Rapport d’Itération n°2

1. Documentation utilisateur

1.1. Mode opératoire

- **Nom du livrable :** Projet JAVA modération
- **Version : 0.2**
- **Date de livraison : 17/03/2025**
- **Description :** Les propriétaires du site r3st0.fr souhaitent se doter d’une application Java qui sera installée sur les postes de travail des modérateurs, permettant de contrôler les avis déposés et de supprimer ou de bloquer la publication des avis ne respectant pas la charte d’utilisation du site ; chaque modérateur devra s’authentifier pour pouvoir utiliser l’application.

Installation :

- Pré-requis :

  - java <= 18

  -un IDE (netBeans, vs code)

  -wamp/xamp avec la BDD importée

- Procédure d'installation :
  1. Récupérer le projet depuis le dépôt gitLab (git clone https://gitlab.com/APeixoto/modo-modo.git)
  2. Importer la BDD fournit avec le projet dans wamp/xamp
  3. Le projet est prêt à être exécuté

Choix de la technologie d’accès aux données :

Nous avons choisi d'utiliser une **couche DAO** dans notre projet pour plusieurs raisons :

- **Simplicité et contrôle** : Le DAO nous permet de gérer directement les requêtes SQL et d’avoir un contrôle total sur l'accès aux données, sans complexité supplémentaire.
- **Architecture adaptée** : L’application étant locale et monolithique, une **API REST** serait inutilement lourde car il n’y a pas de communication entre services ou clients distants.
- **Alternative à JPA** : JPA apporte de la facilité pour des projets complexes, mais il impose un certain cadre et génère des surcharges inutiles pour un projet simple. Le DAO reste plus léger et adapté ici.

- **Bonne pratique** : Le DAO respecte la séparation des responsabilités et facilite la maintenance du code.

