Documentación completa del proyecto TaeMoi App

Taemoi Proyecto Fin de Grado

Carlos Sánchez Román DAW2 Mañana



Versión: v0.2

Fecha: 15/04/2024

Curso: 2023/2024

HOJA DE CONTROL

|  |  |  |  |
| --- | --- | --- | --- |
| Título | Taemoi Proyecto Final de Grado. | |  |
| Entregable | Documentación completa del proyecto. | |  |
| Nombre del Fichero | 2023-2024-IES ALIXAR-DAW2-Carlos Sánchez Román-Taemoi.pdf | |  |
| Autor | Carlos Sánchez Román | |  |
| Versión/Edición | v02r00 | Fecha Versión | 15-06-2024 |
| Aprobado por |  | Fecha Aprobación | DD/MM/AAAA |
|  |  | Nº Total Páginas | 07 |

REGISTRO DE CAMBIOS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Causa del Cambio | Responsable del Cambio | Área | Fecha del Cambio |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

ÍNDICE

Contenido

[1 Introducción. 4](#_Toc169383281)

[2 Descripción del proyecto. 5](#_Toc169383282)

[3 Identificación de las necesidades del proyecto. 6](#_Toc169383283)

[4 Breve análisis/comparativa con las alternativas del mercado. 9](#_Toc169383284)

[5 Stack tecnológico utilizado. 10](#_Toc169383285)

[6 Modelo de Datos. 11](#_Toc169383286)

[7 Prototipo de la Aplicación Web. 14](#_Toc169383287)

[8 Definición API REST publicación de servicios. 17](#_Toc169383288)

[9 Manual de despliegue. 18](#_Toc169383289)

[9.1 Requisitos previos. 18](#_Toc169383290)

[9.2 Despliegue en local. 18](#_Toc169383291)

[9.3 Despliegue en Docker. 33](#_Toc169383292)

[1. FROM mysql:8.0 34](#_Toc169383293)

[2. COPY ./init-db.sh /docker-entrypoint-initdb.d/ 34](#_Toc169383294)

[3. RUN chmod +x /docker-entrypoint-initdb.d/init-db.sh 34](#_Toc169383295)

[4. EXPOSE 3306 34](#_Toc169383296)

[10 Conclusiones y Postmortem. 39](#_Toc169383297)

# Introducción.

La aplicación TaeMoi es una plataforma diseñada para la administración y gestión de información en la escuela de taekwondo "MOISKIMDO TAEKWONDO" en Umbrete. Su objetivo principal es simplificar y optimizar la gestión de datos relacionados con alumnos, eventos y horarios. TaeMoi ofrece una solución integral que incluye un escaparate web accesible al público, donde se puede encontrar información relevante sobre la escuela, así como una sección de inicio de sesión para alumnos y gestores. Los gestores pueden administrar de manera eficiente la información de los alumnos y las actividades de la escuela, mientras que los alumnos pueden consultar sus grupos y horarios de clases.

# Descripción del proyecto.

La aplicación TaeMoi es una plataforma diseñada para administrar y gestionar información relacionada con alumnos, eventos y horarios en la escuela de taekwondo "MOISKIMDO TAEKWONDO" en Umbrete. Desde el seguimiento de datos personales hasta el registro de información relevante, ofrece una solución completa y eficiente para simplificar la gestión de los componentes que forman el negocio.

Además de su funcionalidad principal de gestión interna, TaeMoi también cuenta con un escaparate web accesible al público estándar. Este escaparate ofrece información relevante, como eventos, horarios de clases, y otras noticias relacionadas con la escuela de taekwondo. Así, cualquier persona interesada en la escuela puede acceder a esta información de forma fácil y rápida.

Por otro lado, la plataforma incluye una parte de inicio de sesión reservada para los usuarios gestores del negocio y otra para usuarios que sean alumnos de la escuela. En el caso de la zona protegida para gestores y administradores, los mismos pueden acceder a todas las funcionalidades avanzadas de la aplicación, como la gestión de alumnos, entre otros aspectos fundamentales para la administración eficiente del negocio. En la otra perspectiva, los alumnos pueden iniciar sesión para consultar los grupos a los que pertenecen y los turnos a los que están asociados esos grupos, permitiéndoles un acceso rápido a la consulta de las clases en las que participan.

# Identificación de las necesidades del proyecto.

Para pasar a este apartado, primero vamos a definir el flujo que va a seguir la aplicación para dejar más claras las acciones que se identificarán. El flujo para los distintos roles de la aplicación será el siguiente:

1. Usuario anónimo.

En caso de no tener acceso a la plataforma de gestión y ser un alumno en potencia, se trataría de un usuario anónimo, así que este sería el procedimiento que seguiría:

1. Entrar en la página web.
2. Accede a la página principal de la web donde encontrará información atractiva sobre las actividades del negocio, su localización y un menú para acceder a las distintas subpáginas donde estará la información de contacto, eventos, horario y demás.
   1. Acceso a “Logros”: Página que recoge todos los eventos en los que la escuela ha estado.
   2. Acceso a “Horarios”: Página que recoge una tabla con los horarios y turnos del negocio.
   3. Acceso a “Contáctanos”: Página que contiene un formulario de contacto con opción de pedir información tanto por whatsapp como por correo electrónico.
   4. Acceso a “Login”: Página que contiene un formulario de inicio de sesión para acceder a su información en caso de ser un usuario registrado en la plataforma.
3. Usuario alumno.

En caso de no tener acceso a la plataforma de gestión y ser un alumno dado de alta en la base de datos, se trataría de un usuario alumno, así que este sería el procedimiento que seguiría:

1. Entrar en la página web.
2. Accede a la página principal de la web donde encontrará información atractiva sobre las actividades del negocio, su localización y un menú para acceder a las distintas subpáginas donde estará la información de contacto, eventos, horario y demás.
   1. Acceso a “Logros”: Página que recoge todos los eventos en los que la escuela ha estado.
   2. Acceso a “Horarios”: Página que recoge una tabla con los horarios y turnos del negocio.
   3. Acceso a “Contáctanos”: Página que contiene un formulario de contacto con opción de pedir información tanto por whatsapp como por correo electrónico.
3. Pulsa en el enlace para acceder a la página de login.
   1. Accede a la página de login, que tiene un formulario para iniciar sesión con el usuario gestor o administrador.
   2. Introduce su usuario y contraseña.
   3. Se abre la página principal del alumno con los grupos a los que pertenece
   4. Pulsa en el grupo del que quiera consultar más información.
      1. Accede a la página de turnos del grupo, en el que hay una tabla con los turnos de su correspondiente grupo.
4. Usuario gestor y administrador.

