



**UNIVERSIDAD VALLE DEL MOMBOY; VICERRECTORADO
FACULTAD DE INGENIERÍA; PERIODO 2024A
CARRERA INGENIERÍA EN COMPUTACIÓN
BACKEND; SECCIÓN VIV/FI
UNIDAD III**

e-actividad 3.1:

Sistema de Gestión Universitario

Implementando Funcionalidades para Usuarios/Tokens

Integrantes:

Jonathan Antonio Arellano Márquez C.I: 24.190.278

José Emmanuel Cardoza Ferrer C.I: 31.590.138

Luis Fernando Araujo Giardinella C.I. 26.482.894

Denilson Eduardo Castellanos Garcia C.I. 29.694.566

Tutor: Profesor Roberto Di Michele

Contenido

Introducción.....	3
Funcionalidades Clave.....	4
Integración con Base de Datos.....	4
Visualización de Actividades.....	4
Protección de Datos.....	4
Facilidad de Manejo.....	4
Instalación del Sistema.....	5
Requisitos.....	5
Visual Studio Code (VSCode).....	5
Node.js.....	6
XAMPP.....	7
MySQL Workbench.....	8
GitHub.....	9
Postman.....	10
Despliegue.....	11
Paso 1: Clonando el Repositorio con GitHub.....	11
Paso 2: Ejecutar el Servidor Local MySQL con XAMPP.....	12
Paso 3: Crear la Base de Datos MySQL Local con MySQL Workbench.....	12
Paso 4: Crear el archivo ".env".....	14
Paso 5: Ejecutar el Servidor del SGU.....	15
Paso 6: Ejecutar Postman.....	16
Uso del Sistema.....	17
Consideraciones.....	17
Restricciones.....	18
Registrar un Usuario.....	19
Login.....	19
Visualizar Actividades por Semana.....	20
Visualizar Actividades Futuras de un Profesor.....	20
Manejo de Registros y Datos.....	21
Listar (Ver) Registros - GET.....	21
Aregar un Registro - POST.....	21
Editar un Registro - PUT.....	23
Eliminar un Registro - DELETE.....	24
Migración.....	25
Créditos y Comentarios de Desarrollador.....	28
Diario del Proyecto.....	28
Cambios Planificados para las Siguientes Versiones.....	29
Comentarios de Desarrollo.....	29
Créditos.....	30

Introducción

El presente proyecto “Sistema de Gestión Universitario” (SGU) trata acerca de una continuación al proyecto anterior del mismo nombre, cuyo objetivo es presentar un sistema capaz de listar las actividades pautadas para cada semana dentro de un periodo trimestral de clases. El sistema también mantiene otras funcionalidades como almacenamiento de información referente a profesores, materias, secciones, eventos y trimestres. De igual manera el sistema permite la libre edición de todos los registros referentes a lo mencionado anteriormente.

Este proyecto fue desarrollado con la idea principal de facilitar al usuario preparar un itinerario o incluso informarse de qué actividades futuras se acercan, e incluso permite al usuario mantener un registro de las actividades pasadas. Dentro del proyecto se puede resaltar la accesibilidad a editar, desarrollar y ordenar toda la información necesaria o que hacen referencia a las actividades de la universidad.

En la continuación del proyecto actualmente desarrollado, se han implementado características clave para asegurar la protección de los registros a través de un sistema de usuarios y tokens, este último siendo unas claves generadas por el sistema para identificar qué tipo de usuario está trabajando el SGU, y permitir o negar las operaciones dependiendo de su nivel de permisos. De la misma manera, el SGU también ha mejorado en el ámbito de accesibilidad, obteniendo mejores vistas al listar los datos, y presentando los correspondientes mensajes de advertencia en caso de que sean introducidos datos inválidos o con formato incorrecto.

Funcionalidades Clave

Integración con Base de Datos

El sistema está integrado con una base de datos que almacena toda la información relacionada con eventos, materias, profesores, y trimestres, asegurando la persistencia y seguridad de los datos. De la misma manera el sistema permite listar, agregar, editar y eliminar cualquier dato según el usuario crea necesario.

Visualización de Actividades

Ofrece una vista semanal de las actividades programadas, permitiendo a los usuarios visualizar de manera clara y ordenada los eventos próximos, facilitando así la organización personal y académica. De la misma forma, se ofrece una vista de las actividades programadas en un futuro para un profesor en específico.

Protección de Datos

Se le puede especificar al SGU qué tipo de usuario es el que interactúa con el proyecto, limitando así las funcionalidades dependiendo de la situación, con el fin de proteger los datos registrados y mantener el orden de la jerarquía. Esto es logrado a través de un sistema de registro, donde el usuario de mayor poder puede crear cuentas cuyas credenciales son encriptadas para aún mayor seguridad.

Facilidad de Manejo

A través de validaciones a las entradas que el usuario introduce, el SGU puede sanitizar los datos y presentar un mensaje de advertencia en caso de que los datos ingresados no se encuentren en un formato adecuado, brindando así al usuario una mejor claridad acerca de lo que se necesita solucionar en ese caso.

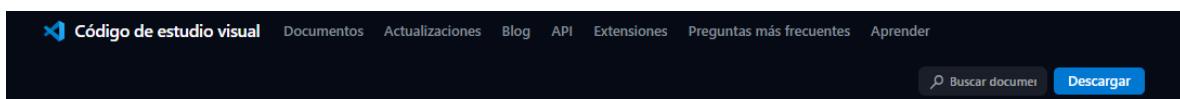
Instalación del Sistema

Requisitos

Visual Studio Code (VSCode)

Es un editor de código fuente desarrollado por Microsoft. Es ligero, potente y soporta una gran variedad de lenguajes de programación y archivos. Para este proyecto, se utilizará para ejecutar el servidor local del SGU.

1. El usuario debe dirigirse al siguiente enlace de descarga:
<https://code.visualstudio.com/download>
2. A continuación, debe hacer clic en el botón de descarga para el sistema operativo que utiliza, sea Windows, macOS o Linux.
3. Tras descargar el archivo de instalación, debe abrirlo y seguir las instrucciones proporcionadas por el asistente de instalación.
4. Una vez completada la instalación, VSCode estará listo para su uso.

Detailed download options for Visual Studio Code across three platforms: Windows, Linux, and Mac.

- Windows:** Includes 'ventanas' (Windows 10, 11) with options for 'Instalador del usuario' (x64, brazo64), 'Instalador del sistema' (x64, brazo64), '.cremaller'a (x64, brazo64), and 'CLI' (x64, brazo64).
- Linux:** Includes '.debutante' (Debian, Ubuntu) with options for '.deb' (x64, brazo32, brazo64), '.rpm' (x64, brazo32, brazo64), '.tar.gz' (x64, brazo32, brazo64), and 'Quebrar' (Tienda instantánea); also includes 'CLI' (x64, brazo32, brazo64).
- Mac:** Includes 'Mac' (macOS 10.15+) with options for '.cremaller'a (chip intel, silicona de manzana, Universal), 'CLI' (chip intel, silicona de manzana), and 'Tienda instantánea' (x64, brazo32, brazo64).

Imagen: Página web de descarga para VSCode

Node.js

Es el entorno de ejecución para JavaScript. En este proyecto, Node.js es necesario para ejecutar el servidor del proyecto SGU, más específicamente nos permitirá alojar los endpoints, permitiendo la interacción con el sistema a través de protocolos HTTP.

- 1. El usuario debe dirigirse al sitio web oficial de Node.js para iniciar la descarga:
<https://nodejs.org/en>
- 2. Luego, debe hacer clic en el botón de descarga que corresponda al sistema operativo que esté utilizando, ya sea Windows, macOS o Linux.
- 3. Una vez descargado el archivo de instalación, se debe de abrir y seguir las instrucciones del asistente de instalación.
- 4. Durante la instalación, es importante asegurarse de marcar la casilla que indica "Instalar herramientas de línea de comandos" para poder utilizar Node.js desde la terminal.
- 5. El usuario puede abrir la terminal y verificar que Node.js se haya instalado correctamente escribiendo el comando **node -v** y/o **npm -v**. De esta forma, debería ver la versión de Node.js y npm que han sido instaladas.



Imagen: Página web de descarga para Node.js

XAMPP

Es una distribución de Apache fácil de instalar que contiene MariaDB, PHP y Perl. Para el propósito de este proyecto, XAMPP se utilizará principalmente para su servidor MySQL, el cual proporcionará el sistema de gestión de bases de datos necesario para almacenar y gestionar la información del proyecto SGU.

1. El usuario debe dirigirse al sitio web oficial de XAMPP para iniciar su descarga:

<https://www.apachefriends.org/index.html>

2. Debe descargar la versión de XAMPP que corresponda a su sistema operativo (Windows, macOS o Linux).

3. Una vez descargado el archivo de instalación, debe seguir las instrucciones del asistente de instalación.

4. Durante la instalación, el usuario debe elegir los componentes que desea instalar (Apache, MySQL, PHP, phpMyAdmin, etc.). Se puede dejar las opciones predeterminadas para una instalación básica.

5. El usuario debe elegir la carpeta de instalación para XAMPP. Por lo general, se recomienda instalarlo en la carpeta predeterminada.

6. Debe completar la instalación y asegurarse de que los servicios de Apache y MySQL se inicien automáticamente al finalizar la instalación.

The screenshot shows the official XAMPP download page. At the top, there's a large orange XAMPP logo with the text "XAMPP Apache + MariaDB + PHP + Perl". Below it, a section titled "¿Qué es XAMPP?" provides a brief overview. To the right is a large image of the XAMPP icon, which is an orange square with a white play button symbol in the center. At the bottom, there are three download links: "Descargar" (Windows 8.2.12), "XAMPP para Linux 8.2.12 (PHP 8.2.12)", and "XAMPP para OS X 8.2.4 (PHP 8.2.4)".

Imagen: Página web de descarga para XAMPP

MySQL Workbench

Es una herramienta visual de diseño de bases de datos que integra desarrollo, administración, diseño, creación y mantenimiento de bases de datos en un único entorno integrado. En este proyecto, MySQL Workbench se utilizará para diseñar, crear y gestionar la base de datos local que almacena toda la información relevante del sistema.

- 1. Dirigirse al sitio web oficial de MySQL Workbench:
<https://dev.mysql.com/downloads/workbench/>
- 2. Descargar la versión de MySQL Workbench que corresponde al sistema operativo en donde el SGU será utilizado (Windows, macOS o Linux).
- 3. Una vez descargado el archivo de instalación, se deben seguir las instrucciones del asistente de instalación.
- 4. Durante la instalación, puedes elegir las opciones predeterminadas. Se recomienda al usuario instalar con las opciones por defecto.

