



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

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
Gestión de Eventos.....	4
Visualización de Actividades por Semana.....	4
Manejo de Trimestres.....	4
Consulta de Eventos Futuros.....	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: Ejecutar el Servidor del SGU a través de VSCode.....	14
Paso 5: Ejecutar Postman.....	15
Uso del Sistema.....	16
Consideraciones.....	16
Visualizar Actividades por Semana.....	17
Visualizar Actividades Futuras de un Profesor.....	17
Manejo de Registros y Datos.....	18
Listar (Ver) Registros - GET.....	18
Aregar un Registro - POST.....	18
Editar un Registro - PUT.....	20
Eliminar un Registro - DELETE.....	21
Créditos y Comentarios de Desarrollador.....	22
Diario del Proyecto.....	22
Comentarios de Desarrollo.....	23
Créditos.....	23

Introducción

El presente proyecto “Sistema de Gestión Universitario” (SGU) trata acerca de un sistema cuyo objetivo principal es listar las actividades pautadas para cada semana dentro de un periodo trimestral de clases. El sistema también posee otras funcionalidades como almacenamiento de información referente a profesores, materias, secciones y eventos. 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.

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.

Gestión de Eventos

Permite a los usuarios crear, editar, y eliminar eventos académicos, tales como clases, seminarios, evaluaciones, y reuniones. Cada evento puede ser detallado con información relevante, incluyendo fechas, horarios, y descripciones.

Visualización de Actividades por Semana

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.

Manejo de Trimestres

Los usuarios pueden definir y gestionar diferentes trimestres o períodos académicos a través de las funcionalidades desarrolladas, asignando eventos a cada uno de ellos para una mejor estructuración del calendario académico.

Consulta de Eventos Futuros

Proporciona una herramienta para consultar los eventos futuros programados para un profesor específico desde una fecha dada, permitiendo así una planificación anticipada y eficiente para cualquier usuario, sea profesor o estudiante.

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.

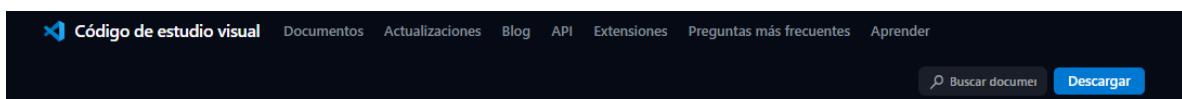


Imagen: Página web de descarga para VSCode

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.

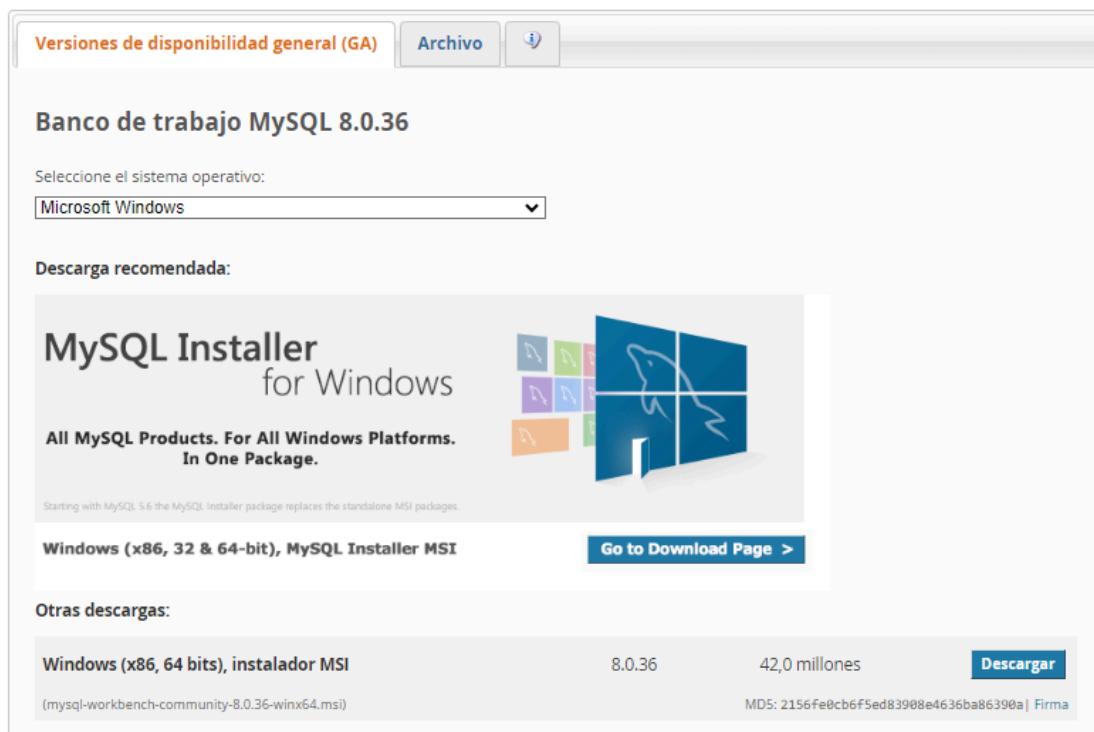


Imagen: Página web de descarga para Mysql Workbench

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.

