Api de login con mongoDB, express y React

Asignatura: Desarrollo Web Avanzado

Villareal, Ariel Reyes

Departamento de Ciencias de la Computación

Universidad de las Fuerzas Armadas Espe

Fecha de entrega: 18/11/2024

Índice

1. Resumen

En este informe se presenta el desarrollo de una API REST para la gestión de usuarios, incluyendo el registro y el inicio de sesión, utilizando tecnologías como Node.js, Express, MongoDB y React. El propósito principal es proporcionar una solución sencilla y segura para la autenticación de usuarios en aplicaciones web. Se emplearon herramientas como bcryptjs para el cifrado de contraseñas y jsonwebtoken para la generación de tokens JWT. La aplicación también incluye un frontend desarrollado con React para interactuar con la API. Las pruebas fueron realizadas usando Postman para verificar la funcionalidad de los endpoints.

2. Introducción

Este informe tiene como objetivo describir el desarrollo de una API REST para el manejo de usuarios (registro e inicio de sesión) mediante tecnologías modernas como Node.js, Express, MongoDB, bcryptjs y jsonwebtoken. Además, se detallará el desarrollo de un frontend en React que interactúa con la API, proporcionando una solución completa de autenticación de usuarios. El desarrollo de aplicaciones web seguras es esencial en el ámbito del desarrollo de software actual. Las soluciones de autenticación son una parte fundamental de muchas aplicaciones modernas. En este informe, se presenta una solución simple pero robusta para la autenticación de usuarios utilizando JWT (JSON Web Tokens) y tecnologías populares como Express y MongoDB. Este enfoque es comúnmente utilizado en aplicaciones web de una sola página (SPA) que requieren un sistema de inicio de sesión seguro. Este informe abarca el desarrollo de la API y el frontend, pero no incluye aspectos como la implementación de seguridad avanzada.

3. Objetivos

3.1.Objetivo General:

Desarrollar una API REST que permita gestionar el registro y el inicio de sesión de usuarios, empleando las tecnologías Node.js, Express, MongoDB, bcryptjs y jsonwebtoken. Además, desarrollar un frontend utilizando React para interactuar con la API.

3.2.Objetivos Específicos:

- Implementar las rutas para el registro y inicio de sesión en la API utilizando Express.
- Usar MongoDB y mongoose para almacenar los datos de los usuarios.
- Implementar el cifrado de contraseñas utilizando bcryptjs.
- Generar tokens JWT para la autenticación de usuarios.
- Desarrollar un frontend básico en React que permita a los usuarios registrarse y acceder a su cuenta.

- Realizar pruebas utilizando Postman para verificar el funcionamiento correcto de las rutas de la API.

4. Desarrollo

4.1.Conceptos base

El desarrollo de una aplicación web moderna implica tanto la creación de una API que maneje la lógica del backend como la creación de un frontend que interactúe con esta API. En este caso, se desarrolló una API REST para manejar operaciones de autenticación (registro e inicio de sesión). Para garantizar la seguridad de las contraseñas, se utilizó bcryptjs para cifrarlas antes de almacenarlas en la base de datos MongoDB. El uso de JWT permite mantener la sesión de los usuarios sin necesidad de almacenar información sensible en el servidor.

4.2.Herramientas de desarrollo

- **Node.js:** Plataforma para ejecutar JavaScript en el servidor.
- **Express:** Framework de Node.js para la creación de la API REST.
- **MongoDB:** Base de datos NoSQL para almacenar la información de los usuarios.
- **Mongoose**: Librería para interactuar con MongoDB a través de objetos.
- **bcryptjs**: Librería para cifrar contraseñas antes de almacenarlas en la base de datos.
- **jsonwebtoken (JWT):** Librería para generar y verificar tokens JWT, que permiten manejar la autenticación de usuarios.
- **React:** Librería de JavaScript para desarrollar la interfaz de usuario.

4.3.Desarrollo de la API REST

Antes de codificar se inicializa express en un directorio de backend y react.

