

Projet AP 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

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

DAILY SCRUM DEBUT (17/03/2025) :

GitLab : https://gitlab.com/Crouan/projet_java_moderation.git

GitLab API : https://gitlab.com/projet6274038/projet_java_moderationapi.git

Pendant la réunion :

→ Attribution des tickets de l'IT3

Planification de l'itération :

Dans un premier temps, nous déciderons tous ensemble des tâches qui seraient à effectuer dans cette troisième itération, c'est à dire celle qui nous semble nécessaire pour avancer plus loin sur le projet. Puis se les attribuer.

Au niveau de la répartition des tâches il a été décidé qu'on s'occuperait tous de nos tâches attribuées. Tout en s'aidant si besoin.

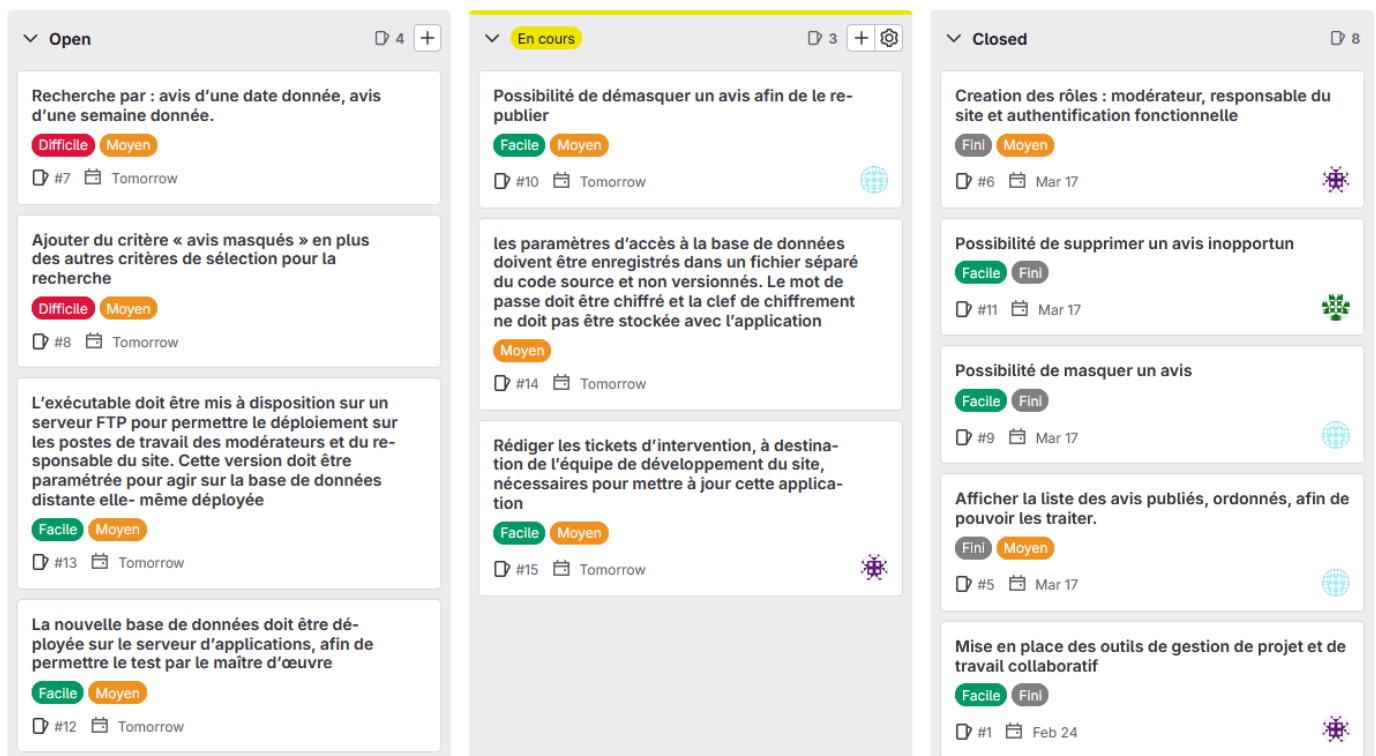
Répartition des tâches du début :

Alexis → Tâche 1

Thomas → Tâche 2

Pierre → Tâche 3

Tableau de Kanban du début :



TÂCHE 01 : Possibilité de démasquer un avis afin de le republier + critère de recherche par avis masquer

Pour cela dans un premier temps, j'ai créé de la même manière que pour les avis publier une API et classe java afin de récupérer tous les avis masquer :

```
public class GetAllAvisMasquer {

    public static List<Object[]> getAvis() {
        // URL de l'API qui récupère les avis des restaurants
        String apiUrl = "http://localhost/ApiResto/getAllAvisMasquer.php";
        List<Object[]> avisList = new ArrayList<>();

        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 "avis" existe dans la réponse
            if (jsonResponse.has("avis")) {
                JSONArray avisArray = jsonResponse.getJSONArray("avis");

                // Extraire les informations de chaque avis et les ajouter dans la liste
                for (int i = 0; i < avisArray.length(); i++) {
                    JSONObject currentAvis = avisArray.getJSONObject(i);
```

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

header('Content-Type: application/json; charset=UTF-8');

$connexion->exec("SET NAMES 'utf8mb4'");

try {
    $reponse = $connexion->prepare("SELECT * FROM critiquer c INNER JOIN etat_avis e ON c.idEA = e.idEA WHERE e.libelleEA = 'masquer'");

    if ($reponse->execute()) {
        $avis = $reponse->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode(["avis" => $avis]);
    } else {
        echo json_encode(["error" => 'Erreur lors de la sélection de données']);
    }
} catch (PDOException $e) {
    echo json_encode(["error" => $e->getMessage()]);
}
?>
```

On teste dans la bdd :

`SELECT * FROM critiqueur c INNER JOIN etat_avis e ON c.idEA = e.idEA WHERE e.libelleEA = 'masquer';`

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

idR	note	commentaire	idU	idEA	date	idEA	libelleEA
4	5	Rapide.	5	1	NULL	1	masquer
7	4	Bon accueil.	1	1	NULL	1	masquer

On test l'URL :

localhost/ApiResto/getAllAvisMasquer.php

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```

avis:
  0:
    idR: 4
    note: 5
    commentaire: "Rapide."
    idU: 5
    idEA: 1
    date: null
    libelleEA: "masquer"
  1:
    idR: 7
    note: 4
    commentaire: "Bon accueil."
    idU: 1
    idEA: 1
    date: null
    libelleEA: "masquer"

