



جامعة سلطان محمد السادس
Université Sultan Moulay Slimane



RAPPORT

TP2 :

Putting everything together

Réalisée par :

- OUMAIMA EL ALAMI



Encadré par :

- Prof. Amal Ourdou



Objectifs du TP :

- Create a registration and authentication web page using Pug
- The user's information should be stored in mongoDB
- Use passport.js for handling authentication
- After authenticating successfully redirect to the books page
- The books are stored in a local variable
- The books page should not be accessible unless the user is authenticated
- Use Tailwind CSS to handle the styling part of the applicatio

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 :

```
> config
> models
> node_modules
▽ views
  └── books.pug
  └── layout.pug
  └── login.pug
  └── register.pug
  └── app.js
  └── index.js
  └── package-lock.json
  └── package.json
```

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).

```
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

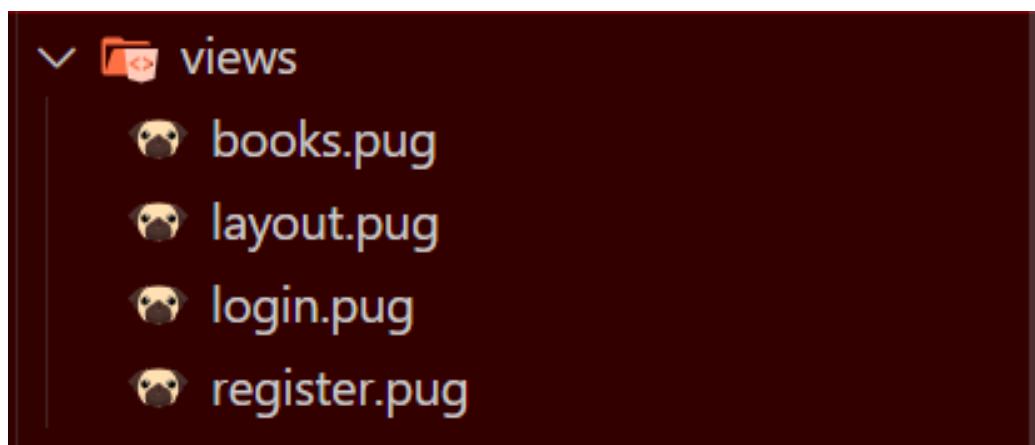
Passport est configuré pour gérer l'authentification locale. En cas de réussite, l'utilisateur est redirigé vers /books.

