

2023/2024	Module DAP – Programmation Orientée Objets –
BTS SIO	Auteur : Groupe 1
2SLAM	Date de rédaction : 18/03/2024

Compte-rendu itération 1

Outils :

Git : <https://gitlab.com/ACANTIN/2slamaprojetapplijavarestog1>

Trello :

<https://trello.com/invite/b/ZdBuzJ6y/ATTI78548614401942bd5e899c209d4d57b2931A8A3E/application-java-resto>

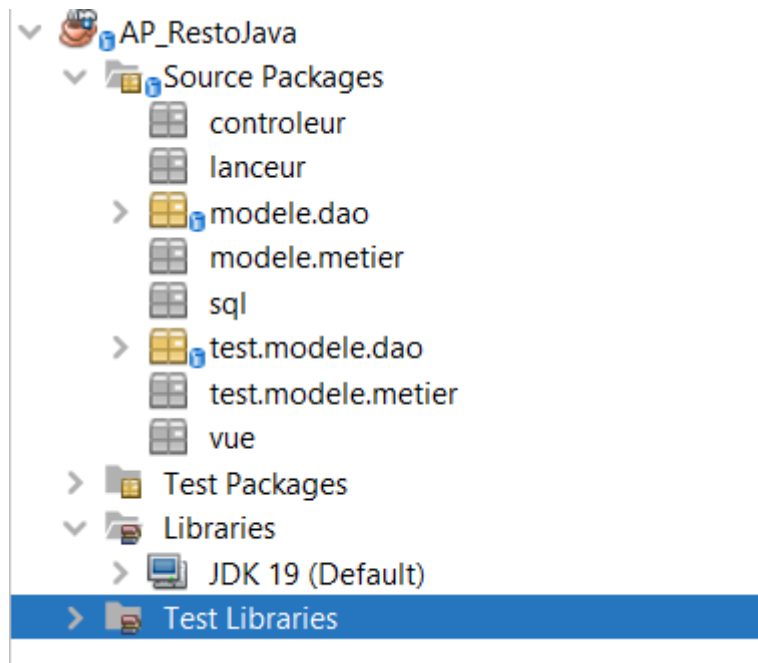
Excalidraw : https://excalidraw.com/#room=29ff0cc9385a1c7a2807,-juEmWtUdQ5LGgpQ_5Lmiw

Excalidraw est un outil de dessin en ligne où sont présente les maquettes

Objectifs généraux :

Nous avons analysé le cahier des charges et nous nous sommes réparti les tâches. Pour la première itération, Malaume Lheureux fait le compte rendu et les maquettes, dans le but de poser les bases de l'interface utilisateur. Alexis Cantin quant à lui, fait la preuve de concept. Pierre Kerlau s'affaire crée le squelette de l'application avec un modèle MVC et à rédiger le rapport d'itération.

Pour le choix de la technologie qui nous permettra de communiquer avec la base de données nous avons choisi de faire une couche DAO.



Itération 2 - Projet Java Resto

G1 : Cantin Alexis, Lheureux Malaume, Kerlau Pierre

Modalité d'accès :

GitLab : https://gitlab.com/2slam_g1_restojava/restojava.git

Gestion des tickets : https://gitlab.com/2slam_g1_restojava/restojava/-/boards

Maquette du projet :

https://excalidraw.com/#room=29ff0cc9385a1c7a2807,-juEmWtUdQ5LGgpQ_5Lmiw

Gestion de projet :

A la suite de problèmes avec le premier dépôt GitLab nous en avons recréé un, le tableau des tickets ont été refaits.

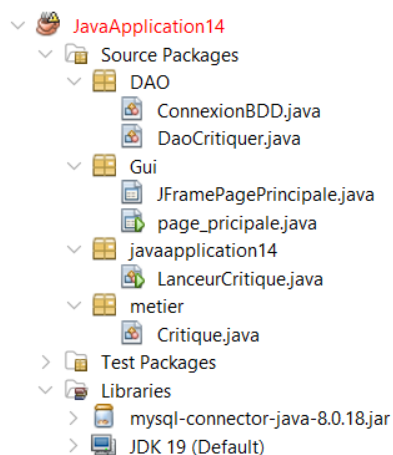
Dans la séance précédente nous avons mis en place le projet en analysant le cahier des charges, en choisissant d'utiliser une couche DAO pour accéder aux données et en construisant une maquette du projet.