```

Ensuite, j'ai dû créer une recherche dans mon JComboBox afin de pouvoir rechercher soit les avis publier ou masquer, pour cela j'ai initialisé le contenu de mon JComboBox:

```

public JFrameListeCommentaire() {
    initComponents();

    comboBoxRecherche.removeAllItems();
    comboBoxRecherche.addItem("Publier");
    comboBoxRecherche.addItem("Masquer");

    loadDataFromPublier();
}

```

J'ai laissé les commentaires publier à afficher par default, puis j'ai créé une fonction loadDataFromMasquer afin d'appeler le résultat de la requête de l'API des commentaires masquer dans le tableau :

```

private void loadDataFromMasquer() {
// Récupérer les avis depuis la classe GetAllAvisMasquer
List<Object[]> avisList = GetAllAvisMasquer.getAvis();

// Définir les colonnes du tableau
String[] columns = {"ID Restaurant", "Note", "Commentaire", "ID Utilisateur", "ID Etat", "Etat"};

// Créer un modèle de tableau avec les colonnes définies
DefaultTableModel tableModel = new DefaultTableModel(columns, 0);

// Ajouter chaque avis récupéré dans le tableau
for (Object[] avis : avisList) {
    tableModel.addRow(avis);
}

// Assigner le modèle au tableau
jTableListeCommentaires.setModel(tableModel);
}

```

Ensuite j'ai ajouté le traitement à faire en fonction de l'option choisi dans le JComboBox :

```

private void jComboBoxRechercheActionPerformed(java.awt.event.ActionEvent evt) {
String selectedOption = (String) jComboBoxRecherche.getSelectedItem();

// Vérifie si selectedOption n'est pas null avant de continuer
if (selectedOption != null) {
    if (selectedOption.equals("Masquer")) {
        loadDataFromMasquer(); // Appeler la méthode pour charger les avis masqués
    } else if (selectedOption.equals("Publier")) {
        loadDataFromPublier(); // Appeler la méthode pour charger les avis publiés
    }
}
}

```

Puis les actions à effectuer quand le bouton Démasquer est utiliser :

```

private void jButtonDemasquerActionPerformed(java.awt.event.ActionEvent evt) {
int selectedRow = jTableListeCommentaires.getSelectedRow();

if (selectedRow != -1) {
// Récupérer l'ID de l'avis sélectionné (l'ID de l'avis est dans la première colonne)
int idR = (int) jTableListeCommentaires.getValueAt(selectedRow, 0); // ID de l'avis dans la colonn

// Récupérer l'ID de l'utilisateur ayant écrit l'avis (en supposant qu'il soit dans la colonne 3)
int idU = (int) jTableListeCommentaires.getValueAt(selectedRow, 3); // ID de l'utilisateur dans la

String currentStateLabel = (String) jTableListeCommentaires.getValueAt(selectedRow, 5); // Libellé

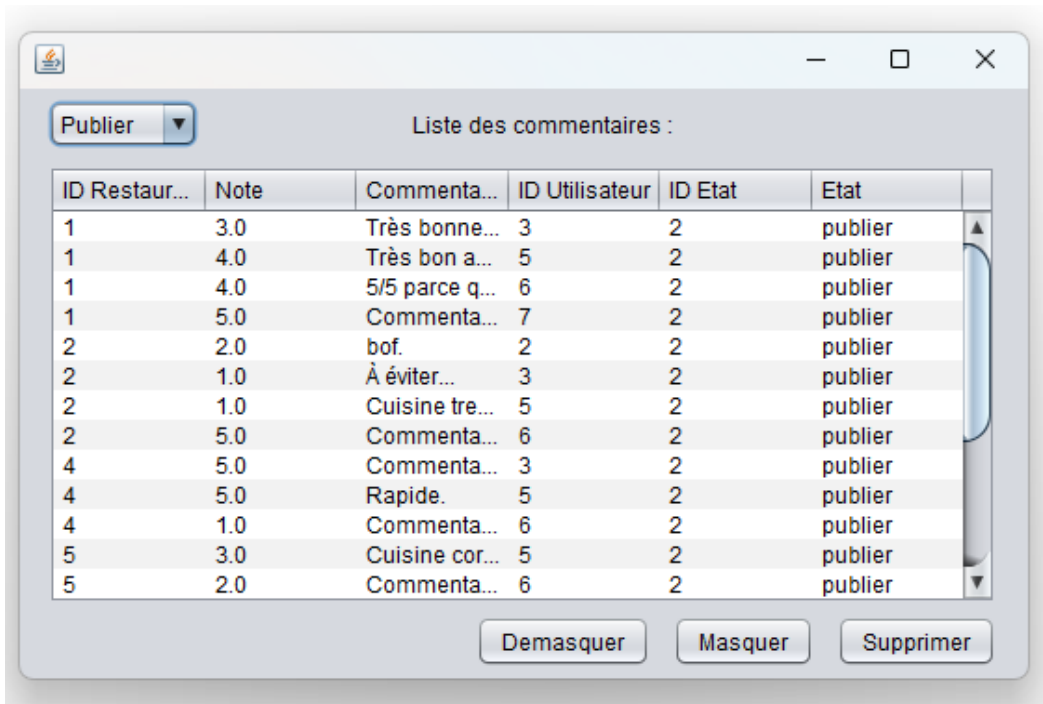
if (!"publier".equalsIgnoreCase(currentStateLabel)) {
    boolean updateSuccessful = UpdateAvis.updateAvisState(idR, 2, idU);

    if (updateSuccessful) {
        // Mettre à jour le tableau pour refléter ce changement
        jTableListeCommentaires.setValueAt(2, selectedRow, 4); // Mettre à jour l'ID Etat (colonne
        jTableListeCommentaires.setValueAt("publier", selectedRow, 5); // Mettre à jour l'état vis
    } else {
        JOptionPane.showMessageDialog(this, "Erreur lors de la mise à jour de l'état de l'avis.");
    }
} else {
    JOptionPane.showMessageDialog(this, "L'avis est déjà publié.");
}
} else {
    JOptionPane.showMessageDialog(this, "Veuillez sélectionner un avis.");
}
}
}

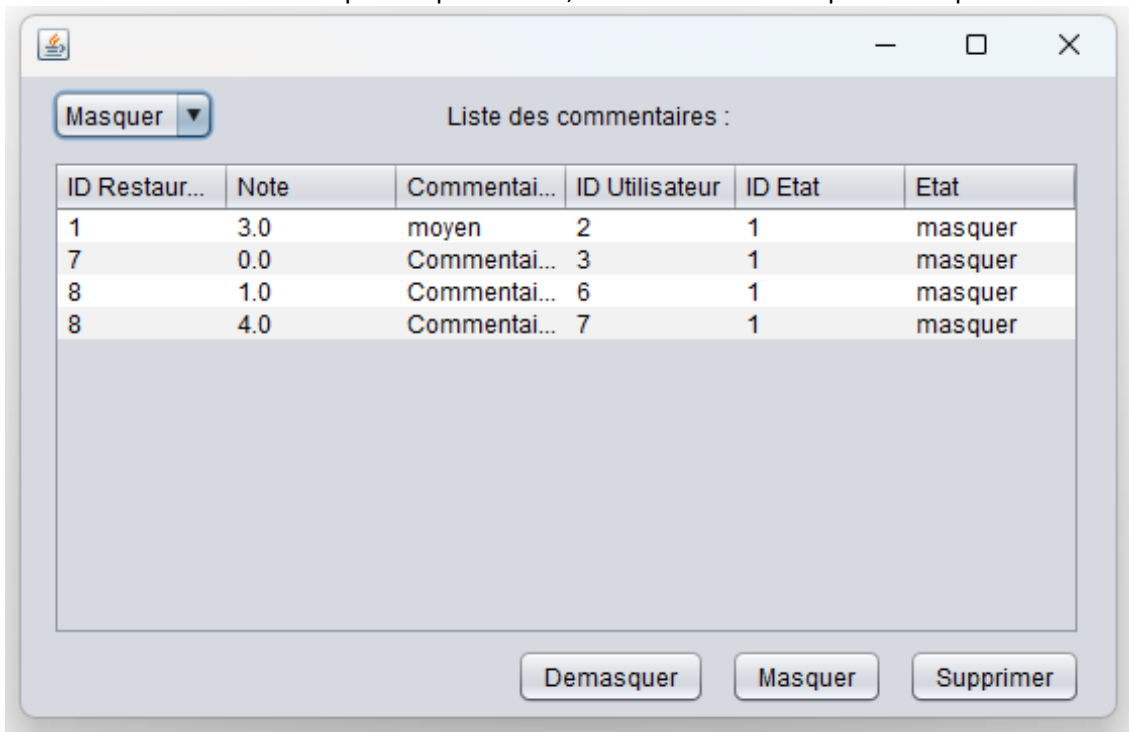
```

