TECNOLÓGICO NACIONAL DE MÉXICO

I.T.E. ENSENADA B.C.

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE ENSENADA.

Manual Tecnico AsesoraTec

Ingeniería en Sistemas Computacionales 6SE

García Romero Ángel al22760920@ite.edu.mx
Pardo Rubalcaba Andres al22760571@ite.edu.mx
Martínez Monge Saúl Guillermo al22760566@ite.edu.mx
González Navarro Oscar Eduardo al22760560@ite.edu.mx

Ensenada B.C a 20 de Mayo de 2025

TECNOLÓGICO NACIONAL DE MÉXICO

I.T.E. ENSENADA

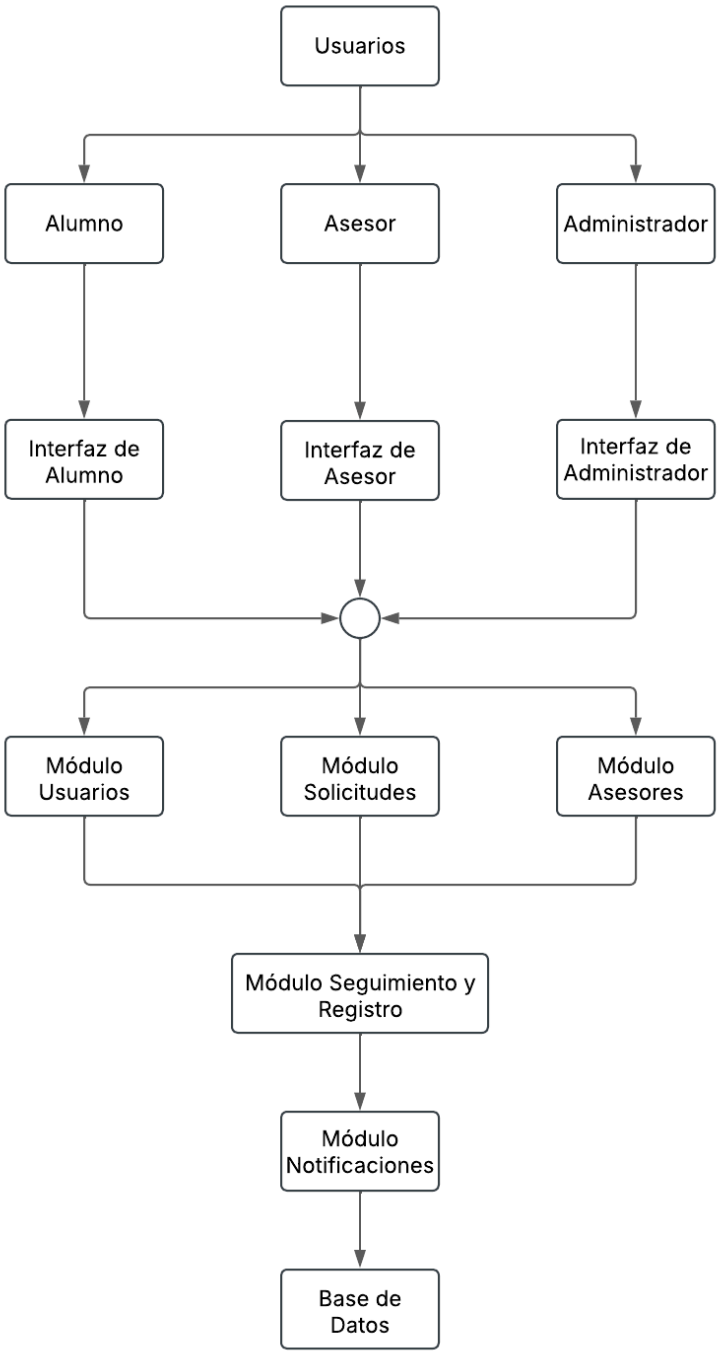
Índice

Arquitectura del sistema

Nuestro sistema consta de un total de cinco módulos, que podemos repartir entre las cuatro capas del modelo de arquitectura en capas, de este modo garantizamos la prueba de cada módulo para la realización de las correcciones correspondientes mientras construímos la aplicación de forma que si es necesario cambiar alguna de las tecnologías a utilizar, pueda cambiarse en la capa de infraestructura, sin afectar el núcleo de la aplicación.

