Universidad de San Carlos de Guatemala
Guatemala
Facultad de Ingeniería en
Ciencias y Sistemas
Prácticas iniciales
Ing. Igor Veliz
Sección F-

Grupo No. 7

Manual Técnico - Sistema de evaluación USAC

Integrantes:

No.	Carnet	Nombre	No. Celular
1	202402587	Angel Raúl Herrera Chilel	56214136
2	202402473	Kimberly Samantha Gómez Chávez	58330770
3	202106435	María Hercilia Flores Alvarez	47694272
4	202200013	Marcos Aarón Toledo Alvarez	45150096
5	202403929	Emiliana Elizabeth Pú Lara	41948533

Tutores:

No.	Nombre	No. Celular
1	Julio Ruano	58925434
2	Gabriel Melgar	43314235

ÍNDICE

MANUAL TÉCNICO - SISTEMA DE EVALUACIÓN USAC

1. Introducción

Este documento describe la arquitectura, componentes y funcionalidades del servidor desarrollado para la aplicación web en versión de prueba de reseñas de catedráticos y cursos de la Facultad de Ingeniería de la USAC. El backend está construido con Node.js y ofrece una API REST que sirve como interfaz entre el frontend (cliente) y la base de datos MySQL.

2. Aplicación web

Es un programa que funciona dentro del navegador, pero que te permite interactuar y hacer tareas complejas.
Se ejecuta en un servidor y el usuario solo necesita un navegador para usarla.

- Permite interacción: no solo lees información, también puedes escribir, modificar, enviar datos y recibir resultados.
- Puede conectarse a bases de datos para guardar y procesar información.
- Se actualiza automáticamente en línea (el usuario no tiene que descargar actualizaciones).

3. Proyecto

El proyecto es una aplicación web en versión de prueba, "Sistema de Evaluación de Catedráticos y Cursos de Ingeniería USAC". Su objetivo es que los estudiantes puedan:

- Publicar opiniones sobre catedráticos y cursos.
- Comentar publicaciones de otros.
- Buscar y filtrar publicaciones.
- Gestionar su perfil y cursos aprobados.

4. Arquitectura del Backend

El backend sigue una arquitectura por capas MVC (Modelo-Vista-Controlador) adaptada:

- Routes: Manejan las solicitudes HTTP y envían respuestas.
- Services: Contienen la lógica de negocio y reglas de aplicación.
- Models: Definen la estructura de datos y interactúan con la base de datos.
- Middleware: Funciones intermedias para autenticación y validación.
- Config: Configuración de la base de datos y otros servicios.
- Utils: Utilidades generales y helpers.

5. Stack Tecnologico

Tecnología	Versión	Propósito
Node.js	18+	Entorno de ejecución JavaScript
Express.js	4.18.2+	Framework web para API REST
MySQL2	3.6.5+	Controlador para MySQL
Sequelize	6.35.1+	ORM para base de datos
bcryptjs	3.0.2+	Encriptación de contraseñas
CORS	2.8.5+	Habilitar peticiones cruzadas

6. Estructura de Carpetas del Backend

```
Backend/
├── src/
│   ├── config/
│   │   ├── database.js     # Configuración de Sequelize
│   │   └── seed.js         # Datos iniciales para la BD
│   ├── middleware/
│   │   └── auth.js         # Middleware de autenticación JWT
│   ├── models/             # Modelos de datos (tablas)
│   │   ├── usuario.js      # Modelo de usuarios
│   │   ├── rol.js          # Modelo de roles
│   │   └── ...             # Otros modelos
│   ├── routes/             # Rutas de la API
│   │   ├── usuarios.routes.js
│   │   ├── asignacion.routes.js
│   │   └── ...
│   ├── service/            # Lógica
│   │   ├── usuarioService.js
│   │   ├── asignacionService.js
│   │   ├── filtroService.js
│   │   └── ...
│   └── utils/
│       └── asyncWrapper.js  # Manejo de errores async
├── package.json
```

7. Funcionalidades del Servidor

7.1 Usuario Administrador

- Gestionar usuarios: Crear, modificar y eliminar usuarios mediante usuarioService.js
- Administrar tablas: Gestión completa de cursos, catedráticos y asignaciones
- Acceso a estadísticas generales: Mediante servicios específicos

7.2 Usuario Estudiante

- Publicar en el foro: mediante publicacionService.js y comentarioService.js.
- Modificar su perfil: mediante perfilService.js.
- Gestionar cursos aprobados: mediante cursoAprobadoService.js.