Celui ci fonctionne de la même manière que le bouton masquer mais à "l'inverse"

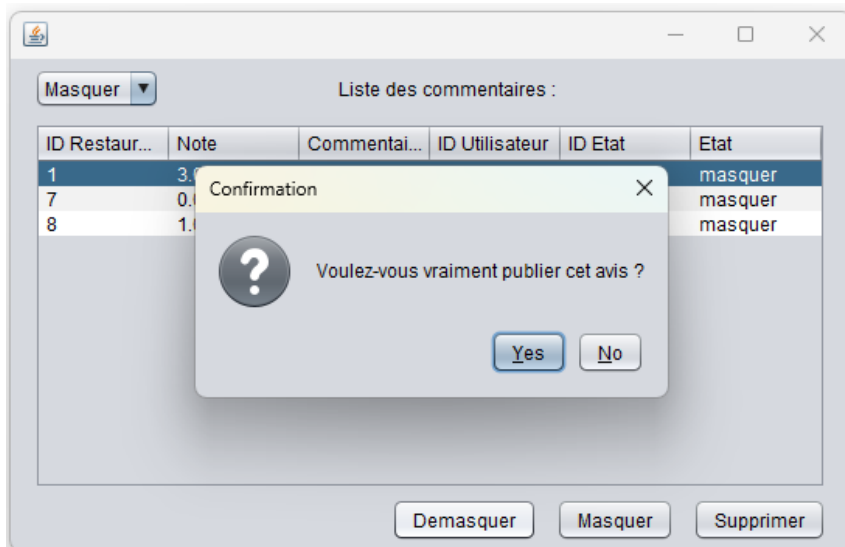
On test ensuite :



On a bien les commentaires publier par default, on choisit ensuite l'option masquer :

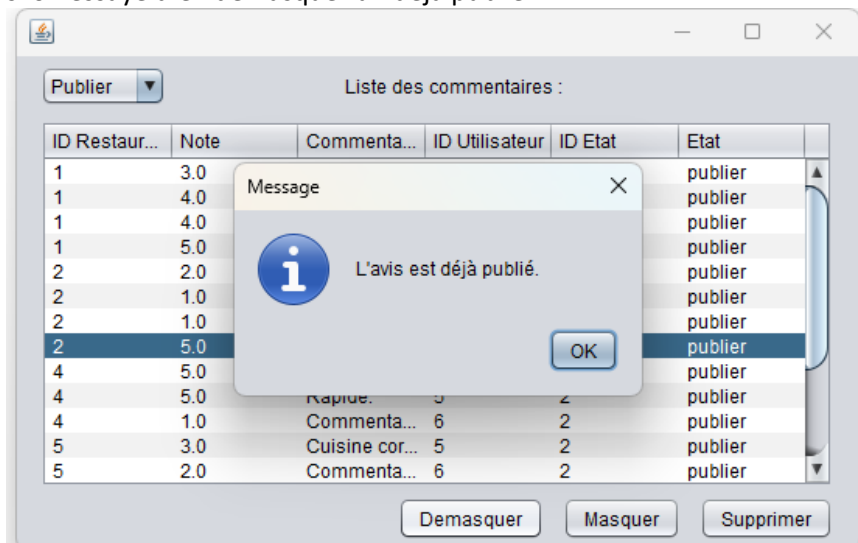


On essaye d'en démasquer un :