Diagramas de la arquitectura

Diagrama de bloques



TECNOLÓGICO NACIONAL DE MÉXICO

I.T.E. ENSENADA

Diagrama de componentes

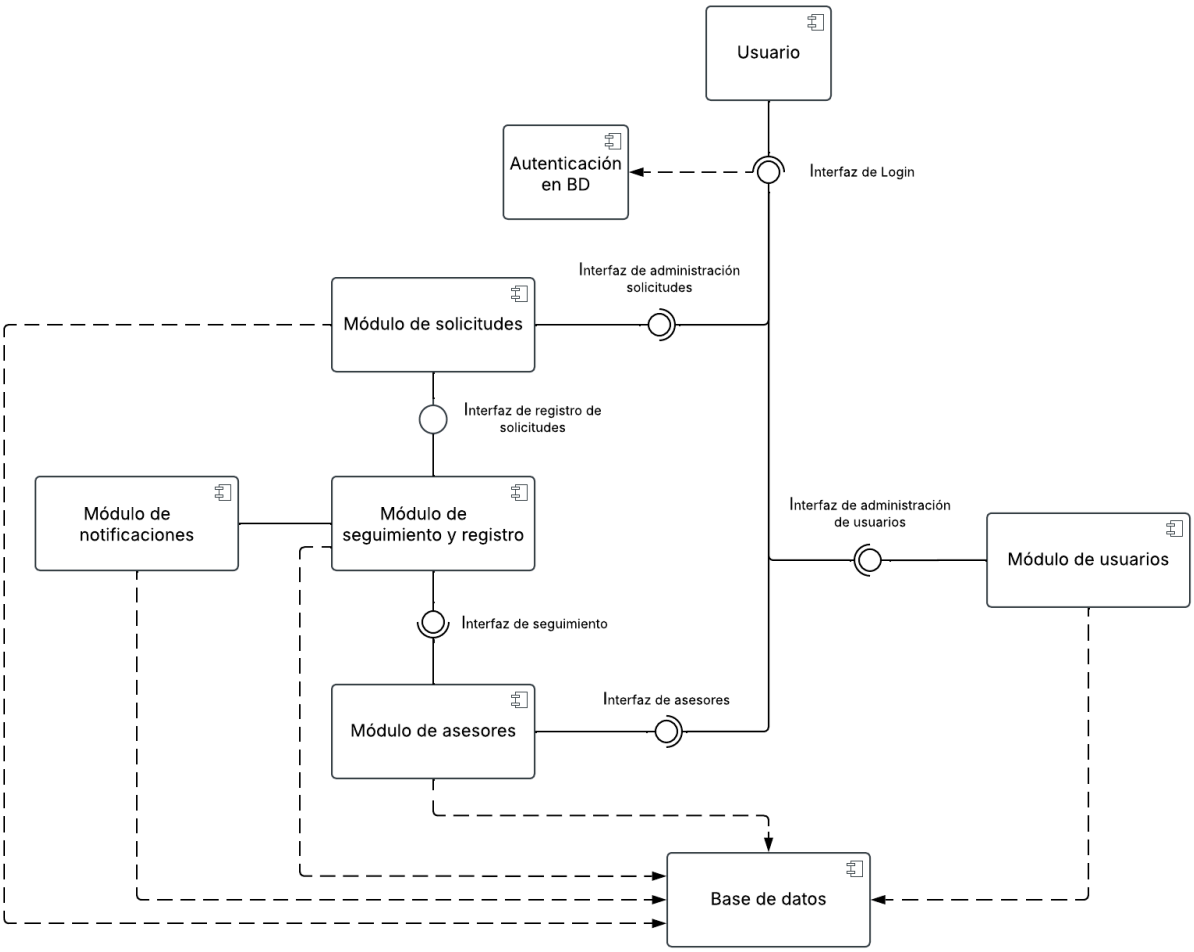
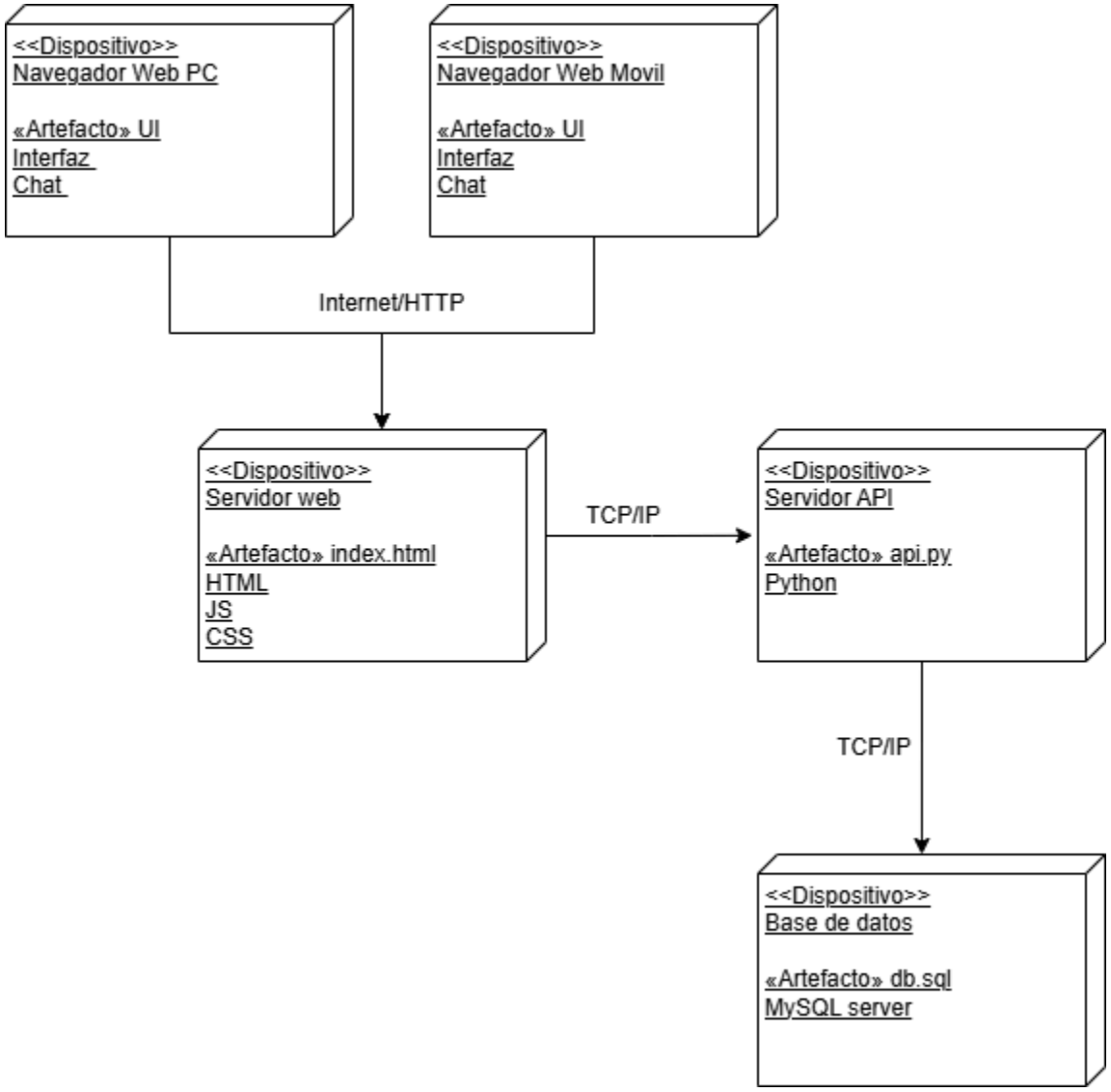