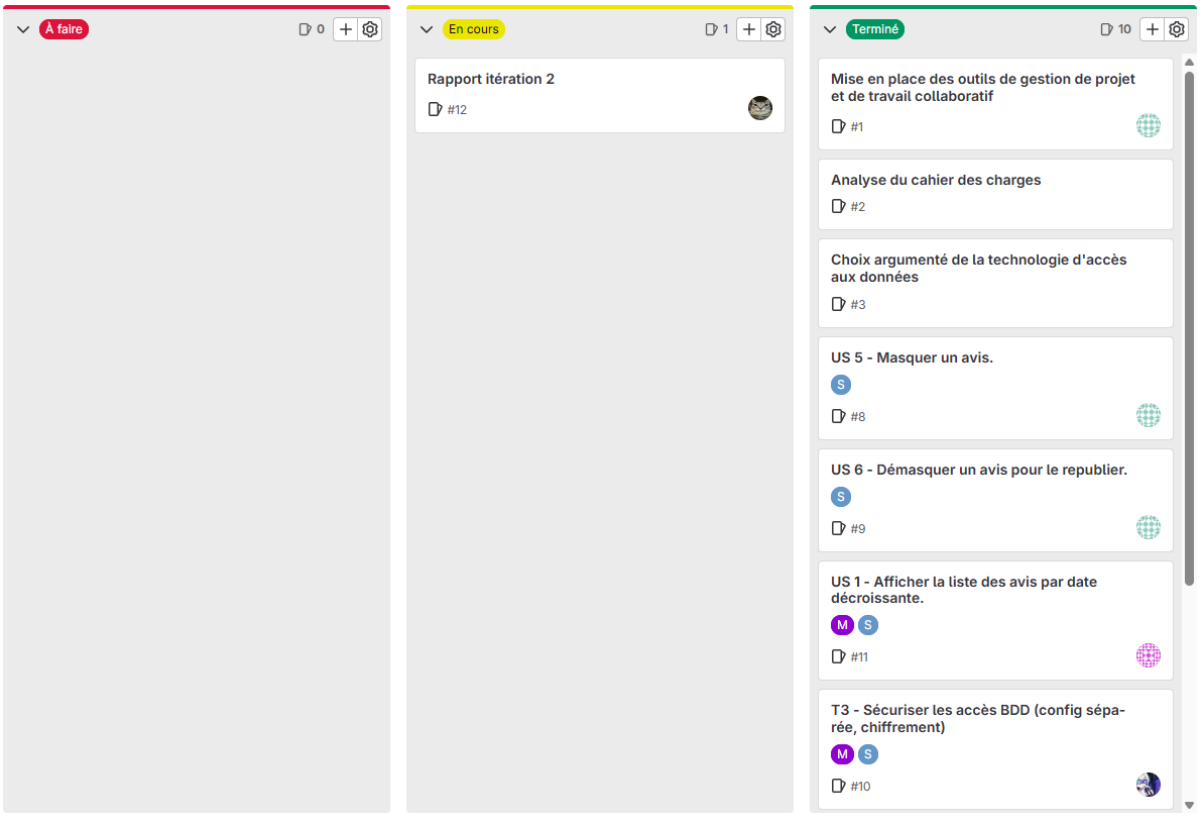
2. Documentation technique

2.1. Gestion de projet

Comptes-rendus des Daily Scrums :

Date	Participants	Avancement	Problèmes rencontrés	Prochaines actions
24/02/2025	Loric Anthony Titouan Stefen	Argumentation du choix de la technologie et P.O.C effectué	Problème de synchronisation avec le fichier restoJdbc dans le dossier « src » et « build »	Maquette à terminer, déterminer les prochains objectifs du projet
03/03/2025	Loric Anthony Titouan Stefen	Avancement sur la détermination de la priorisation des tâches		Commencer **US 5**, **US 6** et **T 3**
10/03/2025	Loric Anthony Titouan Stefen	Organisation des tâches prioritaires		Commencer **T 1**, **US 3** et **T 2**

Répartition des tâches :



Dépôt de code :

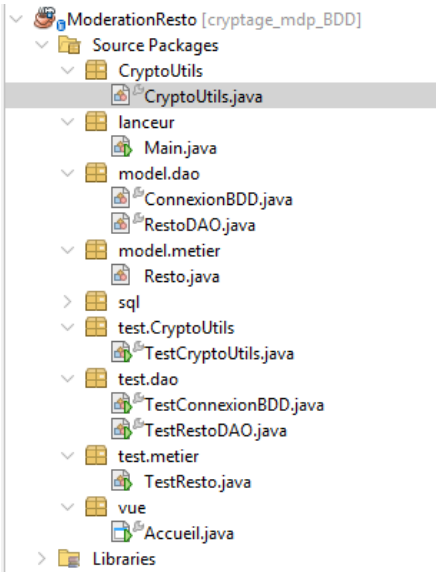
- Adresse du dépôt : [https://gitlab.com/APeixoto/modo-modo]
- Branche de travail de l'itération : [Main]

2.2. Code source

- **Branche Git associée :** [https://gitlab.com/APeixoto/modo-modo/-/tree/main/ModerationResto?ref_type=heads]

- **Évolutions du code :**

- Arborescence du projet après nos modifications et ajouts pour le **T3**:



- Création du package et de sa classe CryptoUtils afin d'ajouter les méthodes de cryptage et décryptage de mot de passe :

```java
package CryptoUtils;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

/**
 *
 * @author loric
 */
public class CryptoUtils {
    private static final String ALGORITHM = "AES";

    public static String encrypt(String data, String secretKey) throws Exception {
        SecretKeySpec keySpec = new SecretKeySpec(secretKey.getBytes(), ALGORITHM);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, keySpec);
        byte[] encryptedData = cipher.doFinal(data.getBytes());
        return Base64.getEncoder().encodeToString(encryptedData);
    }

    public static String decrypt(String encryptedData, String secretKey) throws Exception {
        SecretKeySpec keySpec = new SecretKeySpec(secretKey.getBytes(), ALGORITHM);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, keySpec);
        byte[] decryptedData = cipher.doFinal(Base64.getDecoder().decode(encryptedData));
        return new String(decryptedData);
    }
}
```

