

Back end NODE

Proyecto Final

JESUS MANUEL VARGUEZ BASTO

01 de noviembre de 2023

Tabla de contenido

[Avances. 2](#_Toc155123069)

[Descripción detallada del escenario. 2](#_Toc155123070)

[Objetivo. 2](#_Toc155123071)

[Pre requisitos, 3](#_Toc155123072)

[Back end 3](#_Toc155123073)

[Front end. 3](#_Toc155123074)

[Definición de base y tablas de Libros y Autores. 3](#_Toc155123075)

[Crear base de datos y tablas 3](#_Toc155123076)

[Diagramas. 3](#_Toc155123077)

[Creación de la base de datos 4](#_Toc155123078)

[Tabla Autores 5](#_Toc155123079)

[Definición de campos. 5](#_Toc155123080)

[Creación de tabla autores. 5](#_Toc155123081)

[Tabla Libros 6](#_Toc155123082)

[Definición de campos. 6](#_Toc155123083)

[Creación de tabla libros. 7](#_Toc155123084)

[Inserta registro en las tablas de autores y libros. 7](#_Toc155123085)

[Insertar en autores 7](#_Toc155123086)

[Insertar más de 100 autores. 8](#_Toc155123087)

[Insertar en libros 10](#_Toc155123088)

[Funcionamiento de crud libros. 11](#_Toc155123089)

[Webservice Libros 11](#_Toc155123090)

[Listar Libros 12](#_Toc155123091)

[Parámetros enviados y rutas de acceso 12](#_Toc155123092)

[Resultado 12](#_Toc155123093)

[Ver Libro 13](#_Toc155123094)

[Parámetros enviados y rutas de acceso 13](#_Toc155123095)

[Resultado 13](#_Toc155123096)

[Editar Libro 14](#_Toc155123097)

[Parámetros enviados y rutas de acceso 14](#_Toc155123098)

[Resultado 14](#_Toc155123099)

[Eliminar Libro 15](#_Toc155123100)

[Parámetros enviados y rutas de acceso 15](#_Toc155123101)

[Resultado 15](#_Toc155123102)

[BACK END. 16](#_Toc155123103)

[Modificación en implementación de login en el backend. 16](#_Toc155123104)

[implementación 17](#_Toc155123105)

[Ilustración 1 Diagrama de base de datos tablas Libros Y autores/ relación uno a muchos 3](#_Toc150690497)

[Ilustración 2crear base de datos libros 4](#_Toc150690498)

[Ilustración 3crear tabla autores 5](#_Toc150690499)

[Ilustración 4crear tabla libros 6](#_Toc150690500)

[Ilustración 5 Formas d utilizar el insert 7](#_Toc150690501)

[Ilustración 6 Select a autores 7](#_Toc150690502)

[Ilustración 7 resultado de inserción en autores 8](#_Toc150690503)

[Ilustración 8 Select en autores después de inserción 9](#_Toc150690504)

[Ilustración 9 Insert y Select a libros 10](#_Toc150690505)

[Ilustración 10 Login 10](#_Toc150690506)

# 

# Avances.

## Descripción detallada del escenario.

Para este proyecto se retoma lo de los libros y se implementara acceso y login por token, así como validación de los servicios desde el front end y back end

## Objetivo.

Crear las tablas donde se almacenen los libros y otra para almacenar los autores, deben ser creadas con una relación 1 a muchos.

Crear los diagramas de la base de datos y de las tablas.

Conectar la base de datos con el backend para que el vendedor pueda consultar los datos

## Pre requisitos,

### Back end

Para el desarrollo del back para este proyecto se utilizar NodeJS con el framework Express y mysql.

Dentro de las librerías o dependencias a utilizar se instalarán las siguientes:

* **XPRESS**: facilita la creación de rutas, el manejo de solicitudes y respuestas, y la configuración de middleware.
* **EQUELIZE**: proporcionar una interfaz de JavaScript para realizar operaciones CRUD (Create, Read, Update, Delete).
* **MYSQL2**: es un conector de MySQL para Node.js que permite la conexión y la ejecución de consultas en bases de datos MySQL.
* **DOTENV**: es útil para gestionar configuraciones sensibles, como credenciales de bases de datos, en un archivo separado y no en el código fuente.
* **CORS**: habilita el control de acceso a recursos en otro dominio (CORS) en aplicaciones web. Ayuda a controlar y permitir estas solicitudes cruzadas entre dominios.

### Front end.

Para el front se considera realizarlo en react y material UI REAC

## Definición de base y tablas de Libros y Autores.

### Crear base de datos y tablas

### Diagramas.

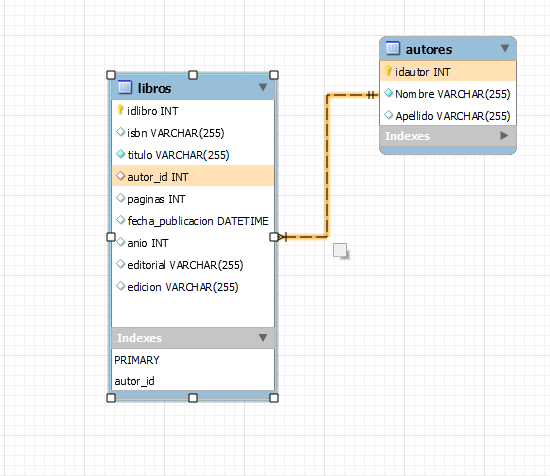


Ilustración Diagrama de base de datos tablas Libros Y autores/ relación uno a muchos

### Creación de la base de datos

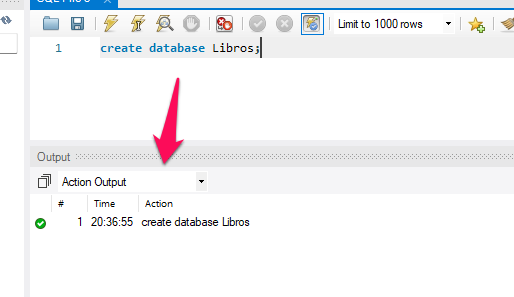


Ilustración crear base de datos libros

### Tabla Autores

#### Definición de campos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Null** | **Key** | **Default** | **Extra** |
| idautor | int | NO | PRI | NULL | auto\_increment,clave autor |
| Nombre | varchar(255) | NO |  | NULL | Nombre autor |
| Apellido | varchar(255) | NO |  | NULL | Apellido autor |

#### Creación de tabla autores.

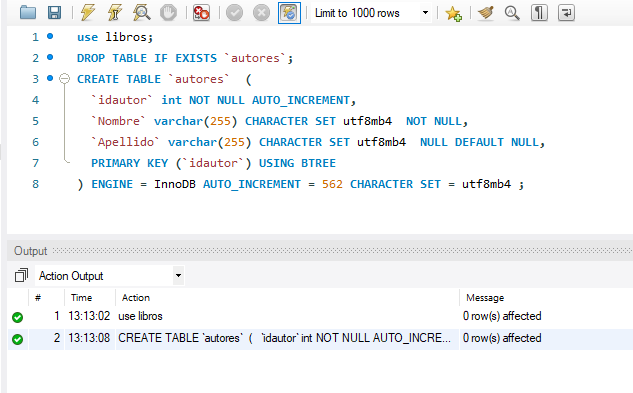


Ilustración crear tabla autores

### Tabla Libros

#### Definición de campos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Null** | **Key** | **Default** | **Extra** |
| id | int | NO | PRI |  | auto\_increment, clave libro |
| isbn | varchar(255) | SI |  | NULL | ISBN |
| titulo | varchar(255) | SI |  | NULL | titulo libro |
| autor\_id | int | si |  | NULL | clave autor,llave forenea |
| paginas | int | SI |  | NULL | paginas |
| fecha\_publicacion | datetime | SI |  | NULL | Fecha de publicacion |
| anio | int |  |  |  | Año |
| editorial | varchar(255) | SI |  | NULL | Editorial |
| edicion | varchar(255) | SI |  | NULL | edicion |