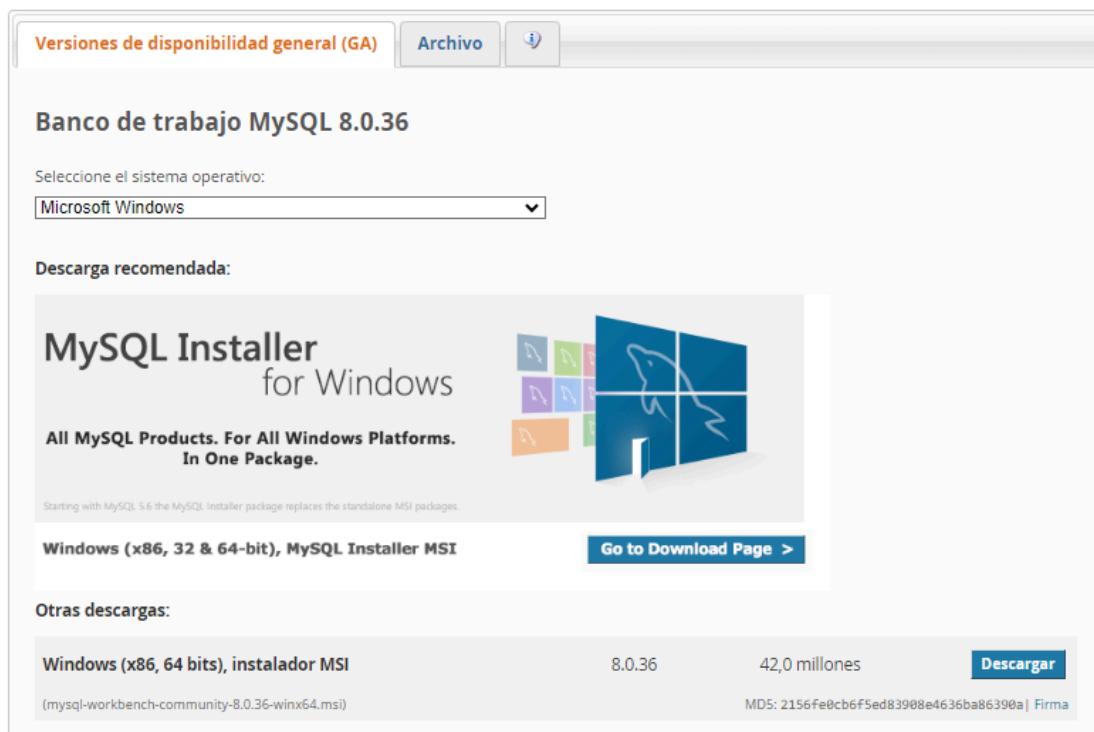


Imagen: Página web de descarga para Mysql Workbench

GitHub

Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se empleará para clonar el repositorio del proyecto localmente, permitiendo el acceso al código fuente y para la fácil implementación del proyecto en la máquina local. Para usarlo deberás seguir los siguientes pasos:

- 1. Crear una cuenta en GitHub: El usuario debe ir al sitio web oficial de GitHub (<https://github.com/>) para crear una cuenta propia.
- 2. Descargar e instalar la aplicación GitHub Desktop: Se necesita la aplicación para instalar el SGU localmente, clonando el repositorio a la máquina local. A través del sitio web de descarga (<https://desktop.github.com/>) el usuario debe seguir las instrucciones para descargar y, una vez descargado, instalar la aplicación.
- 3. Iniciar sesión en la aplicación: Una vez la aplicación ha sido instalada, el usuario puede iniciar sesión para poder clonar repositorios.

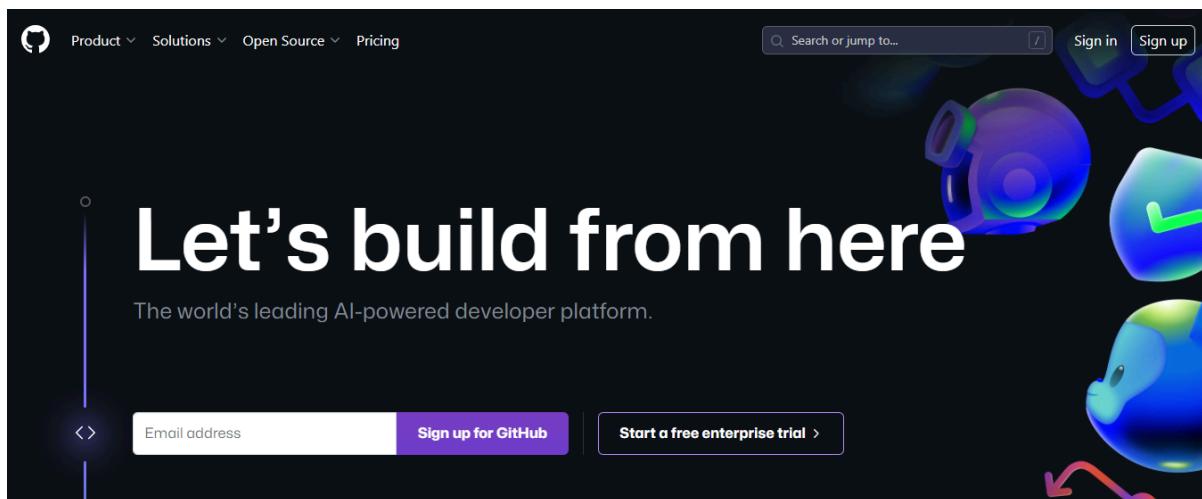


Imagen: Página web de GitHub

Postman

Es una herramienta para probar APIs. Permite enviar peticiones HTTP a una API y recibir respuestas, facilitando la prueba y depuración de los endpoints del servidor. En este proyecto, Postman se utilizará para interactuar con los endpoints del sistema, es decir, es la principal herramienta para el uso del proyecto SGU.

Para descargar Postman, accedemos al sitio web oficial de la herramienta (<https://www.postman.com/downloads/>) y hacemos clic en el botón de descarga correspondiente a su sistema operativo. Una vez descargado el archivo, se debe ejecutar el instalador y seguir las instrucciones del asistente de instalación. Postman está disponible de forma gratuita para Windows, macOS y Linux, y ofrece una interfaz intuitiva para realizar pruebas de API de manera eficiente.

The screenshot shows the official Postman website's download section. At the top, there's a navigation bar with links for 'Producto', 'Precios', 'Empresa', 'Recursos y soporte', and 'Red API pública'. Below the navigation, a large orange button with the Windows logo and the text 'ventanas de 64 bits' is prominently displayed. To the left of this button, there's a note about accepting privacy and terms. Further down, there's a link to 'Notas de lanzamiento' and another note about alternative download options for Mac and Linux. The overall layout is clean and professional.

Imagen: Página web de descarga para Postman

Despliegue

Una vez la máquina local posee los requisitos necesarios, se puede proceder a la instalación y despliegue del sistema. El correcto despliegue brindará seguridad y minimizará los inconvenientes a futuro durante el uso del SGU.

Paso 1: Clonando el Repositorio con GitHub

El proceso de despliegue comienza al obtener los archivos necesarios del sistema a través de la aplicación de GitHub instalada. El usuario puede clonar el repositorio a través de la URL <https://github.com/Luixls/ea3.1-proyecto> y seleccionando la opción correcta:

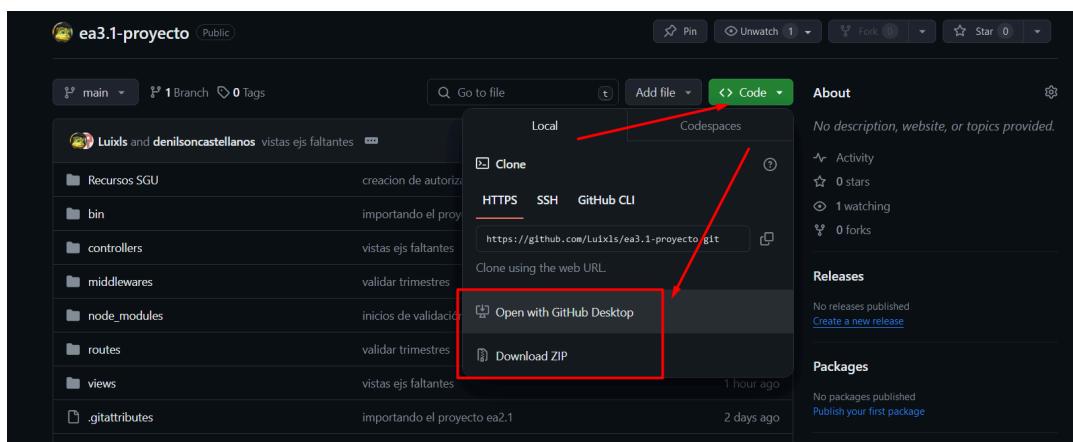


Imagen: Seleccionando la opción para obtener el repositorio con GitHub Desktop. Existe la opción de descargar el archivo zip, pero para este ejemplo, continuaremos con la aplicación GitHub Desktop.

La aplicación detectará que se quiere clonar un repositorio a la máquina local, y presentará el cuadro de confirmación para que el usuario seleccione qué nombre desea colocarle localmente al repositorio y en qué ubicación guardarlo.

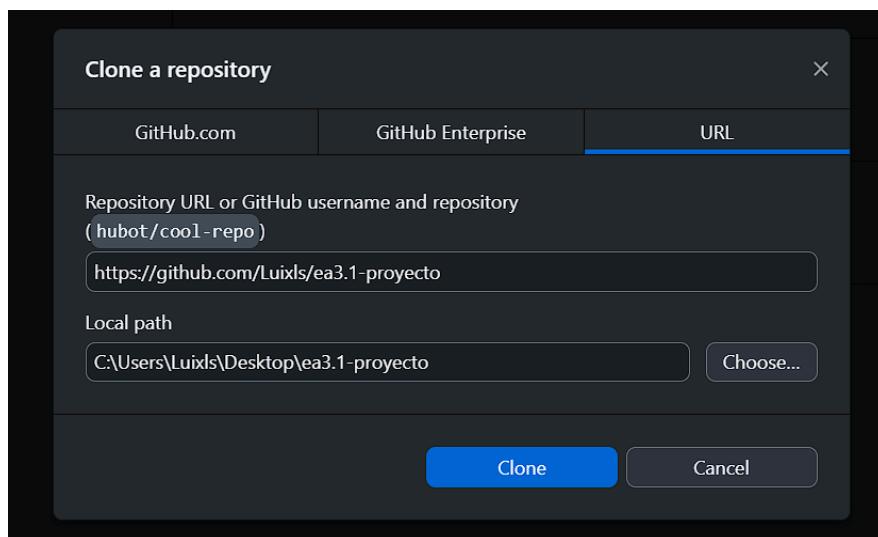


Imagen: Cuadro de confirmación de GitHub Desktop.

