

# Compte rendu TP 2

---

## “Putting Everything Together”

---

**Réalisé par :**

AFIFI SAAD IID3

**Encadré par :**

Pr. Amal OURDOU

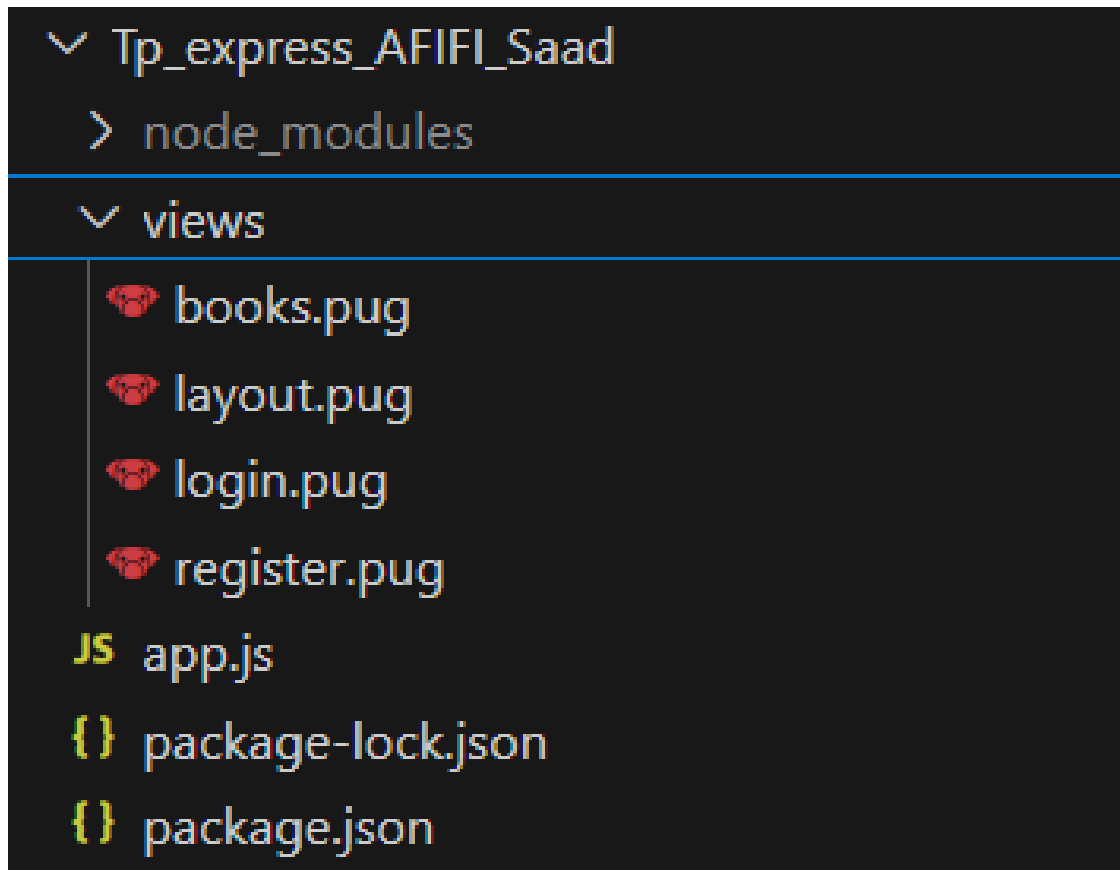


# OBJECTIFS DU TP:

1. L'objectif de ce TP est de combiner l'ensemble des notions vues dans les précédents travaux pratiques afin de construire une application web complète comprenant :
- a. une page d'inscription (register) et une page d'authentification (login) développées avec Pug,
  - b. une base de données MongoDB pour stocker les informations des utilisateurs,
  - c. le module Passport.js pour la gestion de l'authentification,
  - d. une redirection vers une page books affichant une liste de livres (locale) après connexion,
  - e. et enfin, l'utilisation de Tailwind CSS pour styliser les vues.

