Rapport d’Itération n°1

1. Documentation utilisateur

1.1. Mode opératoire

Nom du livrable : initialisation du projet
Version : 0.1
Date de livraison : 03/03/2025
Description : Version préparatoire du projet permettant de s’assurer que la méthode d’accès aux données est fonctionnelle. Elle est composée de 3 classes permettant le test unitaire des classes modèles (Connexion.BDD, RestoDAO et Resto).

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

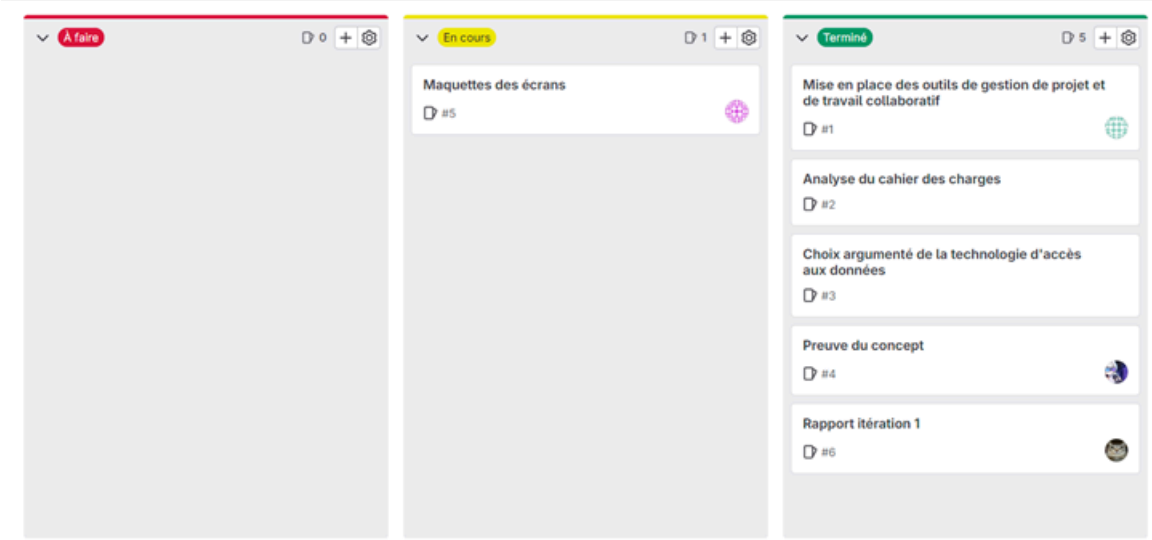
Planification prévisionnelle mise à jour :

[Joindre le diagramme de Gantt mis à jour avec date]

Comptes-rendus des Daily Scrums :

Date	Participants	Avancement	Problèmes rencontrés	Prochaines actions
24/02/2024	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**

Répartition des tâches :



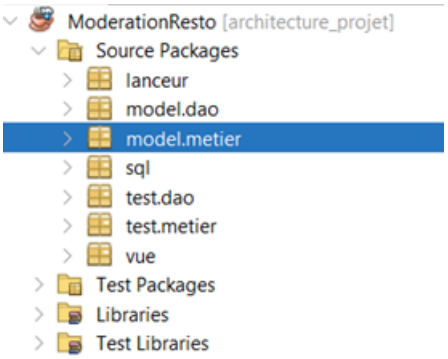
Dépôt de code :

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

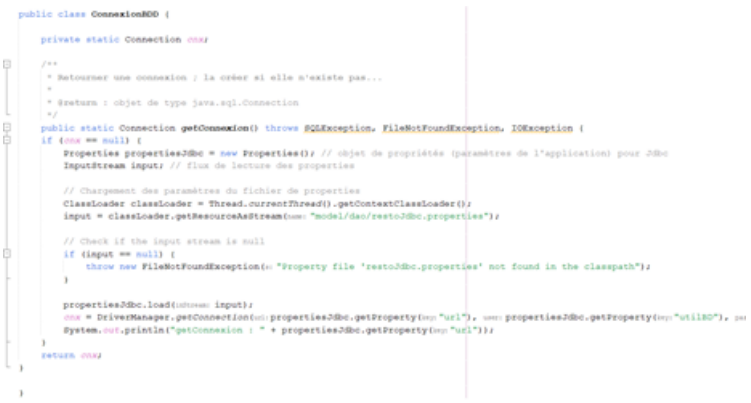
2.2. Code source

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

Évolutions du code :
- création du projet



- Ajout des classe dao et métier



```
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 {
    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();
        input = classLoader.getResourceAsStream(name: "model/dao/restoJdbc.properties");

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

        propertiesJdbc.load(inStream: input);
        cnx = DriverManager.getConnection(url: propertiesJdbc.getProperty(key: "url"), user: propertiesJdbc.getProperty(key: "utilBD"), pas
        System.out.println("getConnexion : " + propertiesJdbc.getProperty(key: "url"));
    }
    return cnx;
}

}
```

```
public class RestoDAO {

        /**
     * Extraire l'ensemble des enregistrements de la table RESTO
     * @return liste d'objets de type Resto
     * @throws SQLException
     */
    public static ArrayList<Resto> getAll() throws SQLException, IOException {
        ArrayList<Resto> lesResto = new ArrayList<>();
        Connection cnx = ConnexionBDD.getConnexion();
        PreparedStatement pstmt = cnx.prepareStatement(string: "SELECT nomR FROM resto");
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            Resto unResto = new Resto(
                    nom: rs.getString(string: "nomR")

            );
            lesResto.add(e: unResto);
        }
        return lesResto;
    }

}
```

```
package model.metier;

/**
 *
 * @author loric
 */
public class Resto {

    private String nom;

    public Resto(String nom) {
        this.nom = nom;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    @Override
    public String toString() {
        return "Resto{" + "nom=" + nom + '}';
    }


}
```