#### Creación de tabla libros.

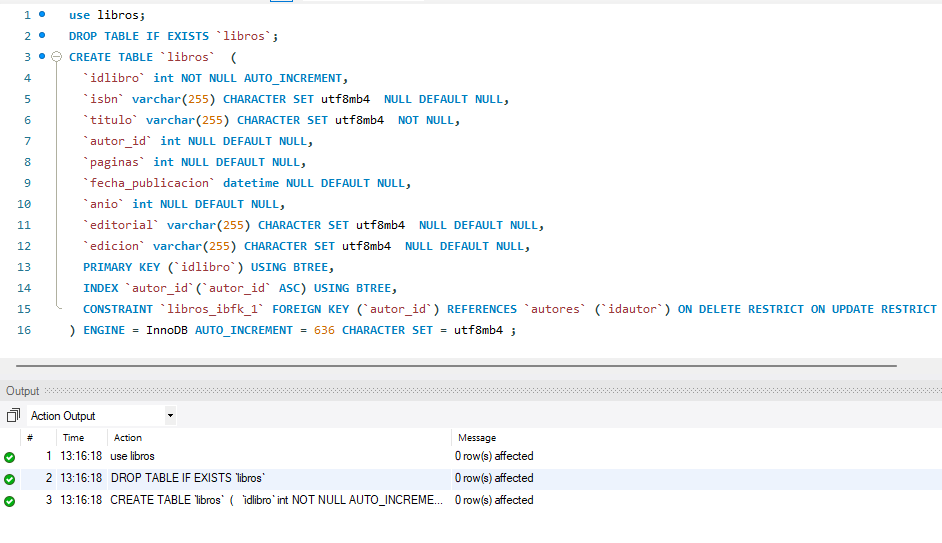


Ilustración crear tabla libros

### Inserta registro en las tablas de autores y libros.

Seleccionamos la base a utilizar, desplegamos las bases disponibles e aplicamos **un insert into** a autores y libros con os datos requeridos

#### Insertar en autores

Definimos la consulta para la inserción de datos en autores pidiendo utilizarlo de la siguiente manera.

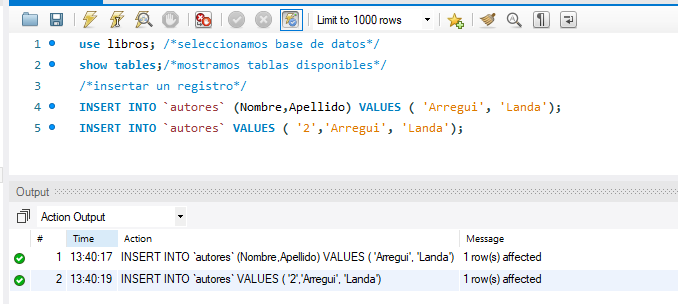


Ilustración Formas d utilizar el insert

El resultado de la inserción quedaría de la siguiente manera.

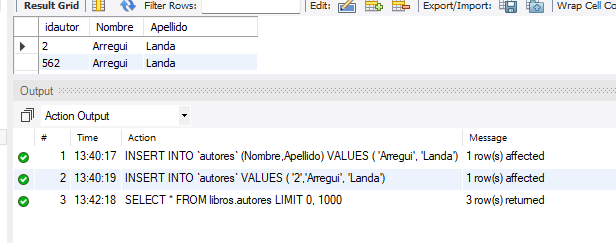


Ilustración Select a autores

#### Insertar más de 100 autores.

Para este caso ya tenemos definido las claves de autor por lo que utilizaremos la inserción donde colocamos todos los valores de cada columna.

Ejemplo: **INSERT INTO `autores` VALUES ( '2','Arregui', 'Landa');**

Procedemos a insertar los datos que tenemos.

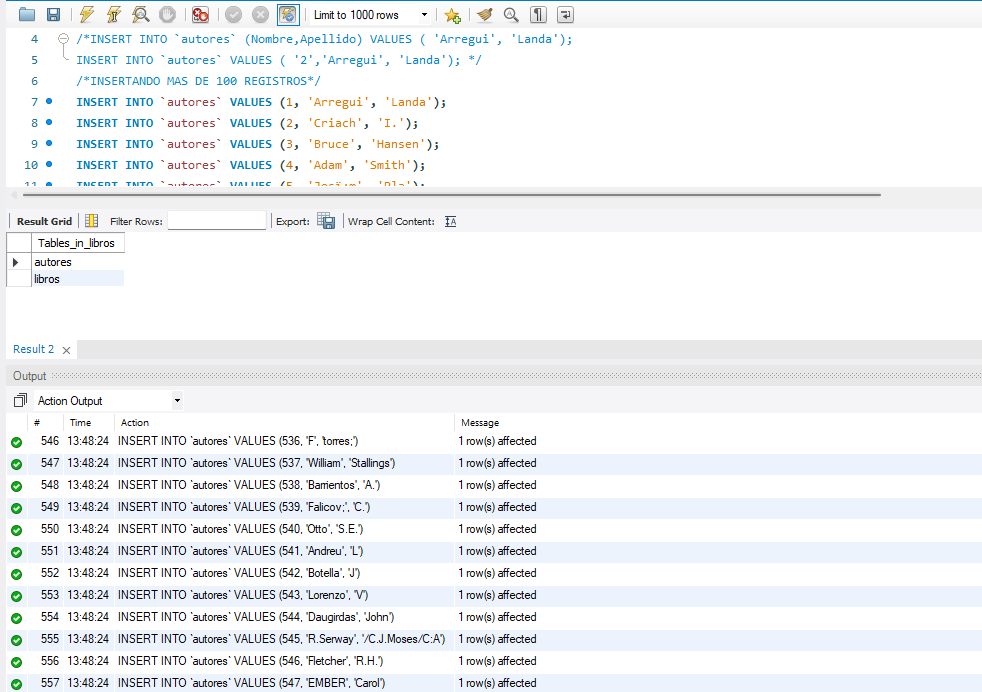


Ilustración resultado de inserción en autores

Se aplican las inserciones sin problema. Aplicamos y select a la tabla autores para verificar.