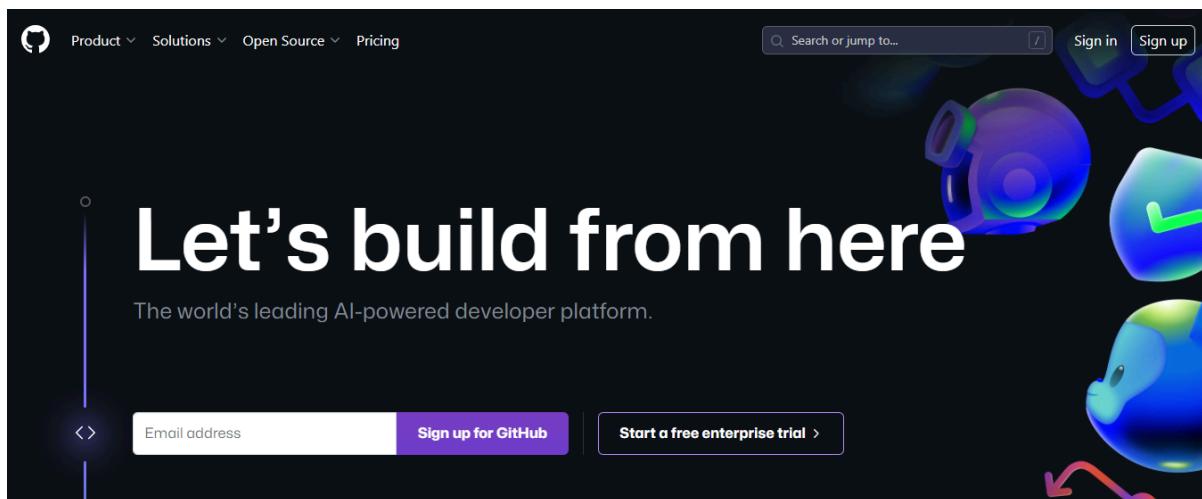


Imagen: Página web de GitHub

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/ea2.1-proyecto> y seleccionando la opción correcta:

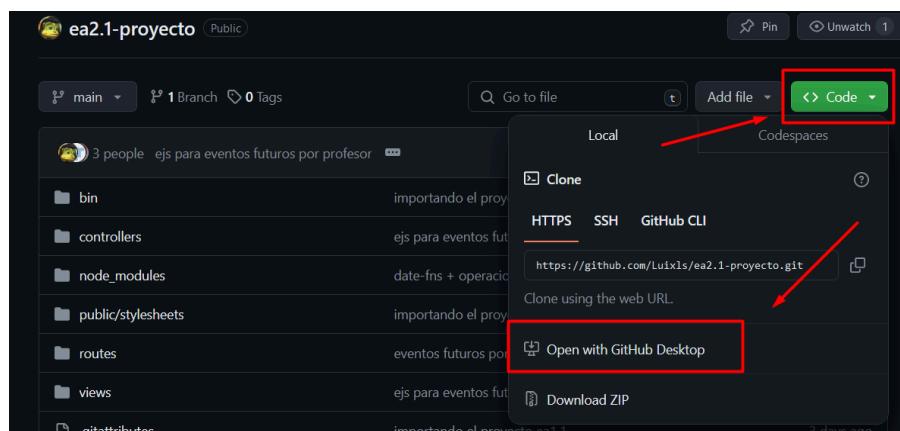


Imagen: Seleccionando la opción para obtener el repositorio con GitHub Desktop.

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 guardararlo.

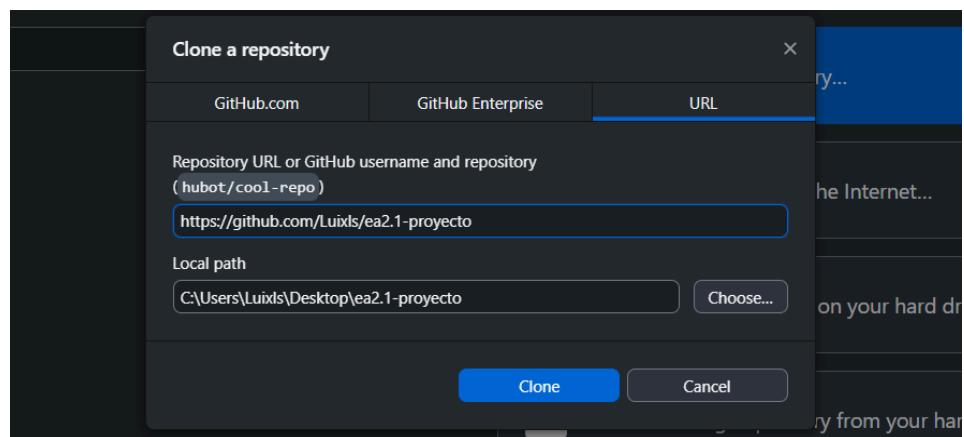


Imagen: Cuadro de confirmación de GitHub Desktop.

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.