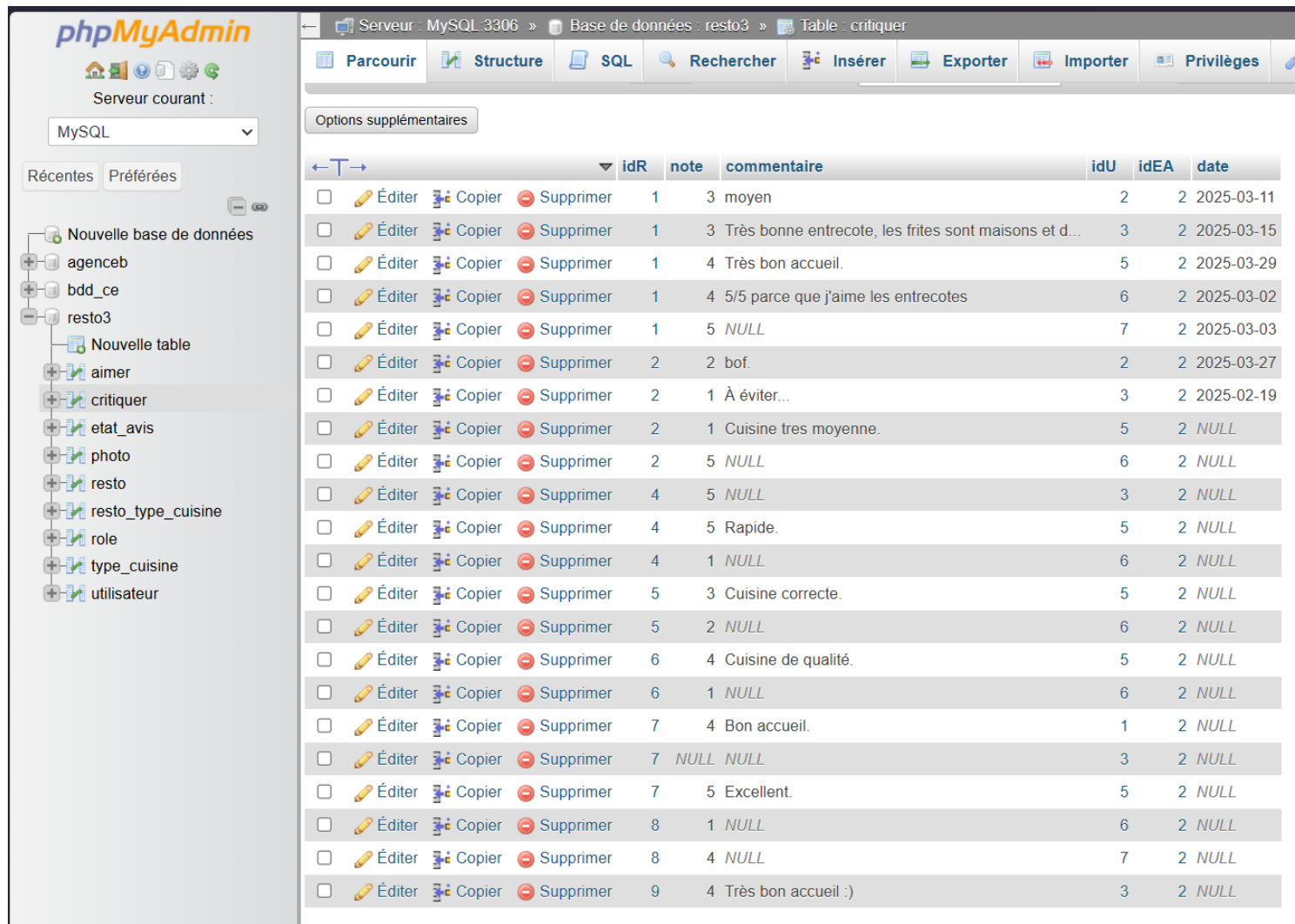
Response from server: {"status": "success", "message": "\u00c9tat de l'avis mis \u00e0 jour"}
L'état de l'avis a été mis à jour.

On constate que cela fonctionne,
si on essaye d'en démasquer un déjà publié :



TÂCHE 02 : Recherche par avis d'une date donnée, avis d'une semaine donnée

J’ai dans un premier temps ajouté un champ “date” à la table “critiquer” :

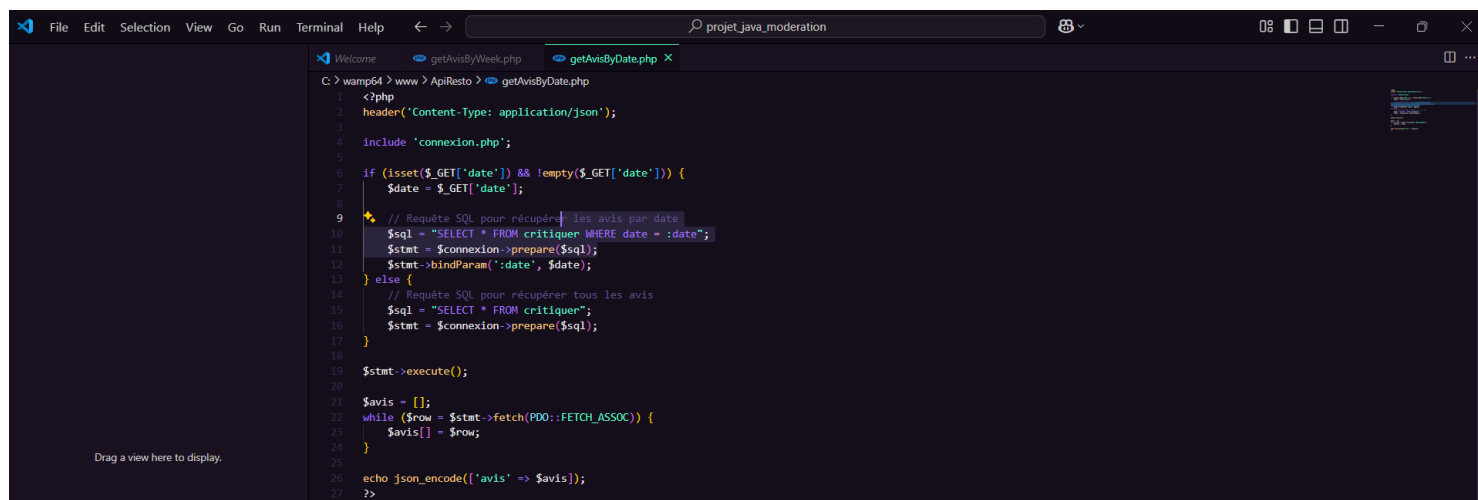


The screenshot shows the phpMyAdmin interface. On the left, the database 'resto3' is selected, and the table 'critiquer' is highlighted. The main area displays the table structure and data. The table has columns: idR, note, commentaire, idU, idEA, and date. The data is as follows:

	idR	note	commentaire	idU	idEA	date
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	3	moyen	2	2	2025-03-11
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	3	Très bonne entrecote, les frites sont maisons et d...	3	2	2025-03-15
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	4	Très bon accueil.	5	2	2025-03-29
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	4	5/5 parce que j'aime les entrecotes	6	2	2025-03-02
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	5	NULL	7	2	2025-03-03
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	2	bof.	2	2	2025-03-27
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	1	À éviter...	3	2	2025-02-19
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	1	Cuisine tres moyenne.	5	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	5	NULL	6	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	4	5	NULL	3	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	4	5	Rapide.	5	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	4	1	NULL	6	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	5	3	Cuisine correcte.	5	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	5	2	NULL	6	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	6	4	Cuisine de qualité.	5	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	6	1	NULL	6	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	7	4	Bon accueil.	1	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	7	NULL	NULL	3	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	7	5	Excellent.	5	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	8	1	NULL	6	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	8	4	NULL	7	2	NULL
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	9	4	Très bon accueil :)	3	2	NULL

