour cela j'ai dû modifier le fichier XML : "activity_main.xml". J'ai divisé l'écran en 3 : un header / une section avec la listeView / un footer avec les boutons de navigation

Voici l'interface dédiée à la liste des restaurants :



Et voici une partie du code de “activity_main.xml” :

activity_main.xml

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schema
    <LinearLayout

        <RelativeLayout
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <FrameLayout
                android:id="@+id/content"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_above="@+id/bottomMenu" />
            <LinearLayout
                android:id="@+id/bottomMenu"
                android:layout_width="match_parent"
                android:layout_height="60dp"
                android:layout_alignParentBottom="true"
                android:orientation="horizontal"
                android:gravity="center"
                android:background="#E0E0E0"
                android:padding="8dp">
                <ImageButton
                    android:id="@+id/menuButton1"
                    android:layout_width="0dp"
                    android:layout_height="match_parent"
                    android:layout_weight="1"
                    android:src="@drawable/ic_launcher_foreground"
                    android:background="@null"
                    android:contentDescription="Menu 1" />
                <ImageButton
                    android:id="@+id/menuButton2"
                    android:layout_width="0dp"
                    android:layout_height="match_parent"
                    android:layout_weight="1"
                    android:src="@drawable/ic_launcher_foreground"
                    android:background="@null"
                    android:contentDescription="Menu 2" />
                <ImageButton
                    android:id="@+id/menuButton3"
                    android:layout_width="0dp"
                    android:layout_height="match_parent"
                    android:layout_weight="1"
                    android:src="@drawable/ic_launcher_foreground"
                    android:background="@null"
                    android:contentDescription="Menu 3" />

            </LinearLayout>
        </RelativeLayout>
```

TICKET 05 : MISE EN PLACE DE LA BDD ET DU DAO

Dans ce Ticket le but Ajout des fichier .sql dans le projet Créé le script SQL pour faire la connexion à la bdd Externe Créé le modèle DAO en PHP qu'on utilisera sous Android.

Ajout du SQL dans le projet

	Table	Action							Lignes	Type	Interclassement	Taille	Perte
☐	aimer	☆	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	16	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
☐	critiquer	☆	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	19	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
☐	photo	☆	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	14	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
☐	resto	☆	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	11	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
☐	utilisateur	☆	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	7	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
	5 tables	Somme							67	InnoDB	utf8mb4_unicode_ci	144,0 kio	0 o

Création du script SQL pour la connexion a la BDD

C: > Users > pierr > OneDrive > Bureau > Resto > connexion.php

```php
<?php
$host = 'localhost';
$dbname = 'resto2';
$username = 'resto_util';
$password = 'secret';

try {
    $pdo = new PDO(dsn: "mysql:host=$host;dbname=$dbname", username: $username, password: $password);
    $pdo->setAttribute(attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo "Erreur de connexion : " . $e->getMessage();
    die();
}
?>
```

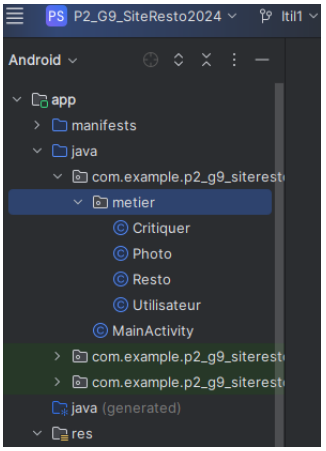
et la récupération des informations dans la bdd pour Android studio

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

require 'connexion.php';

try {
    $stmt = $pdo->query("SELECT idR, nomR, numAdrR, voieAdrR, cpR, villeR, latitudeDegR, longitudeDegR, descR, horairesR FROM resto");
    $lacs = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($lacs);
} catch (PDOException $e) {
    echo json_encode(['error' => 'Erreur de récupération des données : ' . $e->getMessage()]);
}
?>
```