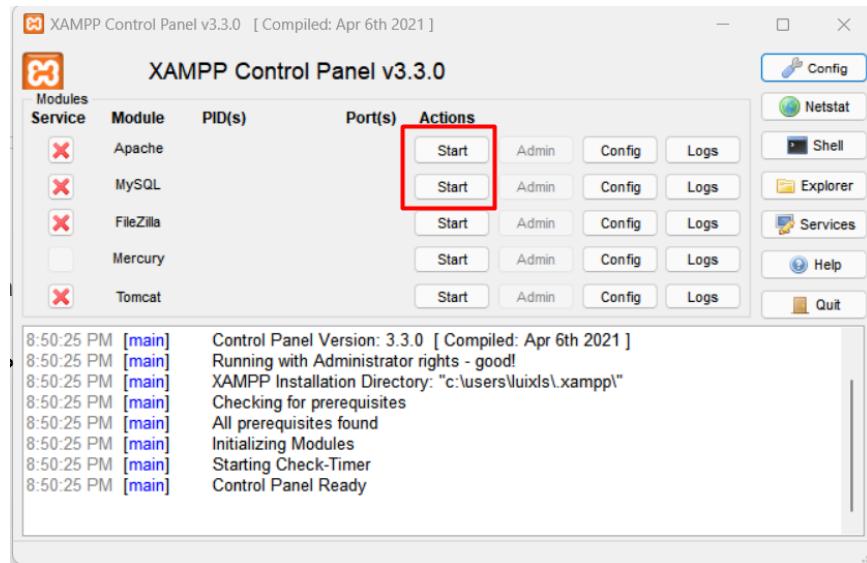


Imagen: Ventana de XAMPP, con los botones para iniciar el servidor resaltados.

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.

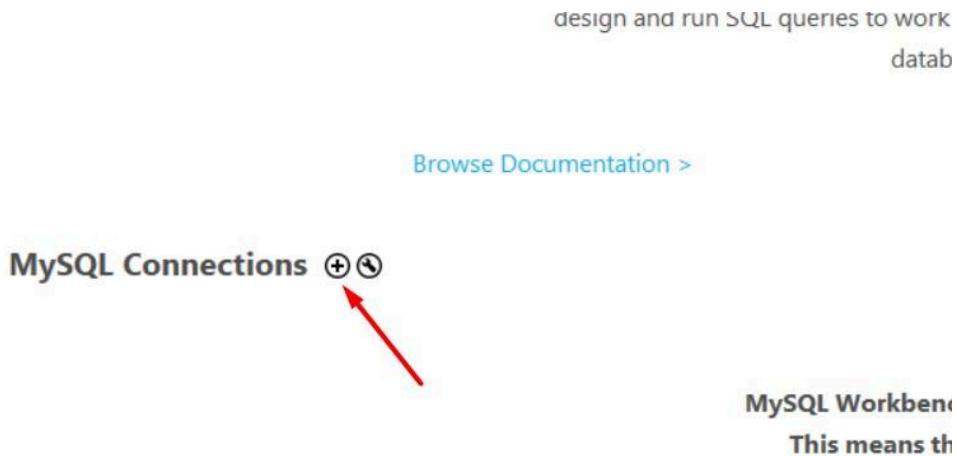


Imagen: Dentro de MySQL Workbench, botón para agregar conexión.

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.

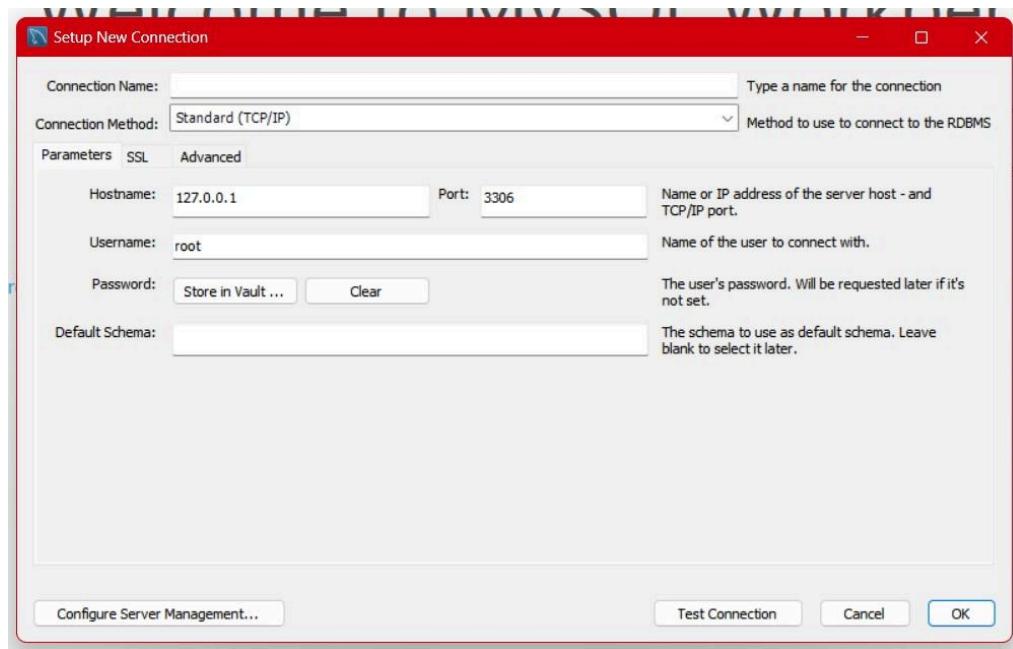


Imagen: Dentro de MySQL Workbench, ventana de configuración de conexión.