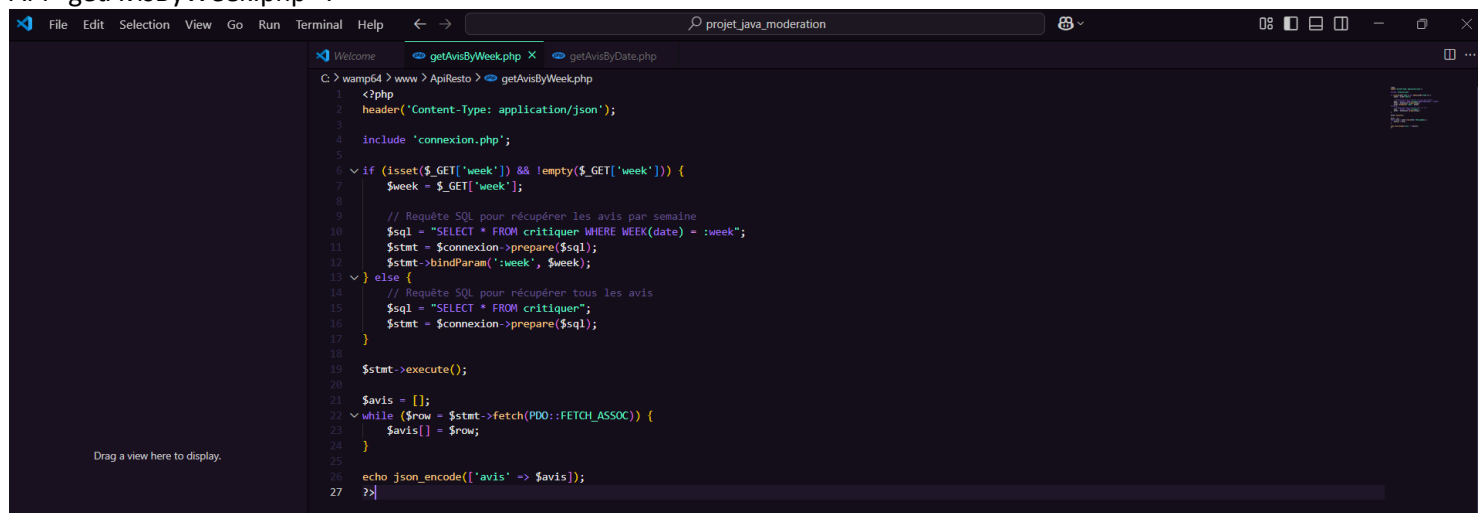
Puis j’ai réalisé les API “getAvisByDate.php” et “getAvisByWeek.php” qui servent à récupérer des avis depuis la table “critiquer” sous forme de JSON, soit pour une semaine spécifique via le paramètre week, soit pour une date précise via le paramètre date, et renvoient tous les avis si aucun paramètre n'est fourni.

API “getAvisByDate.php” :



```
1 <?php
2 header('Content-Type: application/json');
3
4 include 'connexion.php';
5
6 if (isset($_GET['date']) && !empty($_GET['date'])) {
7     $date = $_GET['date'];
8
9     // Requête SQL pour récupérer les avis par date
10    $sql = "SELECT * FROM critiqueur WHERE date = :date";
11    $stmt = $connexion->prepare($sql);
12    $stmt->bindParam(':date', $date);
13 } else {
14     // Requête SQL pour récupérer tous les avis
15    $sql = "SELECT * FROM critiqueur";
16    $stmt = $connexion->prepare($sql);
17 }
18
19 $stmt->execute();
20
21 $avis = [];
22 while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
23     $avis[] = $row;
24 }
25
26 echo json_encode(['avis' => $avis]);
27 >>
```

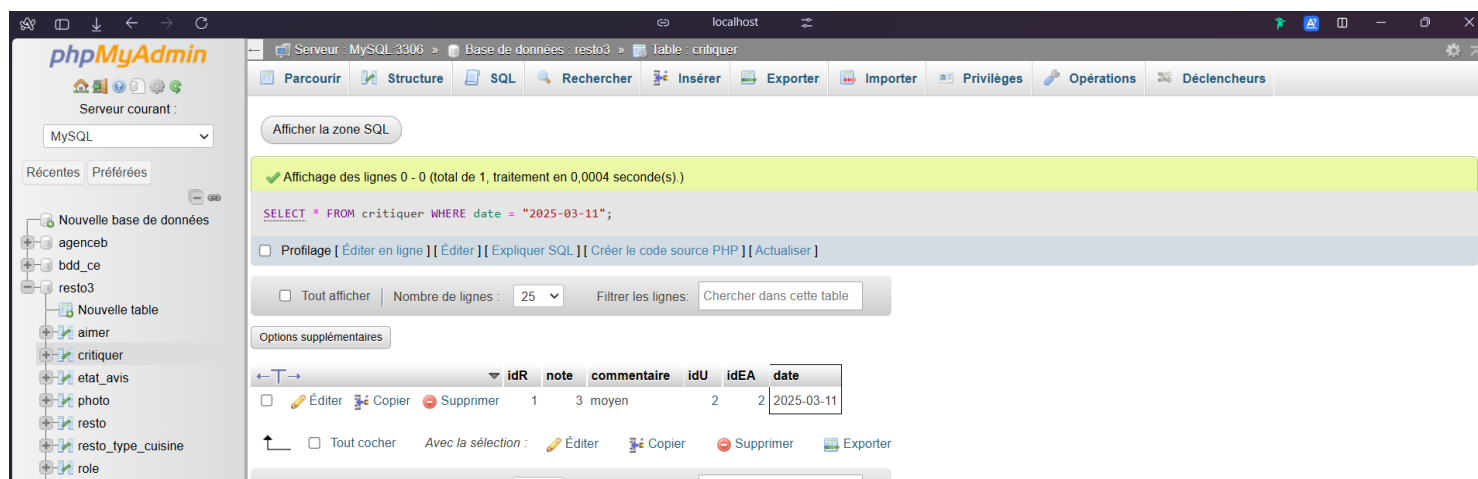
API “getAvisByWeek.php” :



```
1 <?php
2 header('Content-Type: application/json');
3
4 include 'connexion.php';
5
6 if (isset($_GET['week']) && !empty($_GET['week'])) {
7     $week = $_GET['week'];
8
9     // Requête SQL pour récupérer les avis par semaine
10    $sql = "SELECT * FROM critiqueur WHERE WEEK(date) = :week";
11    $stmt = $connexion->prepare($sql);
12    $stmt->bindParam(':week', $week);
13 } else {
14     // Requête SQL pour récupérer tous les avis
15    $sql = "SELECT * FROM critiqueur";
16    $stmt = $connexion->prepare($sql);
17 }
18
19 $stmt->execute();
20
21 $avis = [];
22 while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
23     $avis[] = $row;
24 }
25
26 echo json_encode(['avis' => $avis]);
27 >>
```

