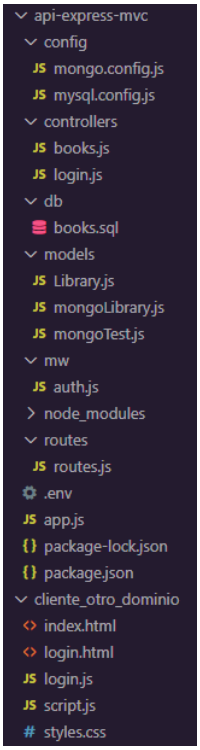
Estructura



config/: Contiene la configuración para conectar con MongoDB y MySQL

controllers/: Maneja las solicitudes del usuario y envía respuestas adecuadas

models/: Interactúa con la base de datos

mw/: Middleware que verifica las solicitudes antes de que lleguen a los controladores (por ejemplo, autenticación JWT)

routes/: Define las rutas que se enlazan con los controladores

db/: Contiene archivos SQL, como el esquema de la base de datos

cliente_otro_dominio/: Carpeta con archivos frontend (HTML, JS y CSS) para interactuar con la API

Adaptación del Modelo de la Librería de MySQL a MongoDB

Para permitir el uso de MongoDB en lugar de MySQL, se realizaron los siguientes cambios:

Configuración en MySQL

```
const mysql = require("mysql2");
const dbConfig = require("../config/mysql.config.js");
```

Configuración en MongoDB

```
const { MongoClient, ObjectId } = require("mongodb");
const dbConfig = require("../config/mongo.config.js");
```

Archivo de configuración mongo.config.js:

```
module.exports = {
    URI: "mongodb+srv://JCARLO:123@cluster0.qopaa.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0",
    DB: "library"
};
```

Funcionalidad con MongoDB

GET

```
listAll = async () => {
  console.log(this.connection)
  const [results, fields] = await this.connection.query(
    "SELECT * FROM books"
  );
  return results;
}
```

PUT

```
create = async (newBook) => {
  try {
    const [results, fields] = await this.connection.query(
      "INSERT INTO books SET ?", newBook
    );
    return results.affectedRows;
  }
  catch (error) {
    return error;
  }

};
```

DELETE

```
delete = async (delBook) => {
  try {
    const [results, fields] = await this.connection.query(
      "DELETE FROM books WHERE id = ?", [delBook.id]
    );
    return results.affectedRows;
  }
  catch (error) {
    return error;
  }
};
```

UPDATE

```
update = async (updBook) => {
  const {id, title, author, year} = updBook
  try {
    const [results, fields] = await this.connection.query(
      "UPDATE books SET title = ?, author = ?, year = ? WHERE id = ?", [title, author, year, id]
    );
    return results.affectedRows;
  }
  catch (error) {
    return error;
  }
};
```

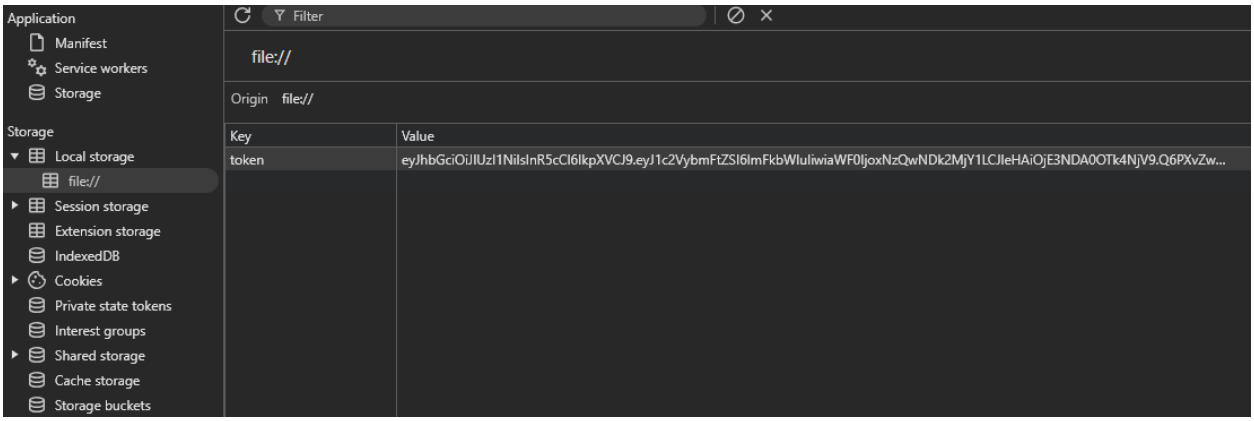
Cambios de autenticación JWT

Frontend

Utiliza el formulario del html (login.html) para iniciar la sesión y envia una solicitud al backend (login.js) para generar el token

```
async function editBook(event) {
    const token = localStorage.getItem("token");
    console.log(token)
```

```
let response = await fetch(apiUrl, {
    method: "DELETE",
    headers: {
        "Content-Type": "application/json",
        "Authorization": `Bearer ${token}`
    },
    body: JSON.stringify(deletedBook)
});
```



Backend

El archivo controller/login.js es responsable de verificar las credenciales de usuario (user/password). Si el usuario y la contraseña coinciden, se genera un token de autenticación con una validez de una hora. En caso de que las credenciales no coincidan, se mostrará un mensaje de error de autenticación

```
const generateToken = async (req, res) => {
    try {
        if (req.body.username !== "admin") {
            return res.json({ error: 'Usuario no encontrado' });
        }

        if (req.body.password !== "admin") {
            return res.json({ error: 'Contraseña incorrecta' });
        }

        const payload = { username: req.body.username };
        const token = jwt.sign(payload, SECRET_KEY, { expiresIn: '1h' });

        return res.json({ token: token });
    } catch (err) {
        console.error('Error en autenticación:', err);
        return res.status(500).json({ error: 'Error interno del servidor' });
    }
};


module.exports = {
    generateToken: generateToken
}
```

Verifica si el header es valida

```
api-express-mvc > .env
1  JWT_SECRET=admin
```

Verifica si el usuario si tiene acceso para hacer operaciones

```
// Configuración de las rutas
router.post('/api/login', login.generateToken)
router.get('/api/books', books.getBooks)

router.post('/api/books', auth.jwtAuth, books.createBook)
router.put('/api/books', auth.jwtAuth, books.updateBook)
router.delete('/api/books', auth.jwtAuth, books.deleteBook)
```