En caso de tener acceso a la plataforma de gestión con una cuenta con permisos, se trataría de un usuario gestor u administrador de la aplicación, así que el procedimiento sería el siguiente:

* Entrar en la página web.
* Accede a la página principal de la web donde encontrará información atractiva sobre las actividades del negocio, su localización y un menú para acceder a las distintas subpáginas donde estará la información de contacto, eventos, horario y demás.
* Pulsa en el enlace para acceder a la página de login.
  + Accede a la página de login, que tiene un formulario para iniciar sesión con el usuario gestor o administrador.
  + Introduce su usuario y contraseña.
  + Se abre la plataforma de gestión de alumnos, donde puede realizar operaciones crud sobre los mismos, eventos, turnos y grupos y realizar el filtrado y la paginación de los mismos.

Una vez definido todo el flujo del proyecto, vamos a identificar los requisitos funcionales a implementar:

Por supuesto, aquí está el listado de requisitos funcionales con los cambios solicitados:

* RF01: Acceso a página principal: El usuario anónimo puede acceder a la página principal de la aplicación ingresando la URL.
* RF02: Acceso a información pública: El usuario anónimo puede acceder a información pública sobre el negocio, como descripción del negocio, eventos, horarios y datos de contacto.
* RF03: Acceso a eventos: El usuario anónimo puede acceder a una página que muestra los eventos en los que ha participado la escuela.
* RF04: Acceso a horarios: El usuario anónimo puede acceder a una página que muestra los horarios y turnos que tiene el negocio.
* RF05: Acceso a contáctanos: Página que contiene un formulario de contacto con opción de pedir información tanto por whatsapp como por correo electrónico.
* RF06: Acceso a contáctanos: Página que contiene un formulario de contacto con opción de pedir información tanto por whatsapp como por correo electrónico.
* RF07: Envío por whatsapp: Uso del api de whatsapp para solicitar información de contacto
* RF08: Envío por email: Servidor de correo interno en el api para enviar un email con la información solicitada al negocio.
* RF09: Inicio de sesión: El usuario gestor y administrador y los usuarios alumnos pueden iniciar sesión utilizando un formulario que requiere un nombre de usuario y una contraseña.
* RF10: Acceso a plataforma de gestión: Después del inicio de sesión, el usuario gestor o administrador es redirigido a la plataforma de gestión de alumnos.
* RF11: Acceso a página de alumno: Después del inicio de sesión, el usuario alumno podrá consultar sus grupos y los turnos asociados.
* RF12: Operaciones CRUD sobre alumnos: El usuario gestor puede realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los alumnos en la plataforma de gestión.
* RF13: Operaciones CRUD sobre grupos: El usuario gestor puede realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los grupos en la plataforma de gestión.
* RF14: Operaciones CRUD sobre turnos: El usuario gestor puede realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los turnos en la plataforma de gestión.
* RF15: Operaciones CRUD sobre eventos: El usuario gestor puede realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los eventos en la plataforma de gestión.
* RF16: Filtrado de alumnos para gestores: El usuario gestor puede filtrar los alumnos en la plataforma de gestión según criterios específicos (como nombre, grado o categoría).
* RF17: Paginación de alumnos para gestores: El usuario gestor puede navegar a través de múltiples páginas de resultados de alumnos en la plataforma de gestión.
* RF18: Acceso a vista administrador: Después del inicio de sesión, el usuario administrador es redirigido a la plataforma de gestión de alumnos con una vista específica para administradores.
* RF19: Operaciones CRUD para administradores: El usuario administrador puede realizar operaciones CRUD sobre los usuarios en la plataforma de gestión, además de las funciones disponibles para el usuario gestor.

# Breve análisis/comparativa con las alternativas del mercado.

Debido a que este proyecto va destinado a la mejora de calidad de la gestión interna de la empresa Moi’s Kim Do Taekwondo y la publicitación de la misma, las alternativas en el mercado actual son inexistentes, por lo que este proyecto se presenta como una innovadora solución integral y personalizada para cubrir las necesidades específicas de una escuela de taekwondo.

El mercado actual de desarrollo web para empresas de artes marciales carece de opciones especializadas que combinen tanto la presentación pública de la escuela como la gestión interna de sus procesos. Aunque existen plataformas genéricas de desarrollo web y gestión empresarial, ninguna se ha adaptado específicamente a las necesidades y particularidades de una escuela de taekwondo como lo propone TaeMoi.

Al no tener competidores directos, TaeMoi tiene la ventaja de establecerse como el primer y único proveedor de una solución integral para la gestión y promoción del negocio. Esto le otorga una posición única en el mercado, permitiéndole establecer estándares y dominar el sector.

Además, al ofrecer una plataforma personalizada que se adapta a las necesidades específicas de Moi’s Kim Do Taekwondo, TaeMoi puede generar un fuerte vínculo con sus clientes, brindando un servicio que no solo resuelve sus problemas inmediatos, sino que también les proporciona un valor añadido a largo plazo.

# Stack tecnológico utilizado.

El proyecto TaeMoi se ha desarrollado utilizando un stack tecnológico específico que se ajusta a las necesidades y características del sistema a implementar. A continuación, se describen las razones y justificaciones para la selección de cada componente del stack tecnológico:

* Backend:



* + Spring Boot : Esta tecnología se ha elegido para el desarrollo del API REST por su robustez, escalabilidad y la facilidad que ofrece para crear aplicaciones independientes y listas para producción en poco tiempo. Spring Boot simplifica la configuración de Spring, permitiendo un desarrollo rápido y eficiente.



* + Java 17 : Se empleará esta tecnología para la programación del backend por su seguridad, rendimiento y amplio soporte.



* + Hibernate : Se hará uso de esta tecnología para el mapeo de los objetos

Java a tablas de la base de datos.

