Projet Resto.fr – Itération 2

Affichage du détail d'un restaurant

User story : En tant qu'utilisateur, je souhaite, à partir de la liste des restaurants, pouvoir consulter la fiche détaillée d'un restaurant.

DAILY SCRUM DEBUT DE L'IT2 (18/11/2024) :

GitLab : https://gitlab.com/KKG_Ammbussh/p2_g9_siteresto2024.git

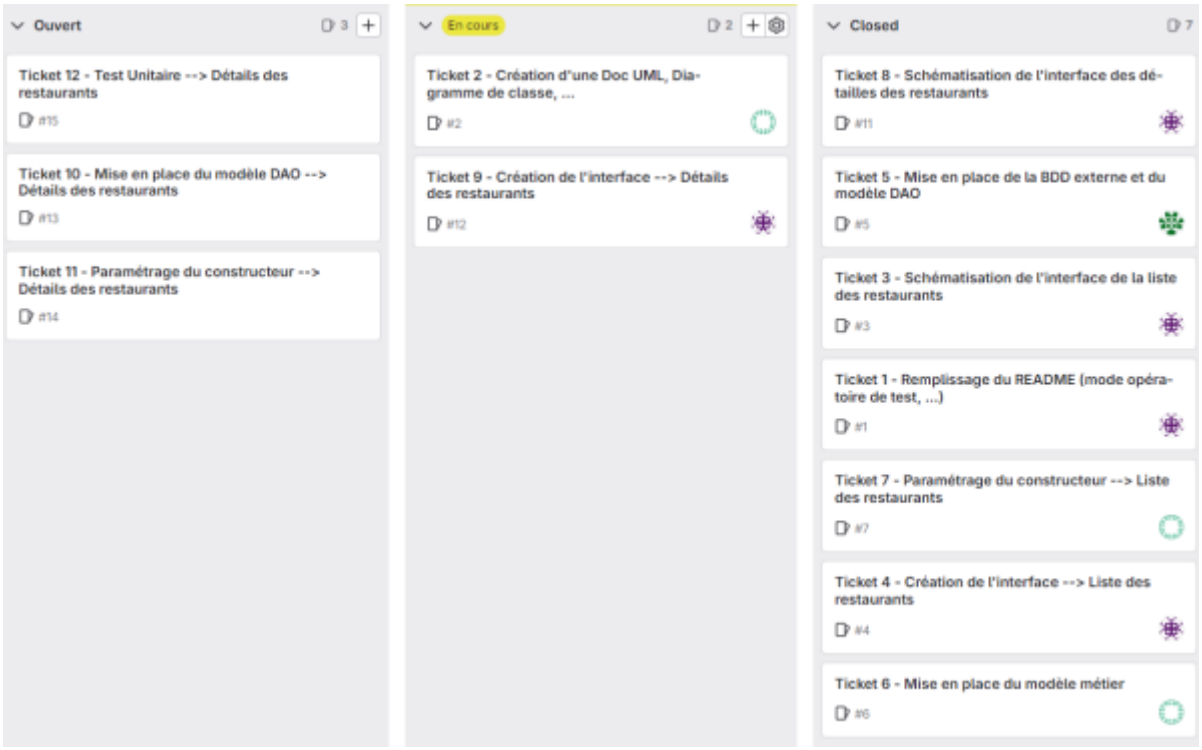
- La branche correspondante à l'itération sur le dépôt est celle avec son numéro, si celle-ci n'est pas présente alors elle correspond au MAIN
- Le README possède normalement le mode opératoire de test

Pendant la réunion :

➔ Attribution des tickets de l'IT2
➔ Création des Ticket de l'IT2

Répartition des tâches du début :
Thibault →Tâche 10, 12
Pierre → Tâche 2, 11
Thomas → Tâche 8, 9