Voici les tests unitaires des deux API :

- Tout d’abords il y a les scripts SQL dans MySQL :

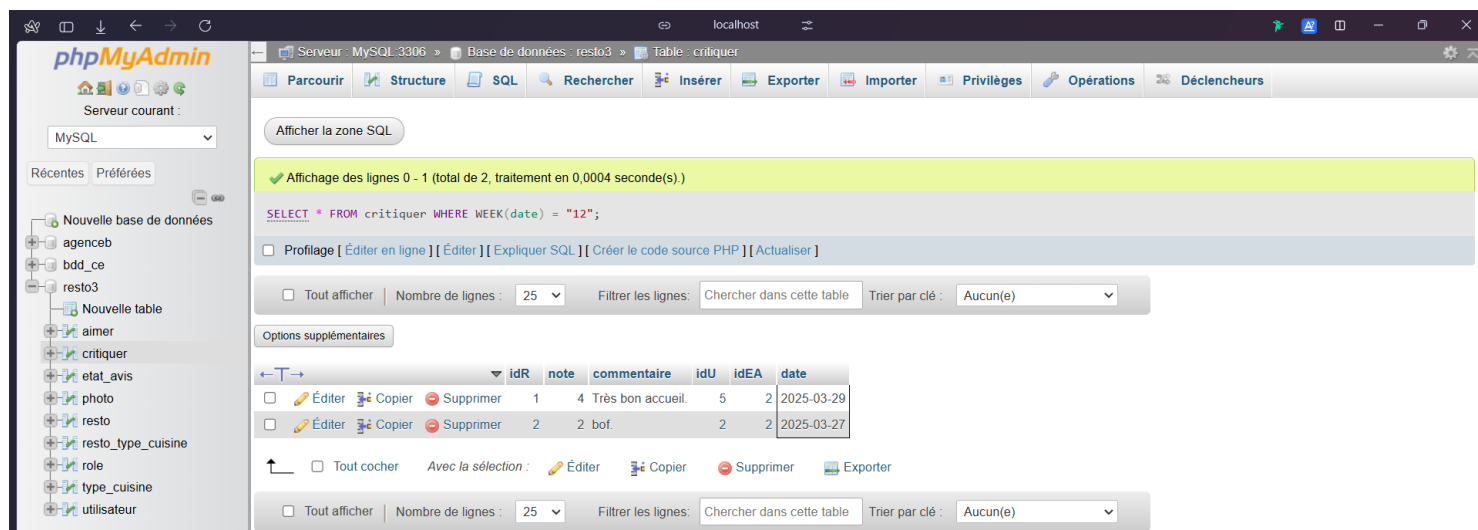


phpMyAdmin interface showing the SQL query and results for the 'critiquer' table.

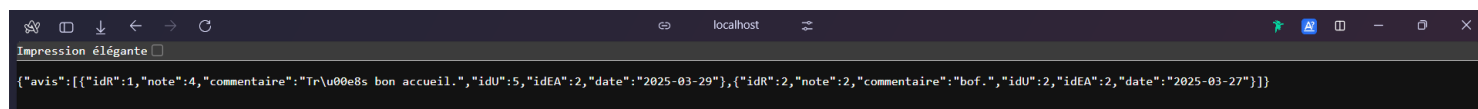
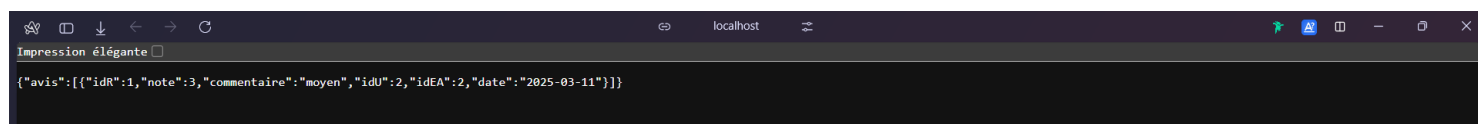
SQL Query: `SELECT * FROM critiqueur WHERE date = "2025-03-11";`

Results:

IdR	note	commentaire	IdU	IdEA	date
1	3	moyen	2	2	2025-03-11



- Puis dans le navigateur et voir ce qu'il retourne



Ensuite, j'ai créé une méthode dans la classe Java permettant de récupérer les avis depuis l'API en effectuant une requête HTTP GET vers le script PHP correspondant. Cette méthode prend en paramètre soit une date (`getAvisByDate`), soit une semaine (`getAvisByWeek`), puis elle établit une connexion avec l'API, lit la réponse JSON et extrait les informations nécessaires (ID du restaurant, note, commentaire, ID utilisateur, état de l'avis, libellé de l'état et date). Ces données sont ensuite stockées dans une liste d'objets pour être réutilisées dans l'application.

