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

UNIVERSITÉ SULTAN MOULAY SLIMANE

ECOLE NATIONALE DES SCIENCES APPLIQUÉS DE KHOURIBGA

RAPPORT DU TP2

PUTTING EVERYTHING TOGETHER

RÉALISÉE PAR :

BENHAMIDA Khadija

ENCADRÉ PAR :

Prof. Amal Ourdou

1. INTRODUCTION GÉNÉRALE

Ce travail pratique a pour but de mettre en œuvre l'ensemble des notions vues en cours sur le framework Express.js, la gestion de l'authentification avec Passport.js, ainsi que la connexion à une base de données MongoDB pour la persistance des utilisateurs.

L'objectif final est de créer une mini-application web nommée "Book Tracker" permettant à un utilisateur de s'enregistrer, de se connecter, puis d'accéder à une page protégée listant des livres, le tout avec un design moderne grâce à Pug et TailwindCSS.

2. OBJECTIFS DU TP

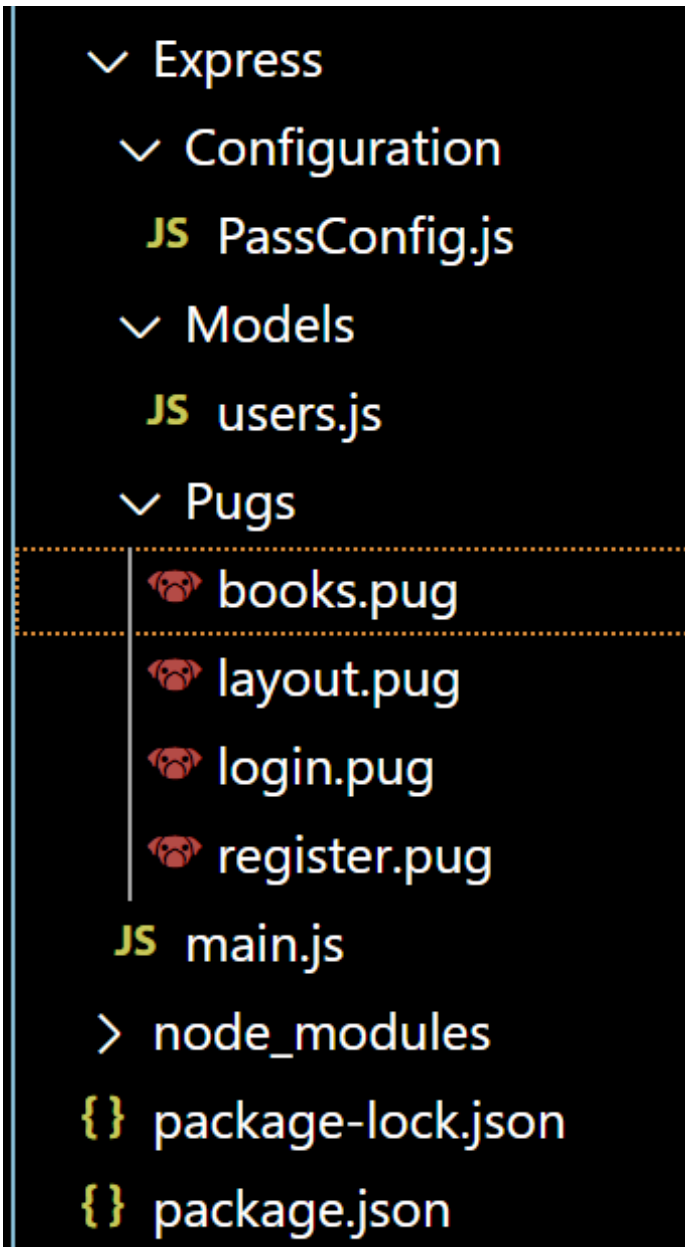
Les objectifs pédagogiques de ce TP sont les suivants :

- Créer un système complet d'authentification et d'inscription.
- Stocker les informations des utilisateurs dans MongoDB.
- Utiliser Passport.js pour la gestion sécurisée de l'authentification.
- Restreindre l'accès à certaines pages pour les utilisateurs connectés uniquement.
- Afficher une liste de livres stockée localement.
- Appliquer un design cohérent et responsive avec Tailwind CSS.