Paso 2: Ejecutar el Servidor Local MySQL con XAMPP

Procedemos a ejecutar el servidor para la base de datos utilizando el programa XAMPP, **se recomienda ejecutar XAMPP como administrador** para minimizar incompatibilidades. Una vez la ventana de XAMPP está presente, podemos iniciar el servidor apretando el botón “Start” para Apache y MySQL.

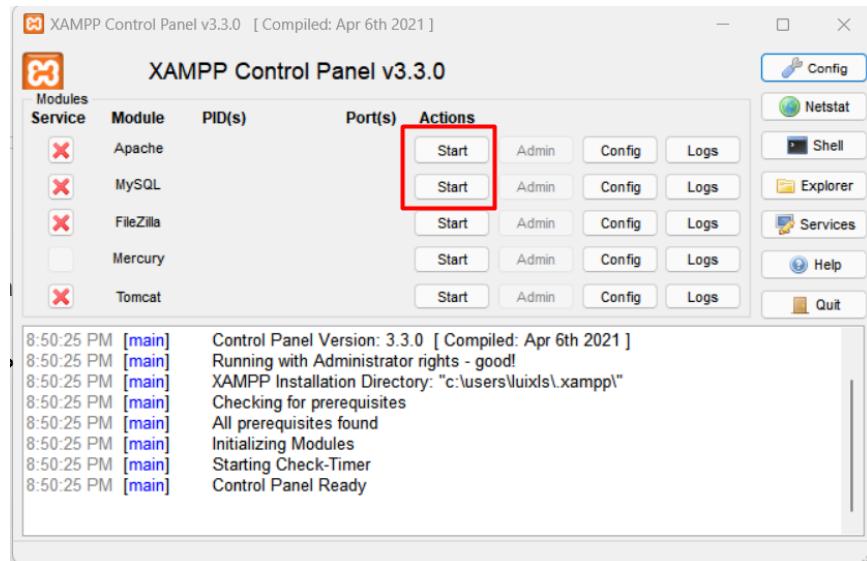


Imagen: Ventana de XAMPP, con los botones para iniciar el servidor resaltados.

Paso 3: Crear la Base de Datos MySQL Local con MySQL Workbench

Con el servidor MySQL ejecutándose, se procede a abrir MySQL Workbench y se establece la conexión al servidor local.

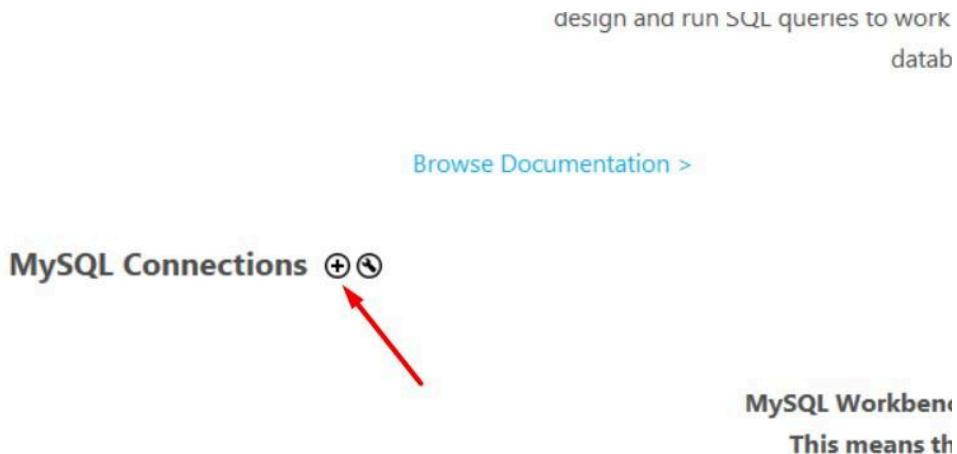


Imagen: Dentro de MySQL Workbench, botón para agregar conexión.

Se recomienda al usuario dejar los valores por defecto al momento de realizar la conexión, para que no haga falta editar archivos dentro del directorio del SGU.

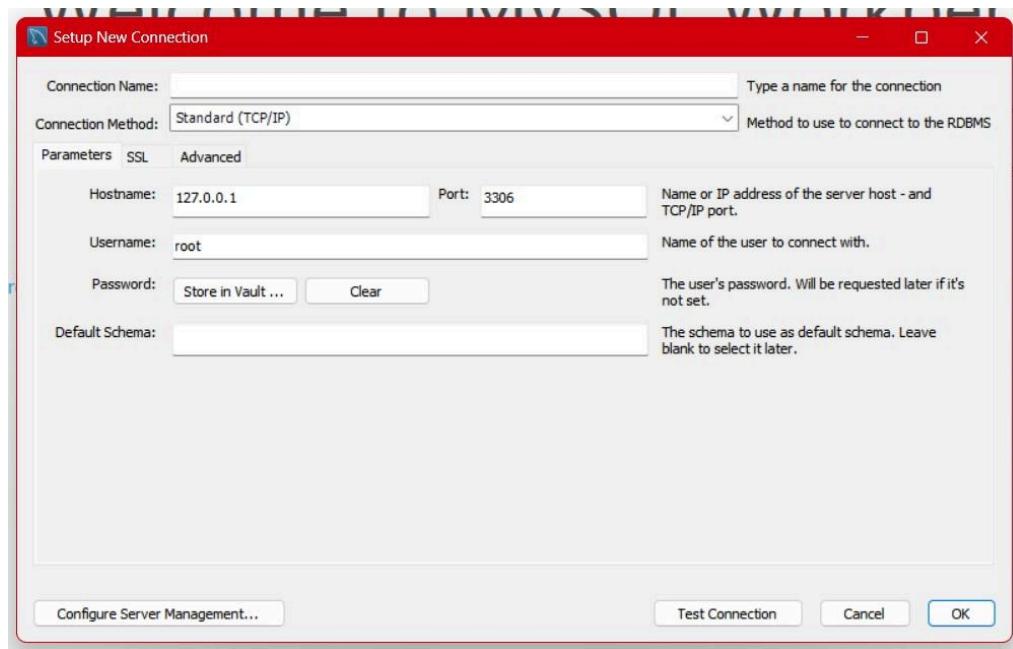


Imagen: Dentro de MySQL Workbench, ventana de configuración de conexión.

Una vez la conexión se establece, procedemos a abrir el script SQL proveído en la carpeta de “Recursos SGU” dentro del directorio del proyecto. Para ello, seleccionamos la opción de “Open SQL Script...” dentro de MySQL Workbench.

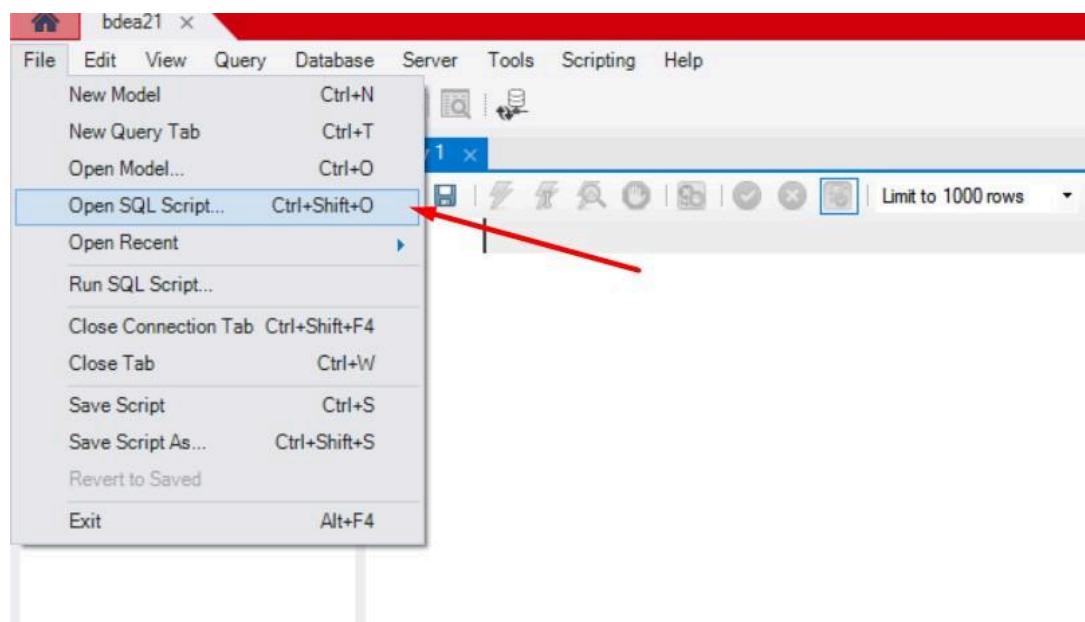
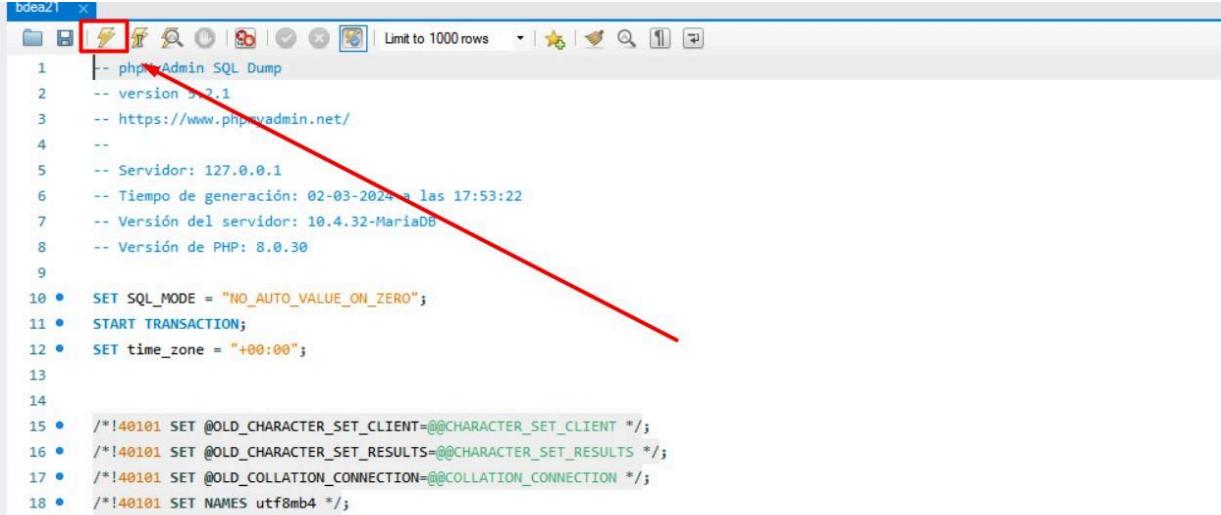


Imagen: Dentro de MySQL Workbench, opción para abrir un archivo script SQL.