Activité du dépôt framagit :

Avant de déposer nos avancées dans la branche main du git, nous ajoutons nos travaux sur la branche [enCoursDeValidation](#) puis nous échangeons pour valider ou modifier.

Documentation :

Pour cette séance, nous avons créé le modèle MVC de notre projet



Notre but était de créer une vue de connexion et une vue d'avis affichant à la fin de la séance, toutes les données présentes dans la table critiquer dans un table.

D'abord nous avons créé les interfaces graphiques de nos deux vues :

- GUI de connexion



- GUI des avis



Ensuite, l'objectif a été d'afficher les critiques dans la table. Pour ce faire, on doit commencer par extraire les données de la table critiquer.

Dans le fichier DAO/DaoCritiquer.java nous ajoutons ce code :

```
28 public static ArrayList<Critique> getAllCritique() throws SQLException {
29     ArrayList<Critique> lesCritiquesTrouvees = new ArrayList<>();
30     Connection cnx = ConnexionBDD.getConnexion();
31     PreparedStatement pstmt = cnx.prepareStatement("SELECT * FROM Critiquer ");
32
33     ResultSet rs = pstmt.executeQuery();
34     while (rs.next()) {
35         Critique unCritique = new Critique(
36             rs.getInt("idR"),
37             rs.getString("commentaire"),
38             rs.getInt("note"),
39             rs.getInt("idU")
40         );
41     };
42     lesCritiquesTrouvees.add(e: unCritique);
43 }
44 return lesCritiquesTrouvees;
45 }
46 }
```

Cela permet de récupérer toutes les données de la table critiquer et de les ajouter dans une ArrayList nommé Critique.

Ensuite nous avons du créer une classe Critique dans Modele.metier/Critique.java avec le constructeur de la classe

```
8 public class Critique {
9
10     private int idR;
11     private String commentaire;
12     private int note;
13     private int idU;
14
15     public Critique(int idR, String commentaire, int note, int idU) {
16         this.idR = idR;
17         this.commentaire = commentaire;
18         this.note = note;
19         this.idU = idU;
20     }
```

Puis créer la méthode toString que l'on crée avec l'outil "Insert code" de NetBeans

```
22 @Override
23 public String toString() {
24     return "Etape{" + "idR=" + idR + ", commentaire=" + commentaire + ", note=" + note + ", idU" + idU + '}';
25 }
```

Enfin, nous ajoutons de la même manière les accesseurs et les mutateurs

```
27 public int getIdR() {
28     return idR;
29 }
30
31 public void setIdR(int idR) {
32     this.idR = idR;
33 }
34
35 public String getCommentaire() {
36     return commentaire;
37 }
38
39 public void setCommentaire(String commentaire) {
40     this.commentaire = commentaire;
41 }
42
43 public int getNote() {
44     return note;
45 }
46
47 public void setNote(int note) {
48     this.note = note;
49 }
50
51 public int getIdU() {
52     return idU;
53 }
54
55 public void setIdU(int idU) {
56     this.idU = idU;
57 }
```

Une fois cela fait, il ne manque plus qu'à faire afficher les données extraites dans la table du fichier `Gui/JFramePagePrincipale.java`

```

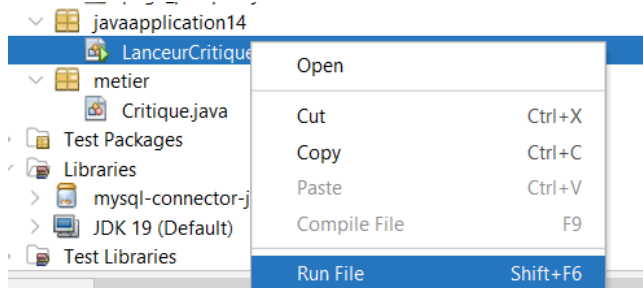
28 public JFramePagePrincipale() {
29     initComponents();
30
31     // Préparation du composant JTable
32     // Définir un modèle de table où les cellules ne sont pas éditables
33     modeleJTableEtapes = new DefaultTableModel() {
34         @Override
35         public boolean isCellEditable(int iRowIndex, int iColumnIndex) {
36             return false;
37         }
38     };
39     //définir les titres des colonnes
40     String[] titres = {"idR", "commentaire", "idU", "note"};
41     modeleJTableEtapes.setColumnIdentifiers(newIdentifiers: titres);
42     // affecter le modele à la JTable
43     jTableEtapes.setModel(dataModel:modeleJTableEtapes);
44
45     // Adapter la largeur des colonnes
46     jTableEtapes.getColumnModel().getColumn(columnIndex: 1).setPreferredWidth(preferredWidth: 400);
47
48     // Préparation du composant JComboBox
49     // affecter un nouveau modele à la JComboBox
50
51 }