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

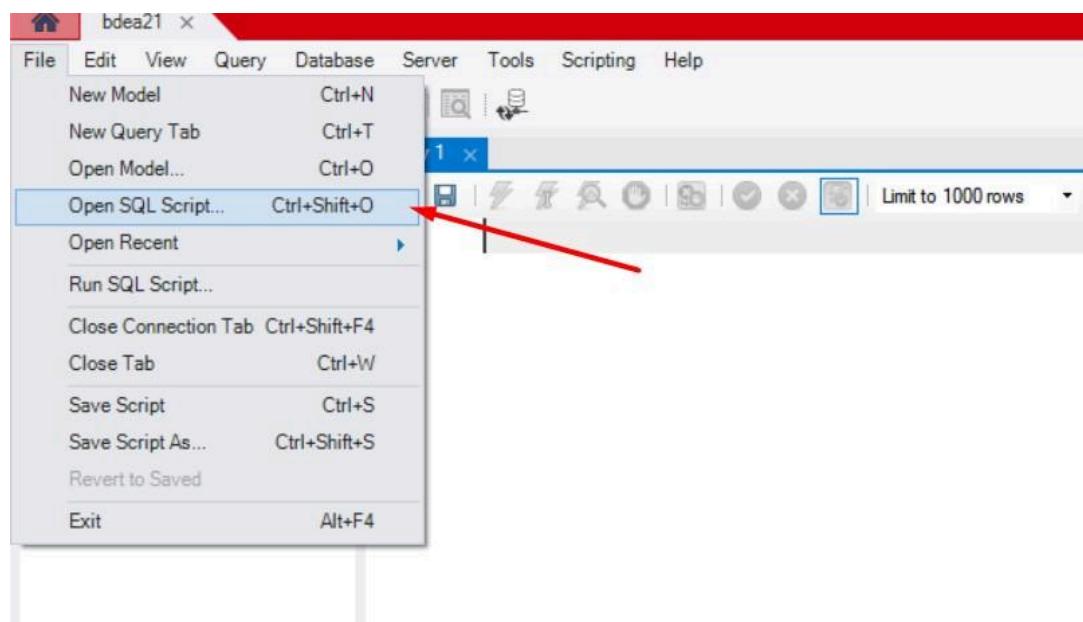
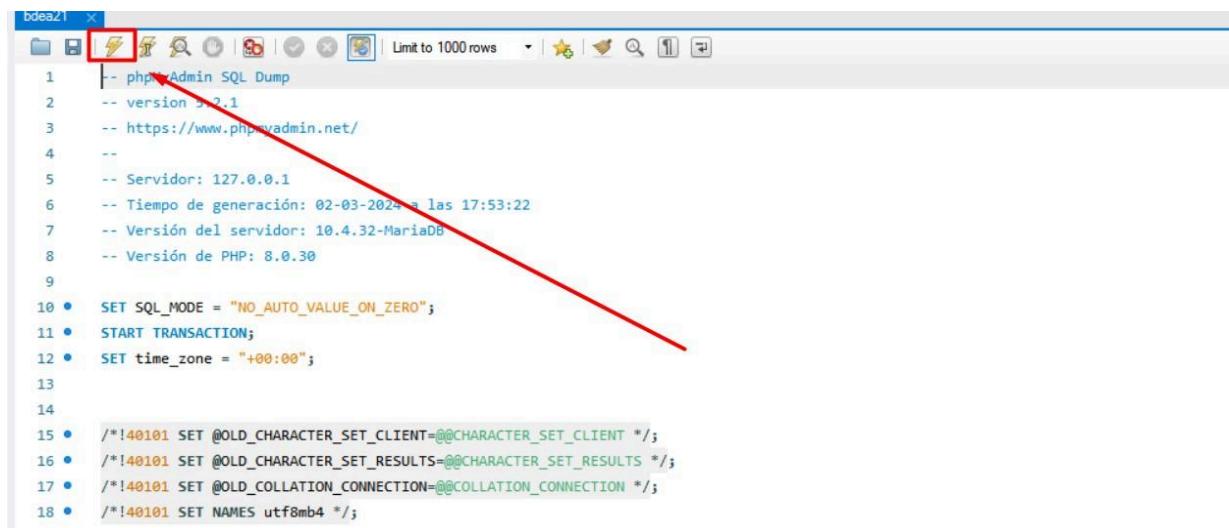


Imagen: Dentro de MySQL Workbench, opción para abrir un archivo script SQL.

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 la creación de la base de datos local.