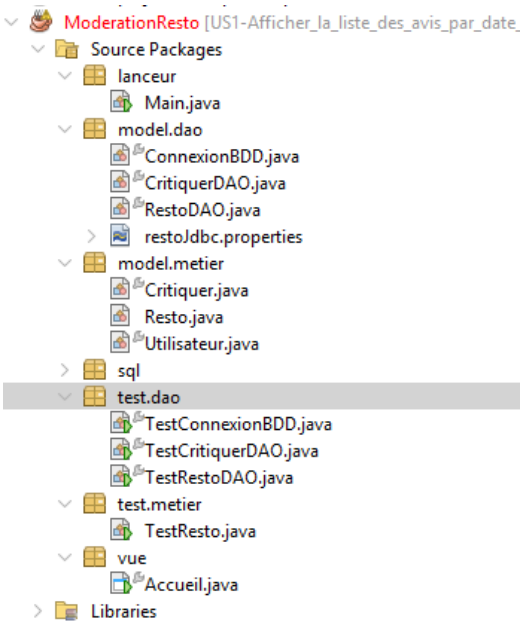
- Utilisation de la méthode lors de la connexion à la base de données :

```java
public class ConnexionBDD {

    private static Connection cnx;

    /**
     * Retourner une connexion ; la créer si elle n'existe pas...
     *
     * @return : objet de type java.sql.Connection
     */
    public static Connection getConnexion() throws SQLException, FileNotFoundException, IOException, Exception {
    if (cnx == null) {
        Properties propertiesJdbc = new Properties(); // objet de propriétés (paramètres de l'application) pour Jdbc
        InputStream input; // flux de lecture des properties

        // Chargement des paramètres du fichier de properties
        ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
        String pathToProperties = "../doc/restoJdbc.properties";
        input = new FileInputStream(pathToProperties);

        // Check if the input stream is null
        if (input == null) {
            throw new FileNotFoundException("Property file 'restoJdbc.properties' not found in the classpath");
        }

        propertiesJdbc.load(input);

        // Récupération de la clé de cryptage et du mot de passe
        String secretKey = propertiesJdbc.getProperty("key");
        String encryptedPassword = propertiesJdbc.getProperty("mdpBD");

        // Décryptage du mot de passe
        String decryptedPassword = CryptoUtils.decrypt(encryptedPassword, secretKey);

        cnx = DriverManager.getConnection(propertiesJdbc.getProperty("url"), propertiesJdbc.getProperty("utilBD"), decryptedPassword);
        System.out.println("getConnexion : " + propertiesJdbc.getProperty("url"));
    }
    return cnx;
}

}
```

- Création du test unitaire de ces nouvelles méthodes :

```java
package test.CryptoUtils;

import CryptoUtils.CryptoUtils;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class TestCryptoUtils {

    public static void main(String[] args) throws Exception {
        // Charger les propriétés depuis le fichier
        Properties properties = new Properties();
        String pathToProperties = "../doc/restoJdbc.properties";
        try (FileInputStream input = new FileInputStream(pathToProperties)) {
            properties.load(input);
        }

        // Récupérer la clé de cryptage depuis les propriétés
        String secretKey = properties.getProperty("key");

        // Mot de passe à tester
        String mdp = "secret";
        String mdpCrypte;
        String mdpDecrypte;

        // Crypter le mot de passe
        mdpCrypte = CryptoUtils.encrypt(mdp, secretKey);
        System.out.println("Mot de passe crypté : " + mdpCrypte);

        // Décrypter le mot de passe
        mdpDecrypte = CryptoUtils.decrypt(mdpCrypte, secretKey);
        System.out.println("Mot de passe décrypté : " + mdpDecrypte);

        // Vérifier que le mot de passe décrypté correspond au mot de passe d'origine
        if (mdp.equals(mdpDecrypte)) {
            System.out.println("Le mot de passe a été correctement décrypté !");
        } else {
            System.out.println("Erreur : Le mot de passe décrypté ne correspond pas au mot de passe d'origine.");
        }
    }
}
```

- Arborescence du projet après nos modifications et ajouts pour le **US1** :



