**1. Installer et Configurer XAMPP (MySQL et phpMyAdmin)**

**a. Téléchargement et installation de XAMPP**

1. Téléchargez et installez XAMPP sur votre machine.
2. Une fois installé, ouvrez **XAMPP Control Panel**.
3. Lancez les services **Apache** (pour le serveur web) et **MySQL** (pour la base de données).

**b. Créer une base de données dans phpMyAdmin**

1. Accédez à **phpMyAdmin** en ouvrant http://localhost/phpmyadmin dans votre navigateur.
2. Créez une nouvelle base de données ecommerce\_db.
3. Créez une table **users** pour stocker les utilisateurs avec les champs suivants :

sql

Copier le code

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

);

**2. Configurer le Backend avec Node.js**

**a. Préparer votre environnement Node.js**

1. Installez **Node.js** si ce n'est pas déjà fait. Téléchargez-le depuis [Node.js](https://nodejs.org/).
2. Créez un répertoire pour le backend. Par exemple, ecommerce\_backend, puis ouvrez un terminal et initialisez le projet Node.js :

bash

Copier le code

mkdir ecommerce\_backend

cd ecommerce\_backend

npm init -y

1. Installez les dépendances nécessaires :

bash

Copier le code

npm install express mysql2 bcryptjs jsonwebtoken cors

**b. Créer le fichier server.js**

Dans ce fichier, nous allons configurer le serveur Express, établir la connexion à MySQL et implémenter les routes pour l'inscription et la connexion des utilisateurs.

Voici le code complet pour le fichier server.js :

javascript

Copier le code

const express = require('express');

const mysql = require('mysql2');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const cors = require('cors');

const app = express();

const port = 5000;

// Middleware

app.use(cors());

app.use(express.json());

// Connexion à la base de données MySQL

const db = mysql.createConnection({

host: 'localhost',

user: 'root', // Utilisateur par défaut de XAMPP

password: '', // Mot de passe vide par défaut

database: 'ecommerce\_db'

});

// Vérification de la connexion

db.connect(err => {

if (err) {

console.error('Erreur de connexion à la base de données:', err);

return;

}

console.log('Connexion à la base de données MySQL réussie');

});

// Route d'inscription (Registration)

app.post('/register', (req, res) => {

const { email, password } = req.body;

// Vérifier si l'email existe déjà

db.query('SELECT \* FROM users WHERE email = ?', [email], (err, results) => {

if (err) {

return res.status(500).json({ message: 'Erreur du serveur', status: false });

}

if (results.length > 0) {

return res.status(400).json({ message: 'L\'email existe déjà', status: false });

}

// Hachage du mot de passe

bcrypt.hash(password, 10, (err, hashedPassword) => {

if (err) {

return res.status(500).json({ message: 'Erreur lors du hachage du mot de passe', status: false });

}

// Insertion de l'utilisateur dans la base de données

db.query('INSERT INTO users (email, password) VALUES (?, ?)', [email, hashedPassword], (err, results) => {

if (err) {

return res.status(500).json({ message: 'Erreur lors de l\'enregistrement', status: false });

}

return res.status(201).json({ message: 'Utilisateur enregistré avec succès', status: true });

});

});

});

});

// Route de connexion (Login)

app.post('/login', (req, res) => {

const { email, password } = req.body;

// Recherche de l'utilisateur dans la base de données

db.query('SELECT \* FROM users WHERE email = ?', [email], (err, results) => {

if (err) {

return res.status(500).json({ message: 'Erreur du serveur', status: false });

}

if (results.length === 0) {

return res.status(400).json({ message: 'Utilisateur non trouvé', status: false });

}

const user = results[0];

// Vérification du mot de passe

bcrypt.compare(password, user.password, (err, isMatch) => {

if (err) {

return res.status(500).json({ message: 'Erreur lors de la vérification du mot de passe', status: false });

}

if (!isMatch) {

return res.status(400).json({ message: 'Mot de passe incorrect', status: false });

}

// Génération du token JWT

const token = jwt.sign({ id: user.id, email: user.email }, 'secretkey', { expiresIn: '1h' });

return res.status(200).json({

message: 'Connexion réussie',

status: true,

token: token

});

});

});

});

// Démarrage du serveur

app.listen(port, () => {

console.log(`Serveur backend démarré sur http://localhost:${port}`);

});

**c. Lancer le serveur**

Dans le terminal, naviguez jusqu'au répertoire du backend et lancez le serveur avec la commande suivante :

bash

Copier le code

node server.js

Le serveur backend sera maintenant accessible à http://localhost:5000.

**3. Configurer le Frontend avec React**

**a. Créer une application React**

Si vous n'avez pas encore créé d'application React, utilisez **Create React App** :

bash

Copier le code

npx create-react-app ecommerce\_frontend

cd ecommerce\_frontend

**b. Installer Axios pour les requêtes HTTP**

Dans le répertoire de votre projet React, installez **Axios** pour faciliter les appels API :

bash

Copier le code

npm install axios

**c. Implémenter les pages d'inscription et de connexion dans React**

**Exemple de page d'inscription (Register.js)**

javascript

Copier le code

import React, { useState } from 'react';

import axios from 'axios';

const Register = () => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await axios.post('http://localhost:5000/register', { email, password });

alert(response.data.message);

} catch (err) {

alert('Erreur : ' + err.response.data.message);

}

};

return (

<form onSubmit={handleSubmit}>

<div>

<label>Email</label>

<input type="email" value={email} onChange={(e) => setEmail(e.target.value)} required />

</div>

<div>

<label>Mot de passe</label>

<input type="password" value={password} onChange={(e) => setPassword(e.target.value)} required />

</div>

<button type="submit">S'inscrire</button>

</form>

);

};

export default Register;

**Exemple de page de connexion (Login.js)**

javascript

Copier le code

import React, { useState } from 'react';

import axios from 'axios';

const Login = () => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await axios.post('http://localhost:5000/login', { email, password });

alert(response.data.message);

localStorage.setItem('token', response.data.token); // Stocker le token dans localStorage

} catch (err) {

alert('Erreur : ' + err.response.data.message);

}

};

return (

<form onSubmit={handleSubmit}>

<div>

<label>Email</label>

<input type="email" value={email} onChange={(e) => setEmail(e.target.value)} required />

</div>

<div>

<label>Mot de passe</label>

<input type="password" value={password} onChange={(e) => setPassword(e.target.value)} required />

</div>

<button type="submit">Se connecter</button>

</form>

);

};

export default Login;

**d. Lancer le frontend React**

Dans le répertoire de votre projet React, lancez l'application avec la commande :

bash

Copier le code

npm start

L'application React sera accessible à http://localhost:3000.

**4. Tester l'Application**

1. Accédez à http://localhost:3000 pour voir votre frontend.
2. Inscrivez un utilisateur via la page d'inscription et connectez-vous via la page de connexion.

4o mini

Haut du formulaire

Bas du formulaire