```

Test :

Maintenant, il est temps de voir si les critiques sont affichés dans la table de la vue `JFramePagePrincipale.java`

Pour se faire, il suffit d'exécuter le fichier `LanceurCritique.java`



Et nous obtenons ceci :

idR	commentaire	idU	note
1	moyen	2	3
1	Très bonne entrecote, les frites sont maisons et délicieuses.	3	3
1	Très bon accueil.	5	4
1	5/5 parce que j'aime les entrecotes	6	4
1		7	5
2	bof.	2	2
2	À éviter...	3	1
2	Cuisine tres moyenne.	5	1
2		6	5
4		3	5

Masquer

Bilan de l'itération :

Pour cette séance, nous avons atteint nos objectifs, c'est-à-dire, créer les vues du projet, ainsi qu'afficher les données présentes dans la table critiquer de notre base de données dans la table de la vue principale de notre projet.

Nos objectifs pour la prochaine séance sont, dans un premier temps, de créer les méthodes de connexion permettant de donner accès à des fonctionnalités en fonction du niveau d'accès du compte. Ensuite, nous voulons créer des boutons permettant de masquer/démasquer et de supprimer les avis. Puis permettre de trier les critiques par date ou par semaine donnée. Enfin, nous voulons pouvoir afficher tous les avis masqués.

Itération 3 - Projet Java Resto

G1 : Cantin Alexis, Lheureux Malaume, Kerlau Pierre

Modalité d'accès :

Gitlab : https://gitlab.com/2slam_g1_restojava/restojava.git

Gestion des tickets : https://gitlab.com/2slam_g1_restojava/restojava/-/boards

Maquette du projet :

https://excalidraw.com/#room=29ff0cc9385a1c7a2807,-juEmWtUdQ5LGgpQ_5Lmiw

Gestion de projet :

à la suite de l'itération 2 ou nous avons fait en sorte que l'application puisse communiquer avec la base de données et qu'elle affiche les commentaires dans la table. Nous avons pour objectif de permettre une connexion et un accès à des outils selon le niveau de permission de l'utilisateur connecté. Nous avons également pour objectif de pouvoir masquer/réafficher des avis, ainsi que les supprimer.

Activité du dépôt framagit :

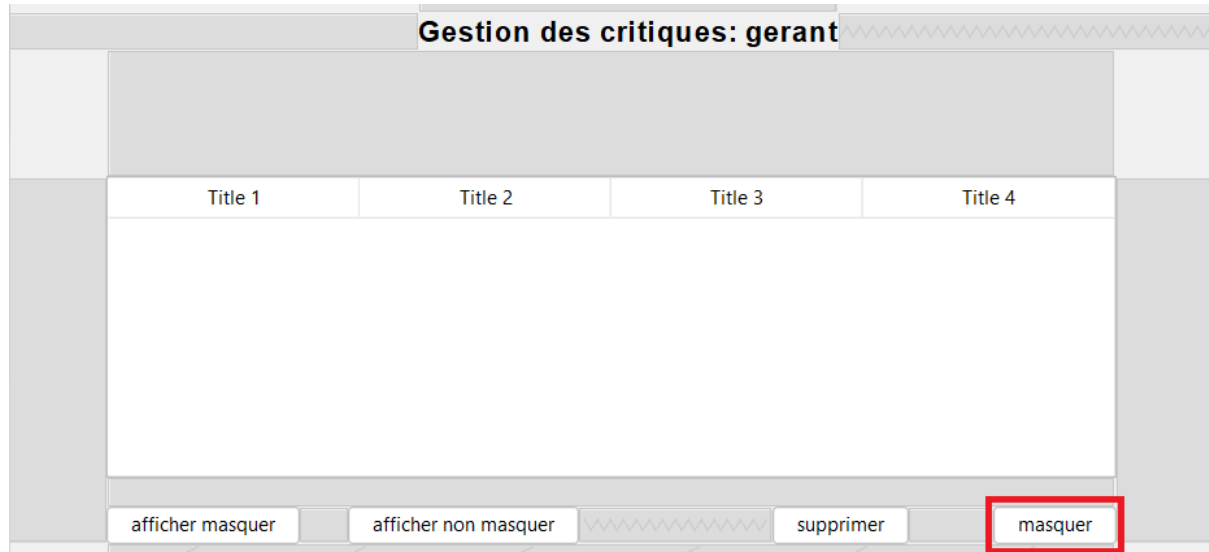
Toutes les activités sont ajoutées soit à la branche [EnCoursDeValidation](#) puis est merge vers la branche [main](#) une fois validé par l'équipe.