3. ENVIRONNEMENT DE TRAVAIL

Outil / Technologie	Version / Description
Node.js	v22.x
Express.js	Framework web backend
Pug	Moteur de templates pour les vues
Tailwind CSS	Framework CSS utilitaire
MongoDB	Base de données NoSQL
Passport.js	Middleware pour l'authentification
VS Code	Environnement de développement
MongoDB Compass	Interface graphique pour MongoDB

4. ARCHITECTURE DU PROJET



Description des dossiers :

- Configuration/ : contient la configuration du module Passport.js.
- Models/ : définit le modèle Mongoose pour les utilisateurs.
- Pugs/ : contient toutes les vues front-end avec le moteur de template Pug.
- main.js : fichier principal qui configure Express, MongoDB et les routes.

5. ANALYSE FONCTIONNELLE

Le système comporte trois grandes parties :

1. Authentification et inscription
2. L'utilisateur peut créer un compte via un formulaire d'inscription (username, password).
3. Les données sont hachées puis stockées dans la base MongoDB.
4. Connexion (Login)
5. Lorsqu'un utilisateur se connecte, Passport.js vérifie les informations et gère la session.
6. Page protégée "Books"
7. Accessible uniquement si l'utilisateur est authentifié.
8. Elle affiche une liste locale de livres.

6. IMPLÉMENTATION ET EXPLICATIONS

A. CONFIGURATION DU SERVEUR (MAIN.JS)

Le serveur Express est initialisé avec :

```
const express = require("express");
const mongoose = require("mongoose");
const passport = require("passport");
const session = require("express-session");
const flash = require("connect-flash");
const path = require("path");
const bcrypt = require("bcryptjs");
```

B. CONNEXION À MONGODB

La base est locale, nommée tp2_auth :

```
mongoose
  .connect("mongodb://127.0.0.1:27017/tp2_auth",
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
```

