Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Segundo Semestre 2023
Practicas Iniciales
Sección: “ F- ”

Manual Técnico

Grupo No.9

Nombre de los estudiantes:

María Patricia Serrano Ramírez	202201989
Valery Nicolle Gálvez García	202200141
Ariel Josué López Gálvez	202200185
Sergio Joel Rodas Valdez	202200271
Lizz Andrea Morelia Castellanos Salazar	201708997

Nombre de los tutores:

Justin Josué Aguirre Román	202004734
Ronaldo Javier Posadas Guerra	202004770

Guatemala, 21 de septiembre de 2023

Introducción

Este manual técnico proporciona una guía detallada y completa para entender la estructura, el funcionamiento y la implementación de nuestra aplicación. Está diseñado para desarrolladores y cualquier persona interesada en comprender en profundidad cómo funciona nuestra aplicación, desde la configuración del *servidor* hasta la gestión de datos en la *base de datos*.

En este manual, se describen en detalle todos los componentes clave de la aplicación, incluyendo el servidor *Express*, las *rutas*, los *controladores* y la conexión a la base de datos *MySQL*. Además, se proporciona información sobre la estructura de la base de datos y los procedimientos almacenados utilizados para gestionar los datos.

La aplicación que se describe en este manual está diseñada para permitir a los usuarios registrar *cuentas*, *crear publicaciones*, *agregar comentarios* y *gestionar cursos*. Este manual está diseñado para ser una herramienta útil tanto para aquellos que están familiarizados con el desarrollo web como para aquellos que desean aprender sobre los conceptos y la implementación detrás de una aplicación web moderna.

Manual Técnico del Servidor

Componentes de la Aplicación:

index.js

Figura 1. Archivo index.js

Este archivo es el punto de entrada principal de la aplicación y configura el servidor web utilizando *Express.js*. Aquí se encuentran las siguientes configuraciones y rutas:

- **Configuraciones del Servidor:**
  - Uso del middleware *cors* para habilitar el control de acceso a recursos compartidos entre diferentes dominios (*CORS*).
  - Uso del middleware *morgan* en modo *dev* para registrar las peticiones *HTTP* en la consola.

- **Configuraciones de Archivos del Servidor:**
  - Configuración de límites de carga para datos *JSON*, datos codificados y datos de texto para limitar el tamaño de los archivos que se pueden cargar.

- **Rutas:**
  - Ruta raíz ("/"): Devuelve un mensaje "*Hello from Backend"* como respuesta.
  - Uso del archivo de rutas *"./Routes/routes.js"* para definir rutas adicionales relacionadas con *usuarios*, *publicaciones*, *cursos* y *comentarios*.

- **Inicio del Servidor:**
  - El servidor se inicia en el puerto 4000 y se conecta a la base de datos utilizando el módulo de conexión definido en *"./DataBase/Database.js"*.

```
const express = require("express");
const cors = require("cors");
const morgan = require("morgan");
const app = express();
const ConexionDB = require("./DataBase/Database")

/*Configuraciones del servidor*/
app.use(cors());/* Politicas del servidor */
app.use(morgan("dev")); /* Controlador de peticiones en consola*/

/* Configuraciones de archivos del servidor*/
app.use(
    express.json({
        limit: "200mb", /* El servidor aguanta archivos no mas de 200 mb */
    })
);
app.use(
    express.urlencoded({
        limit: "200mb", /* El servidor aguanta archivos no mas de 200 mb */
        extended: true,
    })
);
app.use(
    express.text({
        limit: "200mb", /* El servidor aguanta archivos no mas de 200 mb */
    })
);
/* Creacion de las rutas */
app.get("/", (req, res) => {
    res.status(200).send({ message: "Hello from Backend" });
});

app.use("/Grupo9", require("./Routes/routes.js"));

app.listen(4000, () => {
    ConexionDB.Conectar()
    console.log("Servidor levantado con exito en el puerto " + 4000);
});
```

Figura 2. Código index.js

Carpeta Routes

routes.js

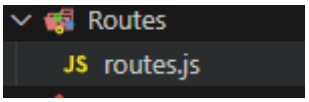

Figura 3. Archivo routes.js

Este archivo contiene las rutas y controladores de la aplicación. Aquí se definen las rutas para diversas operaciones relacionadas con *usuarios*, *publicaciones*, *cursos* y *comentarios*. Algunas de las rutas incluyen:

- Operaciones relacionadas con *usuarios* (*registro*, *inicio de sesión*, *actualización*, *lista de usuarios*, *datos del usuario*, *restablecimiento de contraseña*).

