TP 1 : MISSION PERSISTANCE (CONNEXION BDD)

LE PROBLÈME

À chaque fois que vous redémarrez le serveur (rs avec nodemon), tous vos utilisateurs et tâches disparaissent. C'est inacceptable pour une mise en production.
Votre Mission : Connecter votre API à une vraie base de données SQL pour que les informations survivent au redémarrage.

OBJECTIFS TECHNIQUES

1. Installer les dépendances nécessaires (**TypeORM** et le pilote **SQLite**).
2. Créer un fichier de configuration centralisé pour la base de données (DataSource).
3. Modifier le point d'entrée du serveur (server.js) pour **attendre** la connexion BDD avant d'accepter des requêtes HTTP.

ÉTAPES DE RÉALISATION

Étape 1 : L'Arsenal (Installation)

Nous allons utiliser **SQLite**. C'est une base de données SQL complète qui tient dans un seul fichier. Pas besoin d'installer de serveur lourd (comme Postgres) pour l'instant, mais le code sera 100% compatible.

Dans votre terminal, installez les paquets :

npm install typeorm sqlite3 reflect-metadata

- typeorm : L'ORM qui va gérer le SQL pour nous.
- sqlite3 : Le pilote qui sait parler au fichier de base de données.
- reflect-metadata : Une librairie requise par TypeORM pour fonctionner (même en JS).

Étape 2 : Le Plan de la Base (src/config/data-source.js)

Créez ce fichier. Il doit exporter une instance de DataSource.

Configuration requise :

- **Type :** sqlite
- **Database :** database.sqlite (ce fichier sera créé à la racine)
- **Synchronize :** true (Très important : cela permet à TypeORM de créer les tables

automatiquement. À ne jamais faire en prod !)
- **Logging :** true (Pour voir les requêtes SQL dans la console)
- **Entities :** [] (Laissez un tableau vide pour l'instant, on s'en occupe au TP suivant)

Étape 3 : Le Démarrage Sécurisé (src/server.js)

Actuellement, votre serveur démarre immédiatement. C'est dangereux : si la BDD n'est pas là, l'API ne sert à rien.

Modifiez src/server.js pour :

1. Importer reflect-metadata **tout en haut** du fichier.
2. Initialiser la AppDataSource.
3. **SEULEMENT SI** l'initialisation réussit (.then()), lancez app.listen().
4. Si l'initialisation échoue (.catch()), affichez l'erreur et arrêtez le processus (process.exit(1)).

CRITÈRES DE SUCCÈS (Checklist)

- [ ] Quand je lance npm run dev, je vois un message ✅ Base de données connectée.
- [ ] Un fichier database.sqlite est apparu à la racine du projet.
- [ ] Si je fais une faute de frappe dans la config (ex: type: 'poney'), le serveur **ne démarre pas** et affiche l'erreur.
- [ ] La route GET / ou /api/status répond toujours correctement.

INDICES & ASTUCES

- **Reflect Metadata :** Si vous avez une erreur étrange au démarrage, vérifiez que require('reflect-metadata') est bien la **première ligne** de votre server.js.
- **Le Path :** Le fichier database.sqlite se créera là où vous lancez la commande npm run dev.
- **Synchronize :** Si vous oubliez synchronize: true, aucune table ne sera créée (pas grave pour ce TP, mais bloquant pour le suivant).