Technologie	Rôle
<b>Node.js</b>	Environnement d'exécution JavaScript côté serveur
<b>Express.js</b>	Framework pour la création du serveur web et des routes
<b>Pug</b>	Moteur de templates pour générer les pages HTML
<b>MongoDB + Mongoose</b>	Base de données NoSQL pour stocker les utilisateurs
<b>Passport.js</b>	Middleware de gestion d'authentification
<b>Express-session</b>	Gestion des sessions utilisateurs
<b>Tailwind CSS</b>	Framework CSS utilitaire pour la mise en forme rapide

# PARTIE I : ARCHITECTURE DU PROJET



## PARTIE 2: IMPLÉMENTATION ET EXPLICATIONS

### a) Connexion à MongoDB

Le serveur se connecte à la base locale express\_auth\_tp :

```
mongoose.connect("mongodb://127.0.0.1:27017/express_auth_tp")
  .then(() => console.log("Connecté à MongoDB"))
  .catch(err => console.error(err));
```

### b) Modèle utilisateur

Chaque utilisateur possède un nom d'utilisateur et un mot de passe :

```
const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});
```

```
const User = mongoose.model("User", userSchema);
```

## c) Configuration de Passport.js

Passport est configuré avec une stratégie locale (LocalStrategy) :

```
// Configuration de Passport
passport.use(new LocalStrategy(async (username, password, done) => {
  try {
    const user = await User.findOne({ username });
    if (!user) return done(null, false, { message: "Utilisateur non trouvé" });

    const isMatch = await user.comparePassword(password);
    if (!isMatch) return done(null, false, { message: "Mot de passe incorrect" });

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

## d) Routes principales

Page d'accueil, login, register

```
app.get("/", (req, res) => res.redirect("/login"));

app.get("/login", (req, res) => res.render("login"));
app.get("/register", (req, res) => res.render("register"));
```

Inscription (/register) et login (/login):

```
// Inscription
app.post("/register", async (req, res) => {
  const { username, password } = req.body;
  try {
    const user = new User({ username, password });
    await user.save();
    res.redirect("/login");
  } catch (err) {
    res.send("Erreur : utilisateur déjà existant");
  }
});

// Connexion
app.post("/login",
  passport.authenticate("local", {
    successRedirect: "/books",
    failureRedirect: "/login"
  })
);
```

```
// Protection des routes
function isAuthenticated(req, res, next) {
  if (req.isAuthenticated()) return next();
  res.redirect("/login");
}

// Liste des livres (visible uniquement si connecté)
const books = [
  { title: "L'Alchimiste", author: "Paulo Coelho" },
  { title: "1984", author: "George Orwell" },
  { title: "Le Petit Prince", author: "Antoine de Saint-Exupéry" }
];

app.get("/books", isAuthenticated, (req, res) => {
  res.render("books", { user: req.user, books });
});

app.get("/logout", (req, res) => {
  req.logout(() => res.redirect("/login"));
});

app.listen(9000, () => console.log("serveur lancé : port 9000"));
```

## e) Interface utilisateur (Pug + Tailwind)

### Page register.pug

```
extends layout

block content
  // utiliser tag explicite car lg:grid-cols-2 contient ':'
  div(class="grid grid-cols-1 lg:grid-cols-2 gap-8")
    // Left: illustration / texte
    div(class="hidden lg:flex flex-col justify-center px-8")
      h2(class="text-3xl font-extrabold text-slate-800 mb-4") Rejoins-nous !
      p(class="text-slate-600 mb-6") Crée un compte pour accéder à la liste exclusive des livres
      ul(class="space-y-3 text-sm text-slate-600")
        li ✓ Interface simple et sécurisée
        li ✓ Stockage avec MongoDB
        li ✓ Authentification gérée par Passport.js