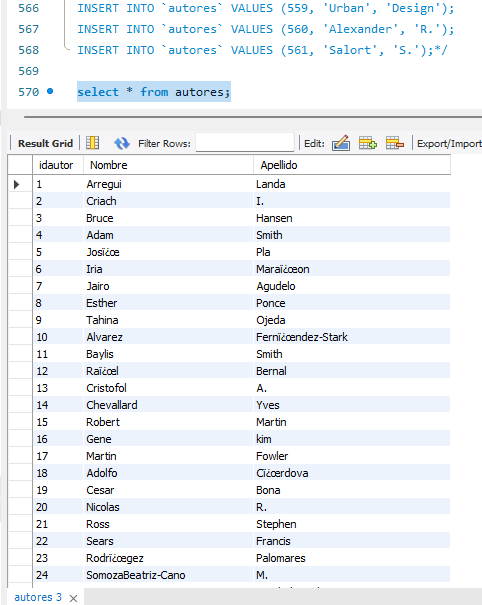


Ilustración Select en autores después de inserción

#### Insertar en libros

Definimos la consulta para la inserción de datos en libros como ya contamos con datos quedaría de la siguiente manera.

Ejemplo: **INSERT INTO `libros` VALUES (2, 'ISBN 10: 8418058013', 'Ciencia en la primera infancia', NULL, NULL, '2023-07-30 00:00:00', 2019, 'Grao', '48');**

Procedemos a insertar los datos que tenemos.

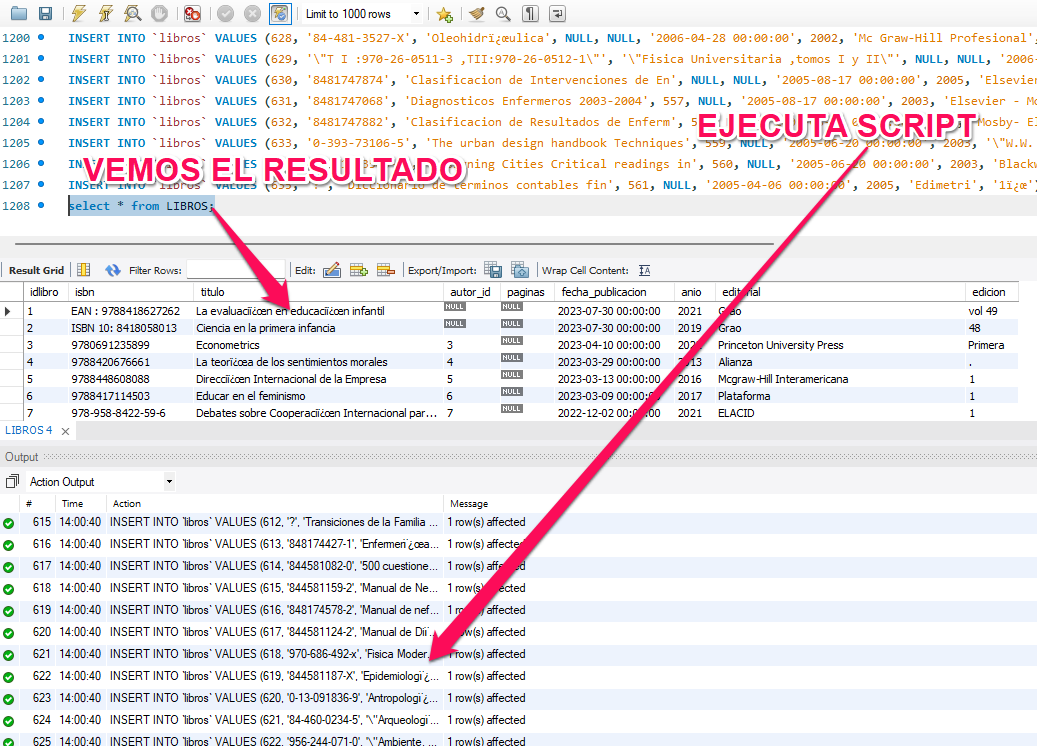


Ilustración Insert y Select a libros

.

Al aplicar los scripts de inserción, podemos visualizar que no hubo problemas y si aplicamos un select los datos se muestran.