Siguiente, simplemente se debe ejecutar dicho archivo apretando el botón correspondiente dentro de MySQL Workbench. La base de datos local será creada con los parámetros necesarios, e incluirá entradas (datos) de ejemplo pre-existentes.



```
bdea21
1 -- phpMyAdmin SQL Dump
2 -- version 5.2.1
3 -- https://www.phpmyadmin.net/
4 --
5 -- Servidor: 127.0.0.1
6 -- Tiempo de generación: 02-03-2024 a las 17:53:22
7 -- Versión del servidor: 10.4.32-MariaDB
8 -- Versión de PHP: 8.0.30
9
10 • SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
11 • START TRANSACTION;
12 • SET time_zone = "+00:00";
13
14
15 • /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
16 • /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
17 • /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
18 • /*!40101 SET NAMES utf8mb4 */;
```

Imagen: Dentro de MySQL Workbench, botón para ejecutar el script con la creación de la base de datos local.

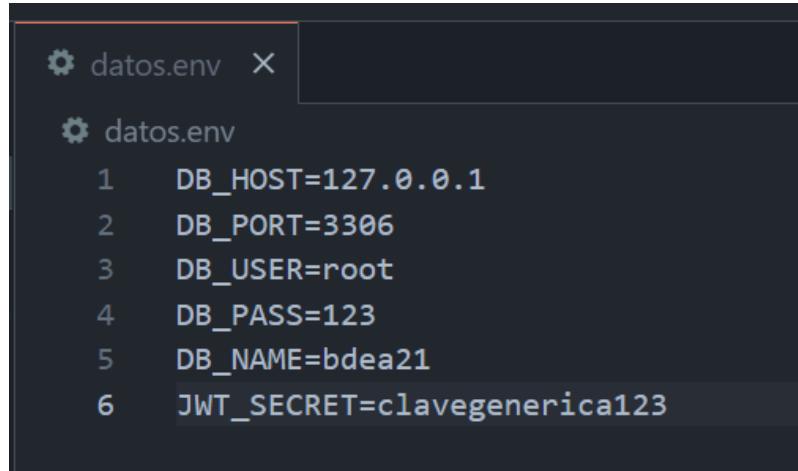
Ejecutar este script **asignará una contraseña a la base de datos**. En este ejemplo, será “123” y se trabajará con esta contraseña por defecto para proseguir.

Paso 4: Crear el archivo “.env”

Al culminar el anterior paso, se debe de crear un archivo llamado “**datos.env**” en la raíz del proyecto, donde serán guardadas las credenciales para obtener acceso a la base de datos local. Una vez creado el archivo, se debe ejecutar VSCode y arrastrar dicho archivo para poder editarla. Ya que se están trabajando los valores por defecto y sabemos que la contraseña asignada a la BD es “123”, se deben introducir las credenciales en el archivo .env de la siguiente manera:

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_USER=root
DB_PASS=123
DB_NAME=bdea21
JWT_SECRET=clavegenerica123
```

Texto en negritas: Las credenciales por defecto a ser introducidas en el archivo **datos.env**.



```
datos.env
1 DB_HOST=127.0.0.1
2 DB_PORT=3306
3 DB_USER=root
4 DB_PASS=123
5 DB_NAME=bdea21
6 JWT_SECRET=clavegenerica123
```

Imagen: Dentro de VSCode, introduciendo las credenciales en el archivo **datos.env**

Paso 5: Ejecutar el Servidor del SGU

Para el siguiente paso, se debe ejecutar el archivo que corresponde al servidor del SGU, localizado en el directorio del proyecto. Para ello, a través de VSCode, debemos elegir la opción de abrir una nueva carpeta, la cual será el directorio del SGU.

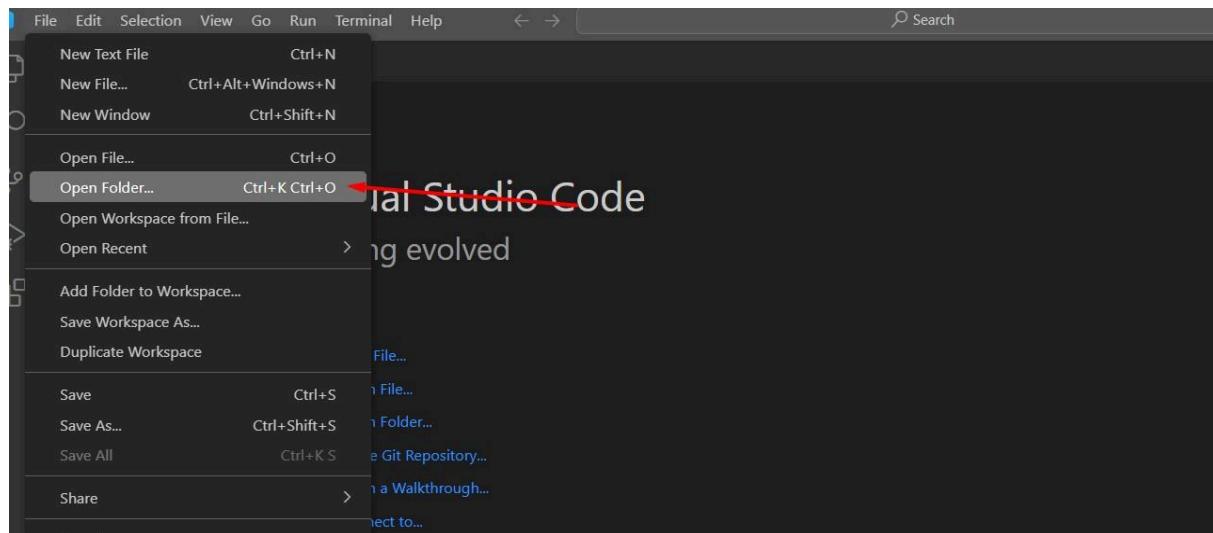


Imagen: Dentro de VSCode, seleccionando la opción de abrir carpeta.

Por último, solo es cuestión de utilizar el comando “**npx nodemon aplicacion.js**” para ejecutar el servidor local del SGU a través de la consola de VSCode. Un mensaje de conexión exitosa será mostrado cuando el servidor arranque.

```

12 // Ejecutar la conexión
13 connection.connect((err) => {
14   if (err) {
15     console.error("CONEXIÓN FALLIDA a la BD MySQL", err);
16     return;
17   }
18   console.log("CONEXIÓN A LA BD MYSQL EXITOSA");
19 });
20 });
21 // Enrutadores
22

```

DEBUG CONSOLE TERMINAL PORTS OUTPUT PROBLEMS

```

PS C:\Users\luixls\Desktop\ea2.1-proyecto> npx nodemon aplicacion.js
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node aplicacion.js`
Servidor corriendo en puerto 3000
CONEXIÓN A LA BD MYSQL EXITOSA
CONEXIÓN A LA BD MYSQL CERRADA CON ÉXITO (PRUEBA COMPLETADA)

```

Imagen: Dentro de VSCode, ejecutando el comando para iniciar el servidor local del SGU.

Paso 6: Ejecutar Postman

El último paso solo consta de ejecutar la aplicación Postman, para empezar a interactuar con el SGU.

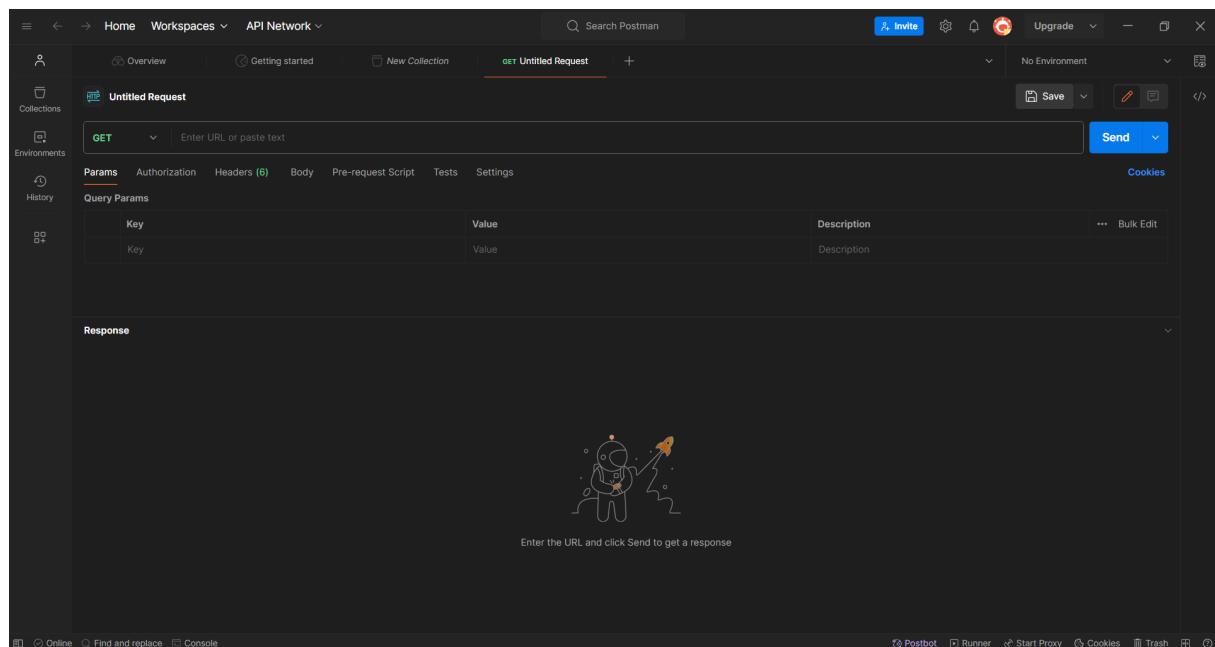


Imagen: Ventana principal de Postman, donde se ejecutarán los protocolos HTTP.

Uso del Sistema

Una vez el SGU ha sido desplegado en una máquina local, se puede proceder a utilizarlo o interactuar con él a través de la herramienta Postman, gracias a los endpoints desarrollados que están al alcance para recibir comandos HTTP (GET, POST, PUT, DELETE). Solo es cuestión de utilizar las URLs correspondientes a cada operación que el usuario final quiera realizar.

Consideraciones

Esta guía de uso de sistema supondrá que el usuario final está ejecutando su servidor local en el puerto por defecto (puerto 3000). Las instrucciones siguientes especificarán las URLs y el comando HTTP a ser utilizado para cada operación y, de ser necesario, el contenido del cuerpo que un comando HTTP debe de poseer. Cabe destacar que las URL siguen un formato parecido entre ellas para facilitar al usuario utilizar el SGU. Esta guía de uso presentará las URL de la siguiente manera:

`http://localhost:3000/eventos/[IDProfesor]/[FechaAAAA-MM-DD]`

Donde las letras en rojo deben ser reemplazadas por los valores correspondientes según los comandos que el usuario desee realizar. De la misma forma, en caso de que un comando requiera de un cuerpo, las letras rojas señalarán qué debe ser reemplazado, como el siguiente ejemplo:

```
{  
  "Nombre": "Nombre",  
  "Fecha": "AAAA-MM-DD",  
  "ID_Materia": "aqui va el # del ID de la materia que le pertenece este evento"  
}
```

Es importante el correcto uso de las URLs y comandos HTTP para evitar confusiones y problemas al momento de utilizar el SGU.

Ciertas acciones o comandos requieren de un “token” que el usuario puede obtener al hacer el login. Dicho token debe ser incluído en el comando enviado por Postman, en la pestaña “Headers”; donde la “Key” debe ser “**auth**” y “Value” debe ser el token del usuario.

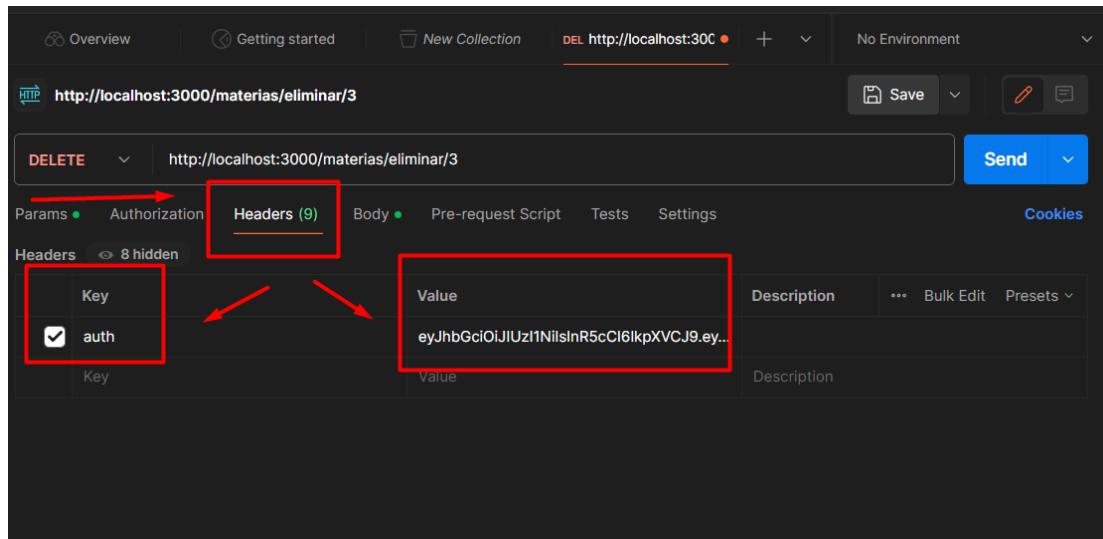


Imagen: En Postman, un ejemplo de cómo debe ir la sección “Headers” para introducir el token del usuario y obtener acceso a funcionalidades que requieren permisos.

Restricciones

Como se ha mencionado anteriormente, ciertas acciones están restringidas solo para aquellos usuarios con el nivel de permiso correspondiente. Por lo general, el sistema sigue el siguiente formato de restricciones:

- **Listar (GET):** Cualquier usuario puede listar los datos, e incluso no hace falta ningún tipo de autenticación para listar los datos.
- **Agregar (POST):** Solo cuentas tipo Director o Profesor pueden agregar datos al sistema.
- **Editar (PUT):** Solo cuentas tipo Director pueden editar libremente, mientras que las cuentas tipo Profesor reciben un mensaje de advertencia, pero pueden editar de todas formas.
- **Eliminar (DELETE):** Solo cuentas tipo Director pueden eliminar datos del SGU.

Estas restricciones se aplican para los datos de las tablas de profesores, materias, secciones y eventos. Cabe destacar que **las cuentas de tipo Profesor no pueden realizar acciones POST, PUT o DELETE en la tabla de trimestres**. Esto es exclusivo de cuentas tipo Director.

Registrar un Usuario

El SGU permite el registro de nuevos usuarios para crear cuentas de tipo “Director” y “Profesor”. Dichas cuentas poseen distintos niveles de permisos, y ciertas funcionalidades estarán disponibles dependiendo de ello. Para registrar un usuario, se debe hacer un comando **POST** a la URL <http://localhost:3000/usuarios/registro>

```
{  
  "nombreUsuario": "NOMBRE DE USUARIO",  
  "contraseña": "CONTRASEÑA AQUÍ",  
  "rol": "Profesor o Director"  
}
```

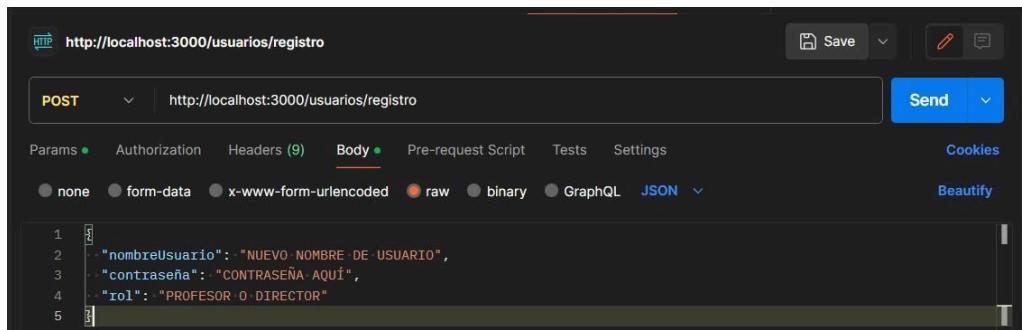


Imagen: En Postman, registrando un usuario.

Login

Realizar un login con una cuenta creada permite que el SGU provea un “token”, la cual es una cadena de texto que el usuario debe guardar e introducir en sus comandos de la manera mencionada en [consideraciones](#) para obtener acceso a las acciones que requieren autenticación.

```
{  
  "nombreUsuario": "NOMBRE DE USUARIO",  
  "contraseña": "CONTRASEÑA AQUÍ"  
}
```

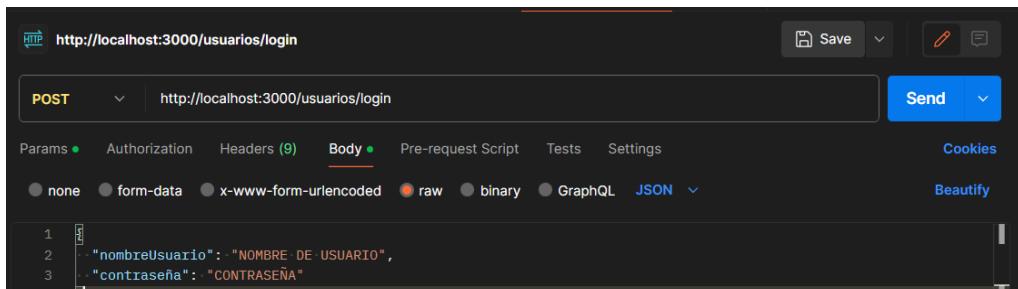


Imagen: En Postman, haciendo login para obtener un token de autenticación.

Visualizar Actividades por Semana

La visualización de las actividades de una semana específica se puede realizar a través de un comando GET a la siguiente URL:

[http://localhost:3000/calendario/actividades/\[NombreTrimestre\]/\[NúmeroDeSemana\]](http://localhost:3000/calendario/actividades/[NombreTrimestre]/[NúmeroDeSemana])

The screenshot shows a Postman request for `http://localhost:3000/calendario/actividades/2024A/1`. The response body contains the following JSON data:

```
Actividades de la Semana: 1 (2024-01-07 - 2024-01-13)

Trimestre: 2024A

[{"ID": 1, "Nombre del Evento": "Clase Unidad I", "Fecha": "2024-01-12", "D\u00eda de la Semana": "viernes", "Materia": "Matem\u00e1tica I", "Profesor": "Juan Perez", "Secci\u00f3n": "VIV/FI"}, {"ID": 4, "Nombre del Evento": "Continuaci\u00f3n Clase Unidad I", "Fecha": "2024-01-13", "D\u00eda de la Semana": "s\u00e1bado", "Materia": "Matem\u00e1tica I", "Profesor": "Juan Perez", "Secci\u00f3n": "VIV/FI"}, {"ID": 6, "Nombre del Evento": "Clase Unidad I", "Fecha": "2024-01-13", "D\u00eda de la Semana": "s\u00e1bado", "Materia": "F\u00edsica I", "Profesor": "Jorge \u00c1lvarez", "Secci\u00f3n": "VIV/FI"}, {"ID": 9, "Nombre del Evento": "Bienvenida al Trimestre", "Fecha": "2024-01-09", "D\u00eda de la Semana": "martes", "Materia": "Humanitas I", "Profesor": "Juana Paredes", "Secci\u00f3n": "VIV/FI"}, {"ID": 10, "Nombre del Evento": "Clase Virtual Unidad I", "Fecha": "2024-01-10", "D\u00eda de la Semana": "mi\u00e9rcoles", "Materia": "Ingenier\u00eda de la Programaci\u00f3n", "Profesor": "Roberto DiMichele", "Secci\u00f3n": "VIV/FI"}]
```

Imagen: Visualización de las actividades pautadas para una semana

Visualizar Actividades Futuras de un Profesor

La visualización de las actividades de una semana específica se puede realizar a través de un comando GET a la siguiente URL:

[http://localhost:3000/eventos/profesor/\[IDProfesor\]/\[FechaAAA-MM-DD\]](http://localhost:3000/eventos/profesor/[IDProfesor]/[FechaAAA-MM-DD])

The screenshot shows a Postman request for `http://localhost:3000/eventos/profesor/6/2024-01-01`. The response body contains the following JSON data:

