

# Projet 3

## Création d'une application de modération

User story : En tant qu'utilisateur, je souhaite, à partir du site restaurant avoir une interface de modération

<b>DAILY SCRUM DEBUT (10/03/2025) .....</b>	<b>2</b>
<b>TÂCHE 01 : MISE EN PLACE DES OUTILS DE GESTION DE PROJETS .....</b>	<b>3</b>
<b>TÂCHE 02 : CHOIX ARGUMENTER DE LA TECHNOLOGIE D'ACCES AU DONNEES .....</b>	<b>5</b>
<b>TÂCHE 03 : PREUVE DE CONCEPT(POC) .....</b>	<b>6</b>
<b>TÂCHE 04 : MAQUETTE .....</b>	<b>8</b>
<b>TÂCHE final : Constitution du rapport d'itération.....</b>	<b>9</b>

## DAILY SCRUM DEBUT (10/03/2025) :

GitLab : [https://gitlab.com/Crouan/projet\\_java\\_moderation.git](https://gitlab.com/Crouan/projet_java_moderation.git)

### Pendant la réunion :

- ➔ Création des Ticket de l'IT1
- ➔ Attribution des tickets de l'IT1
- ➔ Résolution des tickets de l'IT1

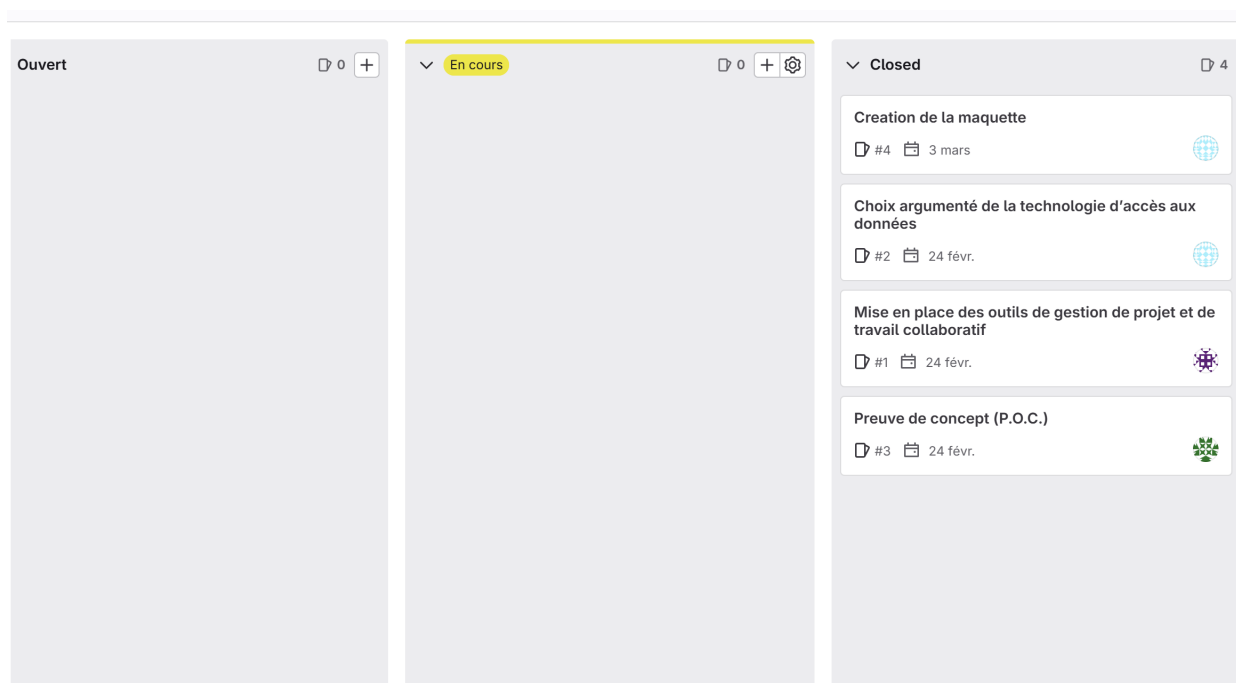
### Répartition des tâches du début :