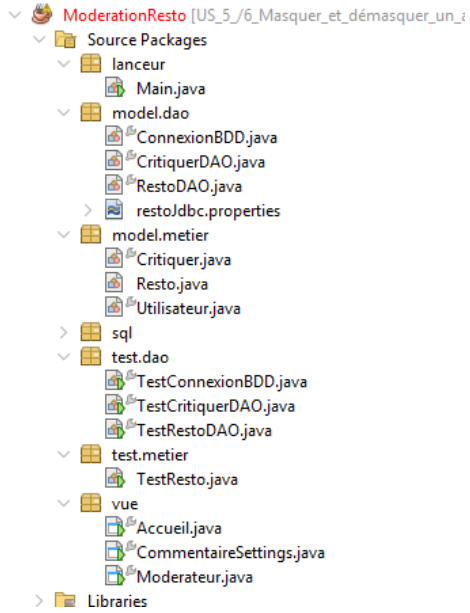
- Création de la classe DAO qui affiche les avis par ordre croissant :

```java
package model.dao;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 *
 * @author Titouan
 */
public class CritiquerDAO {

    public static ArrayList<String> getCommentaireByDate() throws SQLException, IOException {
        ArrayList<String> lesAvis = new ArrayList<>();
        Connection cnx = ConnexionBDD.getConnexion();
        PreparedStatement pstmt = cnx.prepareStatement("SELECT commentaire FROM critiquer WHERE date IS NOT NULL ORDER BY date DESC");
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            String unAvis = rs.getString("commentaire");
            lesAvis.add(unAvis);
        }
        return lesAvis;
    }

}
```

- Et également son fichier de test unitaire :

```java
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import model.dao.ConnexionBDD;
import model.dao.CritiquerDAO;
import model.metier.Critiquer;

/**
 *
 * @author Titouan
 */
public class TestCritiquerDAO {
        /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Test 1 getAll
        System.out.println("\n Test 1 : CritiquerDAO.getCommentaireByDate");
        try {
            ArrayList<String> lesAvis = CritiquerDAO.getCommentaireByDate();
            for (int i = 0; i < lesAvis.size(); i++) {
                System.out.println(lesAvis.get(i));
            }
        } catch (SQLException ex) {
            System.out.println("TestRestoDAO - échec getCommentaireByDate : " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("TestRestoDAO - échec getCommentaireByDate : " + ex.getMessage());
        }

        // Fermeture de la connexion
        try {
            ConnexionBDD.getConnexion().close();
            System.out.println("\nConnexion à la BDD fermée");
        } catch (SQLException ex) {
            System.out.println("TestDaoCategorie - échec de la fermeture de la connexion : " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("TestDaoCategorie - échec de la fermeture de la connexion : " + ex.getMessage());
        }
    }

}
```

- Arborescence du projet après nos modifications et ajouts pour le **US5** et **US6**:



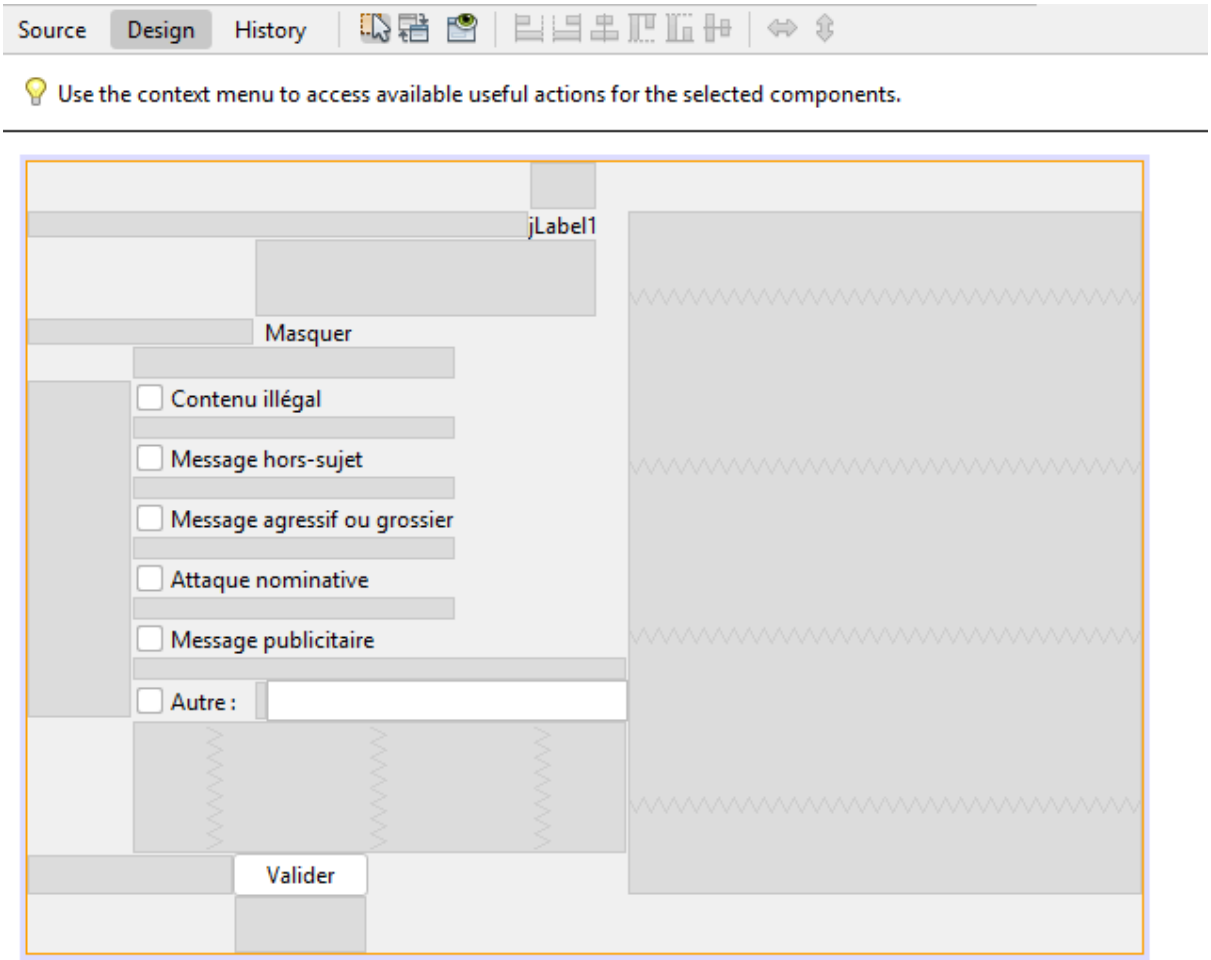
- Modification de la classe critiquerDAO afin d'y ajouter une méthode qui récupère les commentaires masqués :