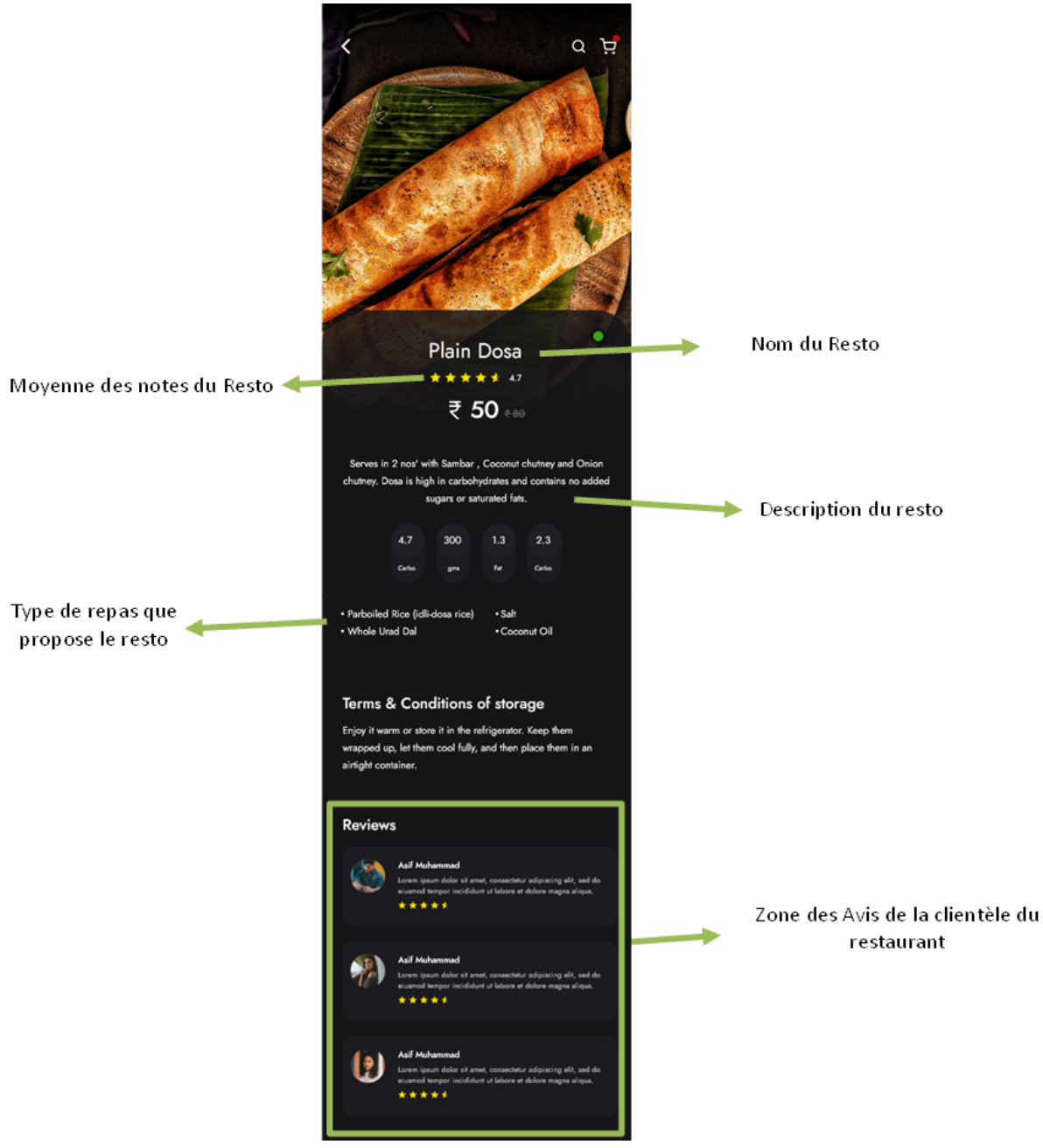
Tableau de Kanban du début :



TÂCHE 08 : SCHEMATISATION DE L'INTERFACE --> DETAIL RESTAURANT

Dans ce ticket, je réalise la deuxième maquette d'interface, dédiée aux détails des restaurants. Cette maquette nous permet de réfléchir à la disposition des boutons et éléments utile pour la suite de l'application. Cette interface doit être claire et épurer pour faciliter la lecture de l'utilisateur.

