MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR DE LA
RECHERCHE SCIENTIFIQUE ET DE L'INNOVATION

UNIVERSITÉ SULTAN MOULAY SLIMANE

ECOLE NATIONALE DES SCIENCES
APPLIQUÉS DE KHOURIBGA

TP2 : Putting everything together

Objectifs du TP :

- Créer une page d'inscription et de connexion.
- Gérer l'authentification avec Passport.js.
- Restreindre l'accès à une page « Books » uniquement aux utilisateurs connectés.
- Utiliser MongoDB pour stocker les utilisateurs.
- Afficher les livres depuis une variable locale.

Réalisé par :

Maryam Ketatni

Sous le suivi de :

Prof. Amal Ourdou

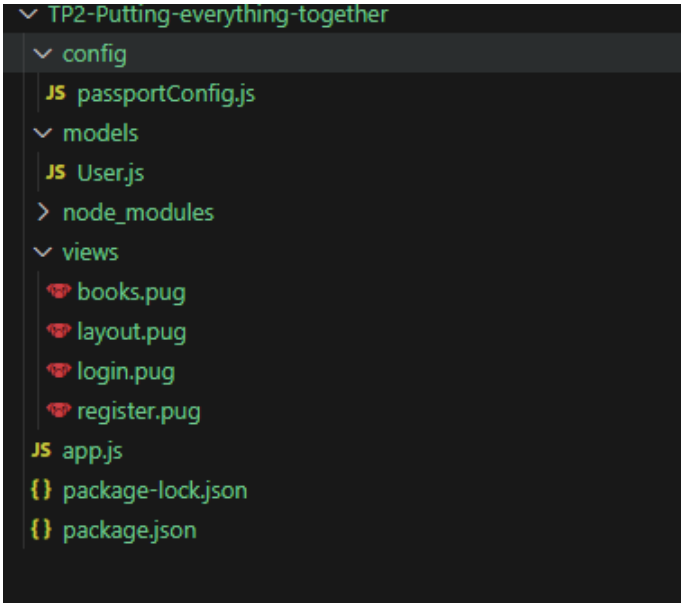
Introduction

Ce travail pratique a pour objectif d'intégrer toutes les notions étudiées dans le module de **développement web avancé avec Node.js et Express.js**. L'application développée permet d'illustrer la création d'un système complet de **gestion d'authentification** avec une interface Pug stylisée via TailwindCSS.

Technologies utilisées

— **Node.js :** environnement d'exécution JavaScript côté serveur.
— **Express.js :** framework minimaliste pour la création du serveur et des routes.
— **MongoDB :** base de données NoSQL pour stocker les utilisateurs.
— **Passport.js :** gestion des sessions et authentification locale.
— **Pug :** moteur de templates pour les vues.
— **TailwindCSS :** framework CSS utilitaire pour le design.

Structure du projet

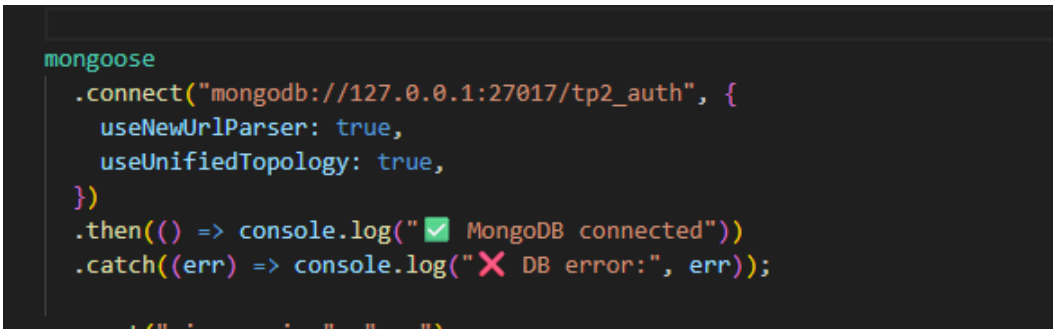


Chaque dossier a un rôle précis :

— **models :** définit le modèle d'utilisateur (Mongoose).
— **config :** contient la configuration de Passport.js.
— **views :** regroupe les vues (Pug).
— **public :** gère les ressources statiques (CSS, images).