TICKET 06 : MODELE METIER

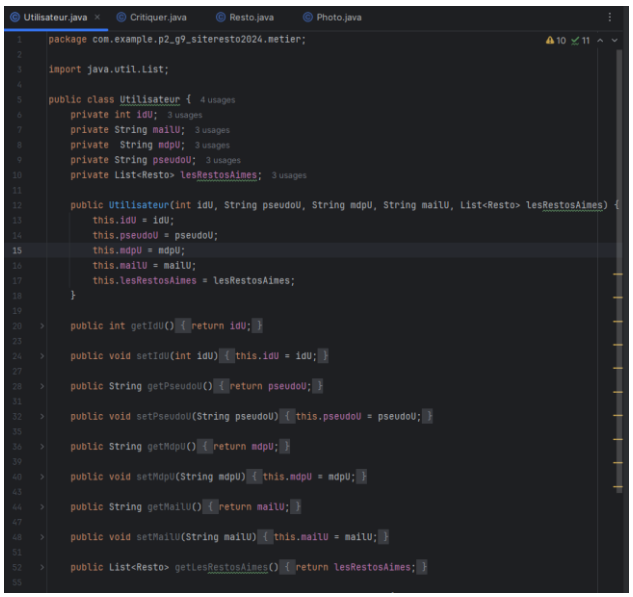
Dans ce ticket, le but était de créer tout le modèle métier de notre app avec ses classes complétées.

Pour cela il fallait savoir dans un premier temps lequel allait être utile, nous avions donc besoin d'uniquement celle avec des informations contenues dedans, soit la classe resto, utilisateur, critique et photo. La classe aimer est une simple table composite donc inutile, elle ne contient aucune information.

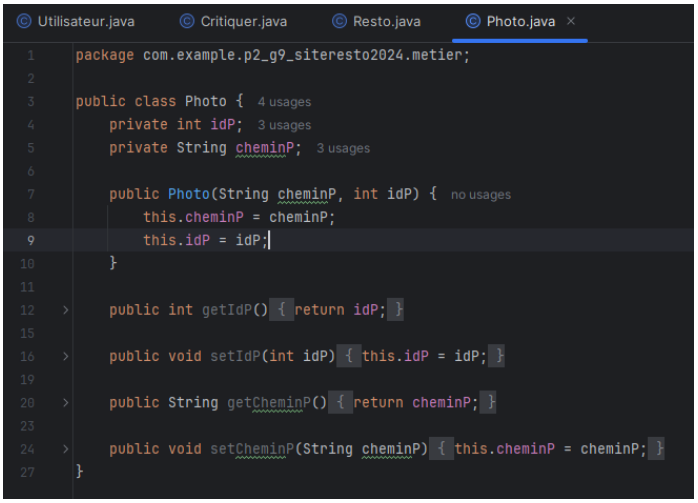


Tout d'abord il faut que l'on crée le package metier :

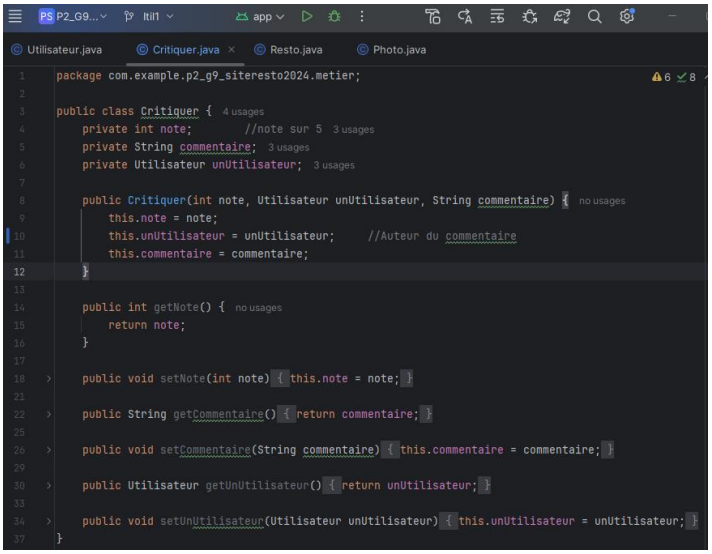
Ensuite j'ai commencé à crée la classe Photo :



J'ai ensuite poursuivi sur la classe Utilisateur qui n'a pas aussi besoin d'utiliser les autres class java :