# Funcionamiento de crud libros.

## Webservice Libros

### Listar Libros

#### Parámetros enviados y rutas de acceso

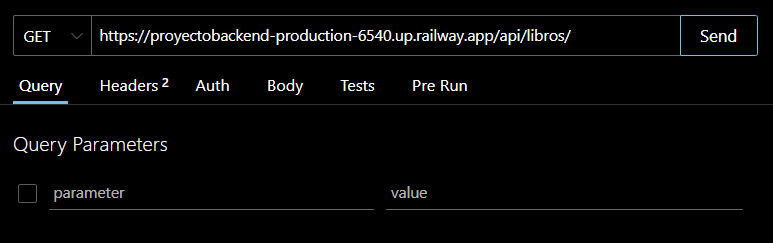


Ilustración end point para ver listado de libros

#### Resultado

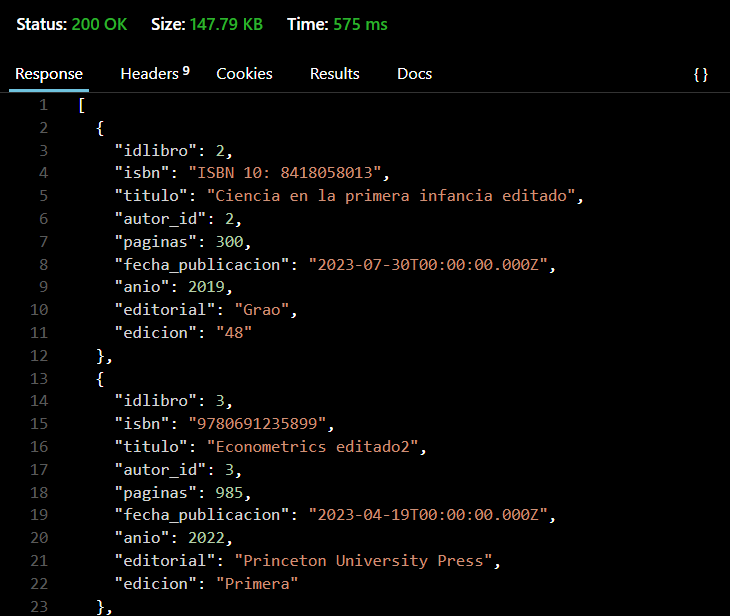


Ilustración resultado Listar Libros

### Ver Libro

#### Parámetros enviados y rutas de acceso

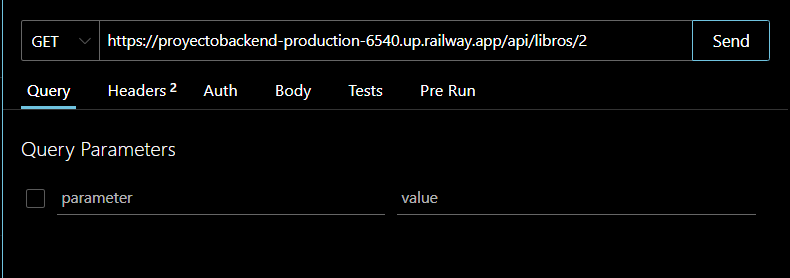


Ilustración ruta para ver un libro se añade clave de libro

#### Resultado

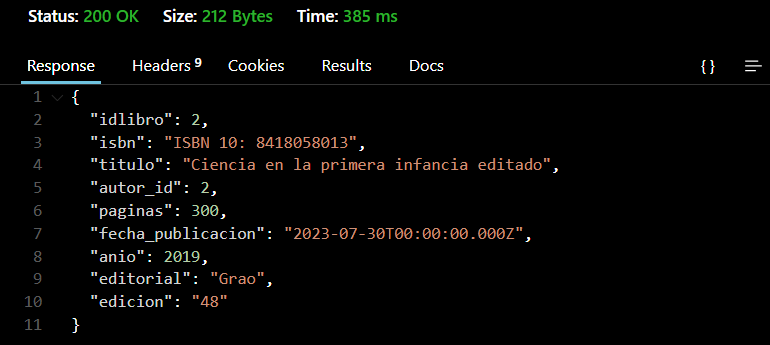


Ilustración resultado de ver el libro 2

### Editar Libro

#### Parámetros enviados y rutas de acceso

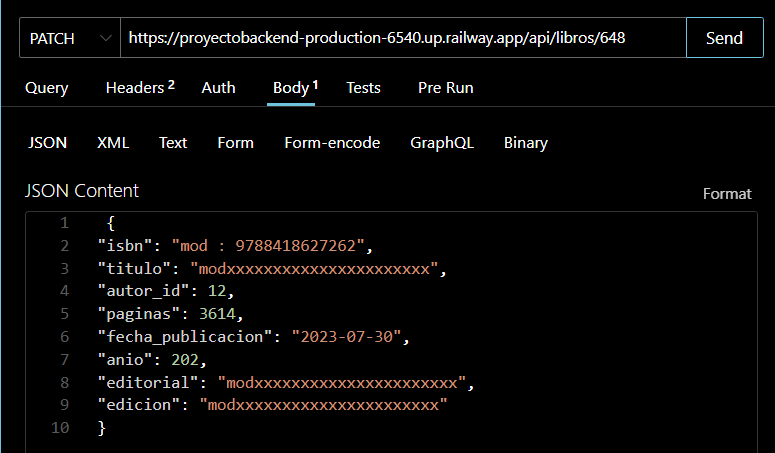


Ilustración url y parámetro json para editar libros

#### Resultado

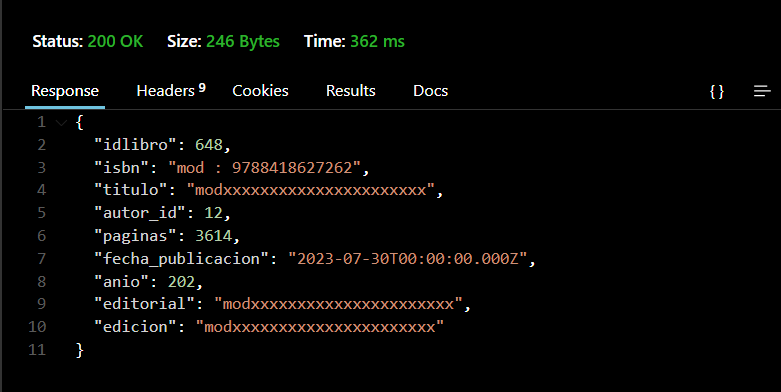


Ilustración resultado de la modificación respondiendo de regreso el json

### Eliminar Libro

#### Parámetros enviados y rutas de acceso

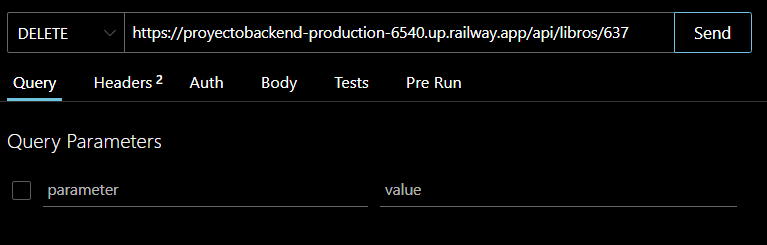


Ilustración ruta para eliminar libro metodo DELETE

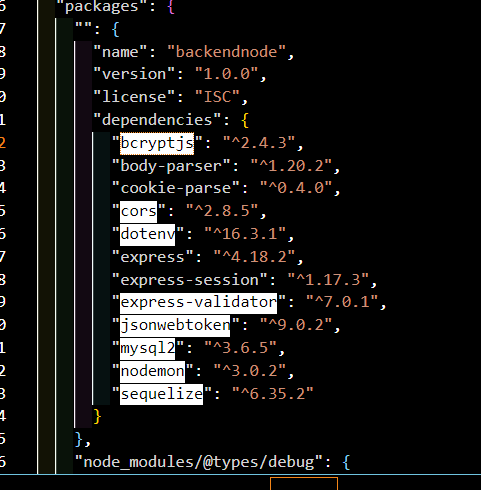
#### Resultado

Para este caso solo se devuelve es estatus 204



# BACK END.

## Modificación en implementación de login en el backend.



Para el back y las modificaciones se requerirá instalar las siguientes dependencias

**Bcryptjs**: para incriptar la contraseña del usuario y nos servirá para comparar y validar la contraseña

**Dotenv**: para crear las variables de entorno

**express-validator**: para validar los parámetros envidos a nuestros servicios

**jsonwebtoken**: para generar la autentificación por token

**mysql2**. Para interactuar con nuestra base mysql y generar un crud

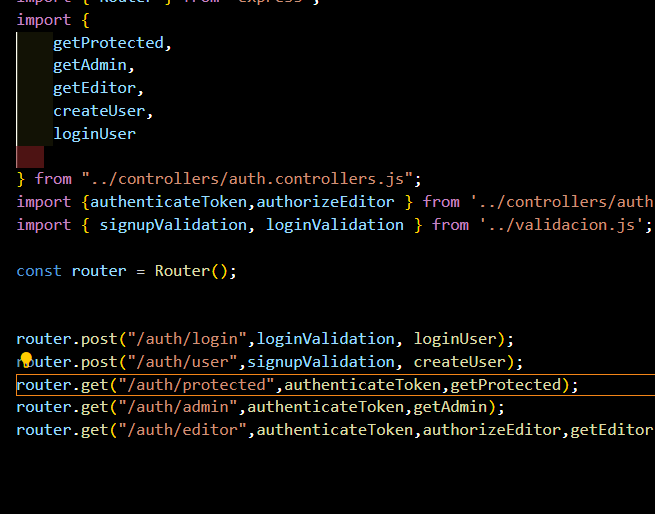
**nodemon**: permite actualizar el servidor cuando se realice un cambio se utilizara en modo desarrollo