Documentation :

Outil masquer avis

Dans un premier temps nous avons créé une nouvelle colonne dans la table critiquer de notre base de donnée nommée “Masque” qui est de valeur 0 pour signifier que l’avis est affiché ou de valeur 1 si l’avis est masqué.

Pour la vue, dans le fichier Gui/JFramePrincipale.java, nous avons créé un JButtonMasquer



Ensuite nous avons créé la méthode DAO setMasqueStatue dans le fichier DAO/DaoCritique.java qui nous servira pour le bon fonctionnement du bouton :

```
/**
 * n° de la dernière étape d'une excursion
 * @param codeExcursion
 * @return nombre d'étapes de l'excursion d'id codeExcursion
 * @throws java.sql.SQLException
 */
public static int setMasqueStatue(int idU, int idR) throws SQLException {
    int nb;
    Connection cnx = ConnexionBDD.getConnexion();
    PreparedStatement pstmt = cnx.prepareStatement(string:"UPDATE critiquer SET Masque = 1 WHERE idU= ? and idR= ?");
    pstmt.setInt(1, idU);
    pstmt.setInt(2, idR);
    nb = pstmt.executeUpdate();
    return nb;
}
```

Ce code permet une connexion à la base de donnée, puis de modifier la valeur de “Masque” par 1 de l’avis sélectionné.

Ensuite nous avons ajouté ce code associé au bouton :

```

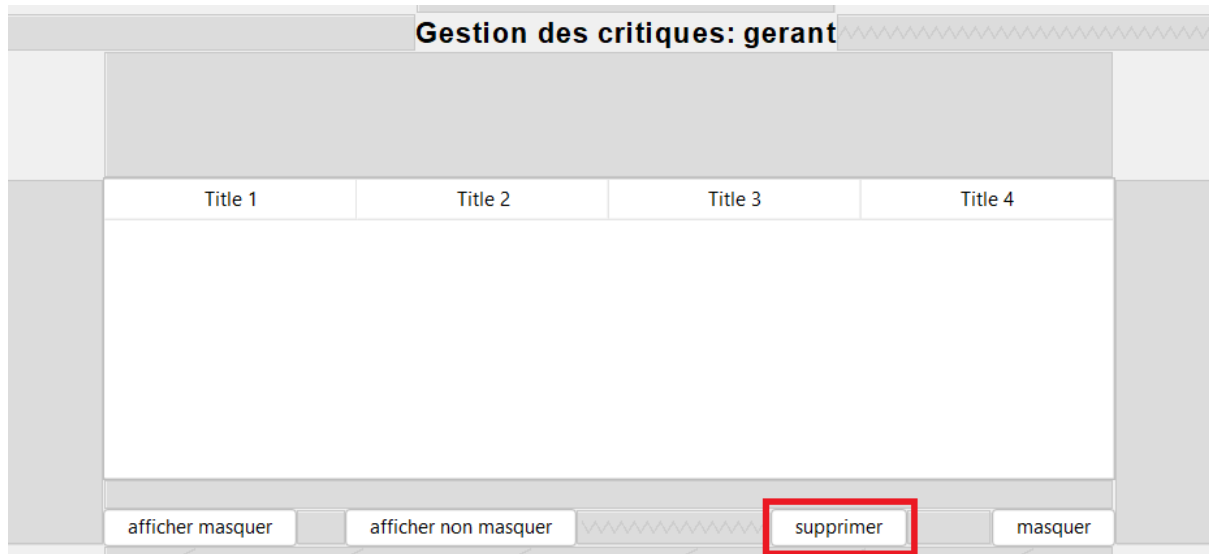
180 private void jButtonMasquerActionPerformed(java.awt.event.ActionEvent evt) {
181     int row= jTableEtapes.getSelectedRow();
182     String idU= jTableEtapes.getModel().getValueAt(rowIndex: row, columnIndex: 2).toString();
183     String idR= jTableEtapes.getModel().getValueAt(rowIndex: row, columnIndex: 0).toString();
184     System.out.print("idR:"+idR+"idU"+idU);
185     try {
186         DaoCritiquer.setMasqueStatus(idU: Integer.parseInt(s: idU), idR: Integer.parseInt(s: idR));
187     } catch (SQLException ex) {
188         Logger.getLogger(name: JFramePagePrincipale.class.getName()).log(level: Level.SEVERE, msg: "changement de masque imp
189     }
190     List<Critique> lesCritiques;
191     try {
192         lesCritiques = DaoCritiquer.getAllCritiqueNonMasque();
193         majAffichageCritique(lesCritiques);
194     } catch (SQLException ex) {
195         Logger.getLogger(name: JFramePagePrincipale.class.getName()).log(level: Level.SEVERE, msg: "getAllCritiqueNonMasque"
196     }
197 }
198