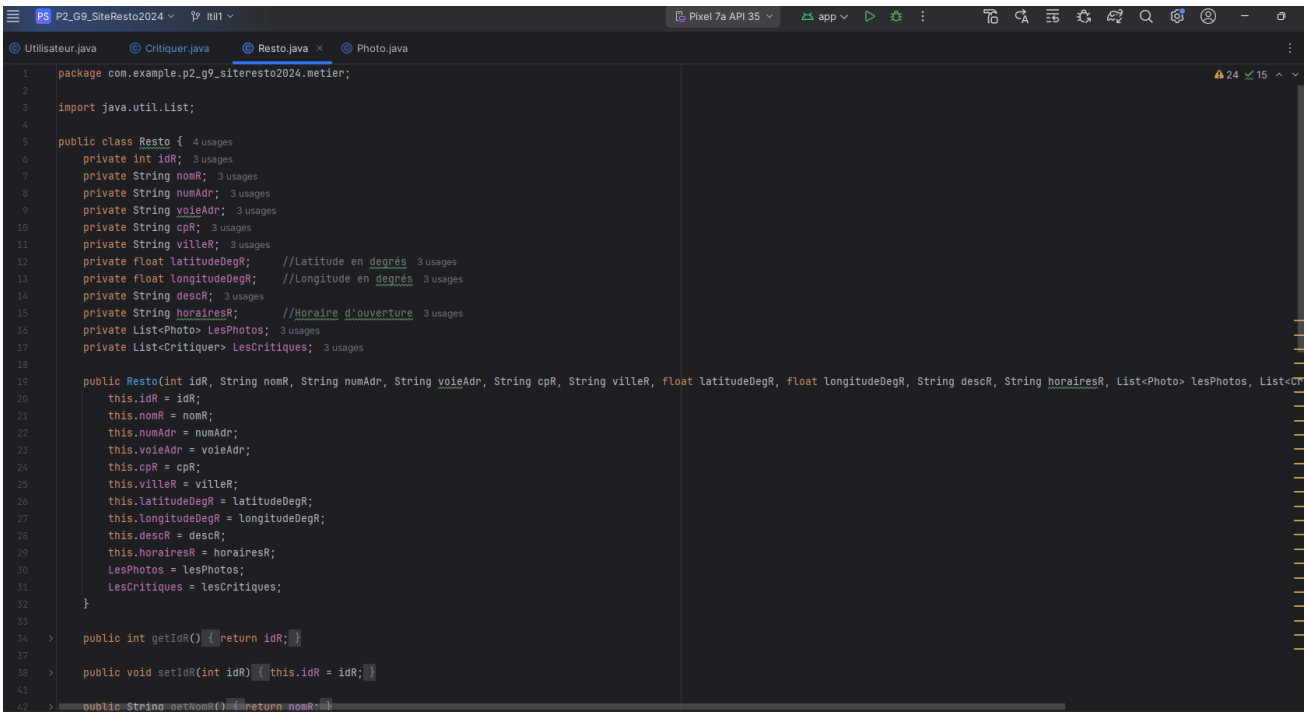


Après j'ai ajouté la classe Critiquer qui elle utilise un utilisateur dedans pour définir l'auteur du commentaire :



```java
package com.example.p2_g9_siteresto2024.metier;

public class Critiquer {
    private int note;        //note sur 5
    private String commentaire;
    private Utilisateur unUtilisateur;

    public Critiquer(int note, Utilisateur unUtilisateur, String commentaire) {
        this.note = note;
        this.unUtilisateur = unUtilisateur;     //Auteur du commentaire
        this.commentaire = commentaire;
    }

    public int getNote() {
        return note;
    }

    public void setNote(int note) { this.note = note; }

    public String getCommentaire() { return commentaire; }

    public void setCommentaire(String commentaire) { this.commentaire = commentaire; }

    public Utilisateur getUnUtilisateur() { return unUtilisateur; }

    public void setUnUtilisateur(Utilisateur unUtilisateur) { this.unUtilisateur = unUtilisateur; }
}
```

Enfin j'ai ajouté la table principal resto à la fin, car celle-ci utilise deux listes (class) une liste de photo du restaurant et la liste des critiques du restaurant :



```java
package com.example.p2_g9_siteresto2024.metier;

import java.util.List;

public class Resto {
    private int idR;
    private String nomR;
    private String numAdr;
    private String voieAdr;
    private String cpR;
    private String villeR;
    private float latitudeDegR;     //Latitude en degrés
    private float longitudeDegR;    //Longitude en degrés
    private String descR;
    private String horairesR;       //Horaire d'ouverture
    private List<Photo> LesPhotos;
    private List<Critiquer> LesCritiques;

    public Resto(int idR, String nomR, String numAdr, String voieAdr, String cpR, String villeR, float latitudeDegR, float longitudeDegR, String descR, String horairesR, List<Photo> lesPhotos, List<C
        this.idR = idR;
        this.nomR = nomR;
        this.numAdr = numAdr;
        this.voieAdr = voieAdr;
        this.cpR = cpR;
        this.villeR = villeR;
        this.latitudeDegR = latitudeDegR;
        this.longitudeDegR = longitudeDegR;
        this.descR = descR;
        this.horairesR = horairesR;
        LesPhotos = lesPhotos;
        LesCritiques = lesCritiques;
    }

    public int getIdR() { return idR; }

    public void setIdR(int idR) { this.idR = idR; }

    public String getNomR() { return nomR; }
```

Bien évidemment sur chaque class, tous les construct ont été défini, ainsi que tous les getter et setter aucune autre méthode n'a été défini à part celle-ci.