**sequelize**: para crear el esquema de las tablas si no existen en la base de datos.

## implementación

para este caso lo implementaremos en el registro de usuario y en el login

se crean las siguientes rutas



Login

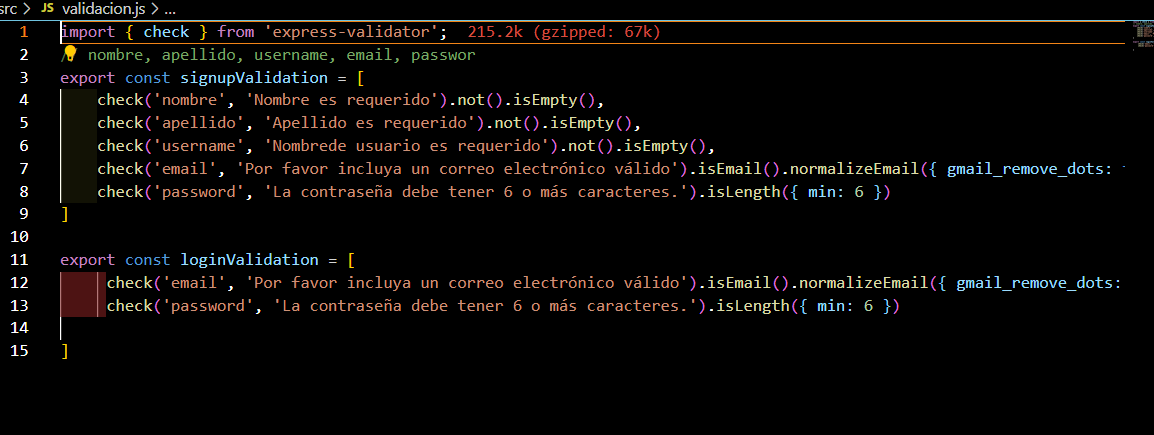
router.post("/auth/login",loginValidation, loginUser);

Registro de usuario

router.post("/auth/user",signupValidation, createUser);

en este punto utilizamos y creamos las dependencias de **express-validator, jsonwebtoken, Bcryptjs**

**para la validación se creo validación.js que valida lo enviado el ws**

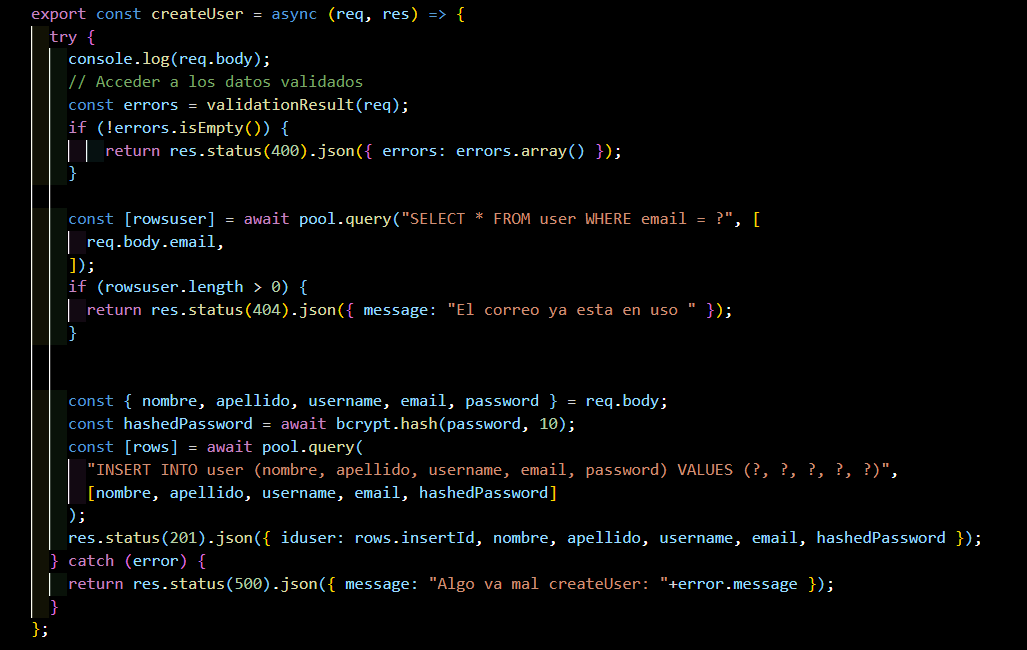
****

## 

Para la validación del token de creo el archivo auth.en donde utilizaremos authenticateToken para validar el token generados. Dentro del mismo esta generateAccessToken que nos generara al accear ala aplicacio en la aplicación



Se creo el archivo auth.controllers.js en donde se encuentra el registro de la cuenta de usuario y el login

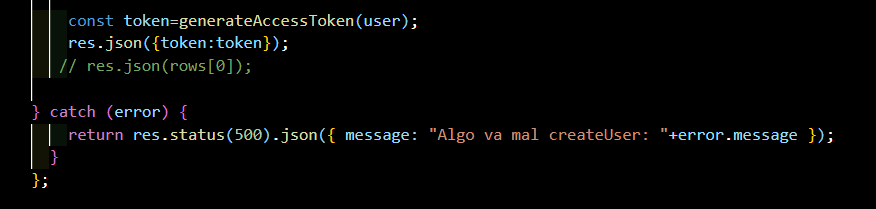


En esta función recibimos los datos enviados validamos el correo para saber si no esta en uso, encriptamos la contraseña y de ultimo registramos las cuentas.

En el login recibimos el correo y la contraseña para, se valida que existe, se valida y compara la contraseña encriptada devuelta por la validación



Y por ultimo generamos el token



La cual nos servirá para verificar y validar dentro del sistema.

Para la interacción con la base se creo el modelo donde definimos los campos a utilizar y que se crean si no existe la tabla.

import { Model, DataTypes, Sequelize } from 'sequelize';

const USER\_TABLE = 'user';

export class User extends Model {

    static config(sequelize) {

        return {

            sequelize,

            tableName: USER\_TABLE,

            modelName: 'User',

            timestamps: false

        }

    }

}

export const UserSchema = {

    iduser: {

        allowNull: false,

        autoIncrement: true,

        Comment:"Clave User",

        primaryKey: true,

        type: DataTypes.INTEGER

    },

    nombre: {

        allowNull: false,

        type: DataTypes.STRING,

        Comment:"Nombre de usuario",

        field: 'nombre'

    },apellido: {

        allowNull: true,

        type: DataTypes.STRING,

        Comment:"Apellido usuario",

        field: 'apellido'

    },username: {

        allowNull: false,

        type: DataTypes.STRING,

        Comment:"nombre de usuario",

        field: 'username'

    },

    email: {

        allowNull: false,

        type: DataTypes.STRING,

        Comment:"correo",

        field: 'email'

    },

    password: {

        allowNull: false,

        type: DataTypes.STRING,

        Comment:"password",

        field: 'password'