```
Eventos Futuros para: Roberto DiMichele

[{"ID": 10, "Nombre del Evento": "Clase Virtual Unidad I", "Fecha": "2024-01-10", "ID Materia": 7, "Nombre Materia": "Ingenier\u00eda de la Programaci\u00f3n"}, {"ID": 11, "Nombre del Evento": "Clase Virtual Unidad III", "Fecha": "2024-02-09", "ID Materia": 7, "Nombre Materia": "Ingenier\u00eda de la Programaci\u00f3n"}]
```

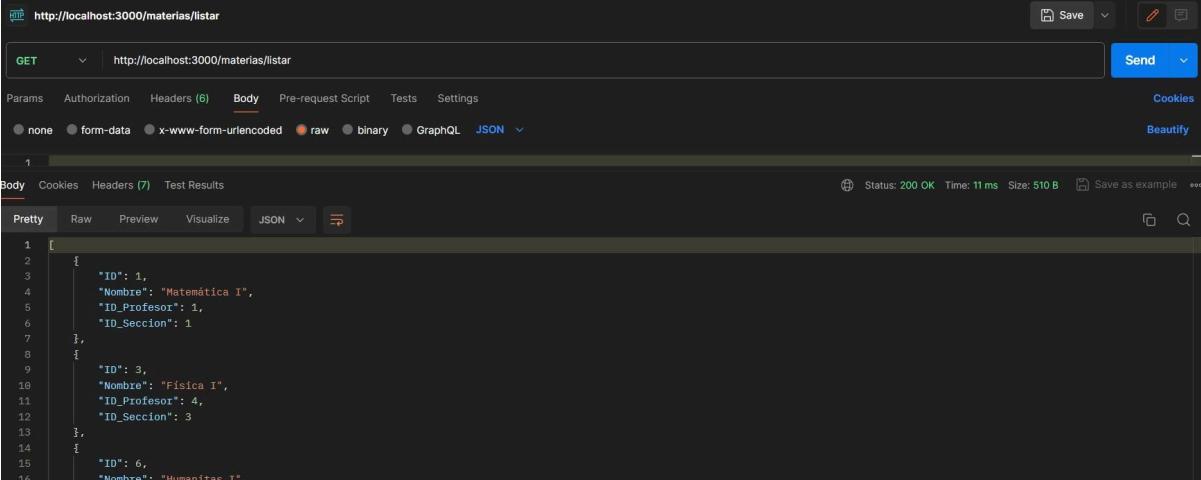
Imagen: Visualización de las actividades futuras de un profesor

Manejo de Registros y Datos

Listar (Ver) Registros - GET

Para listar ver las entradas que el SGU posee sobre un ámbito en específico, se debe realizar un comando **GET** a la URL correspondiente según lo que se quiera listar. Utilizar comandos GET no requieren de ningún permiso o token.

- **Listar Profesores:** <http://localhost:3000/profesores/listar>
- **Listar Materias:** <http://localhost:3000/materias/listar>
- **Listar Secciones:** <http://localhost:3000/secciones/listar>
- **Listar Eventos:** <http://localhost:3000/eventos/listar>
- **Listar Trimestres:** <http://localhost:3000/calendario/listar>



The screenshot shows a Postman interface with a GET request to `http://localhost:3000/materias/listar`. The response status is 200 OK, time is 11 ms, and size is 510 B. The response body is a JSON array containing three objects, each representing a subject:

```
[{"ID": 1, "Nombre": "Matemática I", "ID_Profesor": 1, "ID_Sección": 1}, {"ID": 3, "Nombre": "Física I", "ID_Profesor": 4, "ID_Sección": 3}, {"ID": 6, "Nombre": "Humanitas I"}]
```

Imagen: Listando materias

Agregar un Registro - POST

Para agregar un registro al SGU, se debe utilizar un comando **POST** a la URL correspondiente según las necesidades del usuario. Dicho comando debe tener un cuerpo en formato JSON con los datos necesarios correspondientes, especificados por el usuario según lo que se quiere agregar. Agregar un registro al SGU requiere de un token obtenido al hacer login, siempre y cuando la cuenta del usuario sea de tipo profesor o director.

- **Agregar Profesores:** <http://localhost:3000/profesores/agregar>

```
{  
  "Nombre": "Fulano Detal"  
}
```

- **Agregar Materias:** <http://localhost:3000/materias/agregar>

```
{
    "Nombre": "Nombre de la Materia",
    "ID_Profesor": "aquí va el # del ID del profesor que dará esta materia",
    "ID_Seccion": "aquí va el # del ID de la sección de esta materia"
}
```
- **Agregar Secciones:** <http://localhost:3000/secciones/agregar>

```
{
    "Nombre": "Nombre de la sección",
    "ID_Materia": "aquí va el # del ID de la materia que le pertenece esta sección",
    "ID_Profesor": "aquí va el # del ID del profesor que le pertenece esta materia"
}
```
- **Agregar Eventos:** <http://localhost:3000/eventos/agregar>

```
{
    "Nombre": "Nombre del Evento",
    "Fecha": "AAAA-MM-DD",
    "ID_Materia": "aquí va el # del ID de la materia que le pertenece este evento"
}
```
- **Agregar Eventos Globales:** <http://localhost:3000/eventos/agregar>

```
{
    "Nombre": "Nombre del Evento",
    "Fecha": "AAAA-MM-DD",
    "esglobal": "true"
}
```
- **Agregar Trimestres:** <http://localhost:3000/calendario/agregar>

```
{
    "Nombre": "2024B",
    "Fecha_Inicio": "2024-04-08",
    "Fecha_Final": "2024-07-07"
}
```

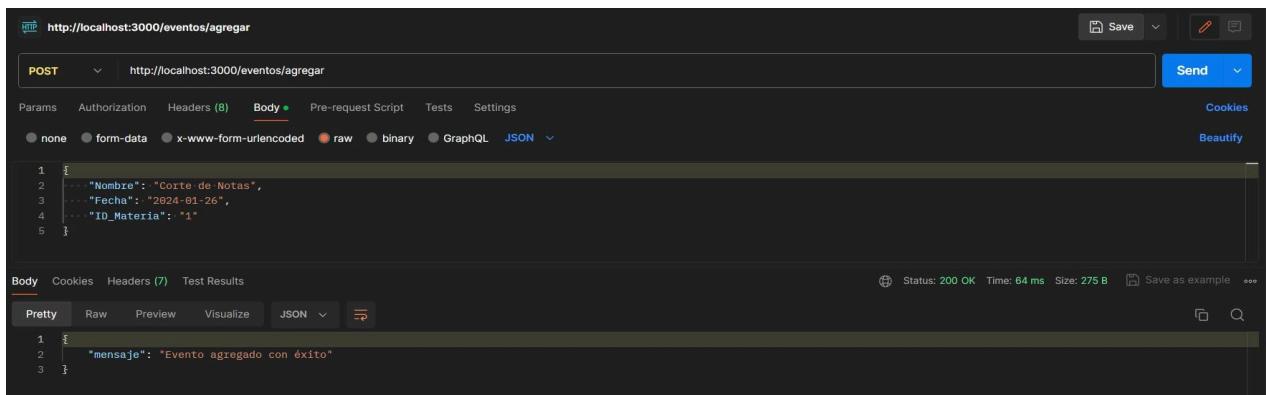


Imagen: Agregando un evento

Editar un Registro - PUT

Para editar un registro al SGU, se debe utilizar un comando **PUT** a la URL correspondiente según las necesidades previamente especificadas por el usuario. Dicho comando debe tener un cuerpo en formato JSON con los datos nuevos que sobreescribirán los anteriores, especificados por el usuario. Editar un registro requiere de un token obtenido al hacer login, y los profesores obtendrán un mensaje de advertencia para utilizar extremo cuidado al momento de editar datos, mientras que una cuenta tipo director no.

- **Editar Profesor:** [http://localhost:3000/profesores/editar/\[IDProfesor\]](http://localhost:3000/profesores/editar/[IDProfesor])

```
{
    "Nombre": "Nombre Nuevo"
}
```

- **Editar Materia:** [http://localhost:3000/materias/editar/\[IDMateria\]](http://localhost:3000/materias/editar/[IDMateria])

```
{
    "Nombre": "Nuevo nombre de la materia",
    "ID_Profesor": "aquí va el # del ID del profesor que dará esta materia",
    "ID_Seccion": "aquí va el # del ID de la sección de esta materia"
}
```

- **Editar Sección:** [http://localhost:3000/secciones/editar/\[IDSeccion\]](http://localhost:3000/secciones/editar/[IDSeccion])

```
{
    "Nombre": "Nuevo nombre de la sección",
    "ID_Materia": "aquí va el # del ID de la materia que le pertenece esta sección",
    "ID_Profesor": "aquí va el # del ID del profesor que le pertenece esta materia"
```

```

}

• Editar Evento: http://localhost:3000/eventos/editar/\[IDEvento\]

{
    "Nombre": "Nuevo nombre del Evento",
    "Fecha": "AAAA-MM-DD",
    "ID_Materia": "aqui va el # del ID de la materia que le pertenece este evento"
}

```

- **Editar Trimestre:** [http://localhost:3000/calendario/editar/\[IDTrimestre\]](http://localhost:3000/calendario/editar/[IDTrimestre])