- `getAvisByDate` :

```

59
60 public static List<Object[]> getAvisByDate(String date) {
61     String apiUrl = "http://localhost/ApiResto/getAvisByDate.php?date=" + date;
62     List<Object[]> avisList = new ArrayList<>();
63
64     try {
65         URL url = new URL(apiUrl);
66         HttpURLConnection connection = (HttpURLConnection) url.openConnection();
67         connection.setRequestMethod("GET");
68
69         BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
70         String inputLine;
71         StringBuilder response = new StringBuilder();
72
73         while ((inputLine = in.readLine()) != null) {
74             response.append(inputLine);
75         }
76         in.close();
77
78         // Afficher la réponse brute pour le diagnostic
79         System.out.println("Response: " + response.toString());
80
81         JSONObject jsonResponse = new JSONObject(response.toString());
82
83         if (jsonResponse.has("avis")) {
84             JSONArray avisArray = jsonResponse.getJSONArray("avis");
85
86             for (int i = 0; i < avisArray.length(); i++) {
87                 JSONObject currentAvis = avisArray.getJSONObject(i);
88
89                 int idResto = currentAvis.optInt("idR", -1);
90                 double note = currentAvis.optDouble("note", 0.0);
91                 String commentaire = currentAvis.optString("commentaire", "Commentaire inconnu");
92                 int idUtilisateur = currentAvis.optInt("idU", -1);
93                 int idEtatAvis = currentAvis.optInt("idEA", -1);
94                 String libelleEtatAvis = currentAvis.optString("libelleEA", "État inconnu");
95                 String avisDate = currentAvis.optString("date", "Date inconnue");
96
97                 Object[] row = {idResto, note, commentaire, idUtilisateur, idEtatAvis, libelleEtatAvis, avisDate};
98                 avisList.add(row);
99             }
100         }
101     } catch (Exception e) {
102         e.printStackTrace();
103     }
104
105     return avisList;
106 }

```

- getAvisByWeek

```
108
109
110 public static List<Object[]> getAvisByWeek(String week) {
111     String apiUrl = "http://localhost/ApiResto/getAvisByWeek.php?week=" + (week != null ? week : "");
112     List<Object[]> avisList = new ArrayList<>();
113
114     try {
115         URL url = new URL(apiUrl);
116         HttpURLConnection connection = (HttpURLConnection) url.openConnection();
117         connection.setRequestMethod("GET");
118
119         BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
120         String inputLine;
121         StringBuilder response = new StringBuilder();
122
123         while ((inputLine = in.readLine()) != null) {
124             response.append(inputLine);
125         }
126         in.close();
127
128         // Afficher la réponse brute pour le diagnostic
129         System.out.println("Response: " + response.toString());
130
131         JSONObject jsonResponse = new JSONObject(response.toString());
132
133         if (jsonResponse.has("avis")) {
134             JSONArray avisArray = jsonResponse.getJSONArray("avis");
135
136             for (int i = 0; i < avisArray.length(); i++) {
137                 JSONObject currentAvis = avisArray.getJSONObject(i);
138
139                 int idResto = currentAvis.optInt("idR", -1);
140                 double note = currentAvis.optDouble("note", 0.0);
141                 String commentaire = currentAvis.optString("commentaire", "Commentaire inconnu");
142                 int idUtilisateur = currentAvis.optInt("idU", -1);
143                 int idEtatAvis = currentAvis.optInt("idEA", -1);
144                 String libelleEtatAvis = currentAvis.optString("libelleEA", "État inconnu");
145                 String avisDate = currentAvis.optString("date", "Date inconnue");
146
147                 Object[] row = {idResto, note, commentaire, idUtilisateur, idEtatAvis, libelleEtatAvis, avisDate};
148                 avisList.add(row);
149             }
150         } catch (Exception e) {
151             e.printStackTrace();
152         }
153
154     }
155     return avisList;
156 }
```

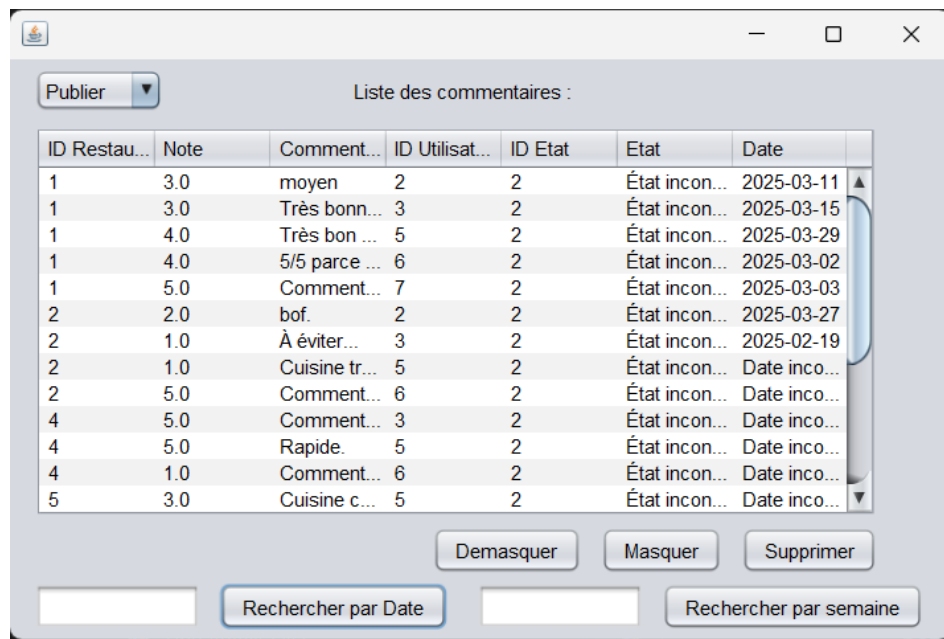
Puis voici le test fonctionnel :

- Lorsque que l'on cherche les avis par date (exemple : 2025-03-11) :

ID Restau...	Note	Comment...	ID Utilisate...	ID Etat	Etat	Date
1	3.0	moyen	2	2	État inconnu	2025-03-11