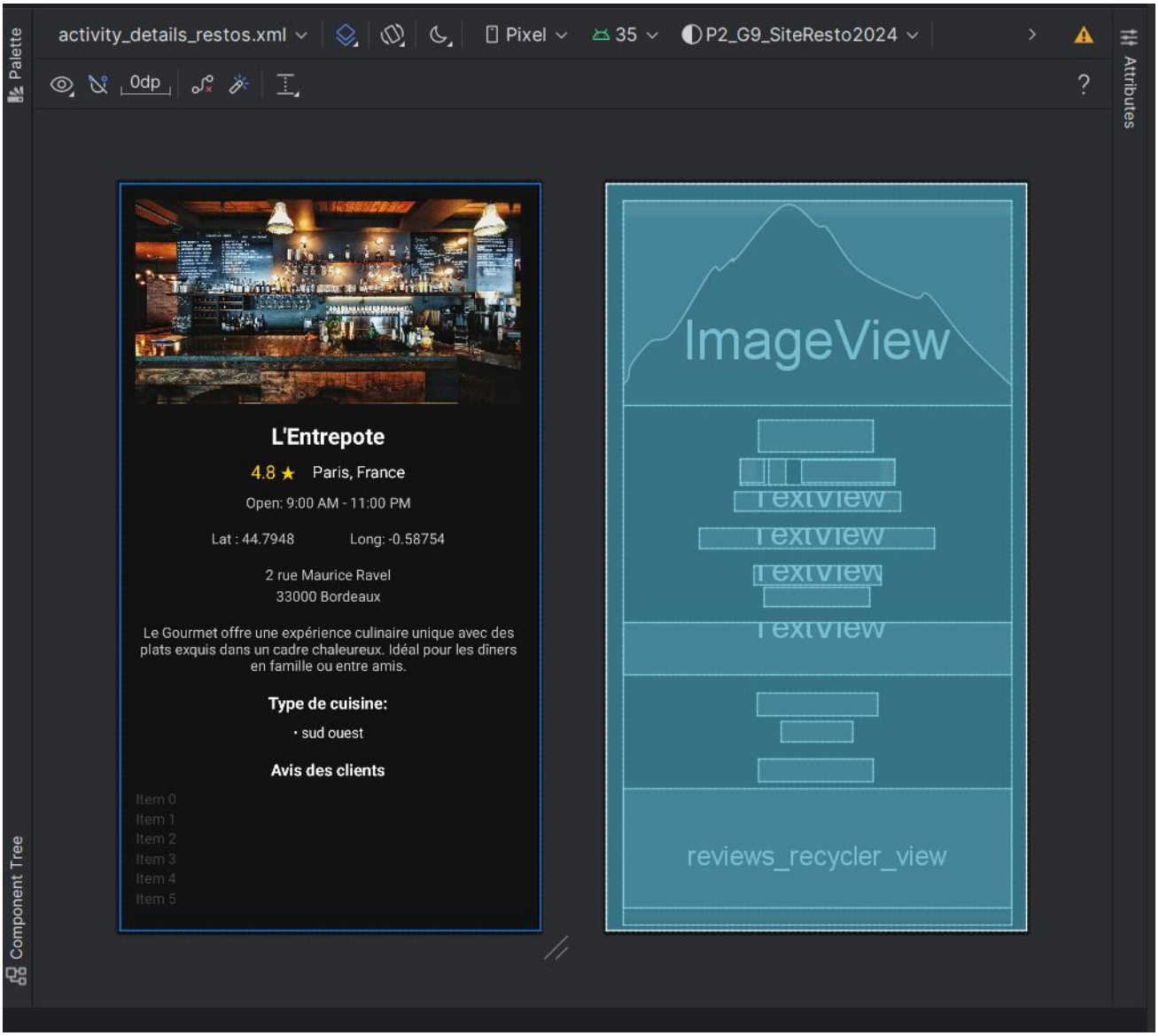
Voici la maquette de l'interface dédiée à la liste des restaurants :