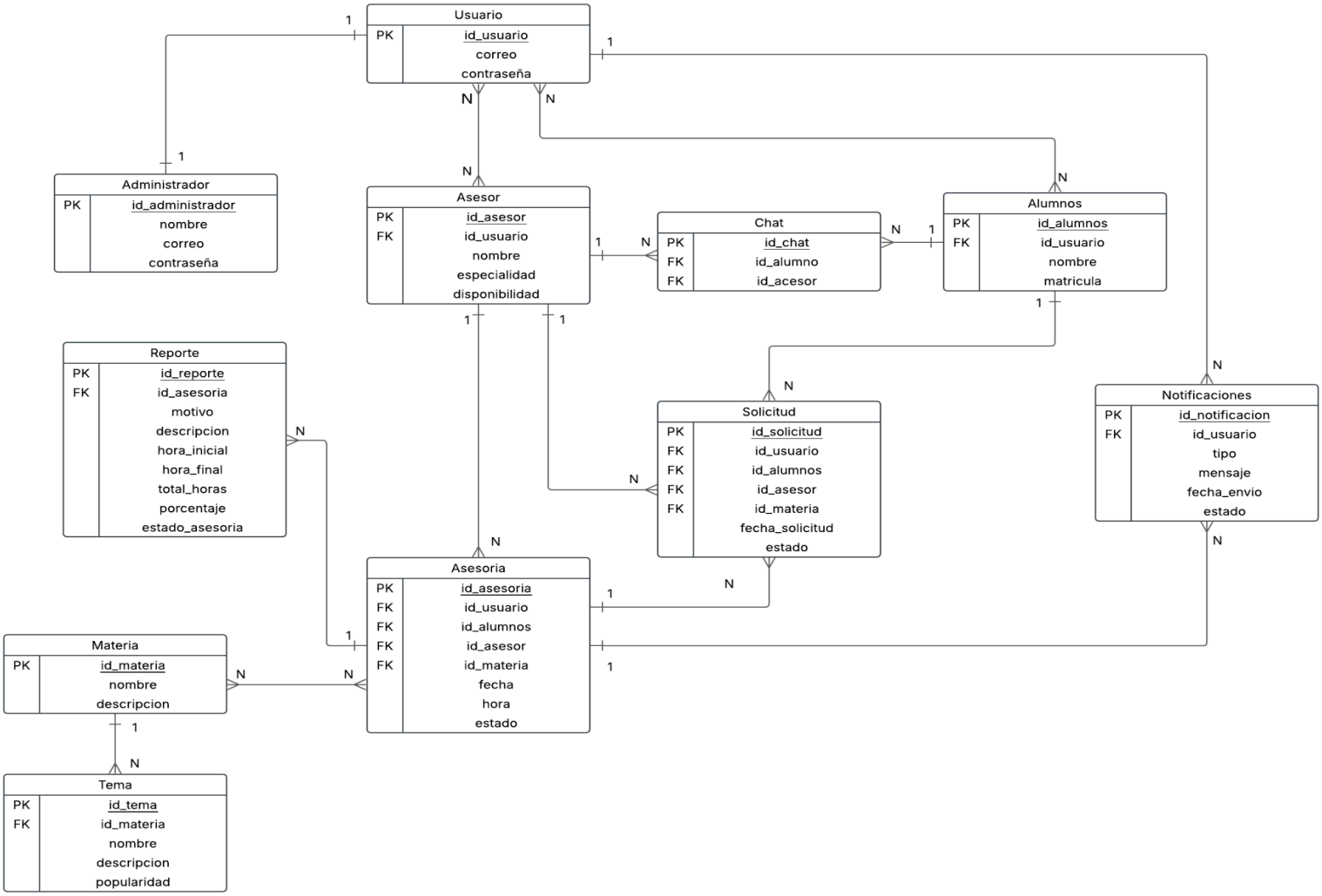
Diagrama de despliegue



TECNOLÓGICO NACIONAL DE MÉXICO

I.T.E. ENSENADA

Diagrama de Entidad-Relación



Estructura de carpetas y módulos

```
gestion_asesorias
├──frontend     #HTML,CSS,JS
│   ├──controllers
│   │   └──CONTROLLER ARCHIVES
│   ├──css
│   │   └──CSS ARCHIVES
│   ├──js
│   │   └──JS API REQUEST ARCHIVES
│   ├──images
│   │   └──IMAGES
│   └──pages
│       └──HTML PAGES
└──backend      #Express,Node.js,MySQL,CORS, Path, JWT, Nodemon
    ├──node_modules
    ├──routes
    │   └──routes.js #API Routes and Methods
    ├──db.js #Database info
    ├──generate_pass.js
    ├──mail.js
    ├──package-lock.json
    ├──package.json
    └──server.js #Server Configuration
```

Librerías y dependencias

Backend

Las dependencias utilizadas en este proyecto en backend son las siguientes:

- Node.js
- Express
- CORS
- body-parser
- mysql2
- jwt-decode
- jsonwebtoken
- nodemon
- dotenv
- crypto
- cookie parser
- multer

Frontend

En el frontend no fueron requeridas dependencias

Instalación de dependencias

Las dependencias necesarias se instalarán automáticamente al hacer en la carpeta backend

backend

- cd backend
- npm install

Instalación de Node.js y NPM

Paso 1: Descargar instalador de Node.js

Página de descarga https://nodejs.org/en/download/

Descargar Node.js
https://phoenixnap.com/kb/wp-content/uploads/2023/12/nodejs-windows-installer-download-page.jpg

Paso 2: Instalar Node.js y NPM desde el instalador

Instalar Node.js y NPM
https://phoenixnap.com/kb/wp-content/uploads/2023/12/nodejs-setup-wizard-welcome-screen.jpg

Paso 3: Verificar la instalación en consola
node -v

```
PS C:\Users\cabal\gestion_asesorias> node -v
v21.6.1
```

npm -v

```
PS C:\Users\cabal\gestion_asesorias> npm
10.2.4
```

Clonado del repositorio

En la carpeta en la que se desea clonar el repositorio

- git init

```
PS C:\Users\cabal\gestion_asesorias> git init
Reinitialized existing Git repository in C:/Users/cabal/gestion_asesorias/.git/
```

- git clone https://github.com/GonzalezNavarroOscar/gestion_asesorias.g

```
PS C:\Users\cabal\gestion_asesorias> git clone https://github.com/GonzalezNavarroOscar/gestion_asesorias.git
Cloning into 'gestion_asesorias'...
remote: Enumerating objects: 3667, done.
remote: Counting objects: 100% (193/193), done.
remote: Compressing objects: 100% (114/114), done.
remote: Total 3667 (delta 100), reused 146 (delta 68), pack-reused 3474 (from 2)Receiving objects:  99% (3631/3667), 4.45 MiB | 456.0
0 KiB/s
Receiving objects: 100% (3667/3667), 4.50 MiB | 443.00 KiB/s, done.
Resolving deltas: 100% (1212/1212), done.
PS C:\Users\cabal\gestion_asesorias>
```