```

Ce code permet de récupérer les information de la ligne sélectionné, puis de masquer l'avis avec la méthode setMasqueStatue.

Outil supprimer avis

Ensuite nous avons créé un outil pour supprimer un avis, pour ce faire nous avons dans un premier temps créé un nouveau bouton dans l'interface utilisateur nommé JButtonSupprimer



Ensuite, nous avons créé une méthode DAO permettant de supprimer un avis dans le fichier DAO/DaoCritique.java :

```

public static int deleteCritique(int idU, int idR) throws SQLException {
    int nb;
    Connection cnx = ConnexionBDD.getConnexion();
    PreparedStatement pstmt = cnx.prepareStatement("delete from critiquer where idU= ? and idR= ?");

    pstmt.setInt(1, idU);
    pstmt.setInt(2, idR);
    nb = pstmt.executeUpdate();
    return nb;
}

```

Enfin, nous avons ajouté ce code associé au bouton JButtonSupprimer dans le fichier Gui/JFramePrincipale.java :

```
private void jButtonSupprimerActionPerformed(java.awt.event.ActionEvent evt) {
    int row= jTableCritique.getSelectedRow();
    String idU= jTableCritique.getModel().getValueAt( row, columnIndex: 2).toString();
    String idR= jTableCritique.getModel().getValueAt( row, columnIndex: 0).toString();

    try {
        DaoCritiquer.deleteCritique( idU: Integer.parseInt( s: idU), idR: Integer.parseInt( s: idR));
    } catch (SQLException ex) {
        Logger.getLogger( name: JFramePageProprio.class.getName()).log( level: Level.SEVERE, msg: "delete de critique impossible", thrown: ex);
    }

    List<Critique> lesCritiques;
    try {
        lesCritiques = DaoCritiquer.getAllCritique();
        majAffichageCritique(lesCritiques);
    } catch (SQLException ex) {
        Logger.getLogger( name: JFramePageProprio.class.getName()).log( level: Level.SEVERE, msg: "getAllCritique", thrown: ex);
    }
}
```

Outil afficher les avis masqués/non masqués

Pour afficher les avis non masqués on crée d'abord une méthode getAllCritiquesNonMasques

```
public static ArrayList<Critique> getAllCritiqueNonMasque() throws SQLException {
    ArrayList<Critique> lesCritiquesTrouvees = new ArrayList<>();
    Connection cnx = ConnexionBDD.getConnexion();
    PreparedStatement pstmt = cnx.prepareStatement( string: "SELECT * FROM critiquer where masque=0 ");

    ResultSet rs = pstmt.executeQuery();
    while (rs.next()) {
        Critique unCritique = new Critique(
            idR: rs.getInt( string: "idR"),
            commentaire: rs.getString( string: "commentaire"),
            note: rs.getInt( string: "note"),
            idU: rs.getInt( string: "idU"),
            masque: rs.getInt( string: "masque")
        );

        lesCritiquesTrouvees.add( e: unCritique);
    }
}
```

Cette méthode sélectionne toutes les critiques qui ont la valeur 0 dans la colonne masque. Ensuite, pour les afficher on crée un bouton dans notre JFrame principale.

Module THALI-MINI

— □ ×

Gestion des critiques: gerant

idR	commentaire	idU	note	masque	
1	moyen	2	3	0	▲
2	bof.	2	2	1	
1	Très bonne entrecote, les frites sont maisons et delicieuses.	3	3	0	
2	À éviter...	3	1	1	
1	Très bon accueil.	5	4	0	≡
2	Cuisine tres moyenne.	5	1	0	
5	Cuisine correcte.	5	3	0	
6	Cuisine de qualité.	5	4	1	
1	5/5 parce que j'aime les entrecotes	6	4	0	
2		6	5	1	▼

afficher masquer

afficher non masquer

supprimer

masquer

Et on y ajoute ce code, qui va appeler la méthode pour afficher les avis non masqués.

```
//affiche les avis non masquer
private void jButtonAffNonMasqActionPerformed(java.awt.event.ActionEvent evt) {

    List<Critique> lesCritiques;
    try {
        lesCritiques = DaoCritiquer.getAllCritiqueNonMasque();
        majAffichageCritique(lesCritiques);
    } catch (SQLException ex) {
        Logger.getLogger(name: JFramePagePrincipale.class.getName()).log(Level.SEVERE, msg: "getAllCritiqueNonMasque", thrown: ex);
    }
}
```

