**Mòdul:** MP08- Desplegament d’aplicacions web **UF:** UF2 – Servidors d’aplicacions web **Professor:** Albert Guardiola

**Data límit d’entrega:** 23/01/2025 23:59

**Mètode d’entrega:** Per mitjà del Clickedu de l’assignatura. Les activitats entregades més enllà

de la data límit només podran obtenir una nota de 5.

**Instruccions:** S’ha d’entregar un únic document amb el nom:

***MP08-UF2-PF-Nom\_Alumne.doc (o pdf)***

Es valorará la presentació.

**ACTIVITAT AVALUABLE PF**

**Resultats de l’aprenentatge:**

RA1. Implanta aplicacions web en servidors d’aplicacions avaluant i aplicant criteris de

configuració per al seu funcionament segur.

El projecte final de M8-UF2 consisteix a lliurar el **backend de la web de llibres que s’ha**

**estat desenenvolupant durant la UF**, amb els següent elements d’arquitectura:

1. Fitxer principal ***app.js*** amb instanciació i configuració del servidor Express i el seu

*middleware*.

1. Fitxer de rutes ***routes/routes.js.***
2. Fitxer de controladors ***controllers/books.js*** (IMPORTANT: el controlador ha de ser totalment agnòstic del tipus de base de dades utilitzada).
3. Dues versions de model de la llibreria (una per MySQL, i una altra per MongoDB): a ***models/Library.js*** i ***models/LibraryMongo.js*.**
4. El *middleware* necessari per a gestionar autenticació per JWT.
   1. Una ruta */api/login* a la que el client podrà fer un POST de login i contrassenya, a partir dels quals el servidor generarà un JWT que retornarà a l’usuari (*res.json(...)*).
   2. El GET haurà d’estar obert a qualsevol usuari.
   3. Tots els altres mètodes (POST, PUT, DELETE) hauran d’estar només disponibles per a clients que incloguin el JWT a les capçaleres de la petició (**veure annexos d’aquest document**).
   4. Les funcions de generació i verificació del token JWT hauran de trobar-se a ***mw/auth.js*.**
5. **S’haurà de fer al client *cliente\_otro\_dominio* els canvis necessaris per**

**implementar l’autenticació amb JWT.**

**Avaluació:**

**60%**: El dia d’avaluació, s’haurà de **mostrar al professor la funcionalitat completa del programa en la versió MySQL.**

-Login i generació del token JWT.

-Llistat, adició, modificació i esborrament de llibres en BBDD.

**40%**S’haurà d’entregar el **codi en un repositori, que haurà d’incloure una carpeta**

***docs* amb un document** que expliqui:

-Quines són les parts de l’arquitectura de l’aplicació (amb captures)

-Com s’ha fet l’ adaptació del model de la llibreria de MySQL a Mongo.

-Captures de la funcionalitat completa usant Mongo (llistat, adició, modificació i esborrament de llibres en BBDD).

-Explicació dels canvis que s’han hagut de fer (*backend i frontend*) per

implementar l’autenticació JWT.

1. **Quines són les parts de l’arquitectura de l’aplicació (amb captures)**

Estructura completa:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**1. Modelo (Modelos de la base de datos)**

Library.js (Modelo para MySQL)

Texto

El contenido generado por IA puede ser incorrecto.

* mongo.config.js (Configuración de MongoDB)

Texto

El contenido generado por IA puede ser incorrecto.

* mysql.config.js (Configuración de MySQL)

Texto

El contenido generado por IA puede ser incorrecto.

2. **Controlador (Lógica de negocio)**

- controllers/books.js (Maneja la lógica para obtener, crear, actualizar y eliminar libros en la base de datos)

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

* auth.js (Maneja la autenticación JWT)

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

3️. **Vista (Frontend - Cliente web)**

* index.html (Interfaz gráfica para interactuar con los libros)

Texto

El contenido generado por IA puede ser incorrecto.

* + script.js (Lógica del frontend para hacer peticiones a la API y actualizar la interfaz)

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

4️. **Rutas y Servidor (Express.js y Middlewares)**

* routes.js (Define las rutas para acceder a los controladores)

Texto

El contenido generado por IA puede ser incorrecto.

* + app.js (Configura el servidor Express, maneja CORS, JSON y las rutas)

Texto

El contenido generado por IA puede ser incorrecto.