Paso 4: Ejecutar el Servidor del SGU a través de VSCode

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

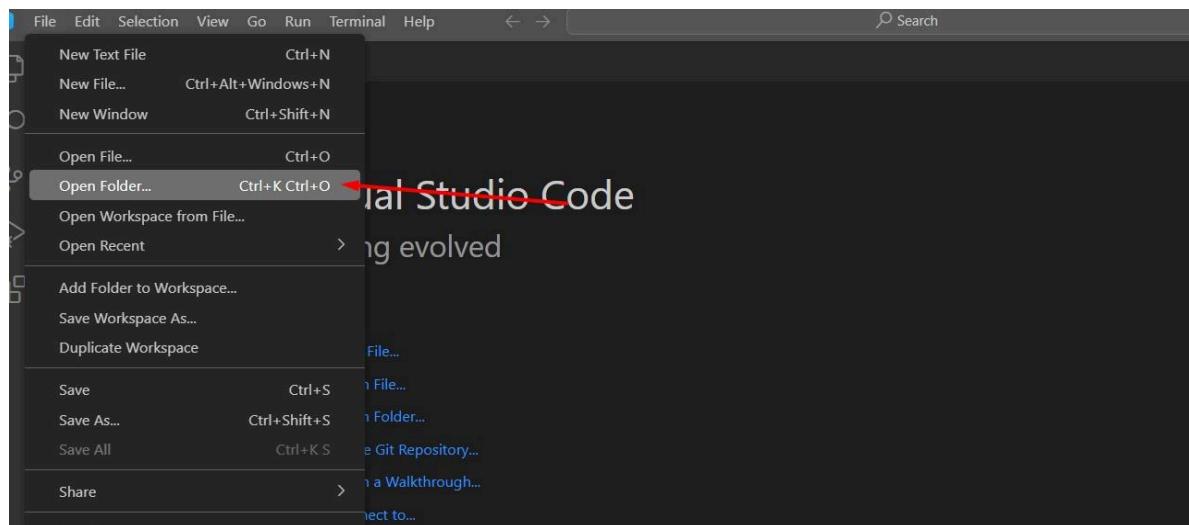
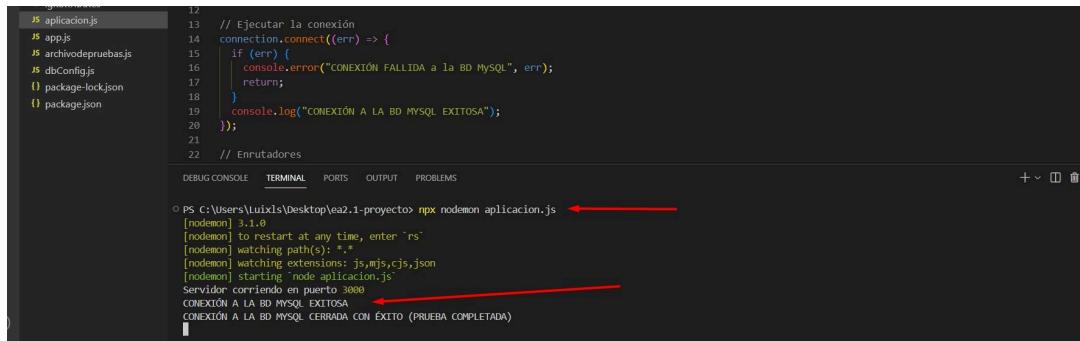


Imagen: Dentro de VSCode, seleccionando la opción de abrir carpeta.

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.



```

JS aplicacion.js
JS app.js
JS archivodepruebas.js
JS dbConfig.js
() package-lock.json
() package.json

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 start 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 EXITO (PRUEBA COMPLETADA)
  