Correr Backend

Proyecto cuenta con un archivo encargado de lanzar el servidor de forma local, que se conecta a la bd en RDS. Para ello, en consola realizar lo siguiente en la carpeta del proyecto:

cd backend

```
PS C:\Users\cabal\gestion_asesorias> cd backend
PS C:\Users\cabal\gestion_asesorias\backend>
```

npm start

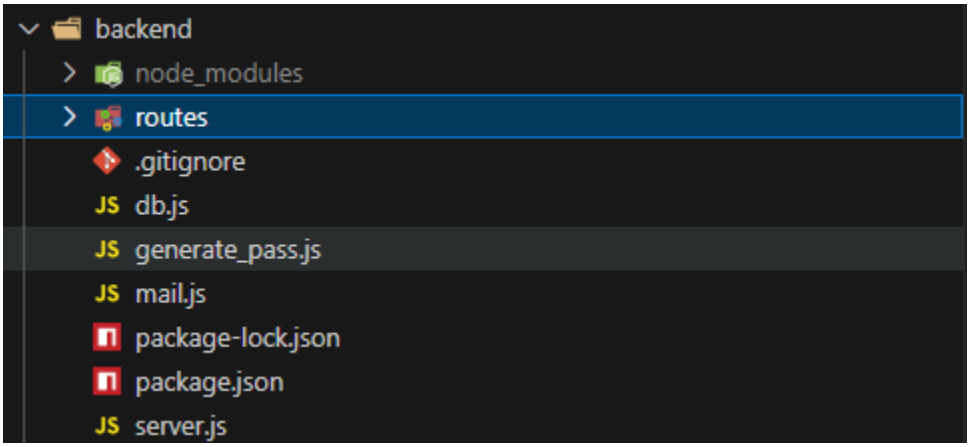
```
PS C:\Users\cabal\gestion_asesorias\backend> npm start

> backend@1.0.0 start
> node server.js


Servidor corriendo en http://localhost:3000
```

Documentación de la API

Rutas

Para poder acceder a las rutas primero ocupamos localizar la carpeta de backend en nuestro proyecto

Dentro de la carpeta de backend nos encontraremos diferentes archivos y carpetas pero en este caso queremos localizar la carpeta de routes



Dentro solo contiene un archivo llamado routes.js que es el que contiene todas las rutas del proyecto

Dentro podemos encontrar cada una de las rutas utilizadas pero para poder tener una mejor gestión están separadas por etiquetas

Ejemplo

```
/*
*
* [###### Funciones de apoyo ######]
*
*/

// Helper para ejecutar queries con promesas
function queryAsync(sql, params) {
    return new Promise((resolve, reject) => {
        db.query(sql, params, (err, results) => {
            if (err) reject(err);
            else resolve(results);
        });
    });
}

// Configuración de Multer para manejar la carga de imágenes
const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null, path.join(__dirname, '../../frontend/images'));
    },
    filename: (req, file, cb) => {
        const fileExtension = path.extname(file.originalname);
        const filename = Date.now() + fileExtension;
        cb(null, filename);
    }
});

const upload = multer({ storage });
```

Cada ruta tiene una pequeña descripción en la parte superior para comprender mejor su función

Ruta	Método HTTP	Autenticación	Parámetros	Descripción	Ejemplo de Llamada
/login	POST	Ninguna	{ "correo": "string", "contraseña": "string" } (Requiere Content-Type: application/json.)	Permite a un usuario autenticarse en el sistema	fetch('/login', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ correo: 'usuario@example.com', contraseña: 'password123' }) });
/registro	POST	Ninguna	`{ "nombre": "string", "correo": "string", "contraseña": "string", "rol": "alumno" \	Permite el registro de un nuevo usuario en la base de datos, asignándole un rol inicial de "alumno" o "asesor".	"asesor", "matricula?": "string" }\<br\> (matriculaes requerida sirol` es "alumno")
/perfil/:id_usuario	GET	JWT (usuario autenticado)	Parámetro de ruta: id_usuario (numérico)	Obtiene la información detallada del perfil de un usuario específico	fetch('/perfil/1', { method: 'GET', headers: { 'Authorization': 'Bearer <YOUR_JWT_TOKEN>' } });
/materias	GET	Ninguna	Ninguna	Recupera una lista de todas las materias disponibles en el sistema, incluyendo su popularidad y detalles adicionales.	fetch('/materias');
/materias_nombres	GET	Ninguna	Ninguna	Recupera únicamente los IDs y nombres de todas las materias	fetch('/materias_nombres');
/agregar-materias	POST	Administrador	multipart/form-data con campos: nombre: string (nombre de la materia) descripcion: string (descripción de la	Permite agregar una nueva materia al sistema, incluyendo la carga de un	const formData = new FormData(); formData.append('nombre', 'Química'); formData.append('descripcion', 'Ciencia de la composición de la