    }

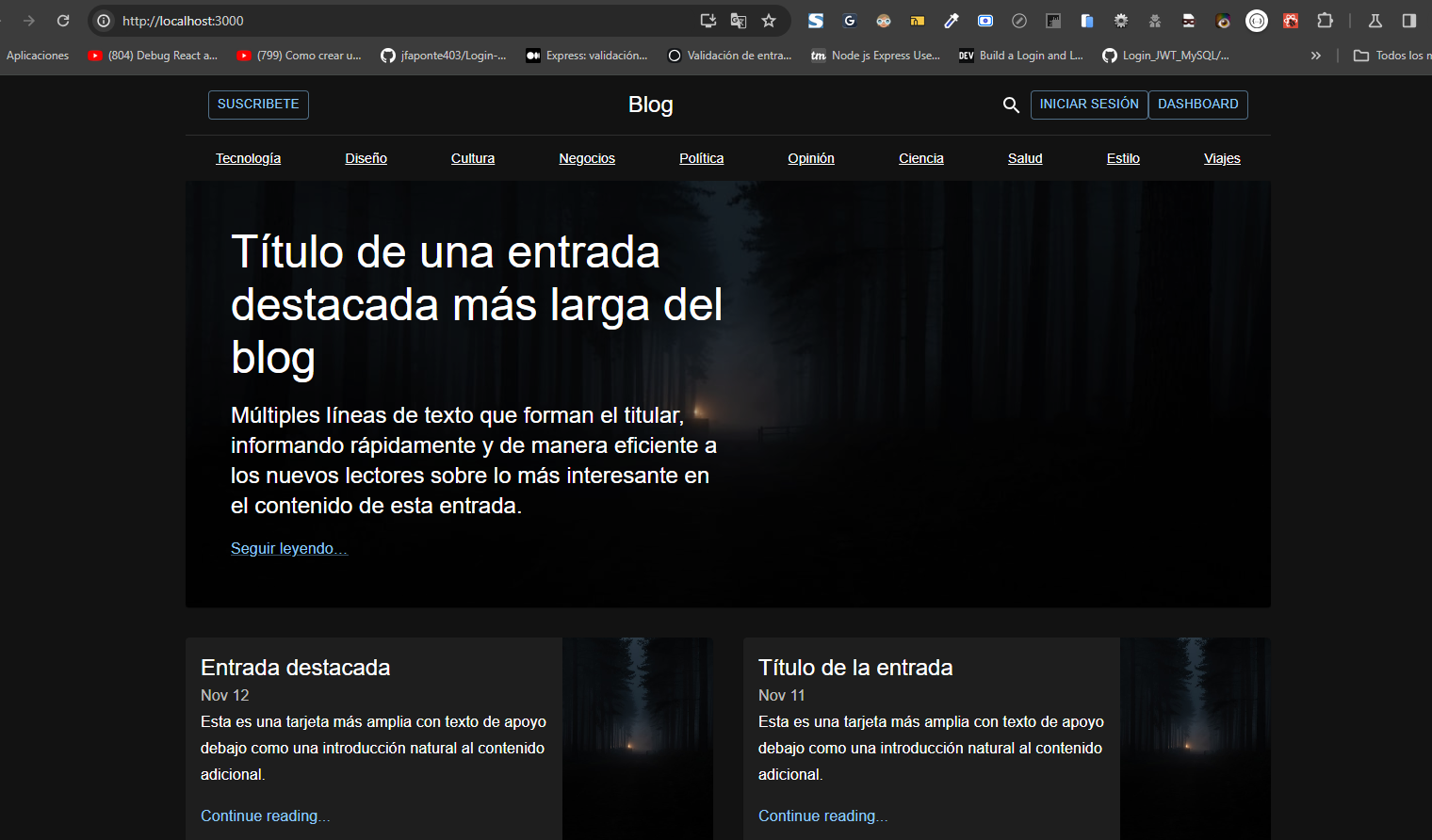
}

## FRONT END.

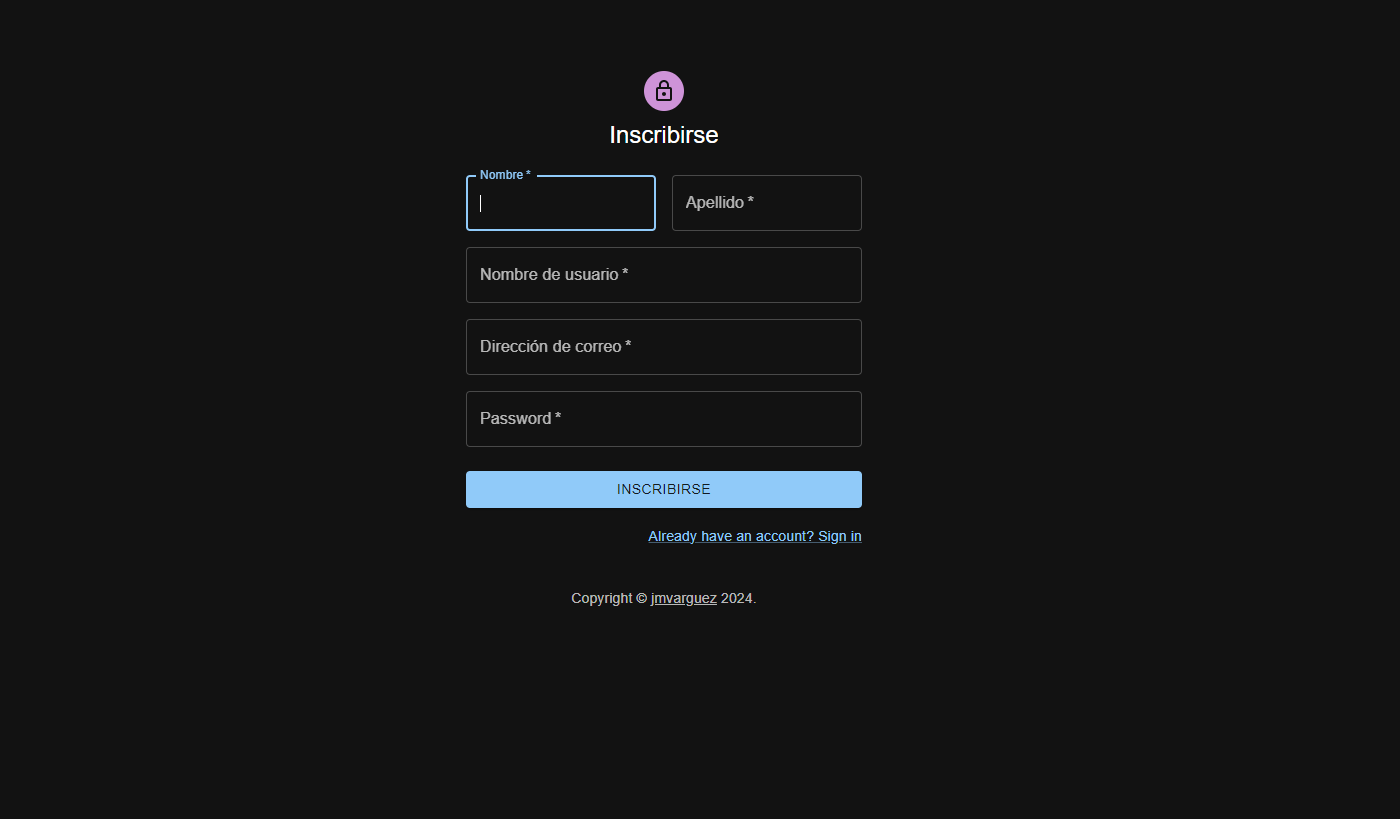
Modificación en implementación de login, para el front end se realizó un cambio en dónde implementaremos material UI y se creara un sitio utilizando plantillas y componente de esta librería

Creando de esta manera una plataforma con dashboard.