- Ajout des classes de test pour les classe dao et métier

```java
public class TestConnexionBDD {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Connection cnx = null;
        try {
            System.out.println(x: "\n Test de connexion");
            cnx = ConnexionBDD.getConnexion();
            System.out.println(x: "ConnexionBDD : connexion réussie");
            cnx.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

```java
public class TestRestoDAO {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Test 1 getAll
        System.out.println(x: "\n Test 1 : DaoResto.getAll");
        try {
            ArrayList<Resto> lesResto = RestoDAO.getAll();
            for (Resto r : lesResto) {
                System.out.println(x: r.toString());
            }
            System.out.println(lesResto.size()+" services trouvés");
        } catch (SQLException ex) {
            System.out.println("TestRestoDAO - échec getAll : " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("TestRestoDAO - échec getAll : " + ex.getMessage());
        }

        // Fermeture de la connexion
        try {
            ConnexionBDD.getConnexion().close();
            System.out.println(x: "\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());
        }
    }

}
```

```java
public class TestResto {

    public static void main(String[] args) {

        System.out.println(x: "TestResto");

        Resto unResto = new Resto(nom: "l'Escale");
        System.out.println(x: unResto.toString());
    }

}
```

2.3. Tests unitaires et d’intégration

Jeux d’essai :
- TestResto -> créer un objet de type « Resto » et l’affiche

- TestConnexionBDD -> affiche un message de confirmation si la connexion est réussie

- TestRestoDAO -> récupéré tous les restaurant de la BDD dans une liste d’objet de type « Resto » et les affichent

Traces d’exécution commentées :
- TestResto

```
ant -f "C:\\Users\\loric\\Documents\\BTS_SIO\\2ème année\\AP\\modo-modo\\ModerationResto" -Dnb.internal.action.name=run.sin
init:
Deleting: C:\Users\loric\Documents\BTS_SIO\2ème année\AP\modo-modo\ModerationResto\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\loric\Documents\BTS_SIO\2ème année\AP\modo-modo\ModerationResto\build\built-jar.properties
Compiling 1 source file to C:\Users\loric\Documents\BTS_SIO\2ème année\AP\modo-modo\ModerationResto\build\classes
compile-single:
run-single:
TestResto
Resto{nom=l'Escale}
BUILD SUCCESSFUL (total time: 0 seconds)
```

- TestConnexionBDD

```
ant -f "C:\\Users\\loric\\Documents\\BTS_SIO\\2ème année\\AP\\modo-modo\\ModerationResto" -Dnb.internal.action.name=run.single -Dj
init:
Deleting: C:\Users\loric\Documents\BTS_SIO\2ème année\AP\modo-modo\ModerationResto\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\loric\Documents\BTS_SIO\2ème année\AP\modo-modo\ModerationResto\build\built-jar.properties
Compiling 1 source file to C:\Users\loric\Documents\BTS_SIO\2ème année\AP\modo-modo\ModerationResto\build\classes
compile-single:
run-single:

 Test de connexion
getConnexion : jdbc:mysql://localhost:3306/resto2
ConnexionBDD : connexion r◆ussie
BUILD SUCCESSFUL (total time: 1 second)
```

- TestRestoDAO

```
run-single:

 Test 1 : DaoResto.getAll
getConnexion : jdbc:mysql://localhost:3306/resto2
Resto{nom=l'entrepote}
Resto{nom=le bar du charcutier}
Resto{nom=Sapporo}
Resto{nom=Cidrerie du fronton}
Resto{nom=Agadir}
Resto{nom=Le Bistrot Sainte Cluque}
Resto{nom=la petite auberge}
Resto{nom=La table de POTTOKA}
Resto{nom=La Rotisserie du Roy L◆on}
Resto{nom=Bar du March◆}
Resto{nom=Trinquet Moderne}
11 services trouv◆s

Connexion ◆ la BDD ferm◆e
BUILD SUCCESSFUL (total time: 1 second)
```

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 2** : Commencer par **US 5**, **US 6** et **T 3** (prioritaires et rapides).
- **Itération 3** : Enchaîner avec **US 1**, **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 :
- Tests d'interaction avec la BDD et priorisation des tâches à venir.

Travail restant à faire :
- la maquette à perfectionner et attribution des tâches en fonction des priorités établies et des compétences de l'équipe.

Difficultés rencontrées :
- Problème d'accès à la BBD dû à la non synchronisation du fichier « restoJdbc » entre le dossier « src » et « build » -> résolue en modifiant directement le fichier présent dans le fichier « build ».