* Frontend:

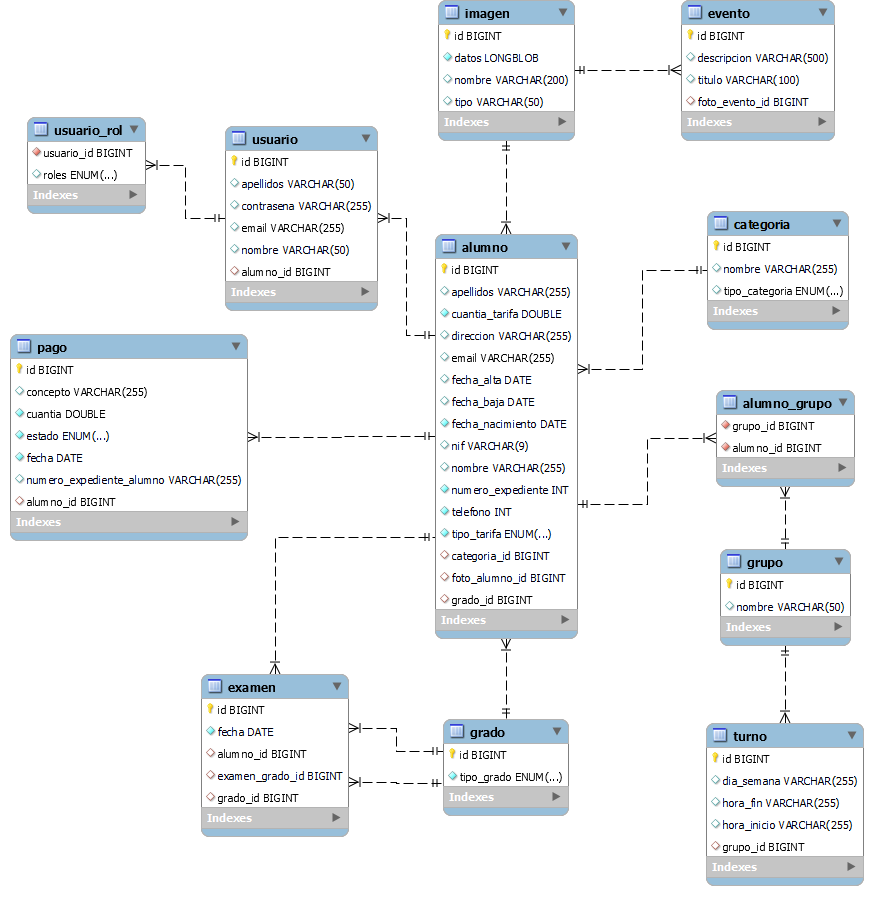


* + Angular17 : Esta tecnología se ha elegido por su arquitectura basada en componentes, que facilita la creación de interfaces de usuario dinámicas y escalables. Se utilizará para las vistas y el consumo de la api.



* + Bootstrap5 : Esta tecnología se ha elegido para el diseño del frontend por su capacidad de crear diseños responsivos y atractivos con menor esfuerzo.
  + Librerías adicionales (Leaflet y ngx-spinner): Leaflet se utiliza para la implementación de mapas interactivos, lo cual es esencial para la localización de eventos y actividades de la escuela. Ngx-spinner se usa para mostrar indicadores de carga, mejorando la experiencia del usuario al proporcionar feedback visual durante operaciones asíncronas.
* Database:
  + MYSQL : Se usará esta tecnología para la base de datos por su alto rendimiento y seguridad, crucial para mantener la integridad de los datos sensibles de la aplicación. 

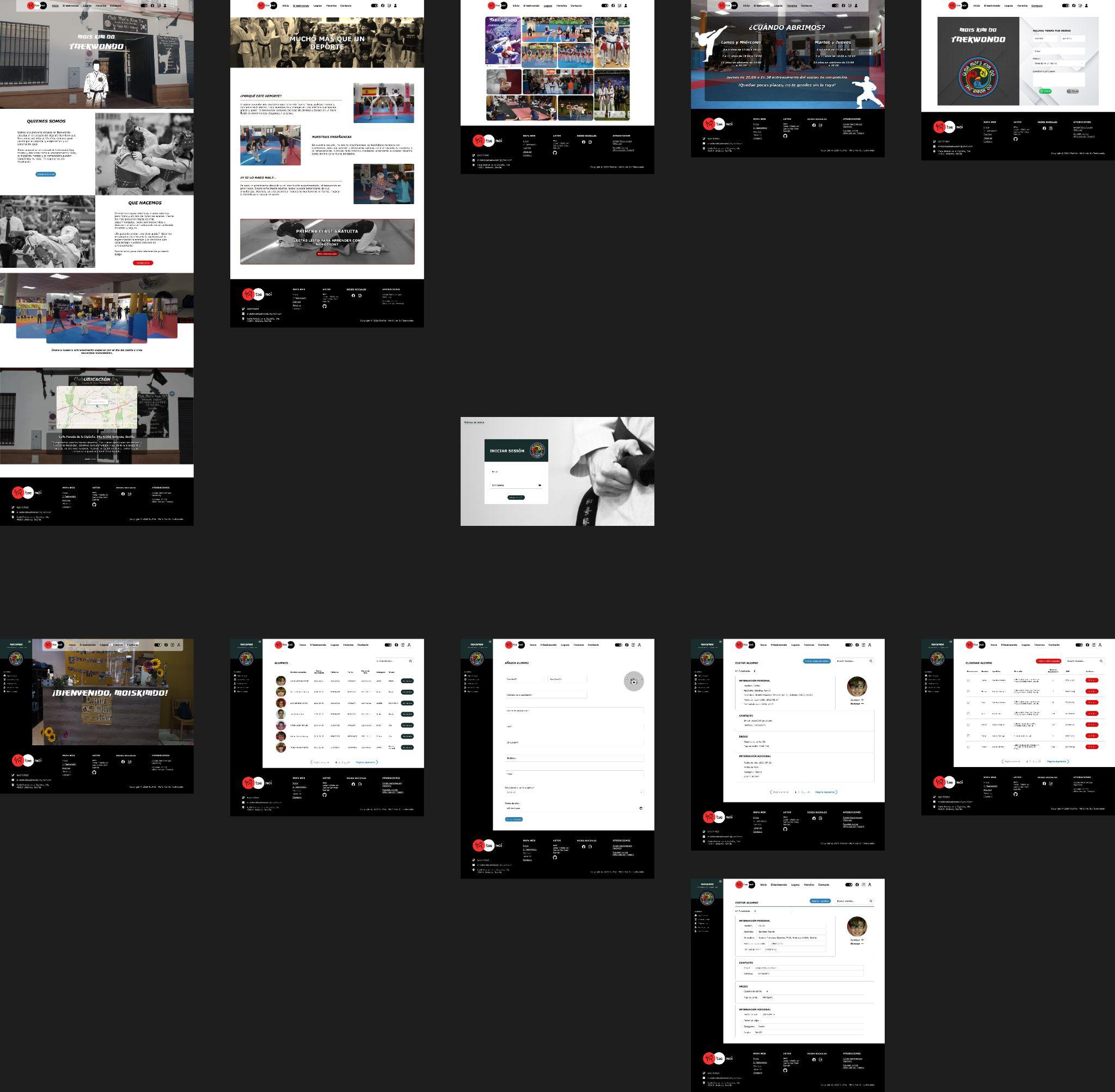
# Modelo de Datos.