- Operaciones relacionadas con *comentarios* (*agregar comentario*, *listar comentarios*).
- Operaciones relacionadas con *publicaciones* (*agregar publicación*, *listar publicaciones*, *listar mis publicaciones*).
- Operaciones relacionadas con *cursos* y *catedráticos* (*listar cursos*, *listar catedráticos*).

```
const express = require("express");
const router = express.Router();
const {addUser,Login,updateUser,ListUser,MydataUser,ForgotPassword} = require("../Controllers/userscontrollers")
const {Listcourses, Listcate} = require("../Controllers/coursecontrollers")
const {addpublication,listpublications,mylistpublications} = require("../Controllers/publicationcontrollers")
const {addcomment,listcomment} = require("../Controllers/commentscontrollers")

/* Usuarios */
router.post('/addUser', addUser);
router.post('/Login', Login);
router.post('/updateUser', updateUser);
router.post('/ListUser', ListUser);
router.post('/MydataUser', MydataUser);
router.post('/ForgotPassword', ForgotPassword);

/* Comentarios */
router.post('/addcomment', addcomment);
router.post('/listcomment', listcomment);

/* Publicaciones */
router.post('/addpublication', addpublication);
router.get('/listpublications', listpublications);
router.post('/mylistpublications', mylistpublications);

/* Cursos */
router.get('/Listcourses', Listcourses);
router.get('/Listcate', Listcate);

module.exports = router;
```

Figura 4. Código routes.js

DataBase

Database.js

Este archivo contiene la configuración y la lógica de conexión a la base de datos *MySQL*. Aquí se definen las credenciales de la base de datos y se utiliza el módulo *mysql* para establecer la conexión. También se exporta la conexión para que esté disponible en otros archivos de la aplicación. La función *Conectar* se utiliza para establecer la conexión a la base de datos.

```
const mysql = require('mysql')

/*Credenciales de su base de datos*/
const conexion = mysql.createConnection(
  {
    host: 'localhost',
    user: 'root',
    password: 'corona2468',
    database: 'usuarios'
  }
);

const Conectar = () => {
    conexion.connect( err => {
        if(err) throw err
        console.log("conectado a la DB");
    })
}

module.exports = {
    Conectar,
    conexion
}
```

Figura 5. Código Database.js

Script.sql

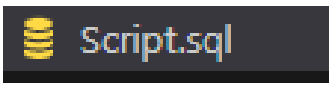

Figura 6. archivo Script.sql

Este archivo *SQL* contiene los comandos para crear la estructura de la base de datos, incluyendo tablas para *usuarios*, *catedráticos*, *cursos*, *publicaciones*, *comentarios* y *asignaciones*. Además, se incluyen comandos *SQL* para insertar datos de ejemplo en estas tablas. Asegúrese de ejecutar estos comandos en su sistema de gestión de bases de datos *MySQL* antes de utilizar la aplicación.

```sql
DROP DATABASE Usuarios;
Create Database Usuarios;
Use Usuarios;

ALTER USER 'root'@'localhost' IDENTIFIED WITH 'mysql_native_password' BY 'Vale0401*';

-- Creacion de la Tabla de Usuarios
Create table USUARIO(
    Carnet BIGINT PRIMARY KEY NOT NULL,
    Nombre VARCHAR(100),
    Apellido VARCHAR(100),
    Correo VARCHAR(50),
    Contrasena VARCHAR(50)
);

-- Creacion de la Tabla de curos
Create table CATEDRATICO(
    idcatedratico INT AUTO_INCREMENT PRIMARY KEY,
    Nombre  VARCHAR(100)
);

-- Creacion de la Tabla de curos
Create table CURSO(
    idcurso INT AUTO_INCREMENT PRIMARY KEY,
    Nombre  VARCHAR(100),
    Creditos INT
);

-- Creacion de la Tabla de Publicacion
Create table PUBLICACION(
    idpublicacion INT AUTO_INCREMENT PRIMARY KEY,
    Descripcion VARCHAR(500),
    Fecha DATE,
    FK_Catedratico INT,
    Fk_Usuario BIGINT,
    Fk_Curso INT,
    FOREIGN KEY (Fk_Usuario) REFERENCES USUARIO(Carnet),
    FOREIGN KEY (FK_Catedratico) REFERENCES CATEDRATICO(idcatedratico),
    FOREIGN KEY (Fk_Curso) REFERENCES CURSO(idcurso)
);
```

Figura 7. Base de datos (SQL)

Controller

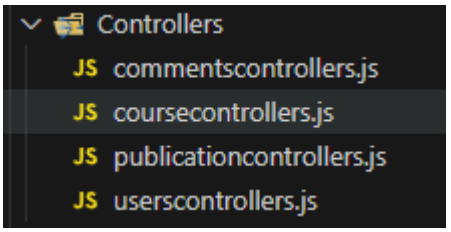

Figura 8. Archivos controllers

userscontrollers.js

Este archivo contiene controladores relacionados con las operaciones de *usuario*, como *registro*, *inicio de sesión*, *actualización de datos*, *listado de usuarios*, *obtención de datos de usuario* y *restablecimiento de contraseña*. Los controladores utilizan el módulo *mysql* para interactuar con la base de datos y responder a las solicitudes *HTTP*.