```

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

Paso 5: Ejecutar Postman

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

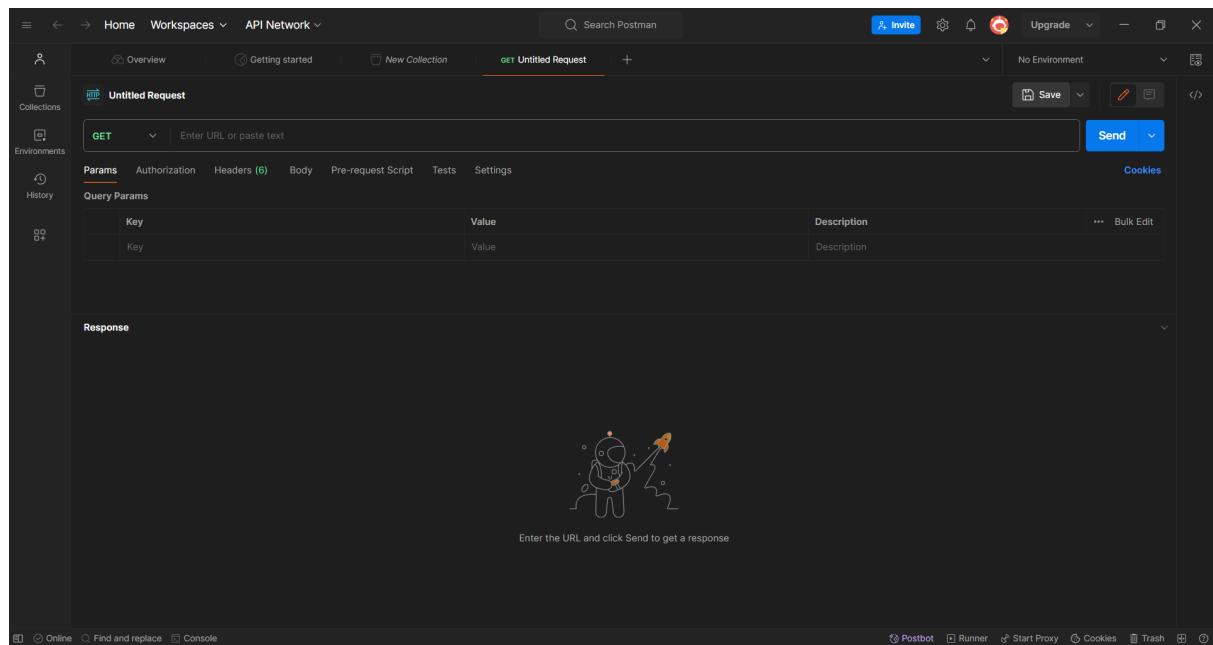


Imagen: Ventana principal de Postman, donde se ejecutarán los protocolos HTTP.

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.

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 the URL `http://localhost:3000/calendario/actividades/2024A/1`. The request method is GET. The response status is 200 OK, time 17 ms, size 3.04 KB. The response body is a table titled "Actividades de la Semana: 1 (2024-01-07 - 2024-01-13)" with the subtitle "Trimestre: 2024A". The table lists the following activities:

ID	Nombre del Evento	Fecha	Día de la Semana	Materia	Profesor	Sección
1	Clase Unidad I	2024-01-12	viernes	Matemática I	Juan Perez	VIV/FI
4	Continuación Clase Unidad I	2024-01-13	sábado	Matemática I	Juan Perez	VIV/FI
6	Clase Unidad I	2024-01-13	sábado	Física I	Jorge Álvarez	VIV/FI
9	Bienvenida al Trimestre	2024-01-09	martes	Humanitas I	Juana Paredes	VIV/FI
10	Clase Virtual Unidad I	2024-01-10	miércoles	Ingeniería de la Programación	Roberto DiMichele	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/\[IDProfesor\]/\[FechaAAAA-MM-DD\]](http://localhost:3000/eventos/profesor/[IDProfesor]/[FechaAAAA-MM-DD])

The screenshot shows a Postman request for the URL `http://localhost:3000/eventos/profesor/6/2024-01-01`. The request method is GET. The response status is 200 OK, time 13 ms, size 2.01 KB. The response body is a table titled "Eventos Futuros para: Roberto DiMichele". The table lists the following future events:

ID	Nombre del Evento	Fecha	ID Materia	Nombre Materia
10	Clase Virtual Unidad I	2024-01-10	7	Ingeniería de la Programación
11	Clase Virtual Unidad III	2024-02-09	7	Ingeniería de la Programación

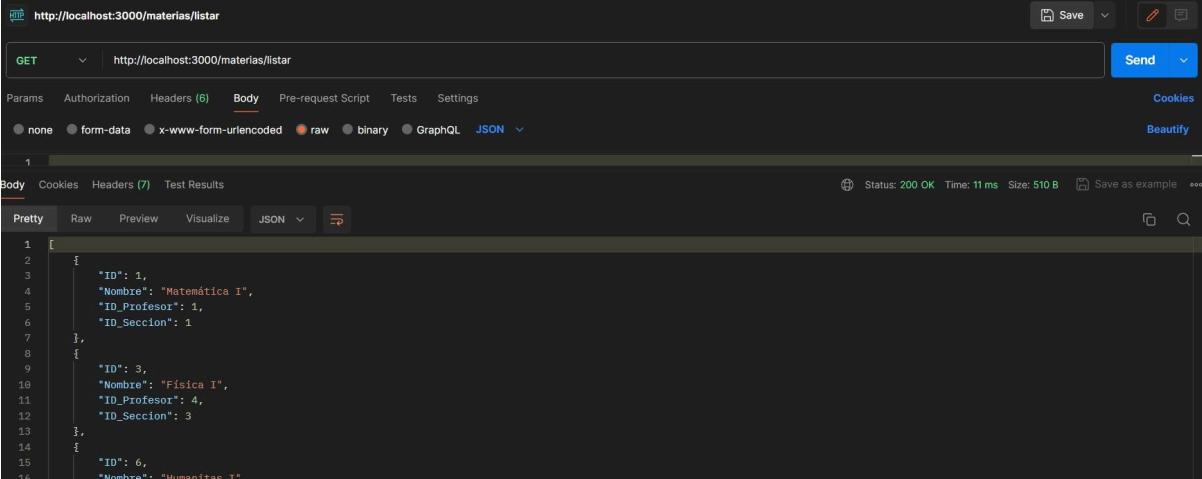
Imagen: Visualización de las actividades futuras de un profesor

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.