* **usuario**
* **id**: Identificador único del usuario (tipo BIGINT).
* **apellidos**: Apellidos del usuario (tipo VARCHAR(50)).
* **contraseña**: Contraseña del usuario (tipo VARCHAR(255)).
* **email**: Correo electrónico del usuario (tipo VARCHAR(255)).
* **nombre**: Nombre del usuario (tipo VARCHAR(50)).
* **alumno\_id**: Identificador del alumno asociado si aplica (tipo BIGINT).
* **usuario\_rol**
* **usuario\_id**: Identificador del usuario (tipo BIGINT).
* **roles**: Roles asociados al usuario (tipo ENUM).
* **alumno**
* **id**: Identificador único del alumno (tipo BIGINT).
* **apellidos**: Apellidos del alumno (tipo VARCHAR(255)).
* **cuanta\_tarifa**: Cuota de la tarifa del alumno (tipo DOUBLE).
* **direccion**: Dirección del alumno (tipo VARCHAR(255)).
* **email**: Correo electrónico del alumno (tipo VARCHAR(255)).
* **fecha\_alta**: Fecha de alta del alumno (tipo DATE).
* **fecha\_baja**: Fecha de baja del alumno (tipo DATE).
* **fecha\_nacimiento**: Fecha de nacimiento del alumno (tipo DATE).
* **nif**: Número de Identificación Fiscal del alumno (tipo VARCHAR(9)).
* **nombre**: Nombre del alumno (tipo VARCHAR(255)).
* **numero\_expediente**: Número de expediente del alumno (tipo INT).
* **telefono**: Teléfono del alumno (tipo INT).
* **tipo\_tarifa**: Tipo de tarifa del alumno (tipo ENUM).
* **categoria\_id**: Identificador de la categoría a la que pertenece el alumno (tipo BIGINT).
* **foto\_alumno\_id**: Identificador de la foto del alumno (tipo BIGINT).
* **grado\_id**: Identificador del grado del alumno (tipo BIGINT).
* **pago**
* **id**: Identificador único del pago (tipo BIGINT).
* **concepto**: Concepto del pago (tipo VARCHAR(255)).
* **cuanta**: Cuantía del pago (tipo DOUBLE).
* **estado**: Estado del pago (tipo ENUM).
* **fecha**: Fecha del pago (tipo DATE).
* **numero\_expediente\_alumno**: Número de expediente del alumno (tipo VARCHAR(255)).
* **alumno\_id**: Identificador del alumno asociado al pago (tipo BIGINT).
* **examen**
* **id**: Identificador único del examen (tipo BIGINT).
* **fecha**: Fecha del examen (tipo DATE).
* **alumno\_id**: Identificador del alumno que tomó el examen (tipo BIGINT).
* **examen\_grado\_id**: Identificador del examen asociado al grado (tipo BIGINT).
* **grado\_id**: Identificador del grado del examen (tipo BIGINT).
* **grado**
* **id**: Identificador único del grado (tipo BIGINT).
* **tipo\_grado**: Tipo de grado (tipo ENUM).
* **imagen**
* **id**: Identificador único de la imagen (tipo BIGINT).
* **datos**: Datos binarios de la imagen (tipo LONGBLOB).
* **nombre**: Nombre de la imagen (tipo VARCHAR(200)).
* **tipo**: Tipo de imagen (tipo VARCHAR(50)).
* **evento**
* **id**: Identificador único del evento (tipo BIGINT).
* **descripcion**: Descripción del evento (tipo VARCHAR(500)).
* **titulo**: Título del evento (tipo VARCHAR(100)).
* **foto\_evento\_id**: Identificador de la foto asociada al evento (tipo BIGINT).
* **categoria**
* **id**: Identificador único de la categoría (tipo BIGINT).
* **nombre**: Nombre de la categoría (tipo VARCHAR(255)).
* **tipo\_categoria**: Tipo de categoría (tipo ENUM).
* **alumno\_grupo**
* **grupo\_id**: Identificador del grupo (tipo BIGINT).
* **alumno\_id**: Identificador del alumno (tipo BIGINT).
* **grupo**
* **id**: Identificador único del grupo (tipo BIGINT).
* **nombre**: Nombre del grupo (tipo VARCHAR(50)).
* **turno**
* **id**: Identificador único del turno (tipo BIGINT).
* **dia\_semana**: Día de la semana en el que se lleva a cabo el turno (tipo VARCHAR(255)).
* **hora\_fin**: Hora de finalización del turno (tipo VARCHAR(255)).
* **hora\_inicio**: Hora de inicio del turno (tipo VARCHAR(255)).
* **grupo\_id**: Identificador del grupo al que pertenece el turno (tipo BIGINT).