publicationcontrollers.js

Este archivo contiene controladores relacionados con las operaciones de *publicación*, como *agregar una nueva publicación*, *listar todas las publicaciones* y *listar las publicaciones* de un usuario específico. También se incluye una función para *guardar catedráticos* en la base de datos cuando se realiza una publicación. Los controladores utilizan el módulo *mysql* para interactuar con la base de datos.

coursecontrollers.js

Este archivo contiene controladores relacionados con las operaciones de *cursos* y *catedráticos*, como *listar todos los cursos* y *listar todos los catedráticos*. Los controladores utilizan el módulo *mysql* para interactuar con la base de datos.

commentscontrollers.js

Este archivo contiene controladores relacionados con las operaciones de *comentarios*, como *agregar un comentario a una publicación* y *listar comentarios de una publicación específica*. Los controladores utilizan el módulo *mysql* para interactuar con la base de datos.

```
const pool = require("../DataBase/Database");

const Login = async (req, res) => {
    const user = req.body;
    try {
        await pool.conexion.query(`SELECT * FROM USUARIO WHERE Carnet = ${user.carne} and Contrasena = '${user.pass}'`, async (err, result) => {
            if (result.length != 0) {
                res.status(200).json({
                    'success': true,
                    'message': result[0]
                });
            } else {
                res.status(400).json({
                    'success': false,
                    'message': "El usuario no esta registrado o la contraseña es incorrecta"
                });
            }
        });
    } catch (error) {
        res.status(200).json({ 'success': false, 'message': 'Existe un error inesperado ' + error })
    }
}
```

Figura 9. Función login - userscontrollers.js

Uso de la Aplicación

La aplicación proporciona una *API* para realizar operaciones de *registro*, *inicio de sesión*, *gestión de usuarios*, *gestión de publicaciones*, *gestión de cursos* y *gestión de comentarios*. Puede utilizar herramientas como *Postman* o aplicaciones *front-end* para interactuar con la *API*.

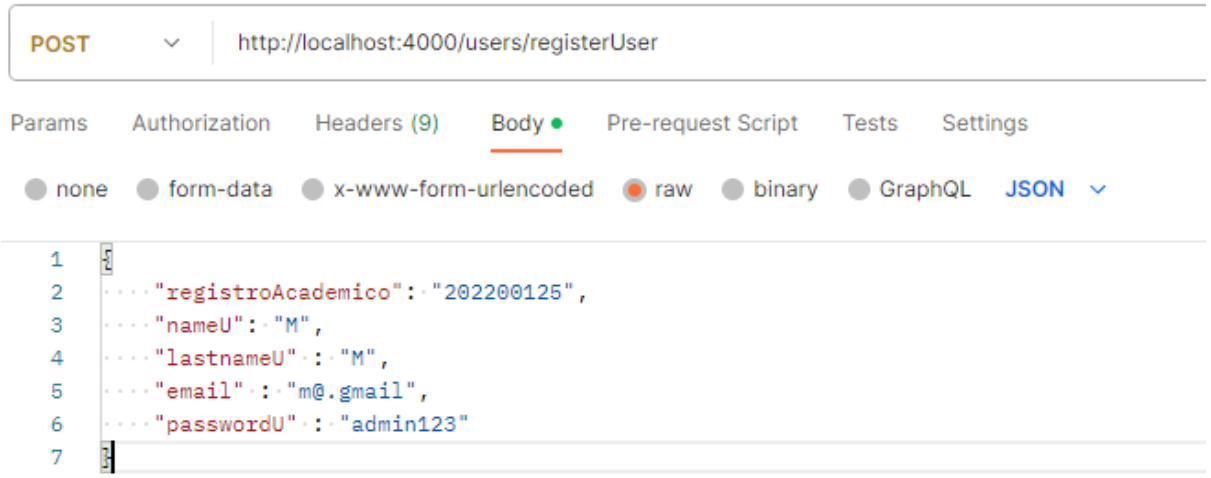

Figura 10. Petición POST - Postman

Conclusiones

1. *Estructura Organizada*: El manual técnico presenta una estructura organizada y detallada de todos los componentes de la aplicación, desde el *servidor* hasta las *rutas* y *controladores*, lo que facilita la comprensión y el mantenimiento del código.

2. *Conexión a la Base de Datos*: El manual explica cómo se establece la conexión a la base de datos *MySQL* y proporciona el archivo *SQL* necesario para *crear las tablas* y *procedimientos almacenados* requeridos por la aplicación.

3. *Gestión de Publicaciones*, *Comentarios*, *Usuarios*, *Cursos*: Se han desarrollado controladores y rutas para gestionar *publicaciones*, *comentarios*, usuarios, *cursos* lo que permite a los usuarios *crear*, *actualizar*, *eliminar* y *buscar* estos recursos de manera efectiva.