Les utilisateurs y sont sauvegardés via le modèle users.js

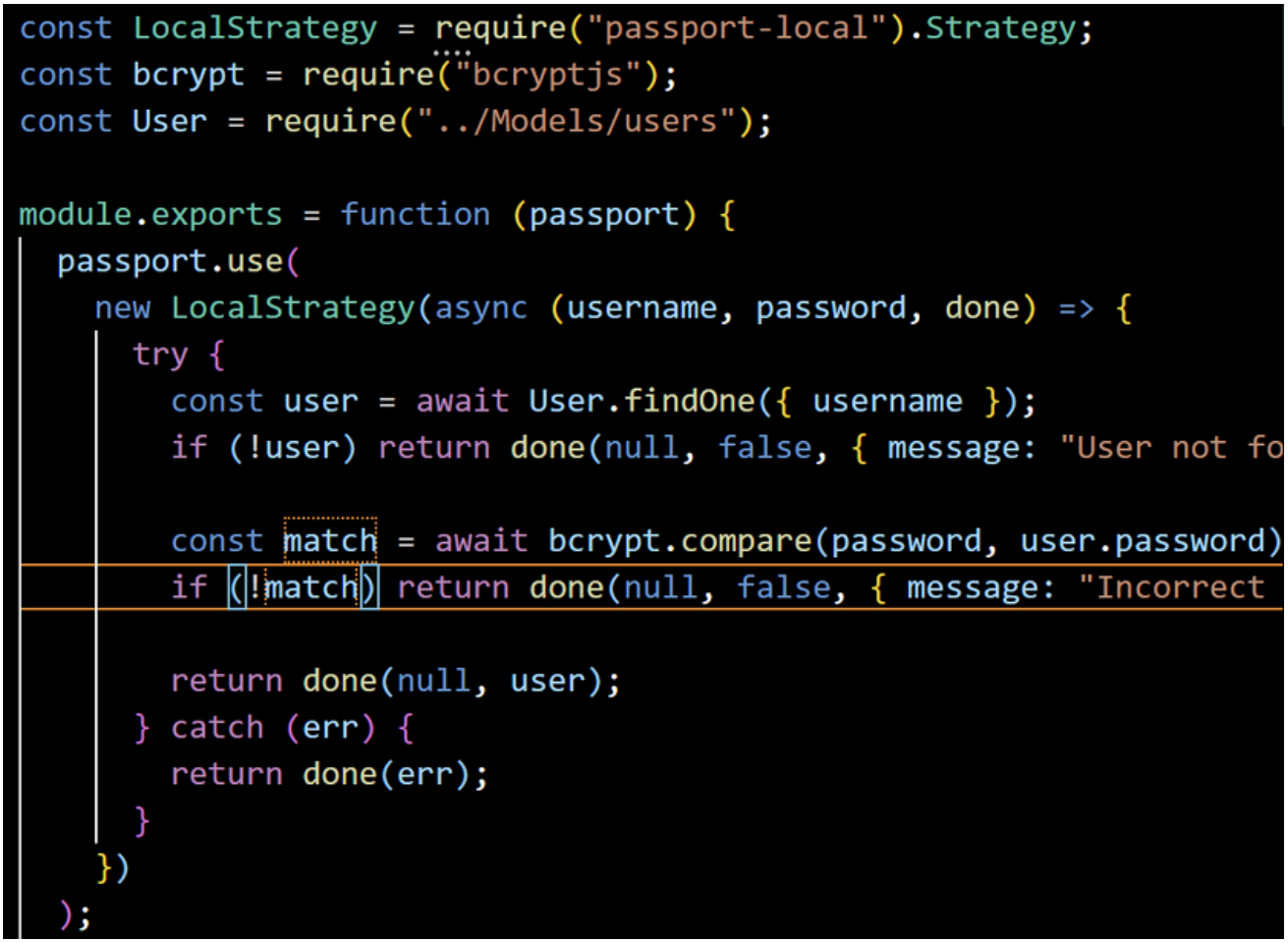
```
_id: ObjectId('690e6f5a2152e216a5030df7')
username : "rihab"
password : "$2b$10$IgV5eDjPUVqTwyEX8NEpYuuT3/PIlF.qefdmty2OacwqDSqJ9q03q"
__v : 0
```

```
_id: ObjectId('690e7060fc5b58cc5dbf1157')
username : "khadija"
password : "$2b$10$q5D1Kyf3qOGLMfj6AIe5cu5xMqrx1uc/OgV3HAumFtQYLgrACM4Qe"
__v : 0
```

C. AUTHENTIFICATION AVEC PASSPORT.JS

PassConfig.js définit la stratégie de connexion “local” :

- Vérification du username dans la base.
- Comparaison du mot de passe haché avec bcrypt.
- Création d’une session utilisateur.



```
const LocalStrategy = require("passport-local").Strategy;
const bcrypt = require("bcryptjs");
const User = require("../Models/users");

module.exports = function (passport) {
  passport.use(
    new LocalStrategy(async (username, password, done) => {
      try {
        const user = await User.findOne({ username });
        if (!user) return done(null, false, { message: "User not fo

        const match = await bcrypt.compare(password, user.password)
        if (!match) return done(null, false, { message: "Incorrect

        return done(null, user);
      } catch (err) {
        return done(err);
      }
    })
  );
```

D. CRÉATION DES VUES PUG AVEC TAILWINDCSS

--> layout.pug
Contient la structure générale et le design global (header, footer, alerts).
--> login.pug & register.pug
Formulaires stylisés avec Tailwind, pour la connexion et l'inscription.
--> books.pug
Page affichant les livres sous forme de cartes (données stockées localement).