TICKET 07 : CONSTRUCTEUR --> LISTE DES RESTAURANT

Dans ce ticket, le but était de modifier le constructeur de notre interface qui affiche la liste des restaurants.
Pour cela, il a fallu autoriser la connexion à internet et créer le client http pour que l'on puisse accéder à notre bdd externe. Qui récupère les informations des restaurants à l'aide de la fonction DAO et les mettre dans des objet métier resto.

Tout d'abord la première chose que j'ai donc faite, c'est toute la partie client http et autoriser la connexion. Pour le client http, il fallait implémenter okhttp dans la librairy avec la version :

app/build.gradle.kts

```
@@ -37,6 +37,7 @@ dependencies {
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
+   implementation (libs.okhttp)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
```

gradle/libs.versions.toml +1 −0 View file @ ddddd7ce

```
@@ -16,6 +16,7 @@ appcompat = { group = "androidx.appcompat", name = "appcompat", version.ref = "a
material = { group = "com.google.android.material", name = "material", version.ref = "material" }
activity = { group = "androidx.activity", name = "activity", version.ref = "activity" }
constraintlayout = { group = "androidx.constraintlayout", name = "constraintlayout", version.ref = "constraintlayout" }
+okhttp = "com.squareup.okhttp3:okhttp:4.10.0"

[plugins]
android-application = { id = "com.android.application", version.ref = "agp" }
```

Ensuite il fallait que j'autorise notre app à accéder à Internet :

app/src/main/AndroidManifest.xml +4 −0 View file @ ddddd7ce

```
<?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" />
+   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
+   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application
        android:allowBackup="true"
@@ -11,6 +14,7 @@
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.P2_G9_SiteResto2024"
+       android:usesCleartextTraffic="true"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
```

Ensuite, je me suis attaqué à la MainActivity.java qui correspond au constructeur de notre layout, j'ai d'abord commencé à faire les imports qui allait me servir :

app/src/main/java/com/example/p2_g9_siteresto2024/MainActivity.java +23 −0 View file @ ddddd7ce

```
package com.example.p2_g9_siteresto2024;

import android.os.Bundle;
+import android.util.Log;
+import android.widget.ArrayAdapter;
+import android.widget.ListView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
@@ -8,6 +11,21 @@ import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

+import com.example.p2_g9_siteresto2024.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;
+
```

Et par la suite, j'ai récupéré le listView du layout pour afficher les restaurants :

```
// Référencement de la ListView pour afficher la liste des restaurants
ListView listViewResto = (ListView) findViewById(R.id.listViewResto);
ArrayList<Resto> lesResto = new ArrayList<>();
```

Après j'ai envoyé ma requête pour récupérer les restaurants :

```
// Création de la requête HTTP vers le fichier PHP (Voir DAO pour les détails)
Request requestClients = new Request.Builder().url("http://10.15.13.176/Resto/get_restos.php").build();
OkHttpClient httpclient = new OkHttpClient();

// Envoi de la requête HTTP en arrière-plan
httpclient.newCall(requestClients).enqueue(new Callback() {
```

Puis j'ai traité la réponse ou non, en la parcourant dans un tableau pour placé les resto un par un dans objet resto qui sont ajouté dans une ArrayList

