



# UNIVERSIDAD AUTÓNOMA DE ZACATECAS “FRANCISCO GARCÍA SALINAS”

UNIDAD ACADÉMICA DE INGENIERÍA ELÉCTRICA  
PROGRAMA ACADÉMICO DE INGENIERÍA DE SOFTWARE

---

## Plataforma de Eventos CECODIC

---

### Manual técnico

Realizado por: Gerardo Maldonado Félix

Fecha: Lunes, 26 de agosto de 2024



**LABSOL**  
**NETWORK**  
Innovation Techs & Open Labs

## Contenido

<b>1. Introducción</b>	2
<b>2. Tecnologías</b>	2
<b>3. Estructura del Proyecto</b>	2
<b>4. Instalación</b>	3
Requisitos Previos:	3
Instrucciones de Instalación:	3
<b>5. Configuración de la Base de Datos</b>	4
<b>6. Variables de Entorno</b>	4
<b>7. Descripción de Componentes – Frontend</b>	5
<b>8. Rutas – Frontend</b>	6
Descripción de Vistas:	6
<b>9. Endpoints de la API – Backend</b>	6
Endpoints GET:	6
Endpoints POST:	7
Endpoints PUT:	7
Endpoints DELETE:	7
<b>10. Modelos de Datos</b>	8
<b>11. Despliegue</b>	9
<b>12. Comandos Útiles</b>	9
Mostrar la vista preliminar en local:	9
Compilar para Producción:	9
Iniciar el Servidor:	9

# 1. Introducción

Este proyecto es una plataforma para la gestión de eventos académicos y científicos, desarrollada con una arquitectura MVC en capas. El frontend utiliza Vue 3 mientras que el backend está construido con Node.js/Express y una base de datos MySQL.

## 2. Tecnologías

Frontend:

- Vue 3

Backend:

- Node.js
- Express
- MySQL

## 3. Estructura del Proyecto

**/instalador:** Contiene los archivos necesarios para instalar el programa.

**/src**

- **/app:** Contiene los componentes y vistas del frontend.
  - **/assets:** Contiene los recursos estáticos como imágenes y estilos generales.
  - **/styles:** Contiene los estilos generales.
  - **/components:** Contiene los componentes por separado.
    - **/buttons:** Contiene los componentes botones.
    - **/queries:** Contiene los componentes para realizar las consultas a la API.
    - **/sesiones:** Contiene los componentes relacionados con el usuario administrador.
  - **/router:** Contiene las rutas de las vistas.
  - **/views:** Contiene las vistas de las páginas que se integran con los componentes.
- **/controllers:** Contiene la lógica de negocio del backend.
- **/database:** Define los modelos de la base de datos.
- **/public:** Contiene el frontend después de compilar, es la carpeta para el despliegue del frontend.
- **/routes:** Define las rutas de la API.
- **/uploads:** Destino para los archivos subidos (iconos y PDFs).

## 4. Instalación

Requisitos Previos:

- npm
- Node.js

Instrucciones de Instalación:

1. Clonar el repositorio:

```
git clone https://labsol.cozcyt.gob.mx/git/GeraMaldonado/cecodic.git
```

2. Dirigirse a la carpeta del proyecto:

```
cd cecodic
```

3. Instalar las dependencias:

```
npm install
```

4. Dar permisos de ejecución al instalador:

```
cd instalador  
chmod +x instalador.sh
```

5. Ejecutar el instalador:

```
./instalador.sh
```

6. Seguir los pasos necesarios del instalador

## 5. Configuración de la Base de Datos

Al ejecutar el archivo instalador/instalador.sh se instalarán los programas necesarios como MySQL, también facilita la configuración de la base de datos al solo tener que crear un usuario para la base de datos y el nombre de la base de datos. La instrucción que se ejecuta es la siguiente:

```
CREATE DATABASE tu_base_de_datos;  
CREATE USER 'tu_usuario'@'localhost' IDENTIFIED BY  
'tu_contraseña';  
GRANT ALL PRIVILEGES ON tu_base_de_datos.* TO  
'tu_usuario'@'localhost';  
FLUSH PRIVILEGES;
```

Claramente se puede realizar de manera manual la creación de la base de datos y el usuario.

## 6. Variables de Entorno

El instalador al finalizar creará un .env con las credenciales necesarias para ingresar a la base de datos. Esto es opcional, también se puede crear manualmente o crear las variables en el sistema.