- **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>



```
1 [
2   {
3     "ID": 1,
4     "Nombre": "Matemática I",
5     "ID_Profesor": 1,
6     "ID_Seccion": 1
7   },
8   {
9     "ID": 3,
10    "Nombre": "Física I",
11    "ID_Profesor": 4,
12    "ID_Seccion": 3
13  },
14  {
15    "ID": 6,
16    "Nombre": "Humanitas I"
17  }
18 ]
```

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 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 Trimestres:** <http://localhost:3000/calendario/agregar>

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

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/eventos/agregar`. The request body is set to `JSON` and contains the following JSON payload:

```

1 {
2   "Nombre": "Corte de Notas",
3   "Fecha": "2024-01-26",
4   "ID_Materia": "1"
5 }

```

The response tab shows a successful `200 OK` status with a response time of `64 ms` and a size of `275 B`. The response body is:

```

1 {
2   "mensaje": "Evento agregado con éxito"
3 }

```

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 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/\[IDEvento\]](http://localhost:3000/eventos/editar/[IDEvento])

```
{  
    "Nombre": "Nuevo nombre del Evento",  
    "Fecha": "AAAA-MM-DD",  
    "ID_Materia": "aquí 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"
}
```

The screenshot shows a Postman interface with a PUT request to `http://localhost:3000/profesores/editar/6`. The request body is a JSON object with a single key-value pair: `"Nombre": "Roberto Di Michele"`. The response tab shows a status of 200 OK, a time of 23 ms, and a size of 276 B. The response body is a JSON object with a single key-value pair: `"mensaje": "Profesor editado con éxito"`.

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])

The screenshot shows a Postman interface with a DELETE request to `http://localhost:3000/calendario/eliminar/4`. The response status is 200 OK, a time of 23 ms, and a size of 279 B. The response body is a JSON object with a single key-value pair: `"mensaje": "Trimestre eliminado con éxito"`.

Imagen: Eliminando un trimestre

Créditos y Comentarios de Desarrollador

Diario del Proyecto

1. Se clonó el proyecto de la ea1.1, y se aplicaron las correcciones indicadas por el tutor.
2. Se creó una BD en MySQL, y se usó XAMPP para el servidor local, y MySQL Workbench para trabajar en la BD.
3. Se instaló el módulo de Node.js "mysql", se realizó la conexión del proyecto a la BD MySQL local, y se testeó la conexión. Por último, se desarrolló la BD.
4. Se analizaron los siguientes pasos a tomar, y se decidió que implementar funcionalidades CRUD para cada tabla se veía como lo más lógico para proseguir.
5. Se introdujo (1) entrada a cada tabla de la BD para que actuaran como datos de prueba.
6. Se implementaron funcionalidades CRUD para la tabla de profesores
7. Se tomó como base las funcionalidades CRUD para profesores para lograr desarrollar las funcionalidades CRUD para el resto de las tablas.
8. Al mismo tiempo, se trabajó en los principios necesarios para lograr la funcionalidad de "calendario de actividades" para cumplir con el enunciado. Esto implicó desarrollar un nuevo enrutador y controlador.
9. Se instaló el módulo el módulo de Node.js "date-fns" para trabajar con fechas de una forma más cómoda.
10. Se decidió desarrollar una nueva tabla para la BD "Trimestres" para guardar en el sistema los trimestres con sus nombres, fecha de inicio y fecha de culminación.
11. Se desarrollaron funcionalidades CRUD para la tabla de trimestres, se desarrolló la lógica para que el sistema calcule semanas a través de la fecha de inicio y fecha final del trimestre, y además, se desarrolló la lógica para mostrar el día de la semana de cada fecha, este paso en gran parte gracias al módulo date-fns.
12. Se desarrolló la lógica para mostrar las actividades por semana, y luego se ajustó para renderizarlo en vista ejs.
13. Se finalizaron todas las funcionalidades CRUD para todas las tablas.
14. Se agregaron estilos CSS para la vista renderizada ejs.
15. Se desarrolló de nuevo la funcionalidad del proyecto pasado para visualizar las actividades futuras de un profesor, incluyendo su renderizado en vista ejs.

Comentarios de Desarrollo

Este proyecto, aunque es una continuación del anterior, sin duda alguna hizo recordar a cada integrante la importancia de planificar por adelantado. Se decidió que los pasos principales a tomar era primero que nada establecer una conexión exitosa a la BD local, luego desarrollar la lógica para esa BD, seguido de desarrollar las funcionalidades CRUD para los ámbitos requeridos y por último, desarrollar la funcionalidad del proyecto que trata sobre el calendario de actividades.

Si se quiere hablar de las partes más exigentes o difíciles de desarrollar, lo primero que se viene a la mente trata sobre manejar fechas; más específicamente calcular inicio y fin de semanas automáticamente. De segundo, el desarrollo de lógica para formatear dichas fechas correctamente para el correcto funcionamiento entre el proyecto con la base de datos también fue uno de los aspectos más desafiantes. Por último, de tercero, se diría que la primera implementación de comandos CRUD también fue un reto a considerar, sin embargo una vez se logró desarrollar dichas funcionalidades, permitir comandos CRUD al resto del proyecto no fue tan difícil.