TÂCHE 09 : CREATION DE L'INTERFACE --> DETAILS DES RESTAURANTS

Dans ce ticket, je réalise une maquette de l'interface dédiée aux détails des restaurants, conformément aux informations que nous avons à disposition (présente dans la bdd). Pour cela j'ai dû créer une Java Class intitulé "details_restos", qui est lié à un nouveau layout ("activity_details_restos.xml"). J'ai voulu réaliser un layout un peux plus travailler graphique afin de réaliser un exemple de design. Cette page a juste pour but d'afficher les informations présente dans la bdd, elle ne présente donc pas ou peu de fonctionnalités.

Voici l'interface dédiée aux détails des restaurants :



TÂCHE 10 : MODELE DAO --> DETAILS DES RESTAURANTS

Dans ce ticket le but est d'ajouter une méthode à notre API pour pouvoir afficher toutes les informations d'un restaurant précis, quand on en sélectionnera 1 sur la liste restaurant.
Pour cela, j'ai dû créer ce fichier nommé getAllById.php, et faire en sorte qu'il se connecte à la base de données et me stock toutes les informations du restaurant souhaité dans un fichier json.
On verra aussi que j'ai modifié l'ancien fichier getAllResto de notre API pour qu'il me récupère les photos des restos dans le cas, où l'on souhaiterait afficher leurs photos sur la liste.

Tout d'abord, la première chose a été de récupérer le fichier connexion.php faites dans l'IT précédente et de l'appelé dans notre méthode :

connexion.php

```php
<?php
$db="resto2";
$dbhost="localhost";
$dbport=3306;
$dbuser="resto_util";
$dbpasswd="secret";

try {
    $connexion = new PDO(dsn: 'mysql:host='.$dbhost.';port='.$dbport.';dbname='.$db.'', username: $dbuser, password: $dbpasswd);
    $connexion->exec(statement: "SET CHARACTER SET utf8");
} catch (PDOException $e) {
    echo "Erreur de connexion : " . $e->getMessage();
    die();
}
?>
```

getAllResto.php

```php
<?php
require 'connexion.php';
```

Ensuite, on créer notre requête SQL et on la prépare à la chercher les informations qu'il nous faut, ATTENTION il ne faut pas oublier de récupérer l'ID.

```php
try {
    // Vérification que la donnée est fournie
    if (isset($_POST['idR'])) {
        $idR = $_POST['idR'];
        $reponse=$connexion->prepare(query: "SELECT * FROM resto r INNER JOIN photo p ON r.idR = p.idR WHERE r.idR = :idR;");
        // Préparer la requête SQL d
        $reponse->bindParam(param: ':idR', var: &$idR);
```

Après il nous faut exécuter la requête, et placer les données qu'on récupère dans une liste qu'on transforme en json par la suite :

```php
        // Exécuter la requête
        if ($reponse->execute()) {
            $data = ["resto " . $idR => $reponse->fetch(mode: PDO::FETCH_ASSOC)];
            echo json_encode(value: $data);

            echo json_encode(value: ['success' => 'Les informations ont été récupérer']);
        } else {
            echo json_encode(value: ['error' => 'Erreur lors de la sélection de données']);
        }
    } else {
        echo json_encode(value: ['error' => 'Paramètres manquants.']);
    }
} catch (PDOException $e) {
    echo json_encode(value: ["error" => $e->getMessage()]);
}
?>
```

Et normalement la méthode est prête à l'emploi.