MainActivity.java

```
30      public class MainActivity extends AppCompatActivity {
33          protected void onCreate(Bundle savedInstanceState) {
54              httpclient.newCall(requestClients).enqueue(new Callback() {
55                  @Override  4 usages
56                  public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {
57                      // Traitement de la réponse HTTP reçue
58                      final String myResponse = response.body().string();
59                      MainActivity.this.runOnUiThread(() -> {
60                          try {
61                              // Conversion de la réponse en un objet JSON principal
62                              JSONObject jsonObjectlesclients = new JSONObject(myResponse);
63
64                              // Extraction du tableau JSON contenant les restaurants
65                              JSONArray jsonArray = jsonObjectlesclients.optJSONArray( name: "clients");
66
67                              // Nettoyage de la liste existante avant d'ajouter les nouveaux éléments
68                              lesResto.clear();
69
70                              // Parcours du tableau JSON pour transformer chaque objet en instance de Resto
71                              for (int i = 0; i < jsonArray.length(); i++) {
72                                  JSONObject jsonObject = jsonArray.getJSONObject(i);
73
74                                  // Récupération des champs nécessaires pour créer un objet Resto
75                                  int idR = jsonObject.getInt( name: "idR");
76                                  String nomR = jsonObject.getString( name: "nomR");
77                                  String numAdr = jsonObject.getString( name: "numAdr");
78                                  String voieAdr = jsonObject.getString( name: "voieAdr");
79                                  String cpR = jsonObject.getString( name: "cpR");
80                                  String villeR = jsonObject.getString( name: "villeR");
81                                  float latitudeDegR = (float) jsonObject.getDouble( name: "latitudeDegR");
82                                  float longitudeDegR = (float) jsonObject.getDouble( name: "longitudeDegR");
83                                  String descR = jsonObject.getString( name: "descR");
84                                  String horairesR = jsonObject.getString( name: "horairesR");
85
86                                  // Création de l'objet Resto et ajout à la liste
87                                  Resto unResto = new Resto(idR, nomR, numAdr, voieAdr, cpR, villeR, latitudeDegR, longitudeDegR, descR, horairesR);
88                                  lesResto.add(unResto);
89                              }
```

jet_SiteRestoAndroid › app › src › main › java › com › example › p2_g9_siteresto2024 › MainActivity › onCreate

À savoir que j'ai temporairement peut-être, créer un construct resto qui n'avais pas besoin de photo et des critiques car pour l'instant ça m'était inutile dans mon affichage.

```
public Resto(int idR, String nomR, String numAdr, String voieAdr, String cpR, String villeR, float latitudeDegR, float longitudeDegR, String descR, String horairesR) {
    this.idR = idR;
    this.nomR = nomR;
    this.numAdr = numAdr;
    this.voieAdr = voieAdr;
    this.cpR = cpR;
    this.villeR = villeR;
    this.latitudeDegR = latitudeDegR;
    this.longitudeDegR = longitudeDegR;
    this.descR = descR;
    this.horairesR = horairesR;
}
```

Pour finir j'ai simplement mis à jour la listview dans le layout et j'ai également préparer l'app en cas d'erreur sur la requête http :

```
                    // Mise à jour de l'adaptateur pour afficher les restaurants dans la ListView
                    ArrayAdapter<Resto> dataAdapter = new ArrayAdapter<>( context: MainActivity.this, android.R.layout.simple_list_item_1, lesResto);
                    listViewResto.setAdapter(dataAdapter);

                } catch (JSONException e) {
                    // Gestion des erreurs liées au traitement du JSON
                    Log.e( tag: "JSONError", msg: "Erreur lors de l'analyse du JSON", e);
                }
            });
        }

        @Override
        public void onFailure(@NonNull Call call, @NonNull IOException e) {
            // Gestion des erreurs de connexion ou d'exécution de la requête HTTP
            e.printStackTrace();
            Log.e( tag: "HTTPRequestError", msg: "Erreur lors de la requête vers get_restos.php", e); // Voir DAO
        }
    });
}
}
```

RESULTAT :

Nous rencontrons actuellement plusieurs problèmes avec Android Studio et GitLab, ce qui nous empêche de tester correctement le programme :

- **De mon côté (Thibault)** : Lorsque je lance le programme, j'obtiens une page blanche sans aucun message d'erreur, ce qui complique le diagnostic.
- **Pour Pierre** : Lorsqu'il clone le projet, celui-ci n'apparaît pas dans ses documents.
- **Pour Thomas** : Le "Run" d'Android Studio ne fonctionne plus depuis un certain temps. Même après avoir réinstallé l'application, le problème persiste.

Nous espérons que le programme fonctionnera si vous parvenez à le tester.
Quelques points importants à noter pour le test :

- Assurez-vous d'avoir bien configuré la base de données dans votre environnement MySQL. Les scripts nécessaires se trouvent dans le dossier **SQL**.
- Copiez également le dossier **dao_resto** dans le répertoire **xampp/htdocs** (ou **wamp/www**, selon votre environnement).