			materia) imagen: file (archivo de imagen)	archivo de imagen	materia.'); formData.append('imagen', fileInput.files[0]); fetch('/agregar-materias', { method: 'POST', body: formData });
/temas	GET	Ninguna	Parámetro de query: materia (string, nombre de la materia)	Obtiene los temas aprobados por el administrador para una materia específica	fetch('/temas?materia=Matemáticas');
/temas_admin	POST	Administrador	{ "id_materia": number }	Obtiene todos los temas (aprobados y no aprobados por el administrador) para una materia específica	fetch('/temas_admin', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_materia: 1 }) });
/temas_actualizar	POST	Administrador	{ "id_tema_old": number, "id_tema_new": number }	Actualiza el id_tema en las tablas Solicitud y Asesoria, reasignando registros de un tema antiguo a uno nuevo	fetch('/temas_actualizar', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_tema_old: 5, id_tema_new: 1 }) });
/agregar-temas	POST	Administrador	{ "id_materia": number, "nombre": "string", "descripcion": "string", "agregado_admin": boolean }	Añade un nuevo tema a una materia específica. Se puede indicar si el tema es agregado directamente por un administrador	fetch('/agregar-temas', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_materia: 2, nombre: 'Electromagnetismo', descripcion: 'Ciencia de la composición de la materia.', agregado_admin: true }) });
/editar-tema	POST	Administrador	{ "id_tema": number, "nombre": "string", "descripcion": "string" }	Permite la modificación del nombre y la descripción	fetch('/editar-tema', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({

				de un tema existente	id_tema: 101, nombre: 'Algebra Lineal Avanzada', descripcion: 'Conceptos avanzados de espacios vectoriales.' }) });
/borrar-tema	POST	Administrador	{ "id_tema": number }	Elimina un tema específico de la base de datos utilizando su ID.	fetch('/borrar-tema', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_tema: 102 }) });
/solicitud	POST	Alumno	{ "id_usuario": number, "id_alumno": number, "id_materia": number, "id_tema": number, "fecha_solicitud": "YYYY-MM-DD", "hora": "HH:MM:SS", "modalidad": "string", "observaciones": "string", "estado": "string" }	Crea una nueva solicitud de asesoría por parte de un alumno para un tema predefinido	fetch('/solicitud', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_usuario: 1, id_alumno: 1, id_materia: 1, id_tema: 101, fecha_solicitud: '2025-06-30', hora: '09:00:00', modalidad: 'Presencial', observaciones: 'Repaso para examen parcial.', estado: 'Pendiente' }) });
/solicitud_personalizada	POST	Administrador	{ "id_usuario": number, "id_alumno": number, "id_materia": number, "nombre_tema": "string", "fecha_solicitud": "YYYY-MM-DD", "hora": "HH:MM:SS", "modalidad": "string", "observaciones": "string", "estado	Crea una nueva solicitud de asesoría permitiendo al alumno proponer un tema personalizado. Si el tema no existe, se creará uno nuevo y se asociará a la solicitud	fetch('/solicitud_personalizada', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_usuario: 1, id_alumno: 1, id_materia: 1, nombre_tema: 'Introducción a la Topología', fecha_solicitud: '2025-07-05', hora: '15:30:00', modalidad: 'Virtual', observaciones: 'Conceptos básicos y ejemplos.', estado: 'Pendiente' }) });
/solicitudes-pendientes	GET	Asesor/Administrador	Ninguno	Recupera una lista de todas las solicitudes de asesoría que se encuentran en estado 'Pendiente'.	fetch('/solicitudes-pendientes');
/solicitude	GET	JWT (asesor)	Parámetro de ruta:	Obtiene las	fetch('/solicitudes-pendientes/2',