Pour afficher les avis masqués on réutilise le code de la méthode
getAllCritiquesNonMasques mais on change la valeur par 1 et on l'appelle
GetAllCritiqueMasque

```
public static ArrayList<Critique> getAllCritiqueMasque() throws SQLException {
    ArrayList<Critique> lesCritiquesTrouvees = new ArrayList<>();
    Connection cnx = ConnexionBDD.getConnexion();
    PreparedStatement pstmt = cnx.prepareStatement("SELECT * FROM critiquer where masque=1 ");

    ResultSet rs = pstmt.executeQuery();
    while (rs.next()) {
        Critique unCritique = new Critique(
            idR: rs.getInt("idR"),
            commentaire: rs.getString("commentaire"),
            note: rs.getInt("note"),
            idU: rs.getInt("idU"),
            masque: rs.getInt("masque")
        );

        lesCritiquesTrouvees.add(unCritique);
    }
}
```

Puis on associe ce code qui va appeler notre méthode GetAllCritiqueMasque à un bouton
JButtonAffNonMas

Module THALI-MINI

Gestion des critiques: gerant

idR	commentaire	idU	note	masque
1	moyen	2	3	0
2	bof	2	2	1
1	Très bonne entrecote, les frites sont maisons et délicieuses.	3	3	0
2	À éviter...	3	1	1
1	Très bon accueil.	5	4	0
2	Cuisine tres moyenne.	5	1	0
5	Cuisine correcte.	5	3	0
6	Cuisine de qualité.	5	4	1
1	5/5 parce que j'aime les entrecotes	6	4	0
2		6	5	1

afficher masquer
afficher non masquer
supprimer
masquer

```
//affiche les avis non masquer
private void jButtonAffNonMasqActionPerformed(java.awt.event.ActionEvent evt) {

    List<Critique> lesCritiques;
    try {
        lesCritiques = DaoCritiquer.getAllCritiqueNonMasque();
        majAffichageCritique(lesCritiques);
    } catch (SQLException ex) {
        Logger.getLogger(name: JFramePagePrincipale.class.getName()).log(Level.SEVERE, msg: "getAllCritiqueNonMasque", thrown: ex);
    }
}
```

Test :

Test pour masquer un avis

Dans un premier temps nous allons lancer le fichier lanceurCritique.java, nous arrivons sur cette page :

Gestion des critiques: gerant

idR	commentaire	idU	note	masque
1	moyen	2	3	0
2	bof.	2	2	1
1	Très bonne entrecote, les frites sont maisons et delicieuses.	3	3	0
2	À éviter...	3	1	1
1	Très bon accueil.	5	4	0
2	Cuisine tres moyenne.	5	1	0
5	Cuisine correcte.	5	3	0
6	Cuisine de qualité.	5	4	1
1	5/5 parce que j'aime les entrecotes	6	4	0
2		6	5	1

On choisit l'avis à masquer, ici le premier avis de la liste et on clique sur masquer

idR	commentaire	idU	note	masque
1	moyen	2	3	0
2	bof.	2	2	1
1	Très bonne entrecote, les frites sont maisons et delicieuses.	3	3	0
2	À éviter...	3	1	1
1	Très bon accueil.	5	4	0
2	Cuisine tres moyenne.	5	1	0
5	Cuisine correcte.	5	3	0
6	Cuisine de qualité.	5	4	1
1	5/5 parce que j'aime les entrecotes	6	4	0
2		6	5	1

Et on voit que la valeur de la colonne masque est passé de 0 à 1, cela signifie que l'avis à été masqué

idR	commentaire	idU	note	masque
1	moyen	2	3	1
2	bof.	2	2	1
1	Très bonne entrecote, les frites sont maisons et delicieuses.	3	3	0
2	À éviter...	3	1	1
1	Très bon accueil.	5	4	0
2	Cuisine tres moyenne.	5	1	0
5	Cuisine correcte.	5	3	0
6	Cuisine de qualité.	5	4	1
1	5/5 parce que j'aime les entrecotes	6	4	0
2		6	5	1