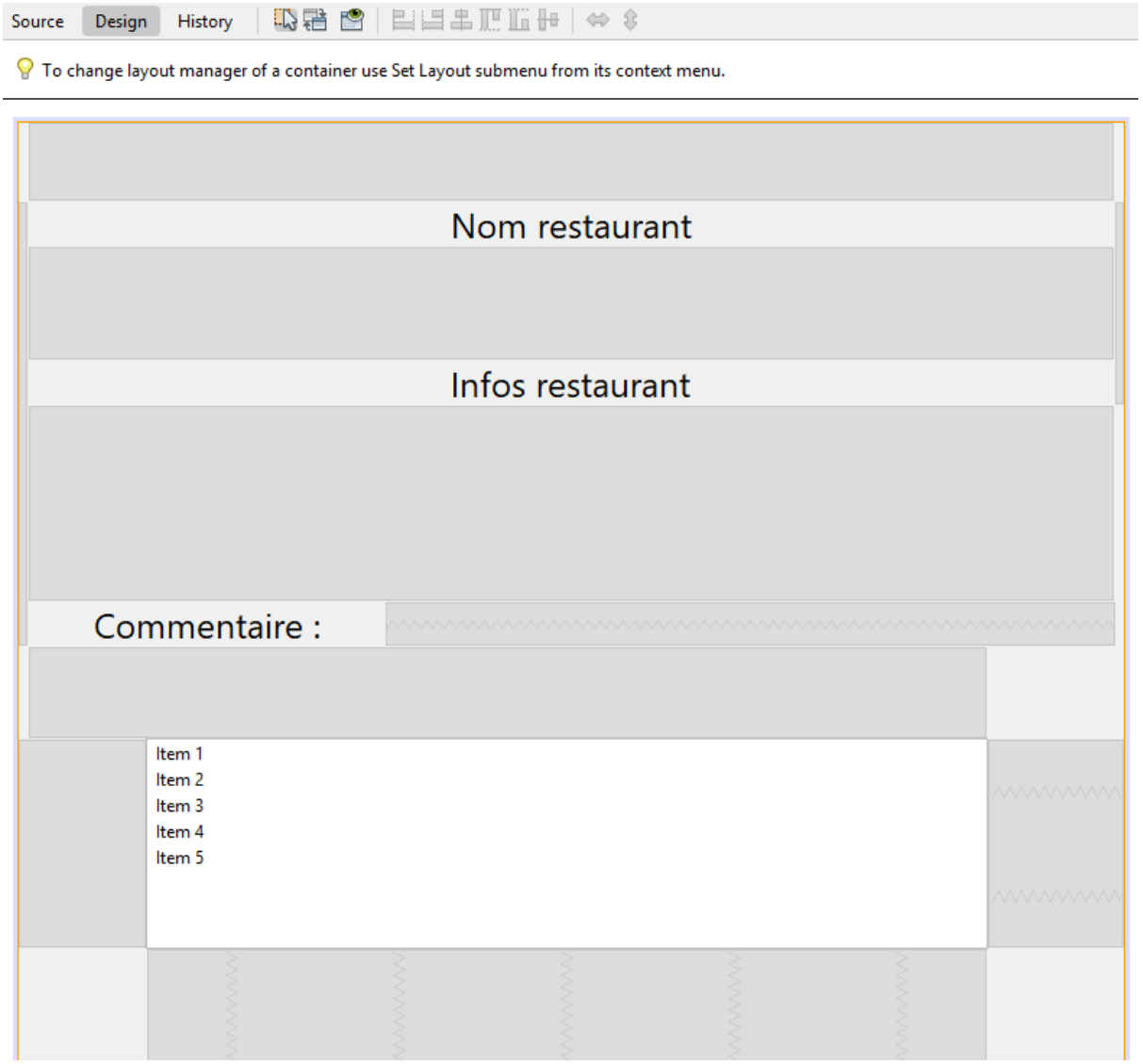
```java
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 *
 * @author Titouan
 */
public class CritiquerDAO {

    public static ArrayList<String> getCommentaireByDate() throws SQLException, IOException {
        ArrayList<String> lesAvis = new ArrayList<>();
        Connection cnx = ConnexionBDD.getConnexion();
        PreparedStatement pstmt = cnx.prepareStatement("SELECT commentaire FROM critiquer WHERE date IS NOT NULL ORDER BY date DESC");
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            String unAvis = rs.getString("commentaire");
            lesAvis.add(unAvis);
        }
        return lesAvis;
    }

    public static ArrayList<String> getCommentaireByDateNoMasq() throws SQLException, IOException {
        ArrayList<String> lesAvis = new ArrayList<>();
        Connection cnx = ConnexionBDD.getConnexion();
        PreparedStatement pstmt = cnx.prepareStatement("SELECT commentaire FROM critiquer WHERE date IS NOT NULL and visible = 1 ORDER BY date DESC");
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            String unAvis = rs.getString("commentaire");
            lesAvis.add(unAvis);
        }
        return lesAvis;
    }

}
```

- Modification dans le test unitaire de la classe DAO des avis afin d'y faire la même chose pour les commentaires masqués :

```java
public class TestCritiquerDAO {
        /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Test 1 getCommentaireByDate
        System.out.println("\n Test 1 : CritiquerDAO.getCommentaireByDate");
        try {
            ArrayList<String> lesAvis = CritiquerDAO.getCommentaireByDate();
            for (int i = 0; i < lesAvis.size(); i++) {
                System.out.println(lesAvis.get(i));
            }
        } catch (SQLException ex) {
            System.out.println("TestRestoDAO - échec getCommentaireByDate : " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("TestRestoDAO - échec getCommentaireByDate : " + ex.getMessage());
        }

        // Test 2 getCommentaireByDateNoMasq
        System.out.println("\n Test 1 : CritiquerDAO.getCommentaireByDateNoMasq");
        try {
            ArrayList<String> lesAvis = CritiquerDAO.getCommentaireByDateNoMasq();
            for (int i = 0; i < lesAvis.size(); i++) {
                System.out.println(lesAvis.get(i));
            }
        } catch (SQLException ex) {
            System.out.println("TestRestoDAO - échec getCommentaireByDateNoMasq : " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("TestRestoDAO - échec getCommentaireByDateNoMasq : " + ex.getMessage());
        }
```

- Création des interfaces graphiques :


Source
Design
History
Use the context menu to access available useful actions for the selected components.
jLabel1
Masquer
Contenu illégal
Message hors-sujet
Message agressif ou grossier
Attaque nominative
Message publicitaire
Autre :
Valider



2.4. Tests unitaires et d'intégration

Jeux d'essai :

- TestCryptoUtils -> Crypte le mot de passe de connexion à la BDD et l'affiche.