s-pendientes/:idUsuario			idUsuario (numérico)	solicitudes pendientes filtradas por las especialidades de un asesor específico.	{ method: 'GET', he... 'Authorization': 'Bearer <YOUR_JWT_TOKEN>' } });
/solicitudes/:id_solicitud	GET	usuario autenticado	Parámetro de ruta: id_solicitud (numérico)	Obtiene los detalles completos de una solicitud específica por su ID.	fetch('/solicitudes/50');
/solicitudes_alumno/:id_usuario	GET	Alumno	Parámetro de ruta: id_usuario (numérico)	Obtiene todas las solicitudes de asesoría realizadas por un alumno específico.	fetch('/solicitudes_alumno/1');
/asesoria	POST	Asesor	{ "id_solicitud": number, "id_usuario": number, "id_alumno": number, "id_asesor": number, "id_materia": number, "id_tema": number, "nombre_tema": "string", "fecha_solicitud": "YYYY-MM-DD", "hora": "HH:MM:SS", "estado": "string", "aula": "string", "modalidad": "string" }	Registra una nueva asesoría, actualiza el estado de la solicitud asociada a 'Aceptada', crea un chat vinculado a la asesoría	fetch('/asesoria', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_solicitud: 50, id_usuario: 1, id_alumno: 1, id_asesor: 2, id_materia: 1, id_tema: 101, nombre_tema: 'Álgebra Lineal', fecha_solicitud: '2025-06-30', hora: '09:00:00', estado: 'Aceptada', aula: 'C-201', modalidad: 'Presencial' }) });
/modificar-asesoria/:id_asesoria	PUT	Administrador	{ "id_alumno": number, "id_asesor": number, "id_materia": number, "id_tema": number, "nombre_tema": "string", "fecha": "YYYY-MM-DD", "hora": "HH:MM:SS", "aula": "string", "modalidad": "string" }	Permite la modificación de los detalles de una asesoría existente.	fetch('/modificar-asesoria/15', { method: 'PUT', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ id_alumno: 1, id_asesor: 2, id_materia: 1, id_tema: 101, nombre_tema: 'Álgebra Lineal Avanzado', fecha: '2025-07-01', hora: '10:00:00', aula: 'Virtual', modalidad: 'Virtual' }) });

/asesorias-proceso	GET	Asesor/Administrador	Ninguno	Recupera una lista de todas las asesorías cuyo estado actual es 'En proceso'.	fetch('/asesorias-proceso');
/detalles-asesoria/:id_asesoria	GET	usuario autenticado	Parámetro de ruta: id_asesoria (numérico)	Obtiene los detalles completos de una asesoría específica por su ID.	fetch('/detalles-asesoria/15');
/eliminar-asesorias/:id_asesoria	DELETE	Administrador	Parámetro de ruta: id_asesoria (numérico)	Elimina una asesoría específica de la base de datos por su ID	fetch('/eliminar-asesorias/15', { method: 'DELETE' });

Creación de base de datos

Creación de BD

```
CREATE DATABASE gestion_asesorias;

Use gestion_asesorias;
```

Tablas

Tabla Usuario

```
CREATE TABLE Usuario (
    id_usuario INT PRIMARY KEY AUTO_INCREMENT,
    correo VARCHAR(100) NOT NULL,
    rol ENUM(
            'alumno',
            'asesor',
            'administrador'
    ) NOT NULL,
    contraseña VARCHAR(255) NOT NULL
);
```

Tabla Administrador

```
CREATE TABLE Administrador (
    id_administrador INT PRIMARY KEY AUTO_INCREMENT,
    id_usuario INT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario) ON DELETE CASCADE
);
```

Tabla Asesor

```
CREATE TABLE Asesor (
    id_asesor INT(11) NOT NULL AUTO_INCREMENT,
    id_usuario INT(20) NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    especialidad VARCHAR(100) DEFAULT 'No definido',
    horarioInicio TIME DEFAULT '00:00:00',
    horarioFin TIME DEFAULT '00:00:00',
    diaInicio VARCHAR(30) DEFAULT 'No definido',
    diaFin VARCHAR(30) DEFAULT 'No definido',
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario) ON DELETE CASCADE
```

```
);
```

Tabla Horario

```
CREATE TABLE Horario(
    id_horario INT(15) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    horario_inicio time DEFAULT '00:00:00',
    horario_fin time DEFAULT '00:00:00',
    dia_inicio varchar(30) DEFAULT 'No definido',
    dia_fin varchar(30) DEFAULT 'No definido',
    id_asesor INT(15),
    FOREIGN KEY (id_asesor) REFERENCES Asesor(id_asesor) ON DELETE CASCADE
    );
```

Tabla Alumno

```
CREATE TABLE Alumno (
    id_alumno INT PRIMARY KEY AUTO_INCREMENT,
    id_usuario INT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    matricula VARCHAR(50) NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario) ON DELETE CASCADE
);
```

Tabla Chat

```
CREATE TABLE Chat (
    id_chat INT PRIMARY KEY AUTO_INCREMENT,
    id_alumno INT NOT NULL,
    id_asesor INT NOT NULL,
    id_asesoria INT NOT NULL,
    FOREIGN KEY (id_alumno) REFERENCES Alumno(id_alumno) ON DELETE CASCADE,
    FOREIGN KEY (id_asesor) REFERENCES Asesor(id_asesor) ON DELETE CASCADE,
    FOREIGN KEY (id_asesoria) REFERENCES Asesoria(id_asesoria) ON DELETE CASCADE
);
```