Dans la méthode getAllByID, on peut retrouver un Inner Join dans la requête, qui nous sert à récupérer les photos pour notre interface, c'est une modif que l'on à également fait dans le getAllResto pour qu'on puisse probablement retrouver les photos des restos dans l'interface liste restaurant :

```php
try {
    $reponse=$connexion->prepare(query: "SELECT * FROM resto r INNER JOIN photo p ON r.idR = p.idR;");
    $reponse->execute();
```

TÂCHE 11 : PARAMETRAGE DU CONSTRUCTEUR --> DETAILS DES RESTAURANTS

Dans ce ticket j'ai dû paramétrer le. constructeur de détail_restaurant.java pour qu'il attribue à chacune de ses balises les bonne informations, qui sont récupérer grâce à un intent généré dans la liste_resto.java. Grâce à ceci quand on cliquera sur un resto dans la liste on sera rediriger vers sa page d'infos avec les bonne informations lui appartenant. Voici comment j'ai fait :

J'ai tout d'abord modifié la partie de lecture du tableau JSON retourné par l'API, pour qu'il me stockent toutes les information importante et plus uniquement le nom et la ville :

```
for (int i = 0; i < jsonArray.length(); i++) {
    JSONObject jsonObject = jsonArray.getJSONObject(i);

    // Récupération des champs nécessaires pour créer un objet Resto
    int idR = jsonObject.getInt( name: "idR");
    String nomR = jsonObject.getString( name: "nomR");
    String numAdrR = jsonObject.getString( name: "numAdrR");
    String voieAdrR = jsonObject.getString( name: "voieAdrR");
    String cpR = jsonObject.getString( name: "cpR");
    String villeR = jsonObject.getString( name: "villeR");
    float latitudeDegR = (float) jsonObject.optDouble( name: "latitudeDegR",  fallback: 0.0); // Conversion en float
    float longitudeDegR = (float) jsonObject.optDouble( name: "longitudeDegR",  fallback: 0.0); // Conversion en float
    String descR = jsonObject.optString( name: "descR",  fallback: "");
    String horairesR = jsonObject.optString( name: "horairesR",  fallback: "");
    //int idP = jsonObject.getInt("idP");
    //String cheminP = jsonObject.getString("cheminP");

    // Création de l'objet Resto et ajout à la liste
    Resto unResto = new Resto(idR, nomR, numAdrR, voieAdrR, cpR, villeR, latitudeDegR, longitudeDegR, descR, horairesR
    //Photo unePhoto = new Photo(cheminP, idP);
    lesResto.add(unResto);
}
```

Ensuite j'ai créé un contrôleur qui réagit quand on clique sur l'un des élément de la liste, à ce moment-là il crée un Intent et stockent les valeurs de l'élément cliqué dedans et il nous transporte vers sa page détail et transfert avec lui le Intent.

```
115         // Gestion du clic sur un élément de la liste
116         listViewResto.setOnItemClickListener((parent, view, position, id) -> {
117             Resto selectedResto = lesResto.get(position);
118
119             // Création de l'Intent pour passer à l'activité de détail
120             Intent intent = new Intent( packageContext: liste_restos.this, detail_restos.class);
121             intent.putExtra( name: "idR", selectedResto.getIdR());
122             intent.putExtra( name: "nomR", selectedResto.getNomR());
123             intent.putExtra( name: "numAdrR", selectedResto.getNumAdr());
124             intent.putExtra( name: "voieAdrR", selectedResto.getVoieAdr());
125             intent.putExtra( name: "cpR", selectedResto.getCpR());
126             intent.putExtra( name: "villeR", selectedResto.getVilleR());
127             intent.putExtra( name: "latitudeDegR", selectedResto.getLatitudeDegR());
128             intent.putExtra( name: "longitudeDegR", selectedResto.getLongitudeDegR());
129             intent.putExtra( name: "descR", selectedResto.getDescR());
130             intent.putExtra( name: "horairesR", selectedResto.getHorairesR());
131             //intent.putExtra("PhotoR", selectedResto.getLesPhotos();
132             startActivity(intent);
133         });
```