4.3.1. Configuración de la base de datos

```
const mongoose = require('mongoose');

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true });
    console.log("MongoDB connected");
  } catch (err) {
    console.error(err.message);
    process.exit(1);
  }
};

module.exports = connectDB;
```

4.3.2. *Modelo de User*

```
const mongoose = require('mongoose');

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

module.exports = mongoose.model('User', UserSchema);
```

4.3.3. *Lógica de la API*

```
const express = require("express");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const User = require("../models/User");
const router = express.Router();

// Registro de usuario
router.post("/register", async (req, res) => {
  const { username, email, password } = req.body;

  try {
    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({ message: "El usuario ya está registrado" });
    }

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

    const user = new User({ username, email, password: hashedPassword });
    await user.save();

    res.status(201).json({ message: "Usuario registrado exitosamente" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Inicio de sesión
router.post("/login", async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });
    if (!user)
      return res.status(404).json({ message: "Usuario no encontrado" });

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch)
      return res.status(400).json({ message: "Credenciales inválidas" });

    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "1h",
    });
    res.json({ token, message: "Inicio de sesión exitoso" });
  } catch (err) {
    res.status(500).json({ message: "Error en el servidor" });
  }
});

module.exports = router;
```

4.3.4. *Definir variable de entorno*

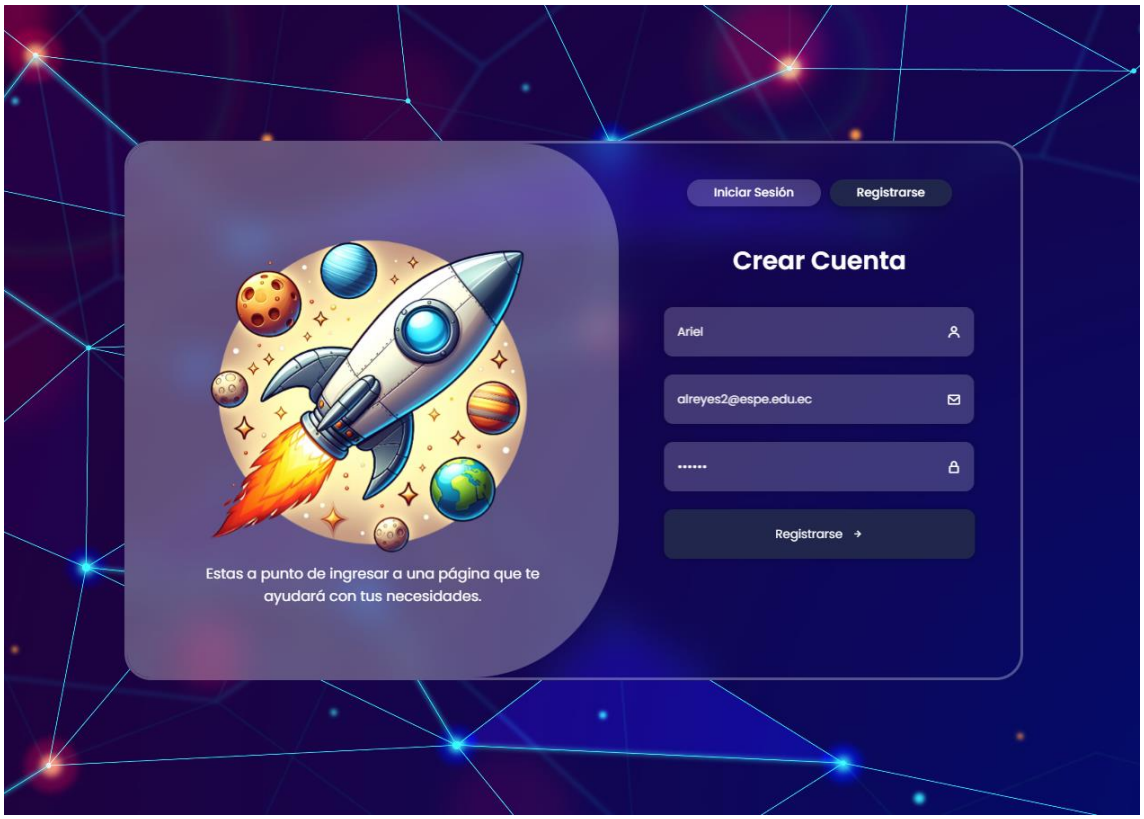
```
MONGO_URI=mongodb://localhost:27017/authDB
JWT_SECRET=your_jwt_secret
```