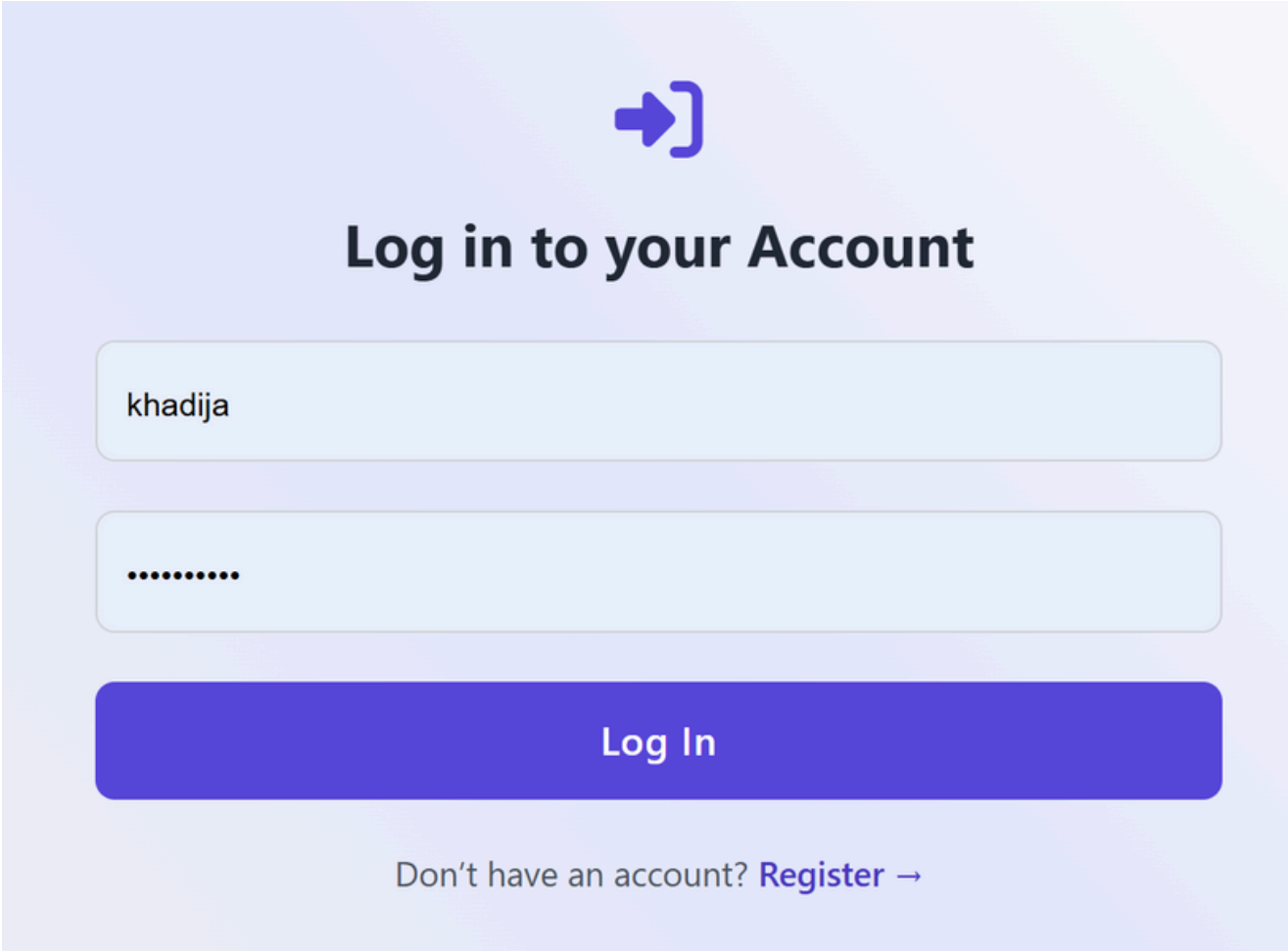
E. SÉCURISATION DES ROUTES

La fonction middleware suivante protège la page /books :

```
function ensureAuthenticated(req, res, next) {
  if (req.isAuthenticated()) return next();
  req.flash("error", "Please login to access this page");
  res.redirect("/login");
}
```