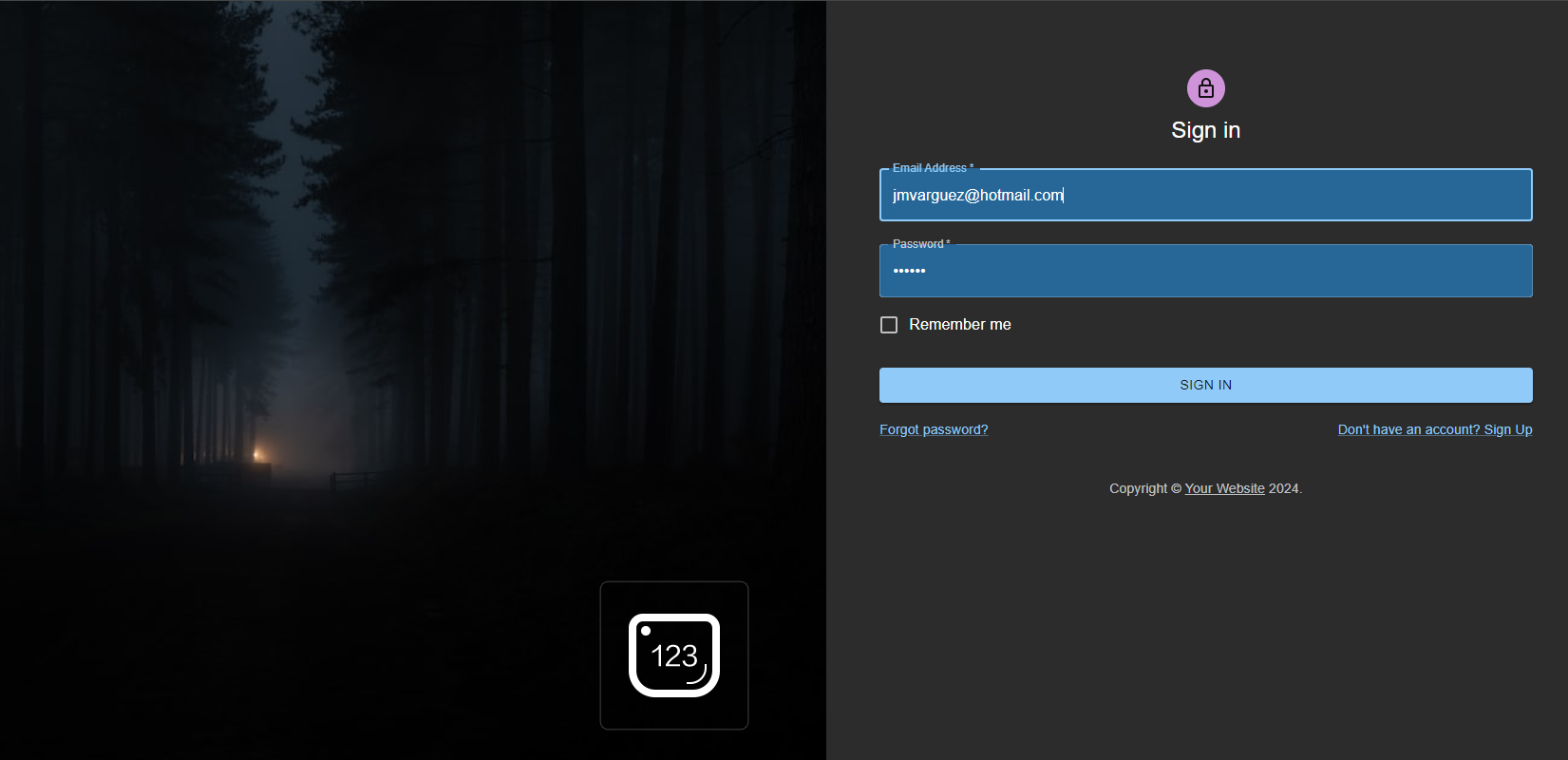
### Página principal.



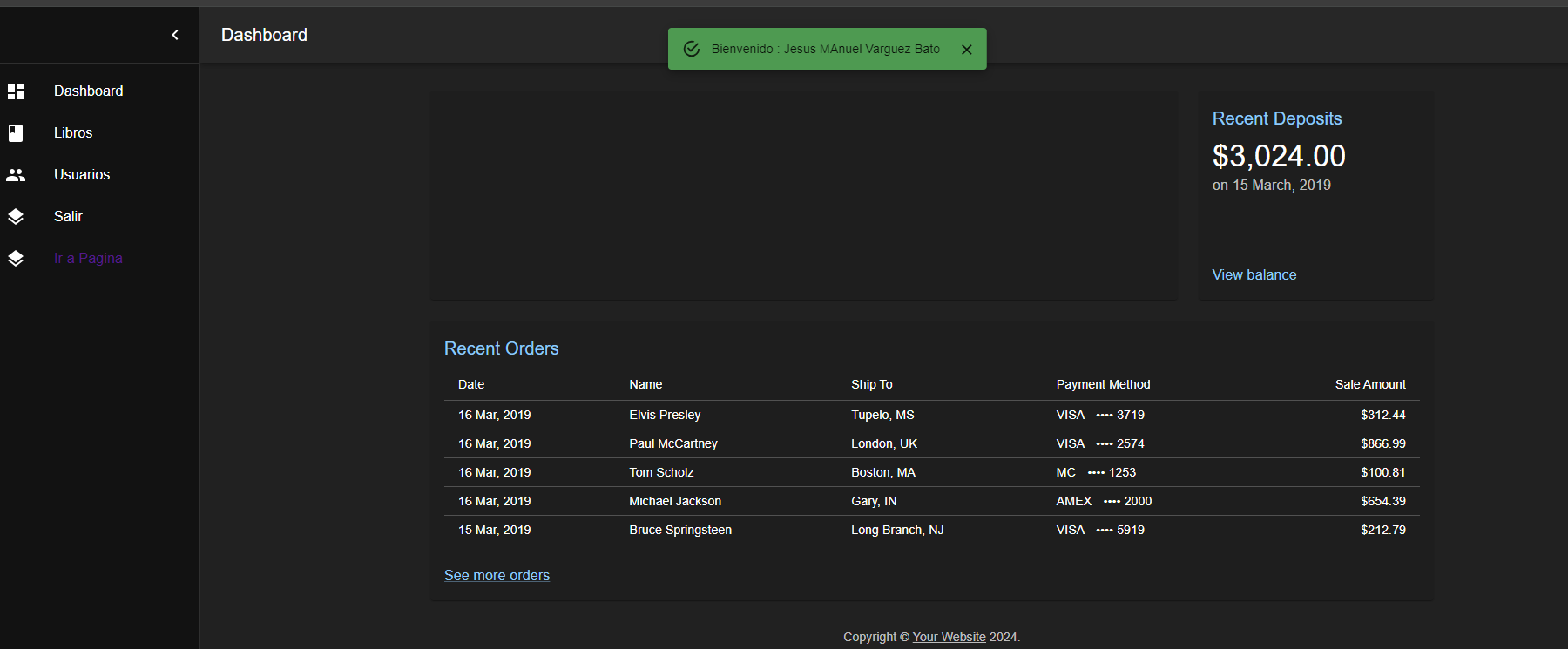
### Registro de cuenta.