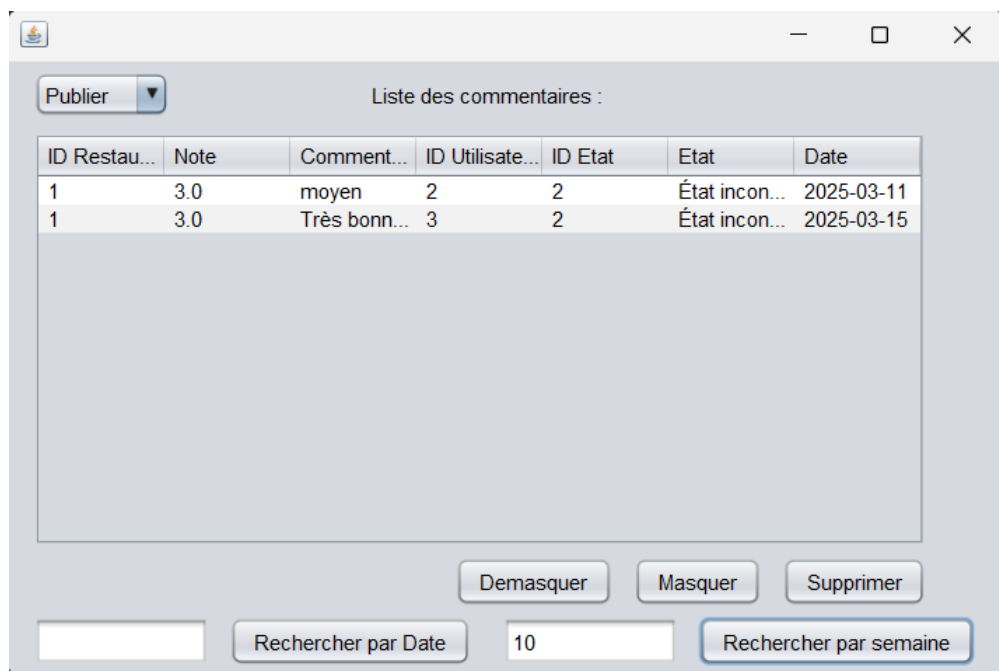
2025-03-11 Rechercher par Date Rechercher par semaine

- Lorsque l'on cherche avec une date null :



Tous les avis s'affichent par défaut.

- Lorsque l'on cherche par semaine (avec un entier qui représente la x ième semaine de l'année) :



Exemple avec la 10ième semaine

Publier ▼

Liste des commentaires :

ID Restau...	Note	Comment...	ID Utilisate...	ID Etat	Etat	Date
1	4.0	Très bon ...	5	2	État incon...	2025-03-29
2	2.0	bof.	2	2	État incon...	2025-03-27

Demasquer Masquer Supprimer

Rechercher par Date 12 Rechercher par semaine

Exemple avec la 12ième semaine

TÂCHE 03 : les paramètres d'accès à la base de données doivent être enregistrés dans un fichier séparé du code source et non versionnés. Le mot de passe doit être chiffré et la clef de chiffrement ne doit pas être stockée avec l'application

J'ai commencé à modifier le Fichier de Connexion pour BDD et pouvoir chiffrer le mot de passe de la connexion à l'utilisateur. Cela n'est pas fini Nous avons des problèmes sur la connexion à la base de données

```
require 'vendor/autoload.php';
use Defuse\Crypto\Crypto;
use Defuse\Crypto\Key;

// Charger les variables d'environnement
$db = getenv(name: 'resto3');
$dbhost = getenv(name: 'localhost');
$dbport = getenv(name: '3306');
$dbuser = getenv(name: 'resto_user');
$encryptedPassword = getenv(name: 'secret');

$encryptionKey = getenv(name: 'DEFUSE_ENCRYPTION_KEY');

try {
    if (!$encryptionKey) {
        throw new Exception(message: "La clé de chiffrement est introuvable.");
    }

    // Déchiffrer le mot de passe
    $key = Key::loadFromAsciiSafeString($encryptionKey);
    $dbpasswd = Crypto::decrypt($encryptedPassword, $key);

    // Connexion à la base de données
    $connexion = new PDO(dsn: "mysql:host=$dbhost;port=$dbport;dbname=$db", username: $dbuser, password: $dbpasswd);
    $connexion->setAttribute(attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
    $connexion->exec(statement: "SET CHARACTER SET utf8");

    echo "Connexion réussie à la base de données.";
} catch (PDOException $e) {
    die("Erreur de connexion : " . $e->getMessage());
} catch (Exception $e) {
    die("Erreur : " . $e->getMessage());
}
```

DAILY SCRUM FIN (23/03/2025) :

Résultat :

Alexis s'est occupé du code Java concernant les filtres masquer et publier et a pu finir ces tâches sans soucis majeur.

Thomas a pu travailler sur la recherche par date et par semaine, ce sont les seules erreurs qu'il a rencontrées, la connexion n'a pas pu être finalisé à cause d'un problème de Bibliothèques qui ne se s'ajoute pas au projet. J'ai aussi réglé tous mes problèmes lors de l'itération précédente qui concernais l'impossibilité d'ajouter des librairies et le problème de variable lors de la connexion.

Pierre lui s'est occupé du chiffrement et du stockage du mot de passe utilisateur pour la connexion à la BDD, Des petites erreurs ont contraint la finalisation de cette tâche mais sera fini durant la semaine.

Pour la dernière itération nous essayeront donc de finir toutes les tâches qu'il nous reste à faire dans la mesure du possible.

Les tâches effectués lors de cette deuxième itération sont : US 6, T 3, US 3 et US 4

Tableau de Kanban de fin :

Thomas Crouan / Projet_Java_Moderation / Issue Boards

Development ▾ Search 🔍 ⚙️ ↗️

Open 2 +

- L'exécutable doit être mis à disposition sur un serveur FTP pour permettre le déploiement sur les postes de travail des modérateurs et du responsable du site. Cette version doit être paramétrée pour agir sur la base de données distante elle-même déployée
Facile Moyen
#13 Yesterday
- La nouvelle base de données doit être déployée sur le serveur d'applications, afin de permettre le test par le maître d'œuvre
Facile Moyen
#12 Yesterday

En cours 2 + ⚙️

- les paramètres d'accès à la base de données doivent être enregistrés dans un fichier séparé du code source et non versionnés. Le mot de passe doit être chiffré et la clef de chiffrement ne doit pas être stockée avec l'application
Moyen
#14 Yesterday
- Rédiger les tickets d'intervention, à destination de l'équipe de développement du site, nécessaires pour mettre à jour cette application
Facile Moyen
#15 Yesterday

Closed 11

- Recherche par : avis d'une date donnée, avis d'une semaine donnée.
Difficile Fini Moyen
#7 Yesterday
- Ajouter du critère « avis masqués » en plus des autres critères de sélection pour la recherche
Difficile Fini Moyen
#8 Yesterday
- Possibilité de démasquer un avis afin de le republier
Facile Fini Moyen
#10 Yesterday
- Creation des rôles : modérateur, responsable du site et authentification fonctionnelle
Finis Moyen
#6 Mar 17
- Possibilité de supprimer un avis inopportun
Facile Fini
#11 Mar 17