El .env contiene los siguientes datos y nombres de variables:

```
HOST=localhost  
DATABASE=tu_base_de_datos  
DB_USER=tu_usuario  
PASSWORD=tu_contraseña
```

## 7. Descripción de Componentes – Frontend

Ruta `/src/app/components/`:

- **About.vue**: Contiene información sobre las instalaciones de la institución.
- **Card.vue**: Contiene información breve de un evento para mostrarlo como tarjeta de presentación.
- **Footer.vue**: Contiene información de contacto de la institución, números de teléfono, ubicación y logos.
- **Header.vue**: Contiene la barra de navegación y el logo de la institución.
- **Paginacion.vue**: Gestiona y organiza los eventos para mostrarlos por páginas (por defecto 9 eventos por página).
- **TablaEvento.vue**: Contiene la información completa de un evento.
- **TablaEventoEditar.vue**: Contiene la información precargada de un evento que se desee modificar.
- **TablaEventoNuevo.vue**: Contiene los campos necesarios para crear un nuevo evento.
- **TablaEventos.vue**: Usando la paginación organiza los eventos en tablas, también con filtros temporales (con rangos de semana y mes) y botones que permiten navegar hacia adelante o hacia atrás.
- **eventosUtils.js**: Contiene los rangos temporales de semana y mes para el filtrado de eventos.
- **util.js**: Contiene las fechas que se usan para generar los rangos y las consultas diarias, formatos de fecha corta y larga, acortadores de texto para limitar la información mostrada, y la ruta del backend a la que apuntan las consultas.

Ruta `/src/app/components/buttons/`:

- **BotonConfirmar.vue**: Componente botón que usa la query de crear evento o actualizar evento, dependiendo en qué componente se use.
- **BotonEditar.vue**: Componente botón que dirige a la ruta editar-evento.
- **BotonEliminar.vue**: Componente botón que usa la query eliminar.
- **BotonNuevoEvento.vue**: Componente botón que dirige a la ruta crear-evento.

Ruta `/src/app/components/queries/`:

- **queries.js**: Componente que contiene las consultas con métodos GET, POST, PUT y DELETE.

Ruta `/src/app/components/sesiones/`:

- **CerrarSesion.vue**: Componente botón para cerrar la sesión del administrador.
- **IniciarSesion.vue**: Componente con los campos para solicitar el inicio de sesión.
- **VistaAdministrador.vue**: Componente que contiene la palabra Administrador y el botón para crear un evento nuevo.

## 8. Rutas – Frontend

Descripción de Vistas:

- `/`: Página de inicio, aquí se muestran las tarjetas con los eventos asignados para ese día/semana.
- `/eventos`: Página donde se muestran los eventos en tabla, se muestran por semana/mes y se pueden visualizar los eventos pasados y futuros.
- `/evento/{id}`: Página que muestra los detalles del evento.
- `/createevento`: Página que permite crear un evento nuevo.
- `/editevento/{id}`: Página que permite editar un evento ya existente.
- `/about`: Página que muestra detalles de la institución (materiales y espacios).

## 9. Endpoints de la API – Backend

Endpoints GET:

- **api/eventos**: Obtiene todos los eventos.
- **api/eventos/fecha/{fecha}**: Obtiene los eventos que coincidan con la fecha ingresada.
- **api/eventos/fecha/{fecha1}/{fecha2}**: Obtiene los eventos que se encuentran dentro de las dos fechas.
- **api/eventos/institucion/{institucion}**: Obtiene los eventos que coincidan con la institución.
- **api/eventos/instituciones**: Obtiene todas las instituciones de los eventos.

- **api/eventos/evento/{id}**: Se obtiene toda la información del evento perteneciente al id.
- **api/eventos/ultimo**: Obtiene el último evento registrado.

#### Endpoints POST:

- **api/eventos/**: Permite crear un evento.
- **api/eventos/users**: Permite crear un usuario.

#### Endpoints PUT:

- **api/eventos/upload/img/{id}**: Guarda una imagen en un evento y guarda la ruta de la imagen.
- **api/eventos/upload/pdf/{id}**: Guarda un documento PDF en un evento y guarda la ruta del documento.
- **api/eventos/evento/{id}**: Permite actualizar la información de un evento.

#### Endpoints DELETE:

- **api/eventos/{id}**: Elimina el evento que coincida con el id ingresado.



## 10. Modelos de Datos

La base de datos consta de dos tablas sin relación. La primera contiene los datos del evento, como nombre, fecha, ubicación, detalles, etc., y se llena mediante el frontend. La segunda tabla es para los administradores, conteniendo nombre, correo y contraseña, y se llena mediante el instalador o directamente desde la base de datos. En el frontend, estos datos solo se consultan y comparan para permitir el acceso como administrador.

eventos		
PK, NN, AI	ideventos	BIGINT
NN	titulo	TEXT
NN	institucion	DATE
NN	fecha	TIME
NULL	hora	TEXT
NULL	lugar	VARCHAR
NULL	resumen	TEXT
NULL	detalles	TEXT
NULL	img	TEXT
NULL	pdf	TEXT

usuarios		
PK, NN, AI	idusuarios	INT
NN	usuario	VARCHAR
NN	correo	VARCHAR
NN	contrasena	TEXT

## 11. Despliegue

Este proyecto es integral, el backend ejecuta el frontend. Si se desea ejecutar independientemente uno del otro, para desplegar el frontend es necesario desplegar el index.html que está dentro de **/src/public**. Para desplegar el backend es necesario ejecutar el index.js que está dentro de **/src**.

Al desplegar el backend también se ejecuta el frontend.

## 12. Comandos Útiles

Mostrar la vista preliminar en local:

```
npm run preview
```

Compilar para Producción:

```
npm run build
```

Iniciar el Servidor:

```
npm start
```