Sur la partie detail_resto.java, on récupère simplement les vues de l'interface detail_resto.xml et l'ont récupère les valeurs du Intent récupéré précédemment. On les met bien évidement dans de nouvelle variable :

```
// Récupération des vues avec un ID
//ImageView imageResto = (ImageView) findViewById(R.id.image_resto);
TextView textViewNomResto = (TextView) findViewById(R.id.nom_resto);
TextView textViewAdresse = (TextView) findViewById(R.id.adresse_resto);
TextView textViewLatitude = (TextView) findViewById(R.id.latitude_resto);
TextView textViewLongitude = (TextView) findViewById(R.id.longitude_resto);
TextView textViewDescription = (TextView) findViewById(R.id.description_resto);
//TextView textViewHoraires = (TextView) findViewById(R.id.horaires_resto);
//TextView textViewNoteResto = (TextView) findViewById(R.id.note_resto);

// Récupération des données transmises via l'Intent
Intent intent = getIntent();

//String imageR = intent.getStringExtra("cheminP");
String nomR = intent.getStringExtra( name: "nomR");
String adresse = intent.getStringExtra( name: "numAdrR") + " " + intent.getStringExtra( name: "voieAdrR") + ", " + intent.getStringExtra( nam
float latitude = intent.getFloatExtra( name: "latitudeDegR",  defaultValue: 0);
float longitude = intent.getFloatExtra( name: "longitudeDegR",  defaultValue: 0);
String descR = intent.getStringExtra( name: "descR");
//String horairesR = intent.getStringExtra("horairesR");
//String noteR = intent.getStringExtra("noteR");
```

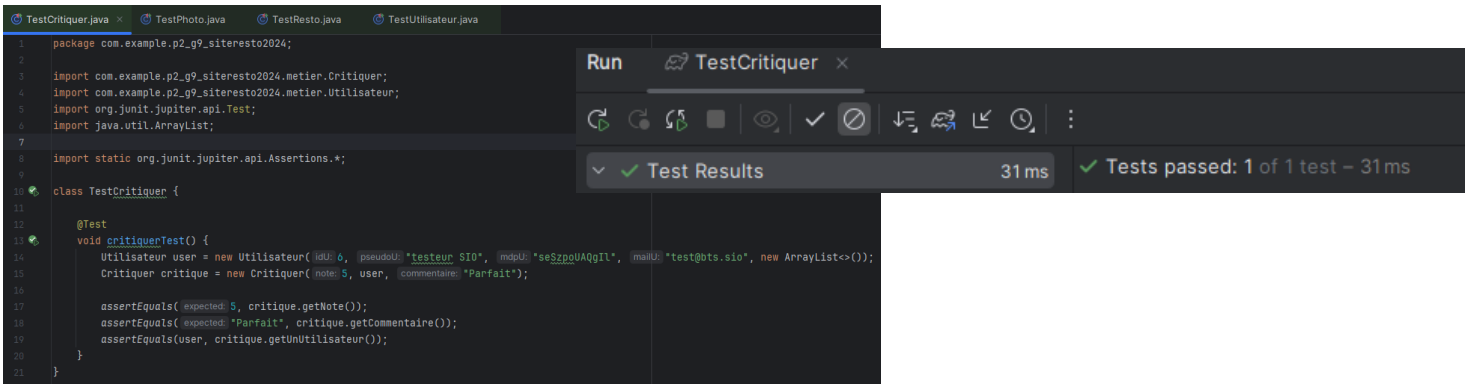
Enfin il nous suffit d'attribuer chaque vues à sa valeur correspondante et le code fonctionne normalement :

```
// Remplissage des champs avec les données

//Drawable drawable = Drawable.createFromPath("imageR");
//imageResto.setImageDrawable(drawable);

textViewNomResto.setText(nomR);
textViewAdresse.setText(adresse);
textViewLongitude.setText(String.valueOf(longitude));
textViewLatitude.setText(String.valueOf(latitude));
textViewDescription.setText(descR);
```