7.3 Usuario Catedrático

- Solo lectura en el foro: Acceso limitado a publicaciones
- Ver estadísticas personales: Número de alumnos y cursos impartidos
- Acceso restringido: Solo a endpoints específicos con autenticación

8. Endpoints Principales de la API

Método	Endpoint	Descripción	Acceso
POST	/api/auth/login	Inicio de sesión	Todos
POST	/api/auth/register	Registro de usuario	Público
GET	/api/publicaciones	Obtener publicaciones	Autenticados
POST	/api/publicaciones	Crear publicación	Estudiantes
GET	/api/catedraticos	Listar catedráticos	Todos
GET	/api/estadisticas	Ver estadísticas	Admin y Catedráticos
PUT	/api/perfil	Actualizar perfil	Propietario

9. Modelo Datos Principales (Modelo usuario)

```
3     import {Rol} from './rol.js';
4
5     export const Usuario = sequelize.define('Usuario', {
6         id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
7         carnet: { type: DataTypes.STRING(9), unique: true, allowNull: false },
8         nombre: { type: DataTypes.STRING, allowNull: false },
9         correo: { type: DataTypes.STRING, unique: true, allowNull: false },
10        password: { type: DataTypes.STRING, allowNull: false },
11        rolId: { type: DataTypes.INTEGER, allowNull: false }
12    }, { tableName: 'usuarios', timestamps: false });
13
14    Usuario.belongsTo(Rol, { foreignKey: 'rolId', as: 'rol' });
```

10. Servicios Principales

UsuarioService (service/usuarioService.js)

- registrar(dto): Crea nuevo usuario con validación de duplicados
- autenticar(correo, password): Verifica credenciales y retorna usuario
- restablecer(registro, correo, nuevaPass): Cambia contraseña con validación
- porId(id): Obtiene usuario por ID
- esEstudiante(id): Verifica si usuario es estudiante
- esCatedratico(id): Verifica si usuario es catedrático

FiltroService (service/filtroService.js)

- usuarios(query):Busca usuarios con filtros personalizados

AsignacionService (service/asignacionService.js)

- estadisticasCatedratico(id): Obtiene estadísticas para catedrático específico

11. Middleware de Autenticación y Autorización

```
import {UsuarioService} from '../services/usuarioService.js';

export function auth(rolPermitido) {
  return async (req, res, next) => {
    const id = req.headers['usuario-id'];
    if (!id) return res.status(401).json({ error: 'Falta usuario-id header' });
    const user = await UsuarioService.porId(id);
    if (!user) return res.status(404).json({ error: 'Usuario no existe' });
    if (rolPermitido && user.rol !== rolPermitido) return res.status(403).json({ error: 'Sin permisos' });
    req.usuario = user;
    next();
  };
}
```

12. Flujos de Autenticación

Registro de Usuario

1. Cliente envía POST a `/api/usuarios/registro` con datos
2. Servicio valida que no exista duplicado (carnet o correo)
3. Encripta password con bcryptjs
4. Crea usuario en base de datos
5. Retorna usuario creado

Inicio de Sesión

1. Cliente envía POST a `/api/usuarios/login` con credenciales
2. Servicio busca usuario por correo
3. Compara password con bcrypt.compareSync()
4. Retorna usuario autenticado

Autorización por Roles

1. Cliente incluye header `usuario-id` en peticiones
2. Middleware auth verifica existencia y permisos
3. Si tiene permisos, agrega usuario a `req.usuari`

13. Consideraciones de Seguridad

- Contraseñas encriptadas con bcryptjs (salt rounds: 10)
- Validación de duplicados en registro
- Autorización por roles en endpoints sensibles
- Validación de parámetros en servicios
- Manejo centralizado de errores

14. Fuentes de documentación

Para el desarrollo y mantenimiento de este proyecto, se recomienda consultar las siguientes fuentes oficiales de documentación:

1. Node.js
    - Documentación oficial: https://nodejs.org/en/docs/
    - Guías y referencias de API: https://nodejs.org/api/
2. Express.js
    - Documentación oficial: https://expressjs.com/
    - Referencia de API: https://expressjs.com/en/4x/api.html
3. Sequelize ORM
    - Documentación oficial: https://sequelize.org/
4. MySQL
    - Documentación oficial: https://dev.mysql.com/doc/
    - Referencia de lenguaje: https://dev.mysql.com/doc/refman/8.0/en/
5. bcryptjs
    - Documentación: https://www.npmjs.com/package/bcryptjs
6. CORS
    - Documentación: https://www.npmjs.com/package/cors
7. dotenv
    - Documentación: https://www.npmjs.com/package/dotenv
    - Uso de variables de entorno: https://github.com/motdotla/dotenv

7. Foto Grupal