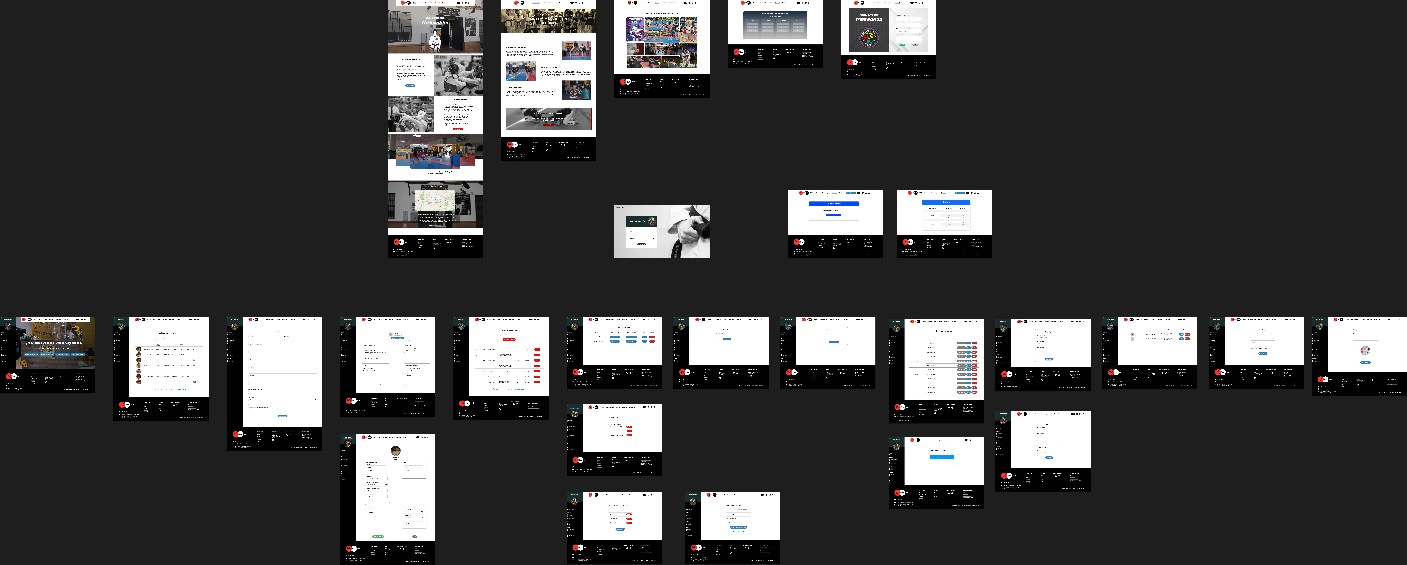
# 

# Prototipo de la Aplicación Web.



(Nota: Esta es la primera versión del prototipo, en el desarrollo del front se han creado nuevas vistas y ajustado diseños mostrados en la captura, pero por falta de tiempo no se ha podido actualizar el prototipo a una versión final.)

Este es el diseño final de figma:



El diseño se constituye por las siguientes pantallas:

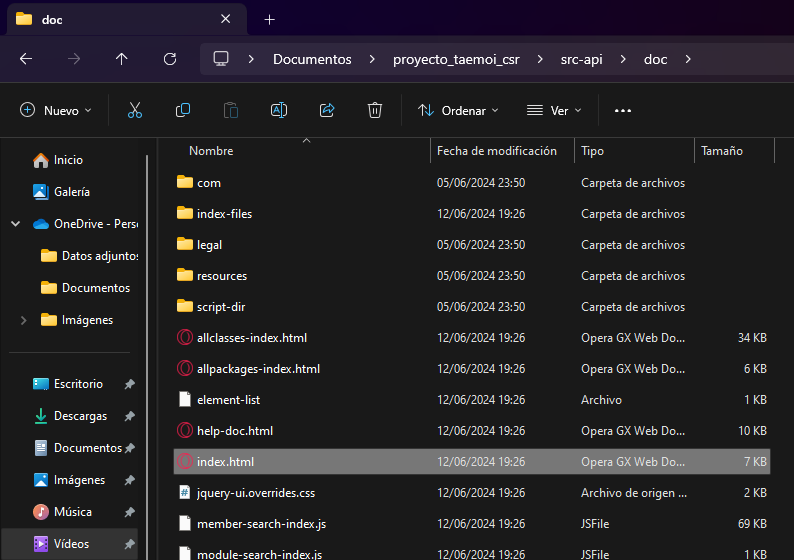
* Escaparate web: El escaparate principal de la aplicación con página principal, página con información sobre el deporte y lo que imparte el negocio, otra con eventos y logros, la de horarios y la página de contacto.
  + Página principal: Información de la escuela, imágenes atractivas, banner con imágenes y mapa con la ubicación.
  + El taekwondo: Valores y enseñanzas del negocio.
  + Logros: Galería de imágenes con los eventos a los que se han asistido o se vienen próximamente, con título y descripción para cada evento.
  + Horarios: Tabla con los horarios de las clases del negocio.
  + Contacto: Formulario de contacto para solicitar más información por whatsapp o correo electrónico.
  + Login: Formulario de login para iniciar sesión con usuario alumno dado de alta o administrador.
* Páginas de usuario: Información que puede consultar un alumno dado de alta, como los grupos a los que pertenece y los turnos que tiene ese grupo.
  + Grupos: Tabla con tarjetas que representan los grupos a los que pertenece el alumno.
  + Turnos: Tabla tipo horario con los turnos que tiene el grupo al que pertenece dicho alumno.
* Páginas de gestor y administrador: Panel de administración con toda la lógica del negocio para realizar las acciones de gestión interna de la empresa.
  + Página principal: Página con un mensaje de bienvenida al usuario que inicia la sesión y botones para acceder a los distintos apartados de gestión.
  + Listado de alumnos: Tabla con un listado de alumnos completo, paginación y filtrado por nombre.
  + Editar alumnos: Tarjetas con la información de cada alumno listados de uno en uno con filtro por nombre y cambio de vista a vista edición con formulario de edición del alumno.
  + Eliminar alumnos: Tabla con listado paginado de alumnos y filtro eliminación individual de alumnos o múltiple con selección.
  + Consultar grupos: Listado de los grupos con botones para eliminar turnos del grupo, añadir o eliminar alumnos del grupo, editar el grupo y eliminarlo.
  + Crear grupos: Formulario para crear un nuevo grupo.
  + Consultar turnos: Listado de los turnos en formato de horario con botones para añadir turnos a un grupo, editar el turno y eliminarlo.
  + Crear turnos: Formulario para crear un nuevo turno y posibilidad de añadirlo a un grupo.
  + Consultar eventos: Listado de todos los eventos con opción para editar el evento y cambiar su imagen o eliminarla y eliminar el evento.
  + Crear eventos: Formulario para crear un evento y añadirle una imagen de forma opcional.

# Definición API REST publicación de servicios.

En este enlace puedes acceder a la documentación completa de Postman con explicaciones del funcionamiento de todos los endpoints y peticiones de prueba:

* <https://documenter.getpostman.com/view/32188944/2sA2rAxgd9>

Y en la carpeta src-api/doc ejecutando el fichero index.html se encuentran todos los métodos comentados con Javadoc.