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/ea2.1-proyecto>

The screenshot shows the GitHub repository page for 'ea2.1-proyecto'. The main area displays a list of commits in the 'main' branch. Each commit includes the author's profile picture, the number of people involved, the commit message, and the date it was made. The commits are as follows:

- 3 people: ejjs para eventos futuros por profesor · b88f517 · 3 days ago
- importando el proyecto ea1.1 · 4 days ago
- ejs para eventos futuros por profesor · 3 days ago
- date-fns + operaciones CRUD para la tabla "trimestres" + m... · 4 days ago
- importando el proyecto ea1.1 · 4 days ago
- eventos futuros por profesor desde fecha determinada · 3 days ago
- ejs para eventos futuros por profesor · 3 days ago
- importando el proyecto ea1.1 · 4 days ago
- date-fns + operaciones CRUD para la tabla "trimestres" + m... · 4 days ago
- importando el proyecto ea1.1 · 4 days ago
- dbConfig.js para importar · 4 days ago
- pequeños fix de formato y comentarios más claros · 4 days ago
- date-fns + operaciones CRUD para la tabla "trimestres" + m... · 4 days ago
- date-fns + operaciones CRUD para la tabla "trimestres" + m... · 4 days ago

To the right of the commit list, there are sections for 'About', 'Releases', 'Packages', and 'Contributors'. The 'Contributors' section is highlighted with a red box and lists four individuals: Luixls, FN2814, denilsoncastellanos, and jonavnet. The 'About' section notes 'No description, website, or topics provided.' and shows activity statistics: 0 stars, 1 watching, and 1 fork.

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 último, 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/ea2.1-proyecto/commits/main/>

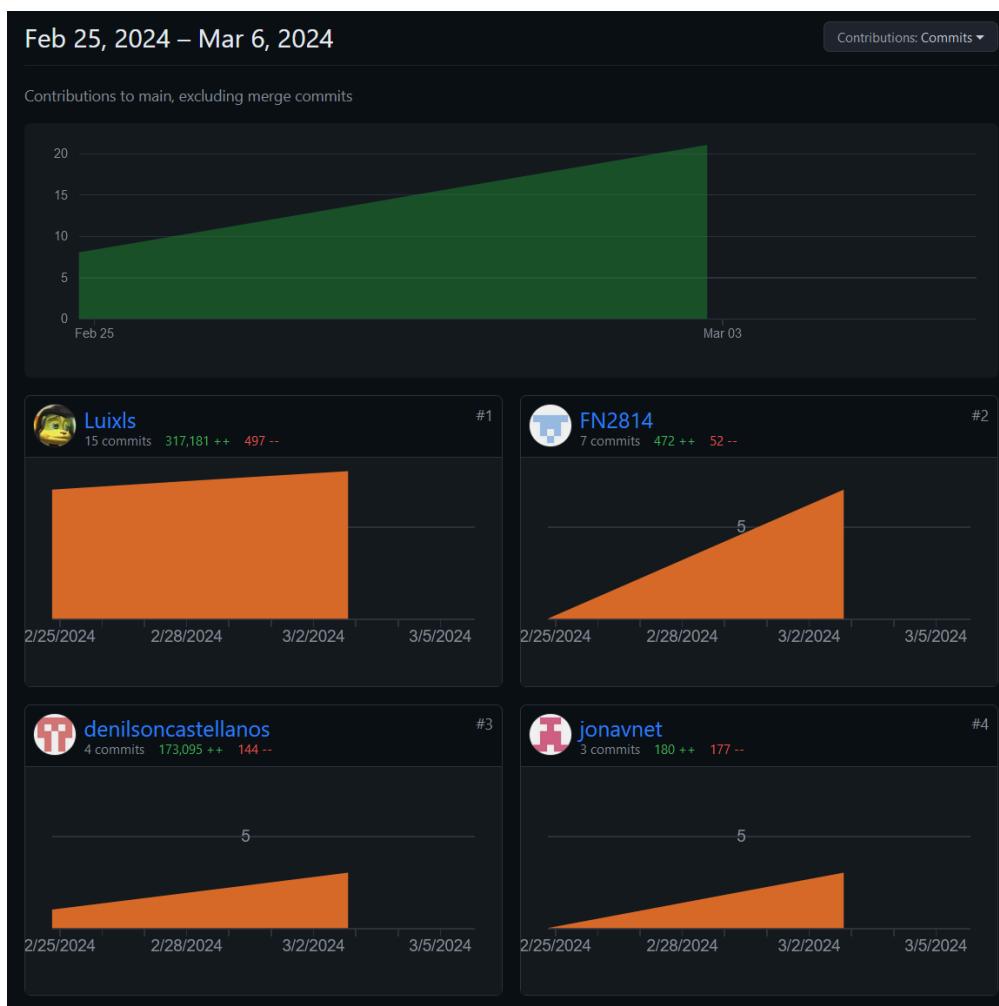


Imagen: Gráfico de participación de los integrantes.