TÂCHE 12 : TEST UNITAIRE DES OBJETS METIER

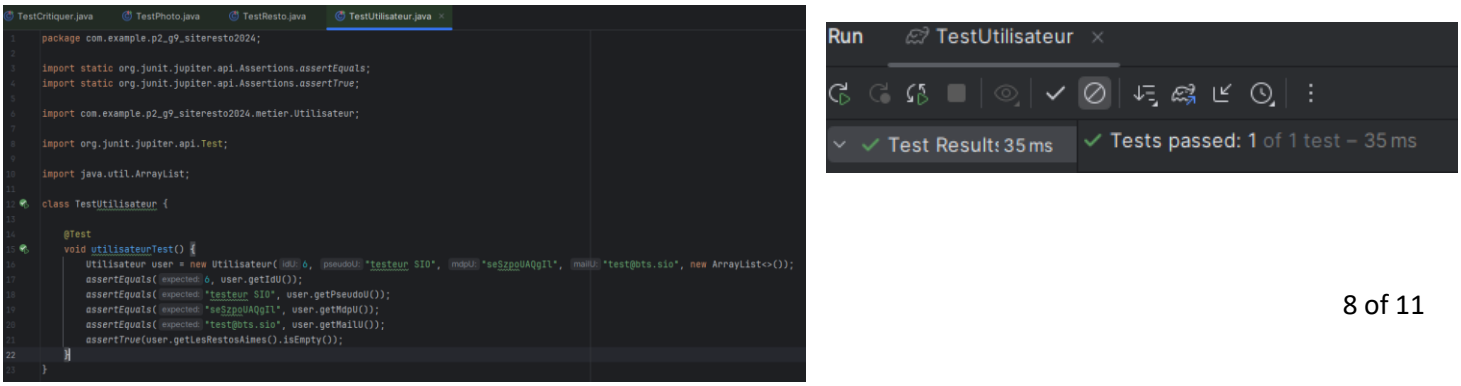
Dans ce ticket j'ai créé les tests unitaires pour nos différentes classes du modèle métier, je me suis inspiré du projet 1 sur le siteResto similaire pour faire mes tests car les modèle sont identique.
J'utiliserai dans ses tests, la méthode assertEquals qui vérifie si les deux valeurs sont égales ou non et répond par false ou true
Vous pourrez retrouver les tests dans le package (aux nombres de 4) :
« Projet_SiteRestoAndroid\app\src\test\java\com\example\p2_g9_siteresto2024 »

Notre premier fichier de test est TestCritique.java, ou l'ont vérifie instancie un utilisateur à qui on attribue une critique :



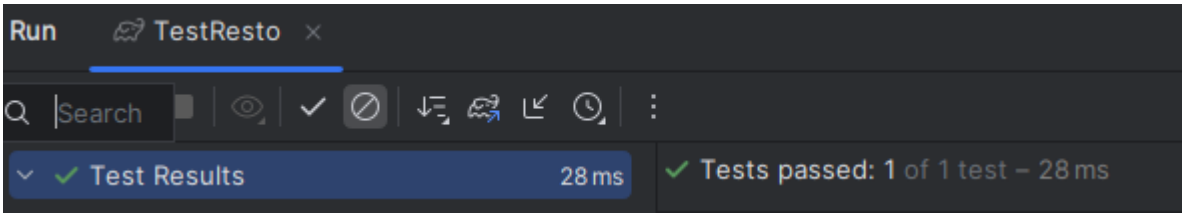
Le second test concernait TestPhotos.java, ou l'ont instancie un simple objet photo uniquement :