# Manual de despliegue.

En este apartado procedemos a explicar el despliegue de la aplicación tanto en local, Docker y Amazon Web Services, por lo que vamos a ello:

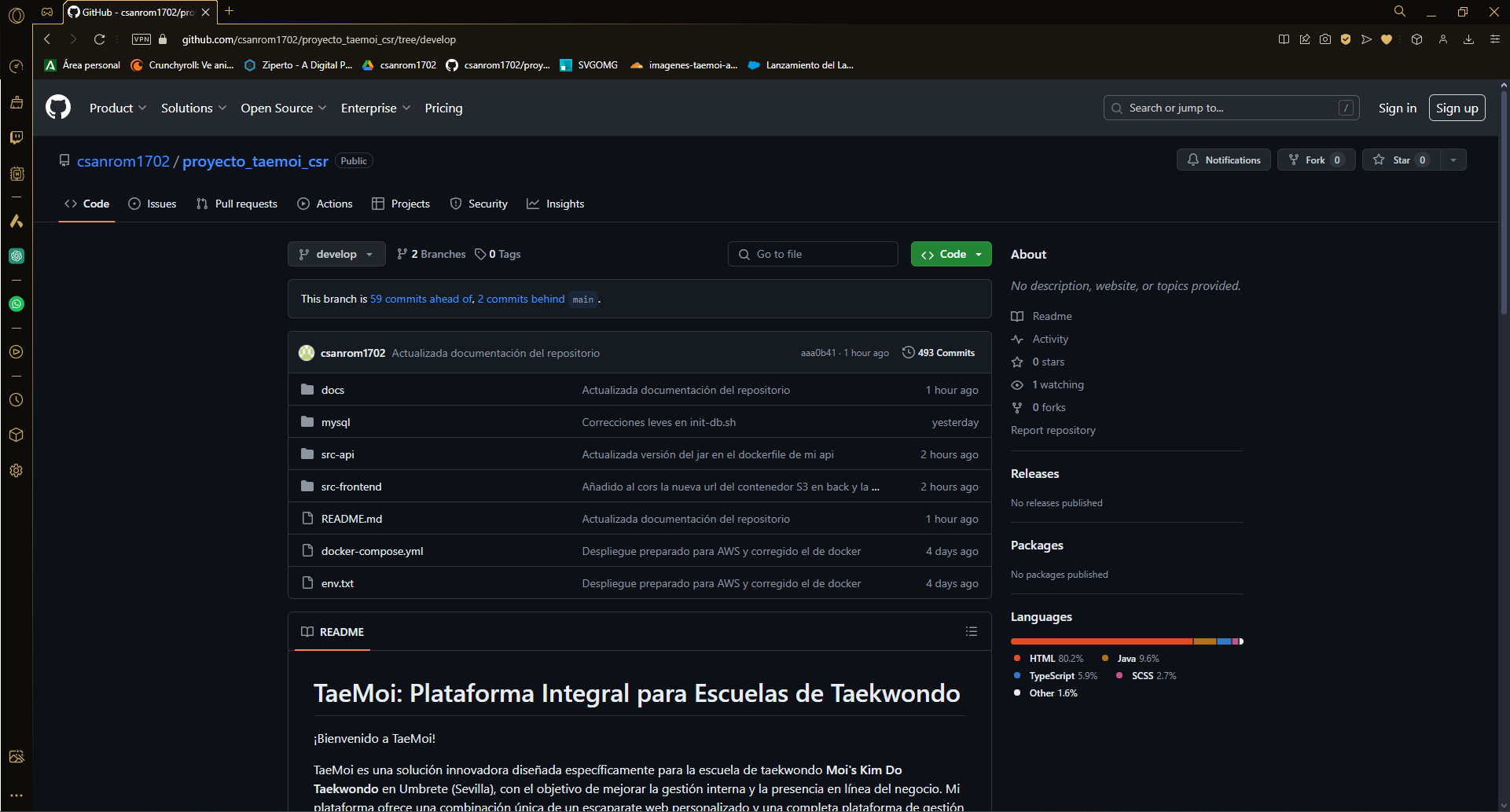
# Requisitos previos.

Antes de empezar con los despliegues, necesitamos tener las siguientes herramientas:

* Git
* Docker Desktop (Windows) o Docker Engine (Linux).
* Npm
* Java 17 y el JAVA\_HOME correctamente configurado.
* MySQL 8.0.36 por preferencia, aunque cualquier versión a partir de la 8 sirve.

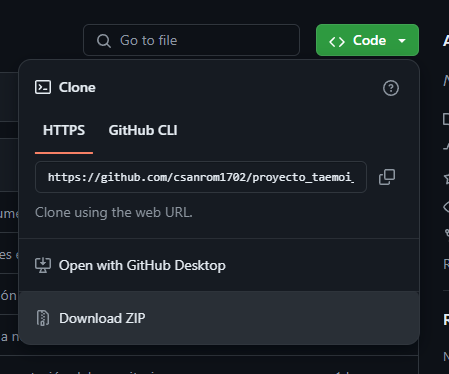
# Despliegue en local.

Para realizar el despliegue en local, lo primero será descargar el proyecto. Para ello entramos a mi repositorio desde este enlace: <https://github.com/csanrom1702/proyecto_taemoi_csr>.

