**Rapport**

Gr : 002

Équipe : 06

Fait par :

**Soulaymane Benaddi,**

**Tahmidul Islam Shaduddin,**

**Loic Arson**

Pour notre mandat 3, notre équipe a travaillé sur la création d'une couche d'accès aux données (DAL) pour connecter notre backend à la base de données principale. Ce rapport explique les étapes principales qu'on a suivies pendant notre travail.

**Objectif**

Le but principal de notre travail était de créer un fichier dal.js avec les requêtes Node.js nécessaires pour le backend de l'application. On a aussi dû créer un fichier test.js pour tester et valider les requêtes sur la base de données.

**Étape 1 : Configuration de l’environnement de travail**

1. **Création des utilisateurs sur MySQL** :
   * On a d'abord désactivé l'authentification Windows par défaut et on l’a remplacée par l'authentification SQL Server. Ça permet de sécuriser les connexions en utilisant des identifiants spécifiques à SQL Server.
   * Ensuite, on a créé des utilisateurs avec des permissions personnalisées pour qu'ils puissent accéder à la base de données.
2. **Installation de l’extension SQL Server sur VS Code** :
   * On a installé cette extension pour pouvoir gérer la base de données directement depuis notre éditeur. C’est super pratique pour exécuter des requêtes et vérifier les changements en temps réel.
3. **Connexion entre Node.js et SQL Server** :
   * On a créé un fichier mssql.js en utilisant la librairie **Knex.js** pour se connecter à la base de données depuis Node.js. Ce fichier permet de configurer les infos comme l'hôte, le nom d'utilisateur, le mot de passe et le nom de la base de données.

**Étape 2 : Développement de la couche DAL**

Après la configuration, on est passé au développement de la couche DAL :

1. **Requêtes de sélection (listerProduit)** :
   * On a créé une fonction pour récupérer un produit spécifique ou une liste de produits.
   * *Résultat* : La fonction retourne un objet ou un tableau d’objets, selon ce qu'on cherche.
2. **Requêtes d’insertion (ajouterProduit)** :
   * Une autre fonction permet d'ajouter un produit dans la base de données.
3. **Requêtes de modification (modifierProduit)** :
   * On a aussi créé une fonction pour modifier un produit existant dans la base de données.
4. **Requêtes de suppression (supprimerProduit)** :
   * Cette fonction permet de supprimer un produit de la base de données en fonction de son ID.

**Étape 3 : Tests et validation**

Enfin, pour être sûrs que tout fonctionne bien, on a créé un fichier dal.test.js avec des tests unitaires pour valider les requêtes de la fonction listerProduit(). Voici comment on a fait :

1. **Installation des packages nécessaires** :
   * On a installé les packages **Jest** et **mock-knex** avec npm pour pouvoir faire les tests correctement.