Afficher uniquement les avis masqués

Pour afficher les avis masqués il suffit de cliquer sur le bouton “afficher masquer”

On obtient ceci :

Gestion des critiques: gerant

idR	commentaire	idU	note	masque
1	moyen	2	3	1
2	bof.	2	2	1
2	À éviter...	3	1	1
6	Cuisine de qualité.	5	4	1
2		6	5	1
8		6	1	1
1		7	5	1

Afficher uniquement les avis non masqués

Pour afficher les avis non-masqués on clique sur le bouton “afficher non masquer”

On obtient ceci :

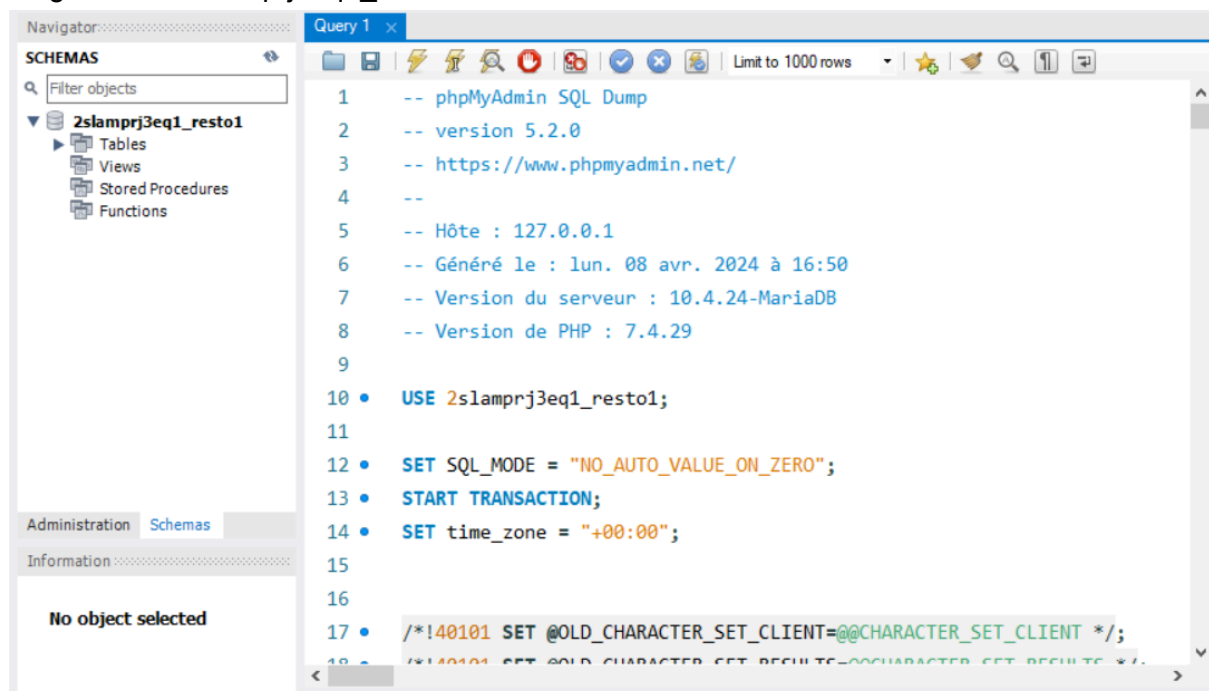
idR	commentaire	idU	note	masque
1	Très bonne entrecote, les frites sont maisons et délicieuses.	3	3	0
1	Très bon accueil.	5	4	0
2	Cuisine tres moyenne.	5	1	0
5	Cuisine correcte.	5	3	0
1	5/5 parce que j'aime les entrecotes	6	4	0