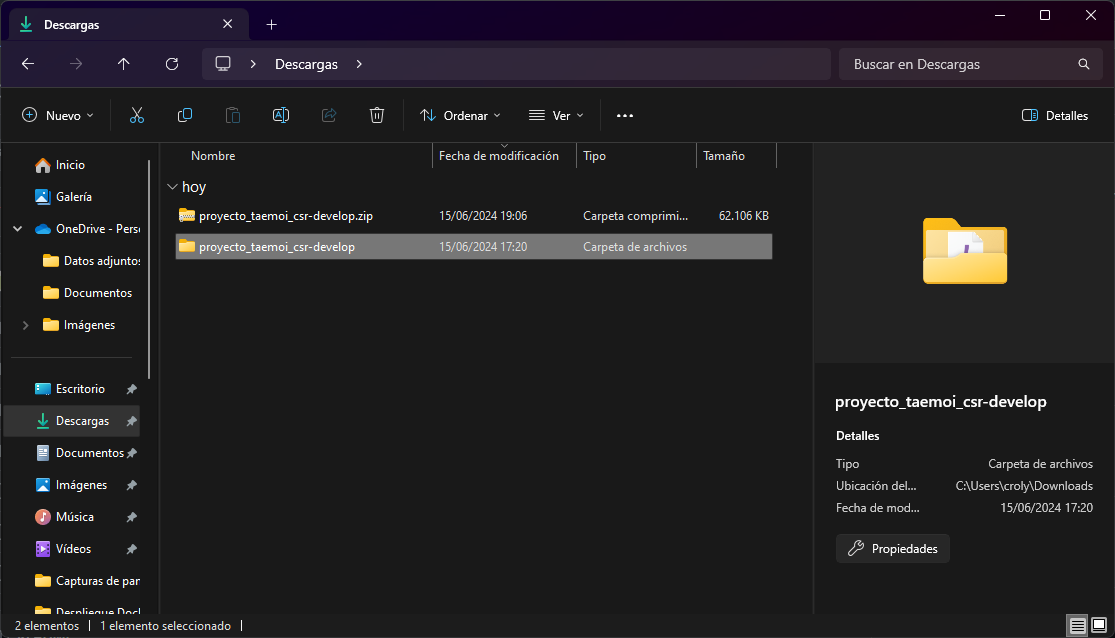




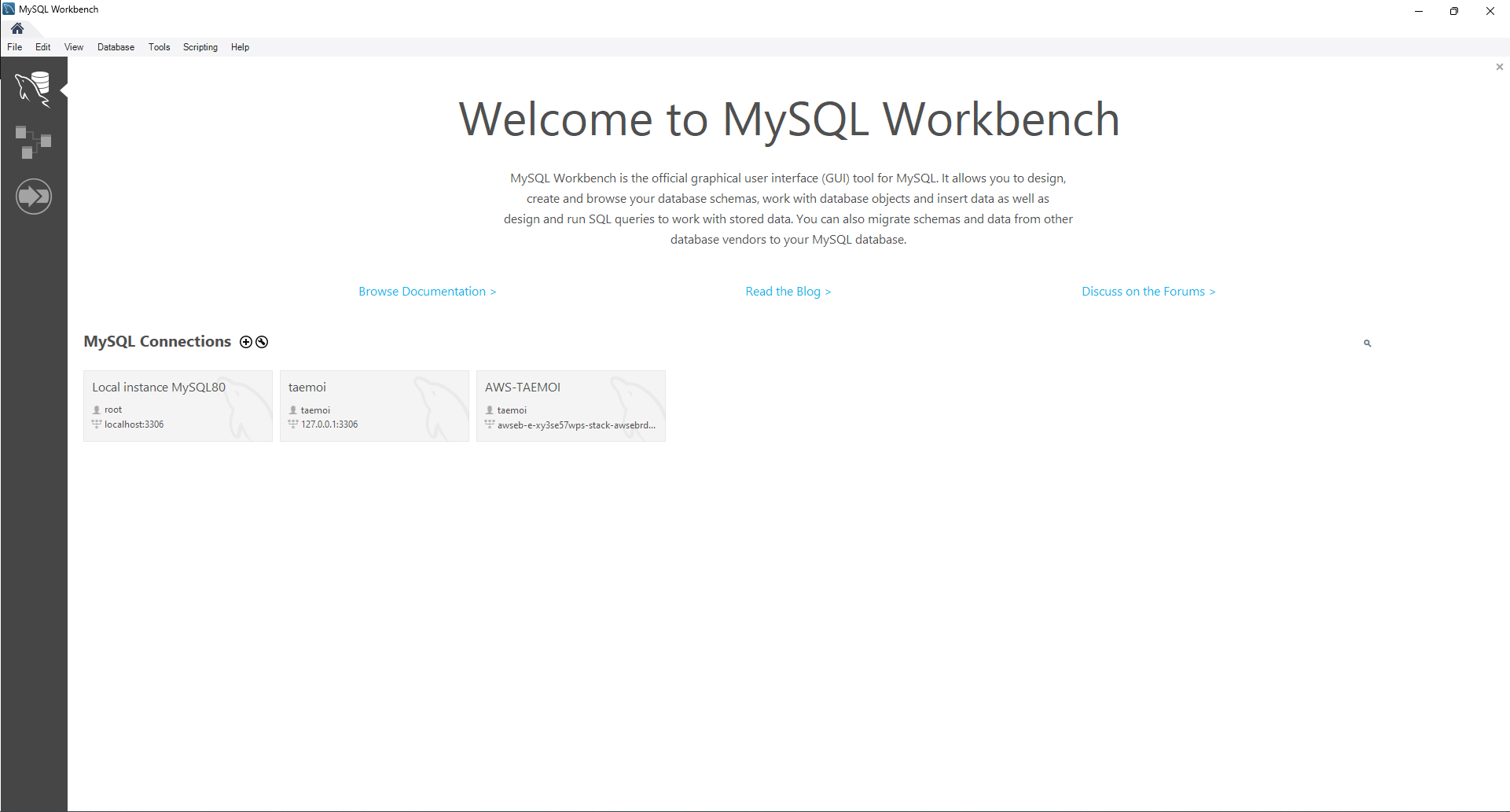
Pulsamos en este icono y le damos a la opción de “Download Zip”.



Abrimos nuestro explorador de archivos y extraemos el proyecto con la herramienta nativa de Windows o cualquier otra herramienta para descomprimir ficheros. En mi caso usaré Winrar.