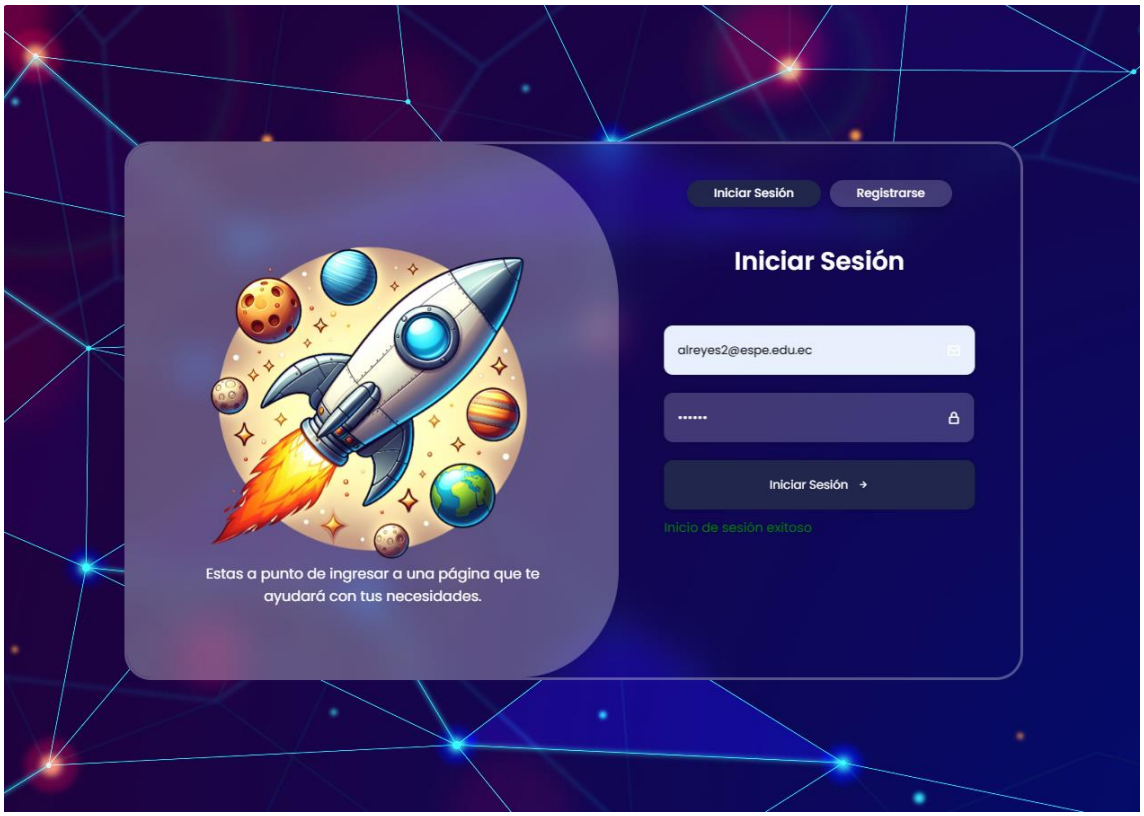
4.3.5. Unificar conceptos y levantar servidores

```
const express = require('express');
const dotenv = require('dotenv');
const connectDB = require('./config/db');
const authRoutes = require('./routes/auth');

// Cargar variables de entorno
dotenv.config();

// Conectar a la base de datos
connectDB();

// Crear la aplicación express
const app = express();

// Habilitar CORS antes de las rutas
const cors = require('cors');
app.use(cors({ origin: 'http://localhost:3000' })); // Permitir solo solicitudes desde localhost:3000

// Habilitar el parsing de JSON en el cuerpo de las solicitudes
app.use(express.json());

// Rutas de la API
app.use('/api/auth', authRoutes);

// Establecer el puerto del servidor
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

4.4.Desarrollo de Frontend

El frontend en React permite que los usuarios interactúen con la API. Se implementaron formularios de registro y login que envían datos a las rutas correspondientes de la API.

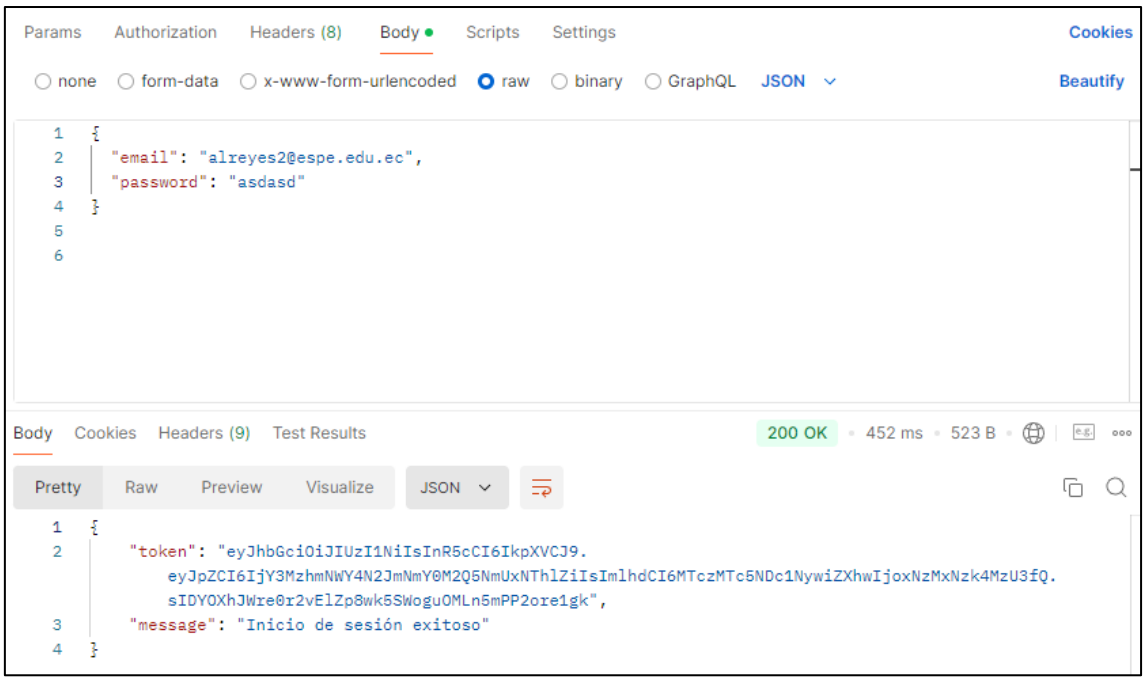




4.5.Resultados

El desarrollo de la API REST y el frontend en React fue exitoso. La API gestiona correctamente las operaciones de registro e inicio de sesión, y el frontend permite a los usuarios interactuar con la API de manera efectiva. Las pruebas realizadas con Postman confirmaron que las rutas están funcionando según lo esperado.

Como se ve en el siguiente grafica el api sirve y el token de autentificación también en la respuesta de postman