    // Right: form card
    div(class="mx-auto max-w-md w-full bg-white shadow-lg rounded-xl p-6")
      h3(class="text-xl font-semibold text-slate-800 mb-4") Création de compte
      form(action="/register" method="POST" class="space-y-4")
        label(class="block")
          span(class="text-sm text-slate-700") Nom d'utilisateur
          input(type="text" name="username" required placeholder="ex: saad_aff" class="mt-1 block w-full rounded-md border-gray-200")
        label(class="block")
          span(class="text-sm text-slate-700") Mot de passe
          input(type="password" name="password" required placeholder="....." class="mt-1 block w-full rounded-md border-gray-200")
        button(type="submit" class="w-full py-2 rounded-md bg-brand text-white font-medium hover:bg-brand-dark transition") S'inscrire
      p(class="text-center text-sm text-slate-500 mt-4")
        | Tu as déjà un compte ?
        a(href="/login" class="text-brand font-medium hover:underline ml-1") Connecte-toi
```

## Rejoins-nous !

Crée un compte pour accéder à la liste exclusive des livres

- ✓ Interface simple et sécurisée
- ✓ Stockage avec MongoDB
- ✓ Authentification gérée par Passport.js

### Création de compte

Nom d'utilisateur

ex: saad\_aff

Mot de passe

.....

S'inscrire

Tu as déjà un compte ? [Connecte-toi](#)

Développé par Saad AFIFI

## Page login.pug

