UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

PRACTICAS INICIALES

CATEDRÁTICO: ING. HERMAN IGOR VELIZ LINARES

MANUAL TÉCNICO

Carlos Sebastián del Cid Ramírez - 202300625

David Arturo Chanquin Velásquez - 202300611

David Norberto Fabro Guzman - 202307499

José Rolando Yaquian Paz - 202201185

Manuel Alejandro López Canel - 202302035

SECCIÓN: F-

GUATEMALA, 18 DE SEPTIEMBRE DEL 2,024

ÍNDICE

INTRODUCCIÓN

Este manual técnico tiene como propósito guiar el proceso de desarrollo, configuración y mantenimiento de una aplicación web orientada a la gestión de usuarios y publicaciones académicas. Provee instrucciones detalladas sobre el uso de los diferentes módulos, desde el registro e inicio de sesión de usuarios hasta la creación y gestión de publicaciones y comentarios.

OBJETIVOS

1. GENERAL

1.1. El objetivo general de este manual es describir las características técnicas de la aplicación web, facilitando la comprensión y el mantenimiento del código.

2. ESPECÍFICOS

2.1. Objetivo 1: Proveer una guía técnica sobre cómo implementar y mantener el sistema de autenticación de usuarios.

2.2. Objetivo 2: Describir la lógica del programa, incluyendo la estructura del código para manejar el flujo de publicación y filtrado de contenido académico.

ALCANCES DEL SISTEMA

El objetivo de este manual es proporcionar una visión técnica clara del sistema, enfocándose en la gestión de usuarios, publicaciones y comentarios. Incluye desde la creación de cuentas y autenticación de usuarios hasta la interacción con publicaciones y perfiles, permitiendo el desarrollo y la modificación eficiente del sistema.

ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**
  - Procesador: Intel i3 o superior.
  - Memoria RAM: 4GB o más.
  - Almacenamiento: 500MB de espacio libre.
  - Conexión a internet para el despliegue de la aplicación y la interacción con la base de datos.
- **REQUISITOS DE SOFTWARE**
  - Node.js versión 14 o superior.
  - MySQL para la gestión de la base de datos.
  - Editor de código como Visual Studio Code.
  - npm (gestor de paquetes de Node.js).

DESCRIPCIÓN DE LA SOLUCIÓN

- La solución propuesta se basa en una aplicación web desarrollada en Node.js y Express, con una base de datos MySQL para almacenar los datos de los usuarios y las publicaciones. El sistema fue diseñado para manejar la autenticación segura de los usuarios, la creación de publicaciones y comentarios, y la visualización de perfiles, siguiendo los requerimientos dados.

LÓGICA DEL PROGRAMA

❖ NOMBRE DE LA CLASE

La lógica del programa está estructurada en múltiples funciones dentro de un archivo central de backend (index.js), usando la arquitectura de Node.js y Express. A continuación, se detallan las librerías, variables globales, y funciones más relevantes.

➢ Librerías

1) express: Para la creación del servidor web y manejo de las rutas HTTP.
2) mysql: Para conectar y manejar la base de datos MySQL.
3) body-parser: Para interpretar las peticiones en formato JSON.
4) cors: Para habilitar el acceso cruzado de recursos en el servidor.
5) fs: Para operaciones básicas con archivos locales.

➢ Variables Globales de la clase

- *dataUsers: Contiene los usuarios registrados.*
- *dataMovies: Contiene las publicaciones de la base de datos. Estas variables globales permiten manejar la información de usuarios y publicaciones de manera persistente.*

➢ Función Main

La función app.listen(PORT) se utiliza para inicializar el servidor y hacer que escuche en el puerto especificado.

```
app.listen(PORT, () => {
  console.log(`Servidor escuchando en el puerto: ${PORT}`);
});
```

➢ **Procedimientos, métodos y Funciones utilizadas**

```
app.post("/register", (req, res) => {
  const { firstName, lastName, email, password, gender, birthDate }
= req.body;
    const userExists = dataUsers.find((user) => user.email ===
email);
  if (userExists) {
    return res.status(400).json({
      success: false,
      message: "Correo registrado anteriormente.",
    });
  }
  const newUser = { firstName, lastName, email, password, gender,
birthDate };
  dataUsers.push(newUser);
  updateDataUsersFile();
  res.status(201).json({
    success: true,
    message: "Usuario registrado exitosamente.",
  });
});
```

Esta función registra un nuevo usuario en el sistema verificando primero si el correo ya está en uso.

```
app.post("/login", (req, res) => {
  const data = req.body;
  const tempuser = dataUsers.find(
      (user) => user.email === data.email && user.password === data.password
  );
  if (!tempuser) {
    res.status(404).send({ success: false, user: null });
  } else {
    res.json({ success: true, user: tempuser });
  }
});
```

Verifica las credenciales del usuario y permite el acceso si son correctas.