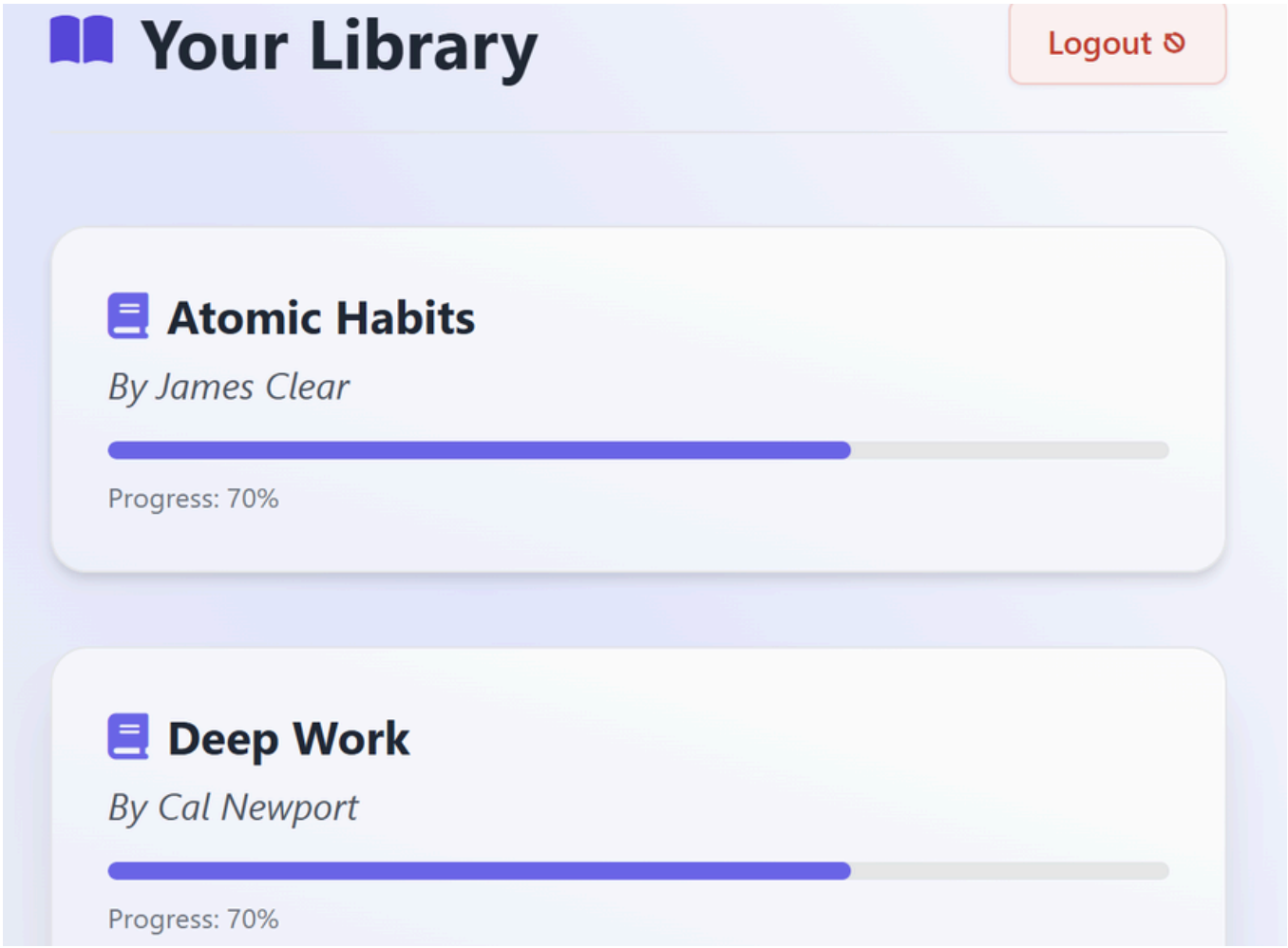
7. RÉSULTATS ET INTERFACES

LOGIN / REGISTER :



PAGE “BOOKS” :

La fonction middleware suivante protège la page /books :



Chaque carte affiche le titre, l’auteur et un indicateur de progression.

9. CONCLUSION

Ce TP a permis de maîtriser la mise en place complète d’une application Express.js :

- Création et gestion d’utilisateurs avec MongoDB
- Authentification sécurisée avec Passport.js
- Utilisation de Pug pour le rendu dynamique des vues
- Intégration du style Tailwind CSS pour un design moderne

Il illustre la synergie entre les technologies backend et frontend modernes dans un environnement Node.js.