→ Estrutura do projeto

gerenciamento-academy/

│── backend/ # Servidor Node.js (Express + SQLite)

│── frontend/ # Aplicação React.js

│── README.md # Documentação

→ Backend (NODE.JS + SQLITE)

mkdir backend

cd backend

npm init -y

npm install express sqlite3 cors bcryptjs jsonwebtoken dotenv

→ Criar o Arquivo .env (Variáveis de Ambiente)

SECRET\_KEY=meusegredo

PORT=5000

→ SERVER.JS

require("dotenv").config();

const express = require("express");

const cors = require("cors");

const sqlite3 = require("sqlite3").verbose();

const bcrypt = require("bcryptjs");

const jwt = require("jsonwebtoken");

const app = express();

app.use(express.json());

app.use(cors());

const SECRET\_KEY = process.env.SECRET\_KEY || "meusegredo";

// Conectar ao SQLite

const db = new sqlite3.Database("./database.db", (err) => {

if (err) console.error(err.message);

console.log("📦 Banco de Dados Conectado!");

});

// Criar tabela de usuários

db.run(

`CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

email TEXT UNIQUE,

password TEXT,

role TEXT DEFAULT 'editor'

)`

);

// Registro de Usuário

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

const { name, email, password } = req.body;

const hashedPassword = await bcrypt.hash(password, 10);

db.run("INSERT INTO users (name, email, password) VALUES (?, ?, ?)",

[name, email, hashedPassword],

(err) => {

if (err) return res.status(400).json({ error: err.message });

res.json({ message: "Usuário criado!" });

}

);

});

// Login de Usuário

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

const { email, password } = req.body;

db.get("SELECT \* FROM users WHERE email = ?", [email], async (err, user) => {

if (!user || !(await bcrypt.compare(password, user.password))) {

return res.status(401).json({ message: "Credenciais inválidas!" });

}

const token = jwt.sign({ id: user.id, role: user.role }, SECRET\_KEY, { expiresIn: "1h" });

res.json({ token, role: user.role });

});

});

// Middleware de Autenticação

const verifyToken = (req, res, next) => {

const token = req.headers["authorization"];

if (!token) return res.status(403).json({ message: "Token não fornecido!" });

jwt.verify(token.split(" ")[1], SECRET\_KEY, (err, decoded) => {

if (err) return res.status(401).json({ message: "Token inválido!" });

req.userId = decoded.id;

req.userRole = decoded.role;

next();

});

};

// Listar Usuários (Apenas Autenticados)

app.get("/users", verifyToken, (req, res) => {

db.all("SELECT id, name, email, role FROM users", [], (err, users) => {

if (err) return res.status(500).json({ error: err.message });

res.json(users);

});

});

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`🚀 Servidor rodando na porta ${PORT}!`));

→ RODAR O BACKEND

cd backend

node server.js

→ Frontend (React.js)

cd ..

mkdir frontend

cd frontend

npx create-react-app .

npm install axios react-router-dom

→ Criar Tela de Login (frontend/src/Login.js)

import { useState } from "react";

import axios from "axios";

import { useNavigate } from "react-router-dom";

export default function Login({ setToken }) {

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

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

const navigate = useNavigate();

const handleLogin = async (e) => {

e.preventDefault();

try {

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

localStorage.setItem("token", data.token);

setToken(data.token);

navigate("/dashboard");

} catch (error) {

alert("Erro ao fazer login");

}

};

return (

<div className="login">

<h2>Login</h2>

<form onSubmit={handleLogin}>

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

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

<button type="submit">Entrar</button>

</form>

</div>

);

}

→ Criar Tela de Dashboard (frontend/src/Dashboard.js)

import { useEffect, useState } from "react";

import axios from "axios";

import { useNavigate } from "react-router-dom";

export default function Dashboard() {

const [users, setUsers] = useState([]);

const navigate = useNavigate();

useEffect(() => {

const fetchUsers = async () => {

try {

const token = localStorage.getItem("token");

const { data } = await axios.get("http://localhost:5000/users", {

headers: { Authorization: `Bearer ${token}` },

});

setUsers(data);

} catch (error) {

alert("Erro ao buscar usuários, faça login novamente.");

localStorage.removeItem("token");

navigate("/");

}

};

fetchUsers();

}, [navigate]);

const handleLogout = () => {

localStorage.removeItem("token");

navigate("/");

};

return (

<div>

<h1>Usuários</h1>

<button onClick={handleLogout}>Sair</button>

<ul>

{users.map((user) => (

<li key={user.id}>{user.name} ({user.role})</li>

))}

</ul>

</div>

);

}

→ Criar Rotas no Frontend (frontend/src/App.js)

import { BrowserRouter, Routes, Route } from "react-router-dom";

import { useState } from "react";

import Login from "./Login";

import Dashboard from "./Dashboard";

export default function App() {

const [token, setToken] = useState(localStorage.getItem("token"));

return (

<BrowserRouter>

<Routes>

<Route path="/" element={<Login setToken={setToken} />} />

<Route path="/dashboard" element={token ? <Dashboard /> : <Login setToken={setToken} />} />

</Routes>

</BrowserRouter>

);

}

→ Rodar o Frontend

Rodar o Frontend