```

{
    "Nombre": "Nuevo nombre del Trimestre",
    "Fecha_Inicio": "AAAA-MM-DD",
    "Fecha_Final": "AAAA-MM-DD"
}

```

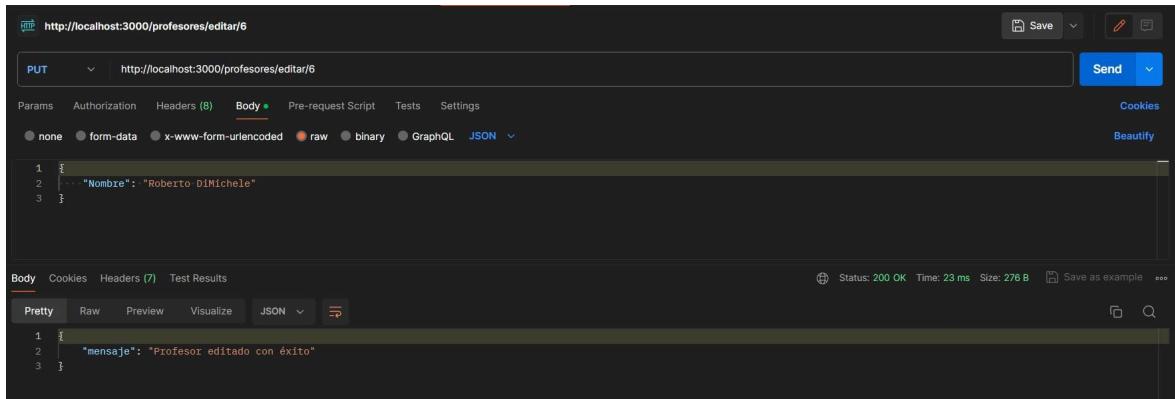


Imagen: Editando un profesor

Eliminar un Registro - DELETE

Para eliminar registros del SGU, se debe ejecutar un comando **DELETE** a la URL correspondiente según la necesidad del usuario.

- **Eliminar Profesor:** [http://localhost:3000/profesores/eliminar/\[IDProfesor\]](http://localhost:3000/profesores/eliminar/[IDProfesor])
- **Eliminar Materia:** [http://localhost:3000/materias/eliminar/\[IDMateria\]](http://localhost:3000/materias/eliminar/[IDMateria])
- **Eliminar Sección:** [http://localhost:3000/secciones/eliminar/\[IDSeccion\]](http://localhost:3000/secciones/eliminar/[IDSeccion])
- **Eliminar Evento:** [http://localhost:3000/eventos/eliminar/\[IDEvento\]](http://localhost:3000/eventos/eliminar/[IDEvento])
- **Eliminar Trimestre:** [http://localhost:3000/calendario/eliminar/\[IDTrimestre\]](http://localhost:3000/calendario/eliminar/[IDTrimestre])

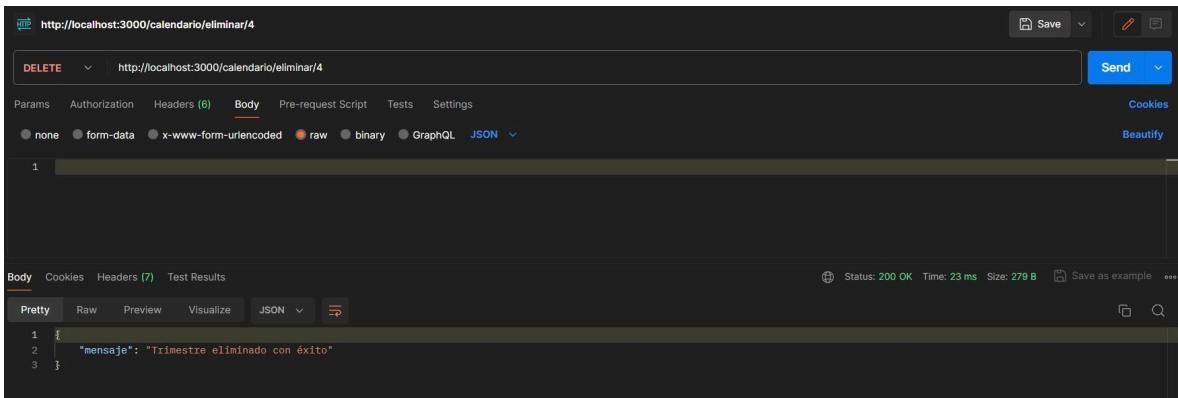


Imagen: Eliminando un trimestre

Migración

Este paso es opcional, bien el usuario podría crear la base de datos manualmente en el servidor, o simplemente exportar e importar en el servidor a usar, sin embargo el siguiente script fue diseñado con la finalidad de migrar una base de datos desde un archivo “SQL” al servidor.

Para ejecutar el archivo “migrate.js” y realizar la migración de la base de datos, siga estos pasos:

1. Ubicación del Archivo SQL: Asegurarse de que el archivo .sql esté ubicado en el directorio especificado en el script. Modificar la ruta del archivo si es necesario.
2. Instalación de Dependencias: Ejecutar `npm install mysql fs path` para instalar las dependencias necesarias para ejecutar correctamente el archivo..
3. Ejecución del archivo: Desde la línea de comandos o terminal, navegar al directorio donde se encuentra el archivo y ejecutar `node <nombre_del_script>.js` para iniciar el proceso. En éste caso el comando sería: `node migrate.js`

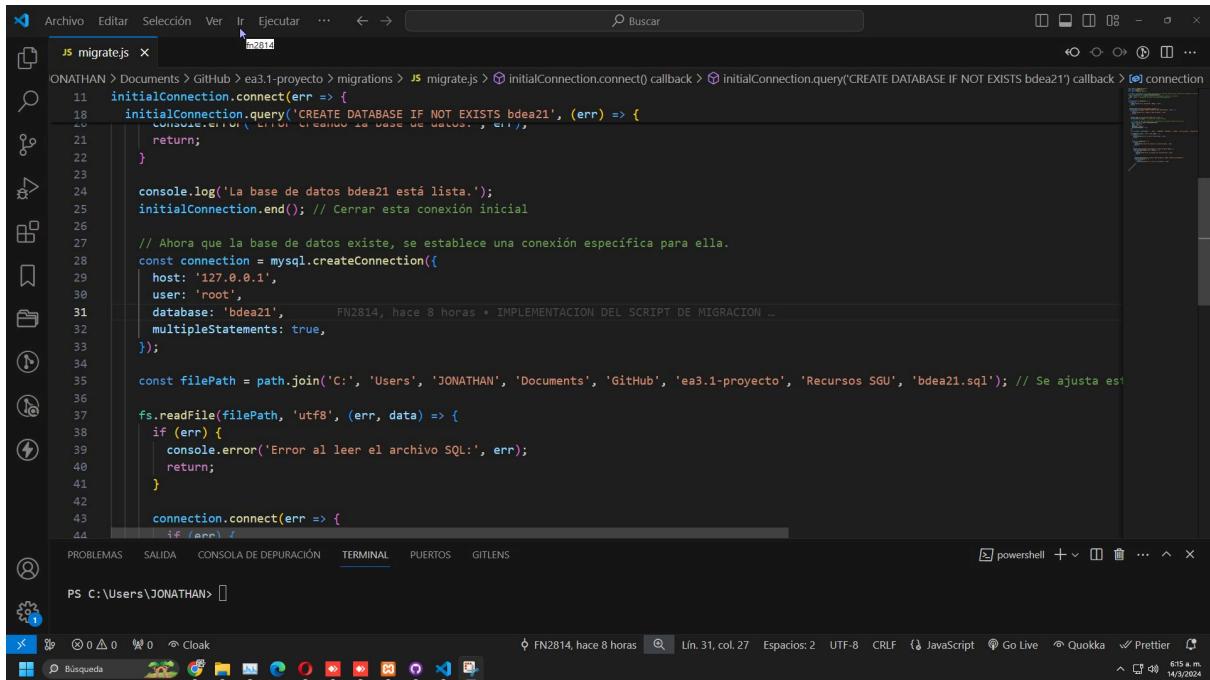
El archivo está ubicado en la carpeta “migrations” de la carpeta raíz del repositorio, además del archivo, el cual tiene un nombre bastante intuitivo respecto a la función:

Migrate.js el cual está desarrollado en lenguaje JavaScript y realiza las siguientes operaciones en el orden indicado:

1. Establecimiento de la Conexión Inicial: Se realiza una conexión a MySQL sin especificar una base de datos para verificar o crear la base de datos **bdea21**.

2. Creación de la Base de Datos: Utilizando el comando CREATE DATABASE IF NOT EXISTS **bdea21**, se verifica la existencia o se crea la nueva base de datos.
3. Cierre de la Conexión Inicial: Una vez creada la base de datos, se cierra la conexión inicial.
4. Establecimiento de la Conexión Específica: Se abre una nueva conexión a MySQL, esta vez especificando la base de datos **bdea21** recién creada, para ejecutar las operaciones de creación de tablas y la inserción de datos.
5. Lectura y Ejecución del Script SQL: El script lee el contenido del archivo SQL ubicado en la ruta especificada y ejecuta las consultas contenidas en él. Este archivo contiene las instrucciones para crear las tablas (`eventos`, `materias`, `profesores`, `secciones`, `trimestres`) y para insertar datos de prueba.
6. Cierre de la Conexión de la Base de Datos: Finalmente, se cierra la conexión a la base de datos una vez completadas todas las operaciones.

A continuación, una explicación más gráfica sobre el uso del archivo:



```

ONATHAN > Documents > GitHub > ea3.1-proyecto > migrations > JS migrate.js > initialConnection.connect() callback > initialConnection.query('CREATE DATABASE IF NOT EXISTS bdea21') callback > connection
  11  initialConnection.connect(err => {
  12    if (err) {
  13      console.error(`Error al crear la base de datos: ${err}`);
  14      return;
  15    }
  16
  17    console.log('La base de datos bdea21 está lista.');
  18    initialConnection.end(); // Cerrar esta conexión inicial
  19
  20    // Ahora que la base de datos existe, se establece una conexión específica para ella.
  21    const connection = mysql.createConnection({
  22      host: '127.0.0.1',
  23      user: 'root',
  24      database: 'bdea21', FN2814, hace 8 horas • IMPLEMENTACION DEL SCRIPT DE MIGRACION ...
  25      multipleStatements: true,
  26    });
  27
  28    const filePath = path.join('C:', 'Users', 'JONATHAN', 'Documents', 'GitHub', 'ea3.1-proyecto', 'Recursos SGU', 'bdea21.sql'); // Se ajusta este
  29
  30    fs.readFile(filePath, 'utf8', (err, data) => {
  31      if (err) {
  32        console.error('Error al leer el archivo SQL:', err);
  33        return;
  34      }
  35
  36      connection.connect(err => {
  37        if (err) {
  38          console.error(`Error al conectar a la base de datos: ${err}`);
  39          return;
  40        }
  41
  42        connection.query(data);
  43      });
  44    });
  45  });
  46
  47  connection.query(`CREATE DATABASE IF NOT EXISTS bdea21`, (err) => {
  48    if (err) {
  49      console.error(`Error al crear la base de datos: ${err}`);
  50      return;
  51    }
  52
  53    console.log(`Base de datos bdea21 creada exitosamente.`);
  54  });
  55
  56  connection.end();
  57
  58  console.log(`La base de datos bdea21 está lista.`);
  59
  60  module.exports = connection;
  
```

Imagen: Tal como se especifica en el comentario del código, se debe especificar de forma correcta la dirección del archivo SQL.