**Alexis** ➔ Tâche 4, 2

**Pierre** ➔ Tâche 2, compte rendu IT1

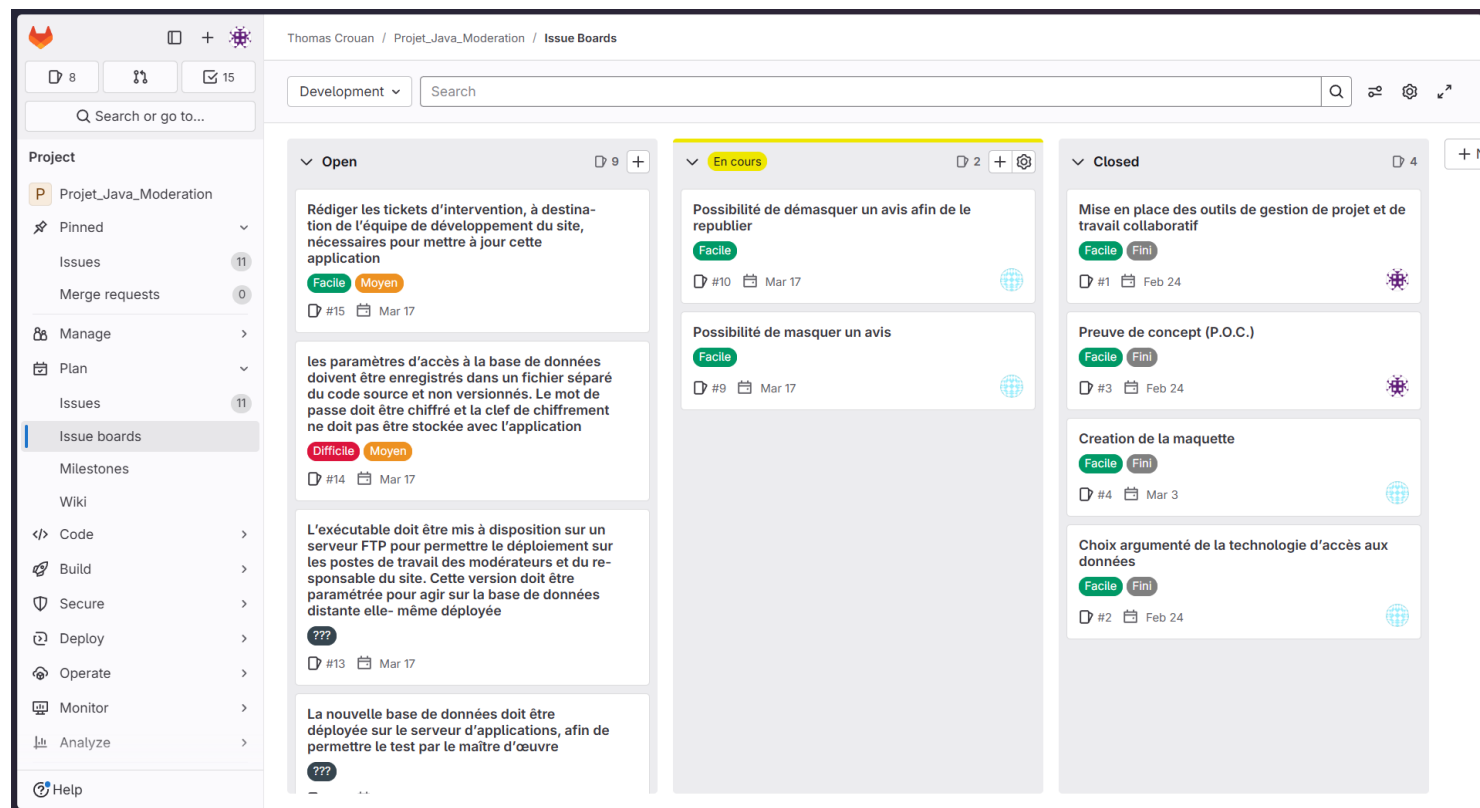
**Thomas** ➔ Tâche 1,3

### Tableau de Kanban du début :



## TÂCHE 01 : Mise en place des outils de gestion de projet et de travail collaboratif

J'ai mis en place le GitLab avec les issues à réaliser dans l'itération 1, ainsi que la création du READ.ME.