```
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 password" });
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    }
28  });
29}
```

Ln 5, Col 32 Spaces: 2 UTF-8 LF { JavaScript Chat quota reached Go Live Go Live

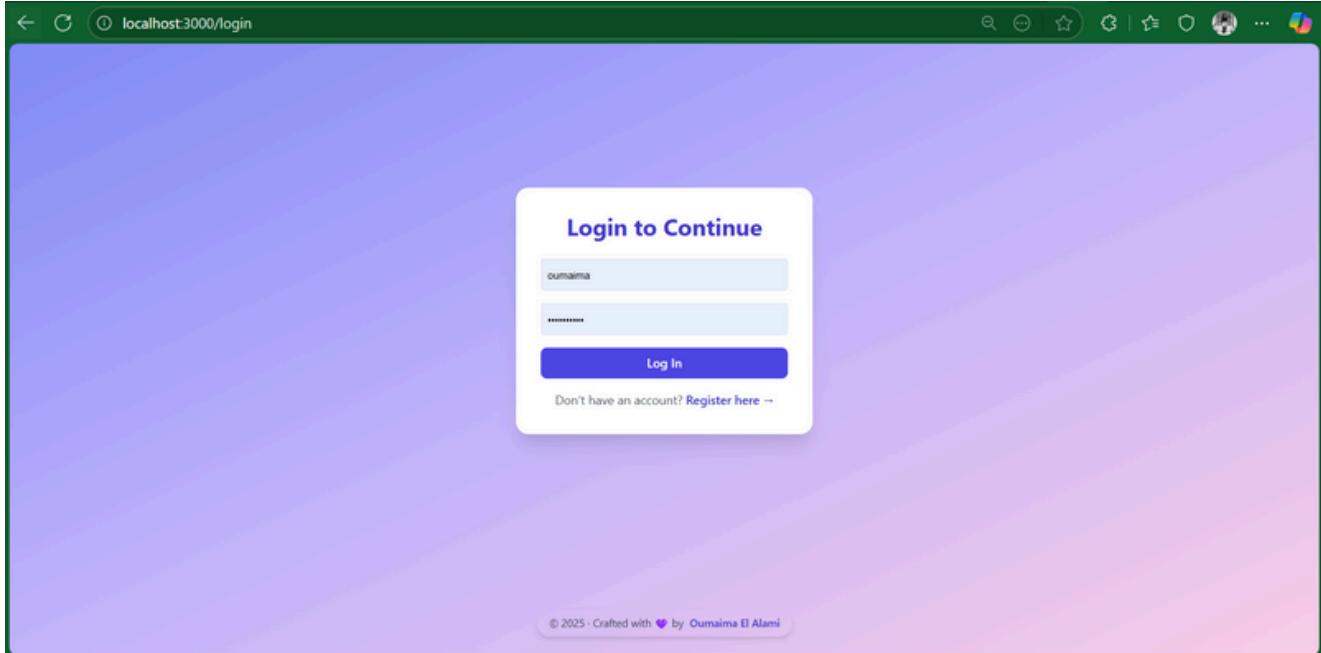
Création des vues Pug :

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

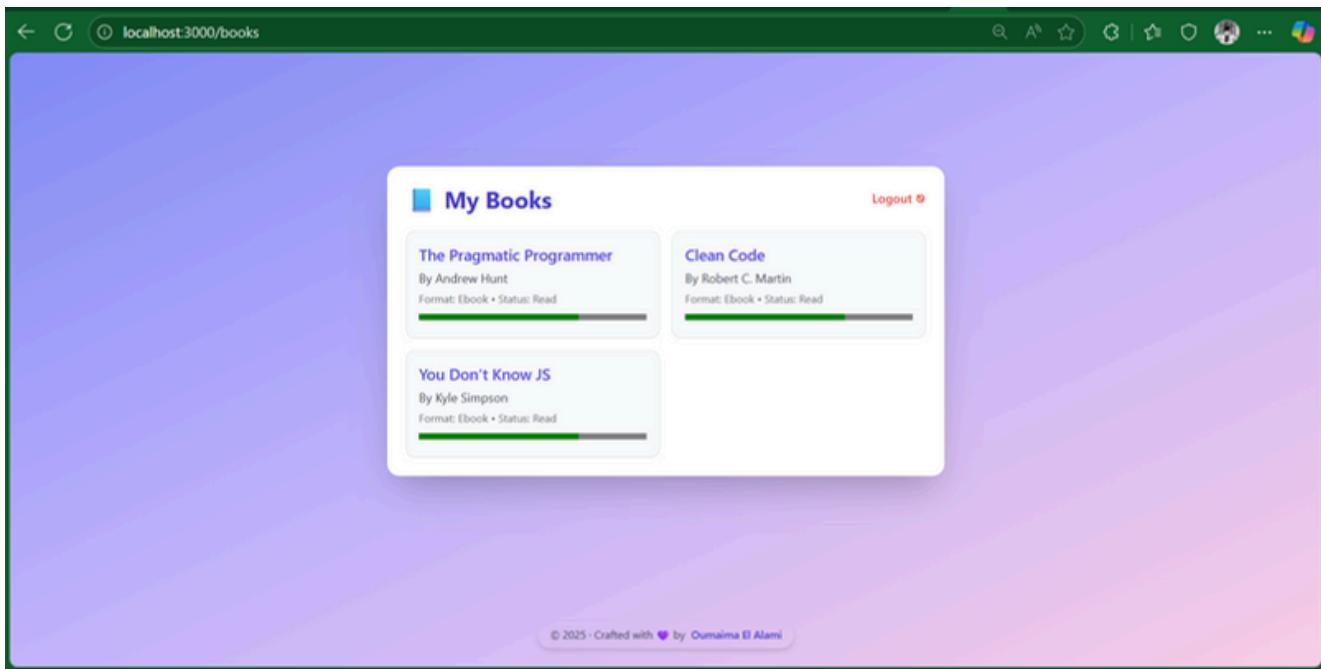


Résultats obtenus

1 Page de connexion



2 Page des livres



3 DB: MONGODB

The screenshot shows the MongoDB Compass interface. The top navigation bar displays the database path: tp2_auth > tp2_auth > users. On the right side of the top bar is a button labeled "Open MongoDB shell". Below the navigation bar, there are tabs for "Documents" (which is selected and highlighted in green), "Aggregations", "Schema", "Indexes", and "Validation". Under the "Documents" tab, there is a search bar with the placeholder "Type a query: { field: 'value' } or [Generate query +](#)". To the right of the search bar are buttons for "Explain", "Reset", "Find" (which is highlighted in green), "Options", and a dropdown menu. Below the search bar is a toolbar with icons for creating a new document, opening, edit, and delete. To the right of the toolbar is a pagination and filtering section showing "25" and "1-1 of 1". The main content area displays a single document in a JSON-like format:

```
_id: ObjectId('690fa82678c01b76ffff1c554')
username : "oumaima"
password : "$2b$10$Jm0Ip8d07pk2nY8VfErIUekDNd/Bpg23nnRC5ydl0HaqIKGbHKtAm"
__v : 0
```