```
C:\> Users > JONATHAN > Documents > GitHub > ea3.1-proyecto > Recursos SGU > bdea21.sql
32  --
33  --
34  -- Estructura de tabla para la tabla `eventos`
35  --
36 CREATE DATABASE IF NOT EXISTS bdea21;
37
38 USE bdea21;
39
40 ALTER USER 'root' @'localhost' IDENTIFIED BY '123';
41
42 CREATE TABLE `eventos` (
43     `ID` int(11) NOT NULL,
44     `Nombre` varchar(255) NOT NULL,
45     `Fecha` date NOT NULL,
46     `ID_Materia` int(11) DEFAULT NULL
47 ) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4_general_ci;
48
49  --
50  --
51  -- Estructura de tabla para la tabla `materias`
52  --
53 CREATE TABLE `materias` (
54     `ID` int(11) NOT NULL,
55     `Nombre` varchar(255) NOT NULL,
56     `ID_Profesor` int(11) DEFAULT NULL,
57     `ID_Seccion` int(11) DEFAULT NULL
58 ) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4_general_ci;
59
60  --
61  --
62  -- Estructura de tabla para la tabla `profesores`
63  --
```

Imagen: En VSCode, archivo script SQL siendo visualizado.

Así como se especifica en ciertos comentarios del archivo “migrate.js”, se deben de ajustar las credenciales conforme a la base de datos, el no realizarlo podría producir contradicciones, en el caso de nuestra prueba se colocó una base de datos sin contraseña para hacer la prueba más “universal”, se descartaron algunas líneas como la #40 del código mostrado en pantalla.

Se destaca el hecho de que éste proceso de migración es opcional, el usuario puede exportar e importar la base de datos manualmente según sus comodidades, o bien usar el script considerando las configuraciones de autenticación.

Créditos y Comentarios de Desarrollador

Diario del Proyecto

1. Se clonó el proyecto de la ea2.1.
2. Se instalaron nuevos módulos de Node.js (jsonwebtoken, dotenv, bcrypt) y se removieron ciertos archivos irrelevantes (que habían sido creados por el generador express al desarrollar el esqueleto).
3. Se creó una regla “.gitignore” para trabajar con un archivo .env local sin que fuera añadido al repositorio.
4. Se desarrollaron los principios del proyecto para cumplir con el enunciado; trabajar con variables de entorno.
5. Se crearon las funcionalidades de registro y login con encriptación de contraseñas de usuario. Esto implicó crear una nueva tabla en la BD local.
6. Se creó el middleware que maneja la autenticación de tokens y la restricción de comandos dependiendo del tipo de usuario.
7. Se desarrolló la lógica para limitar comandos en cada tabla progresivamente.
8. Se desarrolló un middleware de validación de entradas (es decir, sanitizar entradas de usuario) y mostrar mensajes de advertencia en caso de qué tipo de error presenta una entrada.
9. Se implementó la validación de entradas para cada tipo de entradas de datos.
10. Se desarrollaron vistas ejs para todas las URL con endpoints “GET”.
11. Se implementó un script para ejecutar un proceso de migración de la base de datos al servidor.
12. Se creó una funcionalidad para añadir eventos globales al sistema, esto implicó ajustar la BD para el correcto funcionamiento.
13. Ciertos otros arreglos de formato o pequeños cambios fueron implementados a lo largo de los pasos anteriores, pero no lo suficientemente grandes como para mencionarlos específicamente.

Cambios Planificados para las Siguientes Versiones

1. Ajustar la estructura de la BD para permitir y facilitar con las correcciones indicadas a continuación:

"...definir las actividades y evaluaciones que hace en cada sección, pero sin fecha asociada, sólo el número de semana".

"..guardar está información de actividades o evaluaciones de forma general para luego asignarla a cada sección".

2. Implementar la funcionalidad de visualizar actividades pautadas en un futuro **globalmente** desde una fecha indicada por el usuario:

"...ver la planificación académica incluyendo eventos de la facultad, feriados, evaluaciones con sus porcentajes y demás".

3. Trasladar la recolección de datos para ser visualizados por el usuario para que sean manejados por los **enrutadores** en vez de los controladores.
4. Actualizar las funciones para manejar correctamente los cambios a ser realizados en la estructura de la BD.
5. Continuar facilitando la experiencia del usuario (que, a su modo, facilitará realizar pruebas y cambios al proyecto).

Comentarios de Desarrollo

Para los integrantes del equipo, el proyecto fue algo inesperado debido a que por primera vez ningún integrante contaba con al menos un poco de experiencia previa sobre las funcionalidades pautadas por el enunciado.

Las verdaderas dificultades se presentaron al intentar aplicar las correcciones sugeridas para la e-actividad 2.1, ya que este proyecto, al ser una continuación de dicha actividad, estaba más adelantado en su desarrollo y dichas correcciones implican, hasta donde los integrantes del equipo pueden observar, un cambio a la estructura fundamental del proyecto. Sin embargo, esto no es una mala noticia, debido a que los integrantes han notado que un cambio en la estructura de la BD es indispensable, y que dicha estructura que fue desarrollada anteriormente no fue planificada a futuro, grandemente resaltado por los grandes cambios que hay que realizar para aplicar pequeñas correcciones.

Volviendo a entrar en el tema del desarrollo de este proyecto, podemos señalar que la lógica de programación más importante se encuentra en el controlador de usuario, ya que a través del sistema de login se logró limitar ciertas funcionalidades a través de niveles de permisos. De igual manera, el middleware que autentica dichos tokens también es igual de importante al cumplir su funcionalidad de señalar al sistema la legitimidad del token que está siendo introducido.

Créditos

Resaltamos el trabajo en equipo de los integrantes desarrolladores del proyecto. El repositorio puede ser accedido a través de la siguiente URL a GitHub:

<https://github.com/Luixls/ea3.1-proyecto>

The screenshot shows the GitHub repository page for 'ea3.1-proyecto'. The repository was created by FN2814 and jonavnet, with 1 branch and 0 tags. It has 26 commits. The main branch contains files like 'Recursos SGU', 'bin', 'controllers', 'middlewares', 'migrations', 'node_modules', 'routes', 'views', '.gitattributes', '.gitignore', 'README.md', 'aplicacion.js', 'dbConfig.js', 'package-lock.json', and 'package.json'. The 'About' section indicates no description, website, or topics provided. The 'Contributors' section, which lists four contributors (Luixls, denilsoncastellanos, jonavnet, and FN2814), is highlighted with a red box. The 'Languages' section shows JavaScript at 78.7% and EJS at 21.3%. The repository has 0 stars, 1 watching, and 0 forks.

Imagen: Repositorio de GitHub, con los integrantes resaltados.

A continuación, se nombran los integrantes del equipo con su nombre de usuario correspondiente de GitHub:

@jonathanuvm - Jonathan Antonio Arellano Márquez C.I: 24.190.278

@FN2814 - José Emmanuel Cardoza Ferrer C.I: 31.590.138

@Luixls - Luis Fernando Araujo Giardinella C.I. 26.482.894

@denilsoncastellanos - Denilson Eduardo Castellanos Garcia C.I. 29.694.566

Por siguiente, los commits del repositorio del proyecto denotan la participación y/o distribución de las responsabilidades de los integrantes, y de la misma manera denotan cuando una parte del proyecto fue elaborada en conjunto de dos o más integrantes. El historial de commits puede ser accedido a través de la siguiente URL a GitHub:

<https://github.com/Luixls/ea3.1-proyecto/commits/main/>

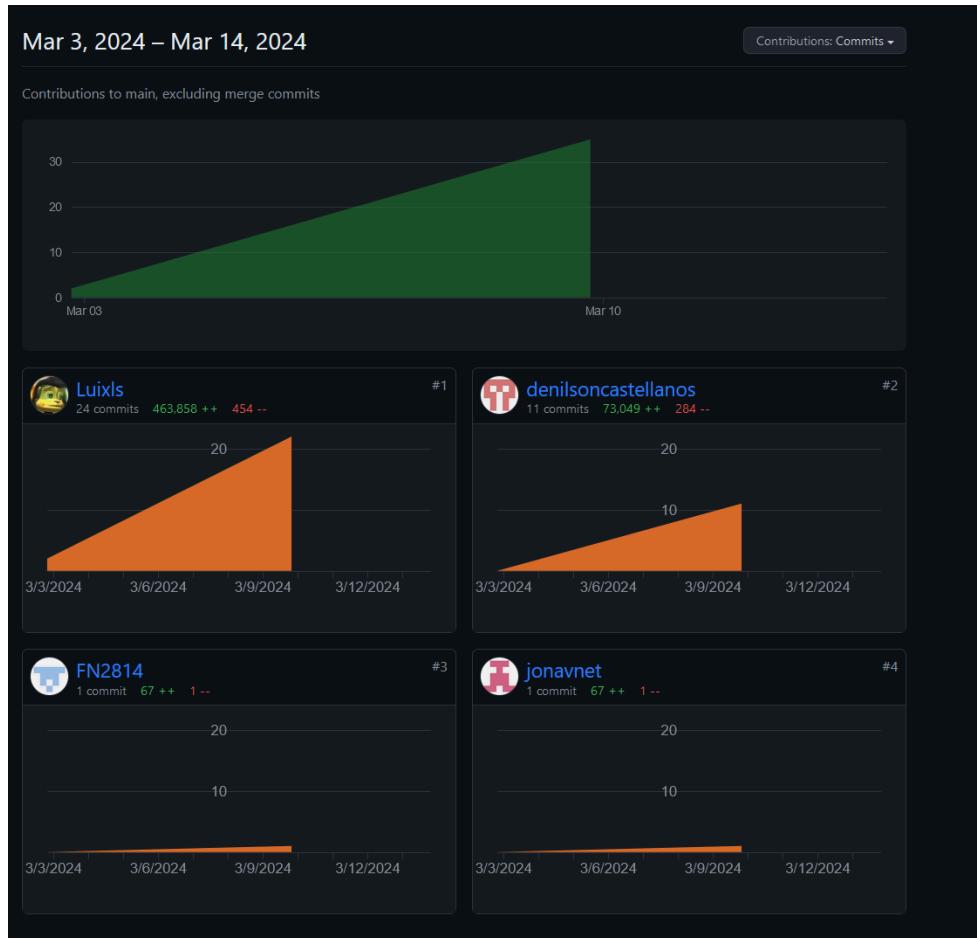


Imagen: Gráfico de participación de los integrantes.

Y para culminar, el video explicativo del proyecto puede ser accedido a través de la siguiente URL a youtube:

<https://www.youtube.com/watch?v=TDjtHZCDGWU>