Connexion à MongoDB

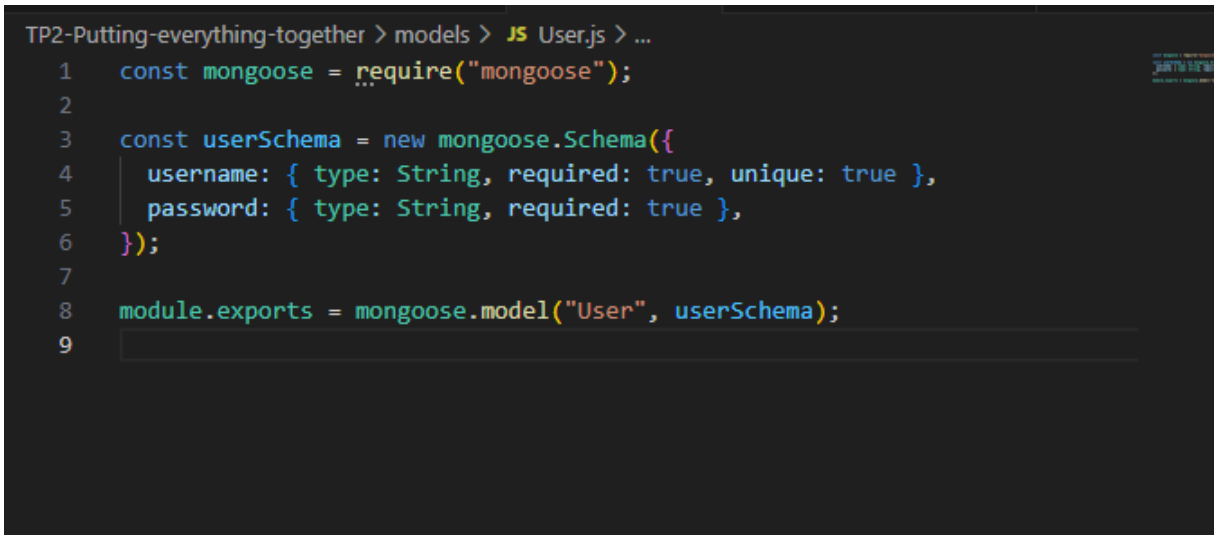
Le serveur Express est connecté à MongoDB via Mongoose :



```
mongoose
  .connect("mongodb://127.0.0.1:27017/tp2_auth", {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => console.log("✅ MongoDB connected"))
  .catch((err) => console.log("❌ DB error:", err));
```

Modèle utilisateur

Le modèle `User.js` contient deux champs : nom d'utilisateur et mot de passe haché (via bcrypt).



```
TP2-Putting-everything-together > models > JS User.js > ...
1  const mongoose = require("mongoose");
2
3  const userSchema = new mongoose.Schema({
4    username: { type: String, required: true, unique: true },
5    password: { type: String, required: true },
6  });
7
8  module.exports = mongoose.model("User", userSchema);
9
```

Configuration Passport.js

Passport est configuré pour gérer l'authentification locale. En cas de réussite, l'utilisateur est redirigé vers `/books`.

TP2-Putting-everything-together > config > passportConfig.js > <unknown> > exports

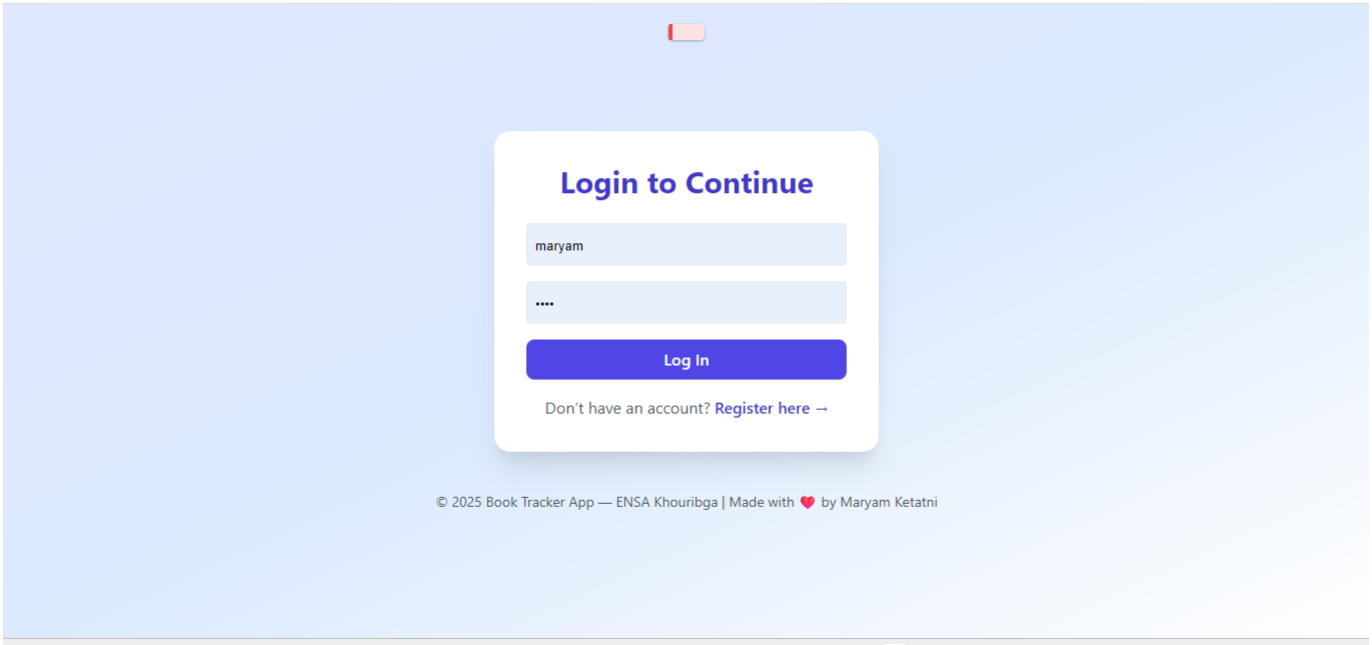
```
1   const LocalStrategy = require("passport-local").Strategy;
2   const bcrypt = require("bcryptjs");
3   const User = require("../models/User");
4
5   module.exports = function (passport) {
6     passport.use(
7       new LocalStrategy(async (username, password, done) => {
8         try {
9           const user = await User.findOne({ username });
10          if (!user) return done(null, false, { message: "User not found" }
11
12          const match = await bcrypt.compare(password, user.password);
13          if (!match) return done(null, false, { message: "Incorrect passwo
14
15          return done(null, user);
16        } catch (err) {
17          return done(err);
18        }
19      })
20    );
21    passport.serializeUser((user, done) => done(null, user.id));
22    passport.deserializeUser(async (id, done) => {
23      try {
24        const user = await User.findById(id);
25        done(null, user);
26      } catch (err) {
27        done(err, null);
28      }
29    });
30  };
31
```