- TestCritiquerDAO -> Affiche les avis de la BDD rangés dans l'ordre décroissant.

- TestCritiquerDAO -> Affucge les avis de la BDD non masqués.

Traces d'exécution commentées :
- TestCryptoUtils :

```
ant -f "C:\\Users\\Stefen R\\OneDrive\\Documents\\NetBeansProjects\\modo-modo\\ModerationResto" -Dnb.internal.action.name=run.single -D
init:
Deleting: C:\Users\Stefen R\OneDrive\Documents\NetBeansProjects\modo-modo\ModerationResto\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\Stefen R\OneDrive\Documents\NetBeansProjects\modo-modo\ModerationResto\build\built-jar.properties
Compiling 1 source file to C:\Users\Stefen R\OneDrive\Documents\NetBeansProjects\modo-modo\ModerationResto\build\classes
compile-single:
run-single:
Mot de passe crypté : fnw7bmWaVAGHA62TyMOyTQ==
Mot de passe décrypté : secret
Le mot de passe a été correctement décrypté !
BUILD SUCCESSFUL (total time: 1 second)
```

- TestCritiquerDAO :

```
 Test 1 : CritiquerDAO.getCommentaireByDate
getConnexion : jdbc:mysql://localhost:3306/resto2
Très bon accueil :)
Excellent.
Bon accueil.
Cuisine de qualité.
Rapide.
Cuisine correcte.
Cuisine tres moyenne.
À éviter...
bof.
5/5 parce que j'aime les entrecotes
Très bon accueil.
Très bonne entrecote, les frites sont maisons et delicieuses.
moyen

Connexion à la BDD fermée
```

- TestCritiquerDAO (avis non masqués) :

```
 Test 1 : CritiquerDAO.getCommentaireByDateNoMasq
Très bon accueil :)
Excellent.
Bon accueil.
Cuisine de qualité.
Rapide.
Cuisine correcte.
Cuisine tres moyenne.
À éviter...
bof.
5/5 parce que j'aime les entrecotes
Très bon accueil.
Très bonne entrecote, les frites sont maisons et delicieuses.
moyen

Connexion à la BDD fermée
```

2.4. Priorisation des tâches

Estimation des complexités manquantes :

ID		Description	Criticité	Complexité (estimée)
US 1		Afficher la liste des avis par date décroissante.	M	S
US 2		Authentification avec rôles et stockage en BDD.	M	L
US 3		Sélectionner les avis par date ou semaine.	M	M
US 4		Filtrer par « avis masqués ».	W	S
US 5		Masquer un avis.	S	S
US 6		Démasquer un avis pour le republier.	S	S
US 7		Supprimer un avis inopportun.	C	S
T 1		Déployer la nouvelle BDD sur le serveur.	M	S
T 2		Mettre l'exécutable sur un FTP connecté à la BDD distante.	M	M
T 3		Sécuriser les accès BDD (config séparée, chiffrement).	S	M
T 4		Rédiger les tickets pour l'équipe R3st0.fr pour gérer les impacts du système de modération.	W	M

A faire :

- **Itération 3** : Enchaîner avec **T 1**, **US 3**, puis **T 2**.
- **Itération 4** : Finaliser avec **US 2**, **US 7**, **US 4**, **T 4**.

2.5. Bilan de l'itération

- **Travail réalisé :**

  - Les tâches **US1**, **US5**, **US6** et **T3** ont été réalisées. La base de données a également reçu des ajouts pour certaines tâches.
  Les tâches qui devaient absolument être complétées sur cette itération sont faites et nous avons pu même en faire une qui était destinée à l'itération suivante.

- **Travail restant à faire :**

  - Le travail de cette itération est fait, il nous faut donc poursuivre sur la prochaine.

- **Difficultés rencontrées :**

  - Aucun problème pertinent retenu durant cette itération.