Ensuite on test la classe utilisateur.java avec notre TestUtilisateur.java, dans celui-ci on fait comme le précédent en instanciant un utilisateur :



Enfin on test le modèle resto qui est le plus important, dedans on instanciera plusieurs listes dont des listes de critique et photo, ainsi qu'un objet utilisateur à attribuer aux critiques. Les listes ne seront pas instancié dans le constructeur mais par la biais de la méthode setLesCritiques et setLesPhotos.

```java
package com.example.p2_g9_siteresto2024;

import com.example.p2_g9_siteresto2024.metier.*;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;
import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

class TestResto {

    @Test
    void restoTest() {
        Utilisateur user = new Utilisateur( idU: 6, pseudoU: "testeur SIO", mdpU: "seSzpoUAQgIl", mailU: "test@bts.sio", new ArrayList<>());
        List<Critiquer> desCritiques = new ArrayList<>();
        desCritiques.add(new Critiquer( note: 5, user, commentaire: "Parfait"));
        desCritiques.add(new Critiquer( note: 3, user, commentaire: "Perfectible ..."));

        List<Photo> desPhotos = new ArrayList<>();
        desPhotos.add(new Photo( cheminP: "cidrerieDuFronton.jpg", idP: 6));
        desPhotos.add(new Photo( cheminP: "bar_de_la-cidrerie.jpg", idP: 7));

        // Créer un restaurant avec des listes vides au départ, puis les initialiser
        Resto resto = new Resto( idR: 4, nomR: "Cidrerie du fronton", numAdr: "", voieAdr: "Place du Fronton", cpR: "64210", villeR: "Arbonne", latitudeDegR: 0, longitudeDegR: 0, descR: "Sans commentaire", horairesR
        resto.setLesPhotos(desPhotos);  // Ajouter des photos
        resto.setLesCritiques(desCritiques);  // Ajouter des critiques

        // Assertions
        assertEquals( expected: 4, resto.getIdR());
        assertEquals( expected: "Cidrerie du fronton", resto.getNomR());
        assertEquals( expected: "Place du Fronton", resto.getVoieAdr());
        assertEquals( expected: 2, resto.getLesPhotos().size());  // Vérifier que les photos ont été ajoutées
        assertEquals( expected: 2, resto.getLesCritiques().size());  // Vérifier que les critiques ont été ajoutées
    }
}
```



RESULTAT :

Clic sur le resto souhaité

Fiche du resto cliqué

DAILY SCRUM DEBUT DE L'IT2 (25/11/2024) :

GitLab : https://gitlab.com/KKG_Ammbussh/p2_g9_siteresto2024.git

- La branche correspondante à l'itération sur le dépôt est celle avec son numéro, si celle-ci n'est pas présente alors elle correspond au MAIN
- Le README possède normalement le mode opératoire de test

Pendant la réunion :

➔ Problématique rencontré
➔ Les tâche de l'it2 à finir
➔ La fusion une fois l'it2 fini sur la branche « main »

Répartition des tâches au final :
Thibault →Tâche 10, 11, 12
Pierre → Tâche 2
Thomas → Tâche 8, 9

Bilan (problématique rencontré et ce qui a été réalisé) :
On s'est retrouvé face à plusieurs problème notamment du retard pour commencer l'it2, et aujourd'hui un souci sur l'IT2.
Le problème étant que les tâche était répartie de façon équivalente normalement mais malheureusement nous avons eu des soucis de compétence pour résoudre certaine tâche chez certain et des soucis de réalisation trop tard pour d'autre qui n'ont pas aidé à nous avancer

Aujourd'hui toutes l'IT2 a été réaliser mais avec du retard, également certaine tâche l'IT2 notamment celle concernant le DAO, non pas été utilisé car une autre solution a été utilisé plus simple

Tableau de Kanban du début :