En la siguiente imagen se puede comprobar que la contraseña si esta encriptada.

5. Conclusiones

El desarrollo de la API REST y la implementación del sistema de autenticación de usuarios utilizando Node.js, Express, MongoDB, bcryptjs y JSON Web Tokens (JWT) han sido exitosos y han permitido cumplir con los objetivos establecidos en el informe. Durante el proceso, se ha logrado crear una solución funcional y segura que permite gestionar el registro e inicio de sesión de usuarios. A través de esta API, los usuarios pueden crear cuentas, iniciar sesión y mantener sesiones autenticadas mediante el uso de tokens JWT.

El desarrollo del frontend con React ha sido efectivo para interactuar con la API de manera fluida y responder adecuadamente a las acciones del usuario. La implementación de pruebas utilizando Postman ha validado que las rutas de la API funcionan correctamente y responden a las expectativas de seguridad y rendimiento.

6. Bibliografía

Node.js Documentation. (n.d.). Recuperado de https://nodejs.org/en/docs/

Express.js Documentation. (n.d.). Recuperado de https://expressjs.com/

MongoDB Documentation. (n.d.). Recuperado de https://www.mongodb.com/docs/

JWT.io. (n.d.). Recuperado de https://jwt.io/

bcryptjs Documentation. (n.d.). Recuperado de https://www.npmjs.com/package/bcryptjs

Github:

https://github.com/ANTHONYNESTORVILLARREALMACIAS/Desarrollo-Web-Avanzado-AnthonyV-ArielR.git