Tabla Mensaje

```
CREATE TABLE Mensaje(
   id_mensaje INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
   id_chat INT NOT NULL,
   contenido TEXT DEFAULT 'Mensaje vacío',
   fecha DATE NOT NULL,
   hora TIME NOT NULL,
   FOREIGN KEY (id_chat) REFERENCES Chat(id_chat) ON DELETE CASCADE
   -> );
```

Tabla Notificacion

```
CREATE TABLE Notificacion (
   id_notificacion INT PRIMARY KEY AUTO_INCREMENT,
   id_usuario INT NOT NULL,
   tipo VARCHAR(50) NOT NULL,
   mensaje TEXT NOT NULL,
   fecha_envio DATETIME NOT NULL,
   estado VARCHAR(50) NOT NULL,
   FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario) ON DELETE CASCADE
);
```

Tabla Materia

```
CREATE TABLE Materia (
   id_materia INT PRIMARY KEY AUTO_INCREMENT,
   nombre VARCHAR(100) NOT NULL,
   imagen VARCHAR(30) NOT NULL,
   descripción TEXT,
   popularidad INT
);
```

Tabla Tema

```
CREATE TABLE Tema (
   id_tema INT PRIMARY KEY AUTO_INCREMENT,
   id_materia INT NOT NULL,
   nombre VARCHAR(100) NOT NULL,
   descripción TEXT,
   popularidad INT,
   FOREIGN KEY (id_materia) REFERENCES Materia(id_materia) ON DELETE CASCADE
);
```

Tabla Asesoria

```
CREATE TABLE Asesoria (
    id_asesoria INT PRIMARY KEY AUTO_INCREMENT,
    id_alumno INT NOT NULL,
    id_asesor INT NOT NULL,
    id_materia INT NOT NULL,
    id_tema INT NOT NULL,
    fecha DATE NOT NULL,
    hora TIME NOT NULL,
    aula VARCHAR(20) NOT NULL,
    estado VARCHAR(50) NOT NULL,
    FOREIGN KEY (id_alumno) REFERENCES Alumno(id_alumno) ON DELETE CASCADE,
    FOREIGN KEY (id_asesor) REFERENCES Asesor(id_asesor) ON DELETE CASCADE,
    FOREIGN KEY (id_materia) REFERENCES Materia(id_materia) ON DELETE CASCADE,
    FOREIGN KEY (id_tema) REFERENCES Tema(id_tema) ON DELETE CASCADE
);
```

Tabla Reporte

```
CREATE TABLE Reporte (
    id_reporte INT PRIMARY KEY AUTO_INCREMENT,
    id_asesoria INT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    descripción TEXT,
    fecha DATETIME NOT NULL,
    hora_inicial TIME NOT NULL,
    hora_final TIME NOT NULL,
    total_horas INT NOT NULL,
    porcentaje INT NOT NULL,
    estado_asesoria VARCHAR(50) NOT NULL,
    id_asesor INT NOT NULL,
    id_alumno INT NOT NULL,
    id_materia INT NOT NULL,
    id_tema INT NOT NULL,
    FOREIGN KEY (id_asesor) REFERENCES Asesor(id_asesor) ON DELETE CASCADE,
    FOREIGN KEY (id_alumno) REFERENCES Alumno(id_alumno) ON DELETE CASCADE,
    FOREIGN KEY (id_materia) REFERENCES Materia(id_materia) ON DELETE CASCADE,
    FOREIGN KEY (id_tema) REFERENCES Tema(id_tema) ON DELETE CASCADE,
    FOREIGN KEY (id_asesoria) REFERENCES Asesoria(id_asesoria) ON DELETE CASCADE
);
```

Tabla Solicitud

```
CREATE TABLE Solicitud (
    id_solicitud INT PRIMARY KEY AUTO_INCREMENT,
    id_usuario INT NOT NULL,
    id_alumno INT NOT NULL,
    hora TIME NOT NULL,
    modalidad VARCHAR(50) NOT NULL,
    observaciones TEXT,
    id_materia INT NOT NULL,
    id_tema INT NOT NULL,
    fecha_solicitud DATE NOT NULL,
    estado VARCHAR(50) NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario) ON DELETE CASCADE,
    FOREIGN KEY (id_alumno) REFERENCES Alumno(id_alumno) ON DELETE CASCADE,
    FOREIGN KEY (id_materia) REFERENCES Materia(id_materia) ON DELETE CASCADE,
    FOREIGN KEY (id_tema) REFERENCES Tema(id_tema) ON DELETE CASCADE
);
```