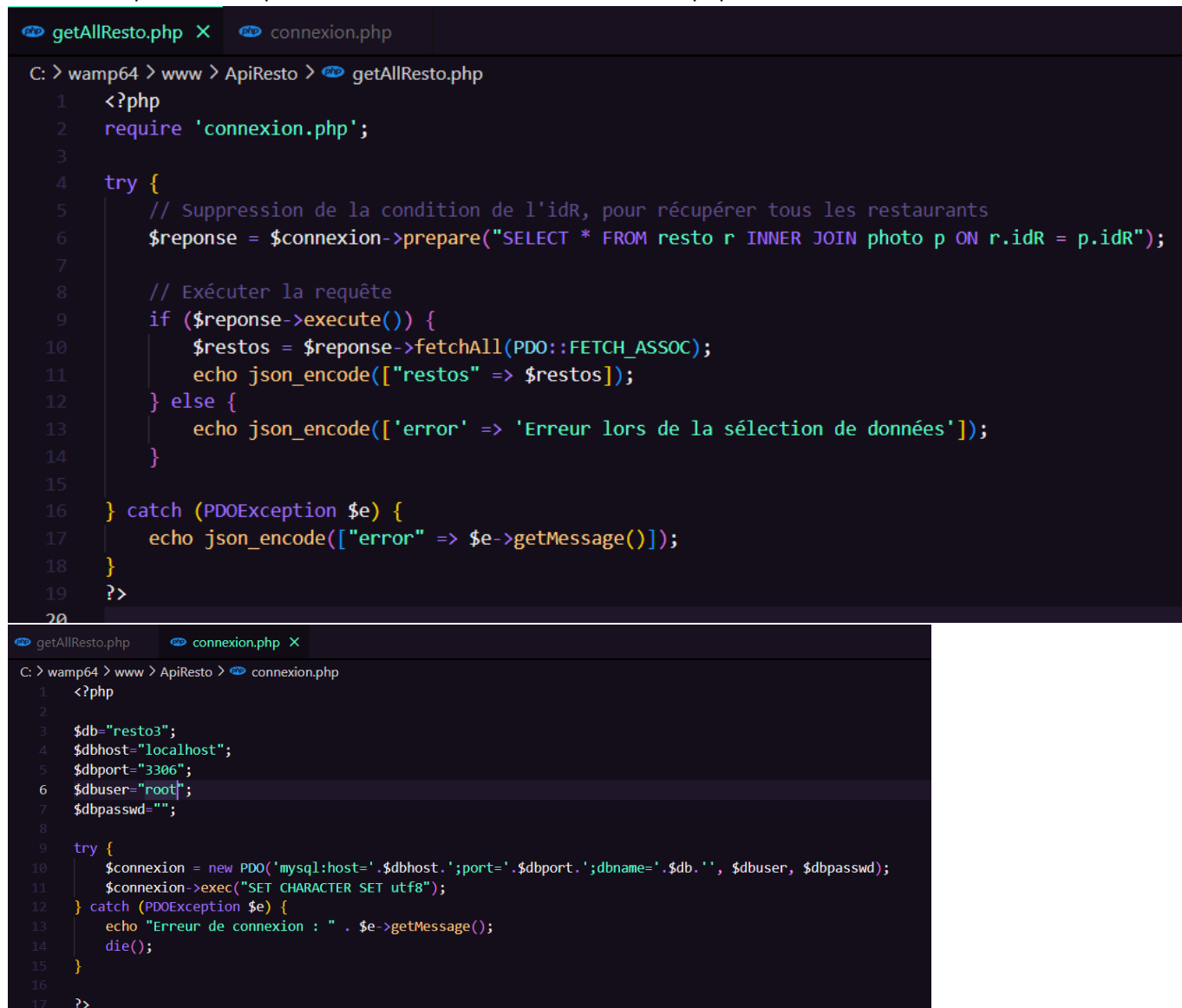
Et une mise en place d'un system de commit et de push, afin de voir claire dans l'arborescence et les modifications du projet.

## TÂCHE 02 : Choix argumenter de la technologie d'accès aux données

L'application Java étant destinée à être installée sur les postes de travail des modérateurs, il nous paraît plus judicieux d'opter pour une API REST. En effet, une API REST présente plusieurs avantages, notamment une facilité de déploiement et de gestion.