**2: ¿Cómo se ha hecho la adaptación del modelo de la librería de MySQL a MongoDB?**

La adaptación de MySQL a MongoDB se hizo en dos partes:

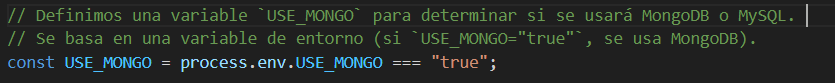
**1. Manteniendo la lógica en books.js**

* 1. En controllers/books.js se agregó la opción de trabajar con MongoDB o MySQL utilizando la variable USE\_MONGO.
  2. Se creó la función getDbCollection() para cambiar dinámicamente entre MongoDB y MySQL sin afectar la lógica principal.

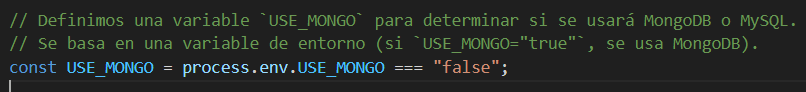
Texto

El contenido generado por IA puede ser incorrecto.

Al momento de usar MongoDb



Al usar Mysql:



**2. Creando un archivo de configuración para MongoDB**

* Se agregó mongo.config.js para almacenar la URI de conexión de MongoDB Atlas.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**3. Ajustando los métodos para manejar \_id en MongoDB**

* 1. En MongoDB, los documentos no tienen un id numérico como en MySQL, sino un \_id en formato ObjectId.
  2. Se modificaron las consultas en books.js para usar ObjectId(req.body.id) en las operaciones **update y delete**.

Texto

El contenido generado por IA puede ser incorrecto.

**📌 Pregunta 3: Capturas de la funcionalidad completa usando MongoDB (listado, adición, modificación y eliminación de libros)**

USO DE MONGODB(ATLAS):

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

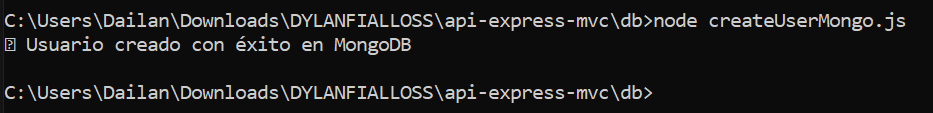
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

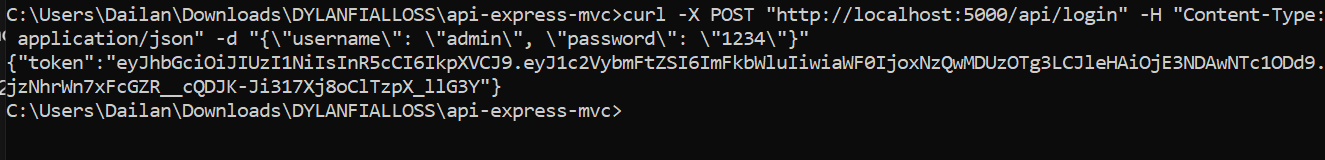
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Creacion de usuario(Mongo) para la verificacion en JWT:



Creacion del token con authJWT:



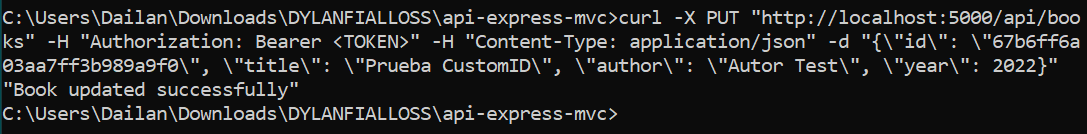
Libro correctamente creado con el token en funcionamiento:

Crear Libro:

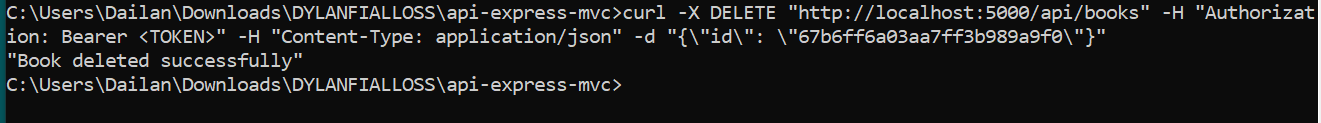
Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Modificar Libro:



Eliminar Libro:



AHORA VISUALMETNE COMO UN USUARIO:

Nos registramos con el usuario(en este caso dylan:1234)

Sale que ya existía porque estaba haciendo varias pruebas.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

**INICIAMOS SESION:**

NO SE EXAMENTE COMO MOSTRARTELO PERO AQUI ESTA UNA VEZ INICIADO SESION:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Aqui se muestra el usuario creado y guardado en el gestor de base de datos MongoDb:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**ANEXO 1. Ciclo de vida del token JWT**

****

1. Comenzaríamos desde el cliente, haciendo una petición POST para enviar el usuario y contraseña, y realizar el proceso de login.
2. Se comprobaría que ese usuario y su contraseña son correctos, y de serlos, generar el token JWT …
3. …para devolverlo al usuario.
4. A partir de ahí la aplicación cliente, con ese token, haría peticiones solicitando recursos, siempre con ese token JWT dentro de un encabezado, que sería Authorization: Bearer XXXXXXX, siendo Bearer el tipo de prefijo seguido de todo el contenido del token.
5. En el servidor se comprobaría el token mediante la firma, para verificar que el token es seguro, y, por tanto, podemos confiar en el usuario.
6. Tras verificar que el token es correcto y saber quién es el que ha hecho la petición, podemos aplicar entonces el mecanismo de control de acceso, saber si puede acceder o no, y si es así, responder con el recurso protegido, de manera que lo podría recibir de una forma correcta.

De esta forma podríamos **implementar el proceso de autenticación**, y hacerlo, además, con estos JSON Web Token.

**ANEXO 2. Función de generación del token a partir de usuario y contraseña**

const jwt = require('jsonwebtoken'); const bcrypt = require('bcrypt'); const db = require('./db');

// Clave secreta (guárdala en .env)

const SECRET\_KEY = process.env.JWT\_SECRET || 'supersecreto123';

// Función para autenticar usuario y generar JWT const generateToken = async (username, password) => {

try {

const [rows] = await db.execute('SELECT id, username, password FROM users WHERE username = ?', [username]);

if (rows.length === 0) {

return { error: 'Usuario no encontrado' };

}

const user = rows[0];

// Comparar contraseña con la almacenada en MySQL

const isMatch = await bcrypt.compare(password, user.password); if (!isMatch) {

return { error: 'Contraseña incorrecta' };

}

// Generar JWT con datos del usuario

const payload = { username: user.username };

const token = jwt.sign(payload, SECRET\_KEY, { expiresIn: '1h' });

return { token };

} catch (err) {

console.error('Error en autenticación:', err); return { error: 'Error interno del servidor' };

}

};

module.exports = generateToken;

**ANEXO 3. Función de verificación del JWT incluido en las cabeceras**

const jwt = require('jsonwebtoken');

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

const token = req.header('Authorization');

if (!token) {

return res.status(401).json({ error: 'Acceso denegado. Token requerido.' });

}

try {

const decoded = jwt.verify(token.replace('Bearer ', ''), process.env.JWT\_SECRET);

next();

} catch (err) {

res.status(401).json({ error: 'Token inválido o expirado.' });

}

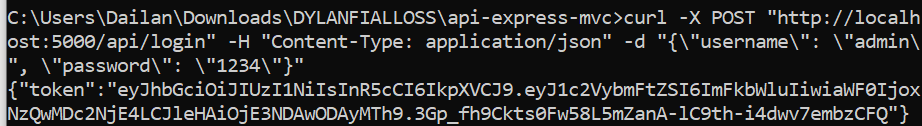
};

module.exports = jwtAuth;

Uso en mysql

Creacion del token:

curl -X POST "http://localhost:5000/api/login" -H "Content-Type: application/json" -d "{\"username\": \"admin\", \"password\": \"1234\"}"



Token temporal: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNzQwMDc2NjE4LCJleHAiOjE3NDAwODAyMTh9.3Gp\_fh9Ckts0Fw58L5mZanA-lC9th-i4dwv7embzCFQ"

Get:

curl -X GET "http://localhost:5000/api/books" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNzQwMDc2NjE4LCJleHAiOjE3NDAwODAyMTh9.3Gp\_fh9Ckts0Fw58L5mZanA-lC9th-i4dwv7embzCFQ" Texto

El contenido generado por IA puede ser incorrecto.