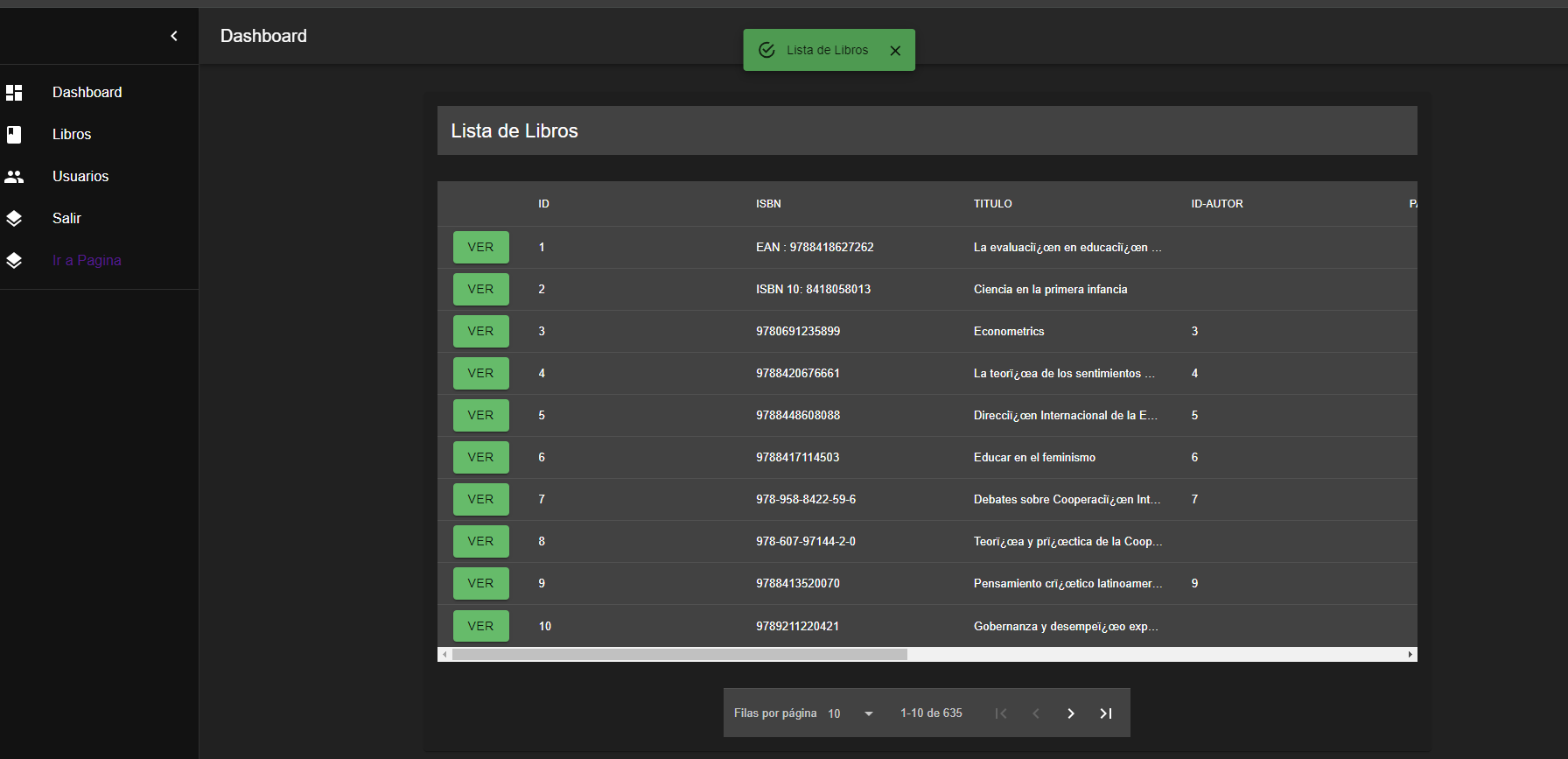
### Login.



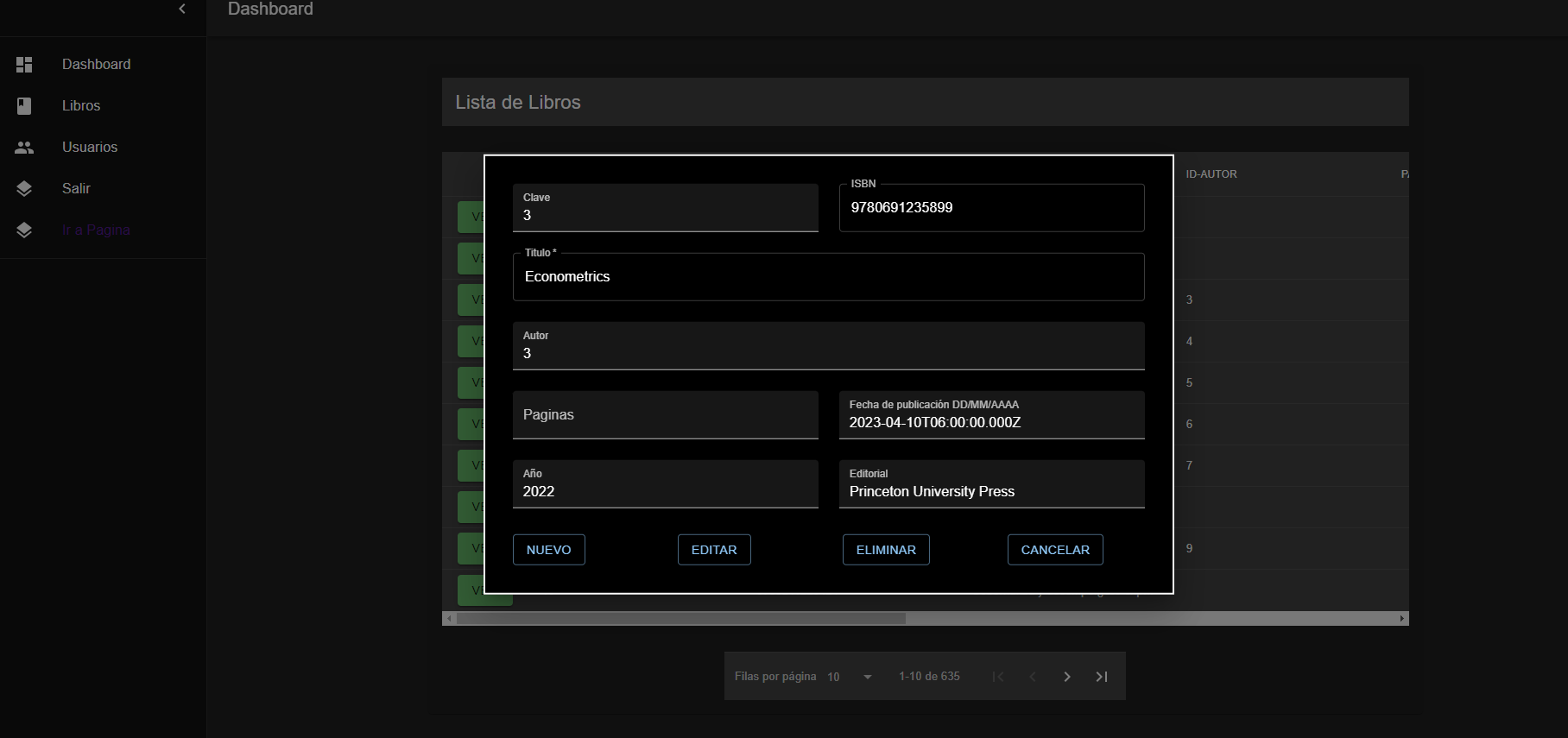
### Dashboard.



### Listado de libros.



### Crud libros.



Se implementara autentificación con token a cada servicio crud de librería, para la implantación de seguridad.

Ya implementado trataremos de implementarlo en dokers