Création des vues Pug

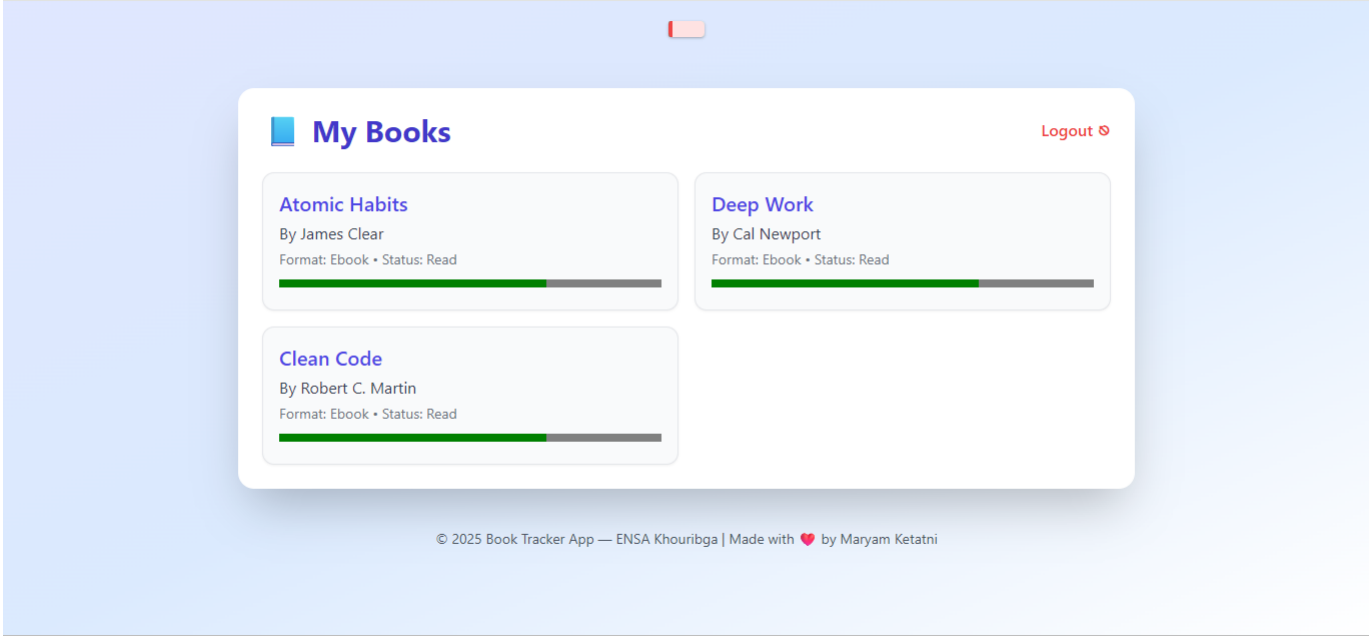
— **layout.pug** : gabarit principal (structure + style).
— **login.pug** : formulaire de connexion.
— **register.pug** : création de compte.
— **books.pug** : liste protégée des livres.

Résultats obtenus

1 Page de connexion



2 Page des livres



3 DB :MONGODB