Supprimer un avis

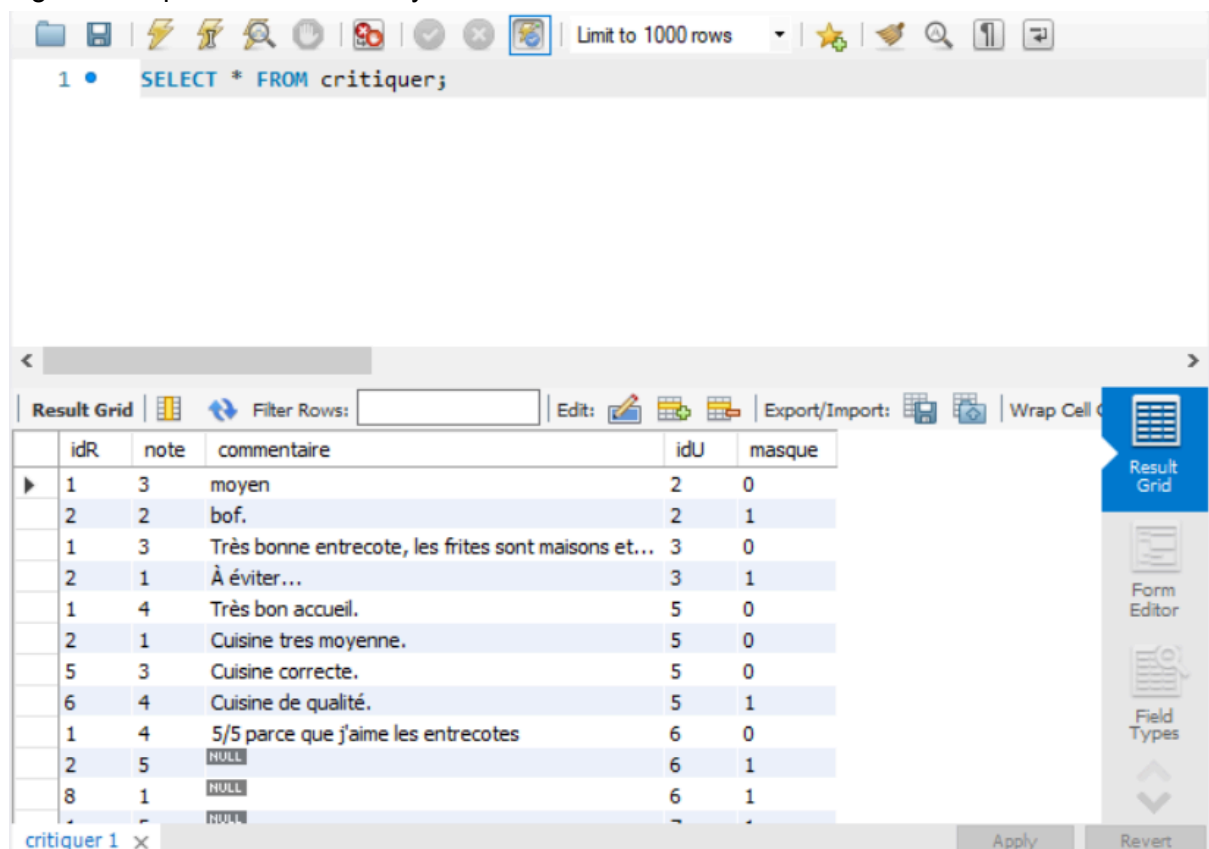
Pour supprimer un avis il suffit de sélectionner un avis puis de cliquer sur le bouton supprimer


```
1 • CREATE DATABASE 2slamprj3eq1_resto1;
```

Ensuite nous importons la base de donnée que l'on a utilisé pour le développement en ajoute la ligne "USE 2slamprj3eq1_resto1;".



Enfin pour tester si la base de donnée est bien importé on fait une simple requête et on regarde ce que nous renvoie MySQL Workbench.



Il nous reste à déployer le projet depuis Netbeans vers le serveur. Pour ce faire nous commençons par exporter le projet en fichier JAR.




Project Properties -> Build -> Packaging -> Build JAR after compiling

Puis nous nous connectons au serveur avec FileZilla et il nous reste juste à transférer le projet.

(Je n’ai pas d’image car Alexis s’en est occupé et pour éviter de faire une erreur je ne préfère pas refaire la manipulation).

Pour télécharger notre projet il faut aller à cette adresse : <http://10.15.253.250/2slamprj3eq1/>

Index of /2slamprj3eq1

Name	Last modified	Size	Description
<hr/>			
 Parent Directory		-	
 dist/	2024-04-08 17:54	-	
 restojava/	2024-04-08 17:43	-	

Puis il suffit de télécharger le fichier “dist”.
Une fois le fichier téléchargé nous n’avons plus qu’à lancer le fichier JAR, puis nous arrivons cette interface :



Bilan de l’itération :

Pour cette itération, nous avons fait plusieurs user story et nous avons déployé notre application. La principale difficulté à été de faire le code pour la connexion à l’application, que nous n’avons pas pu finir. Une autre difficulté, bien moindre à été de comprendre comment exporter le projet au format JAR.