## TÂCHE 03 : Preuve de concept (P.O.C.)

J'ai dans un premier temps mis mon API et mon fichier connexion.php dans un dossier dans mon www :



```

C: > wamp64 > www > ApiResto > getAllResto.php
1  <?php
2  require 'connexion.php';
3
4  try {
5      // Suppression de la condition de l'idR, pour récupérer tous les restaurants
6      $reponse = $connexion->prepare("SELECT * FROM resto r INNER JOIN photo p ON r.idR = p.idR");
7
8      // Exécuter la requête
9      if ($reponse->execute()) {
10         $restos = $reponse->fetchAll(PDO::FETCH_ASSOC);
11         echo json_encode(["restos" => $restos]);
12     } else {
13         echo json_encode(['error' => 'Erreur lors de la sélection de données']);
14     }
15 } catch (PDOException $e) {
16     echo json_encode(["error" => $e->getMessage()]);
17 }
18 ?>
19
20
C: > wamp64 > www > ApiResto > connexion.php
1  <?php
2
3  $db="resto3";
4  $dbhost="localhost";
5  $dbport="3306";
6  $dbuser="root";
7  $dbpasswd="";
8
9  try {
10     $connexion = new PDO('mysql:host='.$dbhost.';port='.$dbport.';dbname='.$db.'', $dbuser, $dbpasswd);
11     $connexion->exec("SET CHARACTER SET utf8");
12 } catch (PDOException $e) {
13     echo "Erreur de connexion : " . $e->getMessage();
14     die();
15 }
16
17 ?>
  
```

Puis j'ai récupéré l'API dans le code java de l'application et demandé d'afficher tous les noms des restaurants présents dans la bdd :

```
import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class GetAllRestaurants {

    public static void main(String[] args) {
        // URL de votre API PHP
        String apiUrl = "http://localhost/ApiResto/getAllResto.php";

        try {
            // Créer une URL à partir de l'URL de l'API
            URL url = new URL(apiUrl);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET"); // méthode GET pour récupérer des données

            // Lire la réponse de l'API
            BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuilder response = new StringBuilder();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            // Convertir la réponse en objet JSON
            JSONObject jsonResponse = new JSONObject(response.toString());

            // Vérifier si la clé "restos" existe dans la réponse
            if (jsonResponse.has("restos")) {
                JSONArray restos = jsonResponse.getJSONArray("restos");

                // Afficher les informations de chaque restaurant
                for (int i = 0; i < restos.length(); i++) {
                    JSONObject currentRestaurant = restos.getJSONObject(i);
                    String nom = currentRestaurant.optString("nomR", "Nom inconnu");
                    System.out.println("Nom du restaurant : " + nom);
                }
            } else {
                System.out.println("Aucun restaurant trouvé dans la réponse.");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Voici le résultat dans le terminal :

```
run:
Nom du restaurant : l'entrepote
Nom du restaurant : l'entrepote
Nom du restaurant : le bar du charcutier
Nom du restaurant : Sapporo
Nom du restaurant : Cidrerie du fronton
Nom du restaurant : Cidrerie du fronton
Nom du restaurant : Cidrerie du fronton
Nom du restaurant : Agadir
Nom du restaurant : Le Bistrot Sainte Cluque
Nom du restaurant : la petite auberge
Nom du restaurant : La table de POTTOKA
Nom du restaurant : La Rotisserie du Roy Lion
Nom du restaurant : Bar du Marché
Nom du restaurant : Trinquet Moderne
BUILD SUCCESSFUL (total time: 0 seconds)
```

## TÂCHE 04 : Maquette

Pour la maquette, étant donné que nous utilisons un système de vue MVC, nous avons dû créer un package de vue dans lequel nous avons pu inclure nos différents JFrame. À l'heure actuelle, nous en avons deux : un pour la connexion de l'utilisateur et un autre pour afficher la liste de tous les commentaires en vue de la modération future.

La maquette présente deux interfaces utilisateur distinctes, chacune dans un cadre gris avec une bordure violette.

La première interface, intitulée "Connexion :", est centrée et contient les éléments suivants :

- Le texte "Login :" suivi d'un champ de saisie rectangulaire blanc.
- Le texte "Mot de passe :" suivi d'un champ de saisie rectangulaire blanc.
- Un bouton rectangulaire blanc avec l'inscription "Connexion".

La seconde interface, intitulée "Liste des commentaires :", est également centrée et contient :

- Un tableau à 4 colonnes avec des en-têtes "Title 1", "Title 2", "Title 3" et "Title 4".
- Une zone de contenu vide en dessous du tableau.
- Une barre de boutons en bas avec quatre boutons rectangulaires blancs : "Modifier", "Supprimer", "Ajouter" et "Bloquer".