```
extends layout

block content
  div(class="flex flex-col items-center")
    div(class="w-full max-w-md bg-white shadow-lg rounded-xl p-6")
      h3(class="text-2xl font-semibold text-slate-800 mb-4") Connexion
      form(action="/login" method="POST" class="space-y-4")
        label(class="block")
          span(class="text-sm text-slate-700") Nom d'utilisateur
          input(type="text" name="username" required placeholder="Ton nom d'utilisateur" class="mt-1 block w-full rounded-md border)
        label(class="block")
          span(class="text-sm text-slate-700") Mot de passe
          input(type="password" name="password" required placeholder="....." class="mt-1 block w-full rounded-md border-gray-200)
        button(type="submit" class="w-full py-2 rounded-md bg-brand text-white font-medium hover:bg-brand-dark transition") Se connecter
      div(class="mt-4 text-center text-sm text-slate-500")
        | Pas encore inscrit ?
        a(href="/register" class="text-brand font-medium hover:underline ml-1") Créer un compte
```

### Connexion

Nom d'utilisateur

Ton nom d'utilisateur

Mot de passe

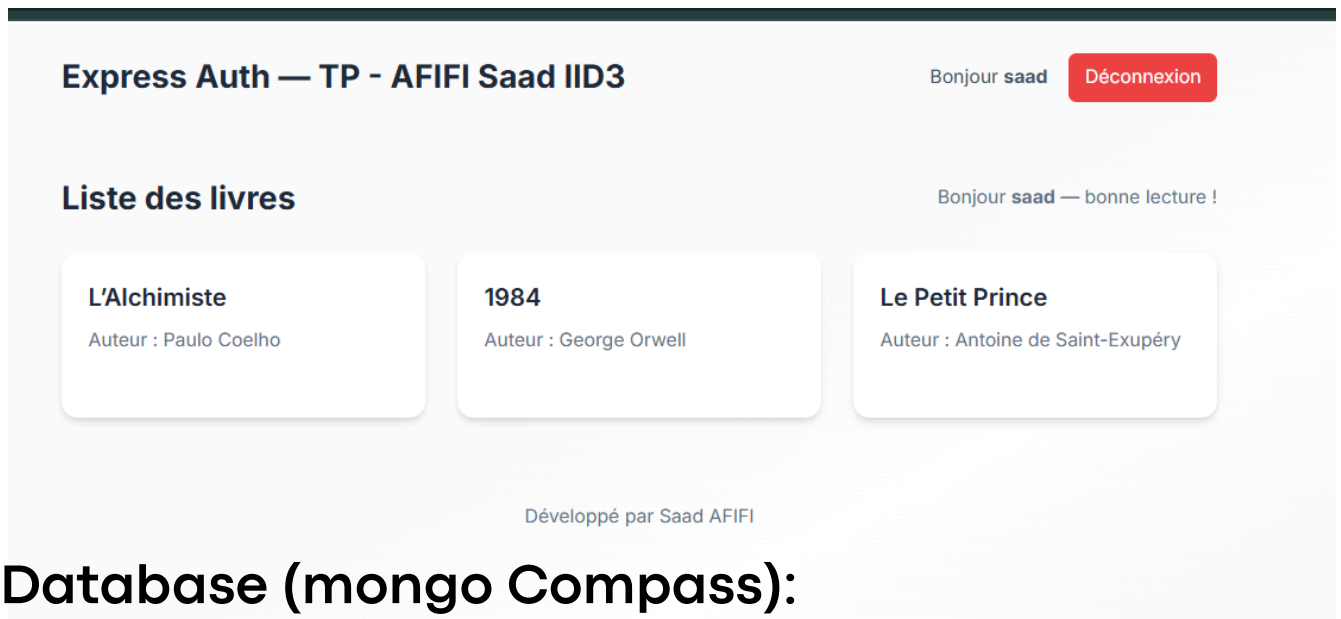
.....

Se connecter

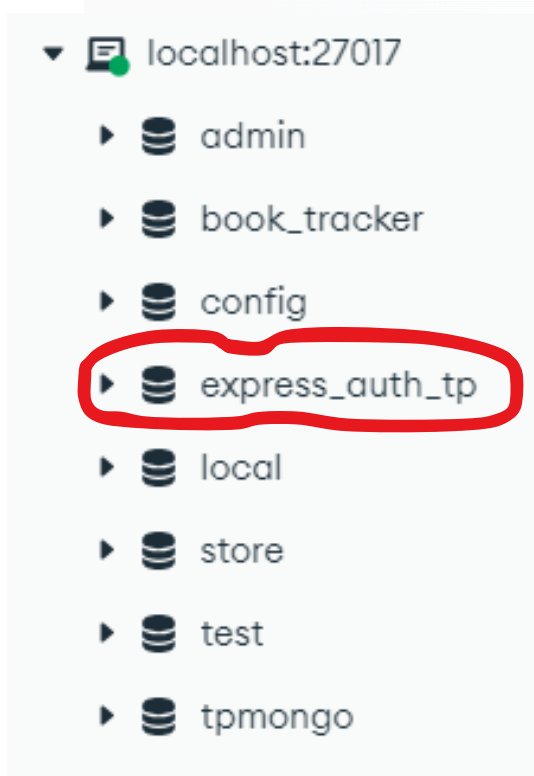
Pas encore inscrit ? [Créer un compte](#)

Développé par Saad AFIFI

```
block content
  div(class="space-y-6")
    div(class="flex items-center justify-between")
      h2(class="text-2xl font-bold text-slate-800") Liste des livres
      p(class="text-sm text-slate-500") Bonjour #[strong #{user.username}] – bonne lecture !
    div(class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6")
      each book in books
        div(class="bg-white rounded-xl shadow-md p-5 flex flex-col justify-between")
          div(class="mb-4")
            h3(class="text-lg font-semibold text-slate-800") #{book.title}
            p(class="text-sm text-slate-500 mt-2") Auteur : #{book.author}
          div(class="mt-3 flex gap-2")
            if books.length === 0
              div(class="text-center text-slate-500 mt-6") Aucun livre disponible pour l'instant.
```



## f) Database (mongo Compass):



# CONCLUSION :

**Ce TP a permis de mettre en pratique :**

- la création d'un serveur Express avec gestion des vues Pug,
- l'intégration d'une base de données MongoDB via Mongoose,
- la configuration d'un système complet d'authentification sécurisée avec Passport.js,
- et l'utilisation de Tailwind CSS pour concevoir une interface moderne et réactive.

**Cette approche "full-stack" illustre comment les composants serveur, base de données, middleware et front-end interagissent dans une application web professionnelle.**