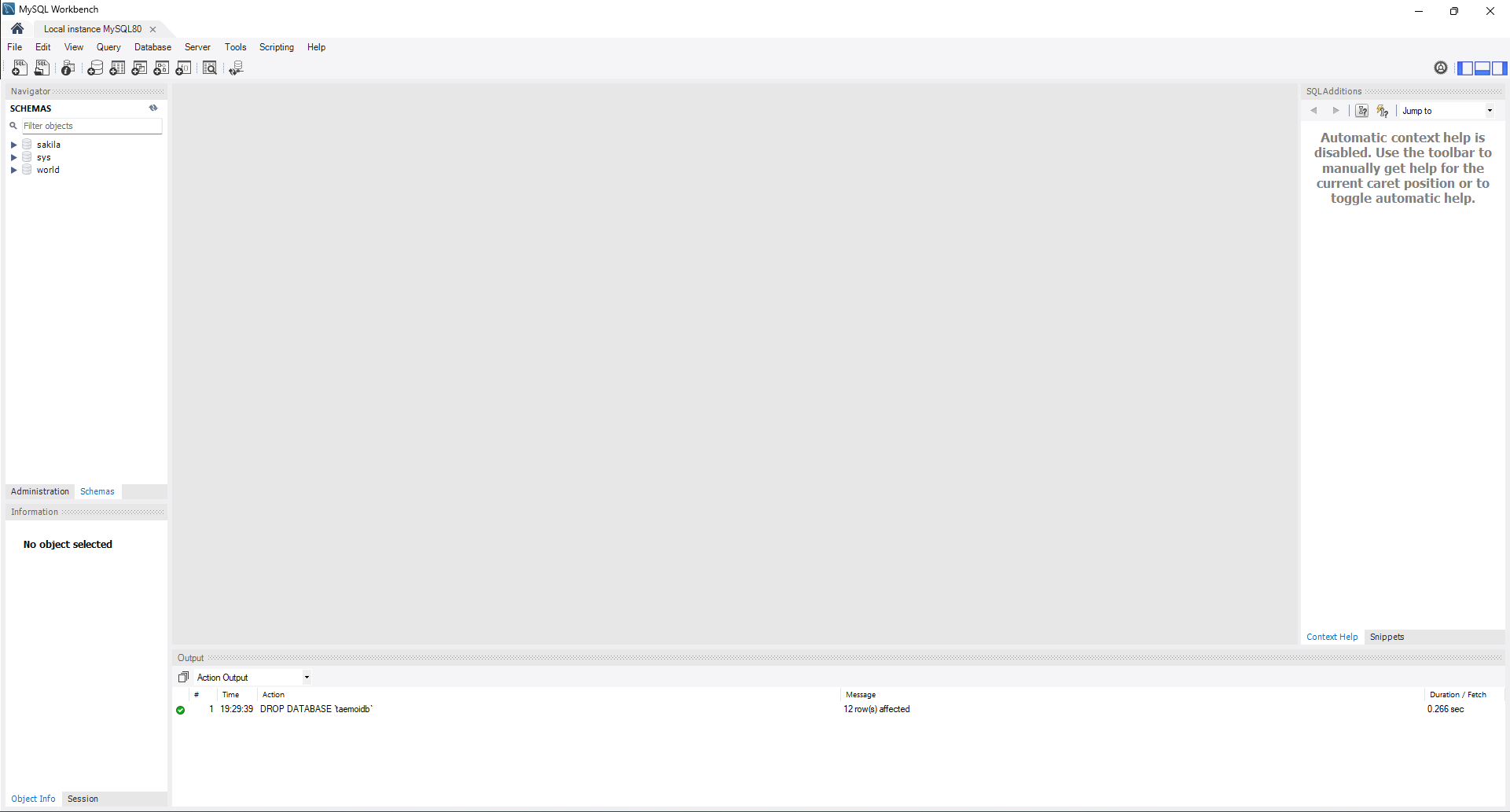


Abrimos MySQL Workbench (también se puede desde la terminal de MySQL)

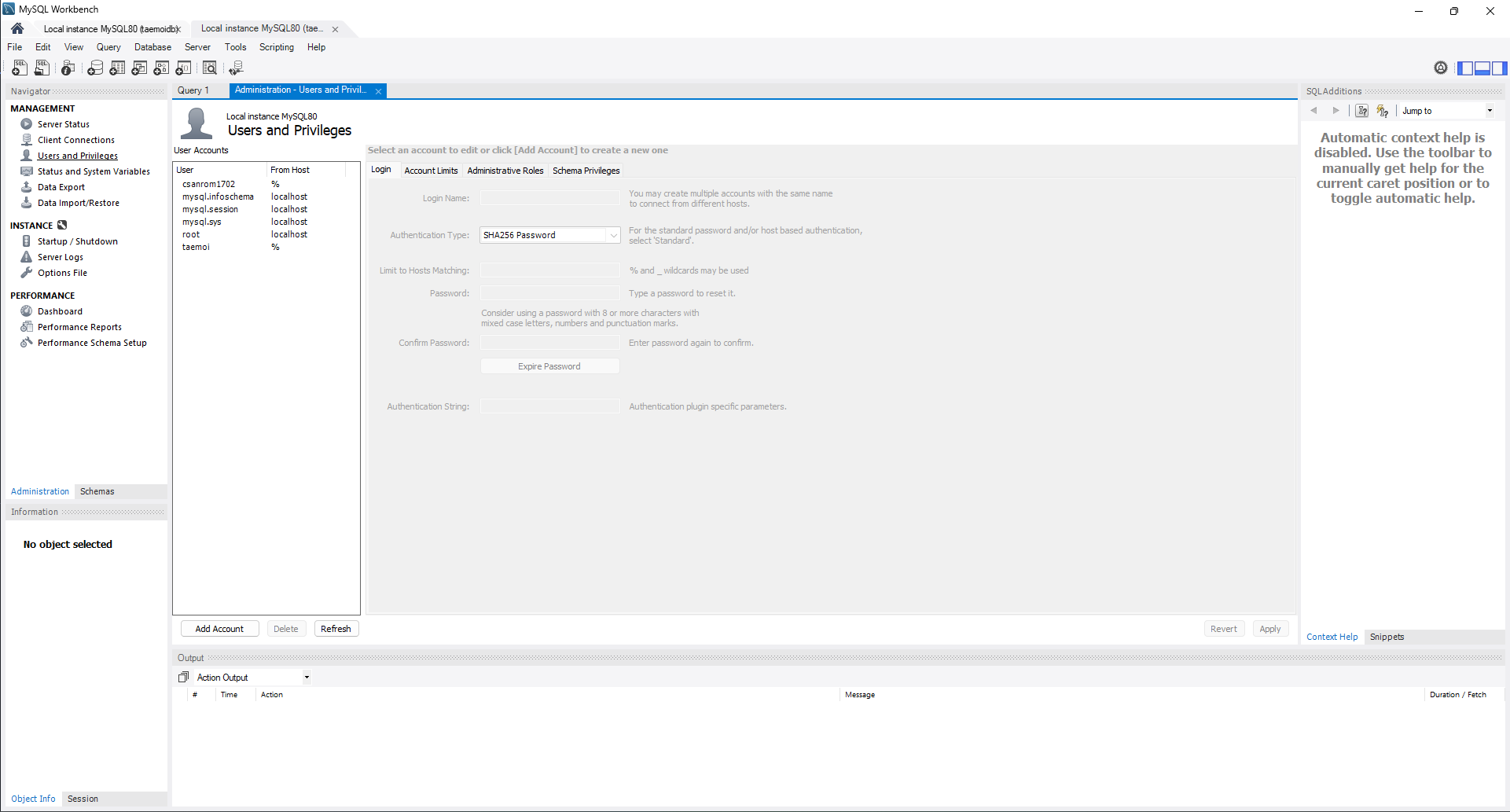


(Los pasos de crear un usuario se puede saltar si ya tienes un usuario con cuenta de permisos totales sobre cualquier base de datos que tenga una contraseña de mínimo 8 caracteres).

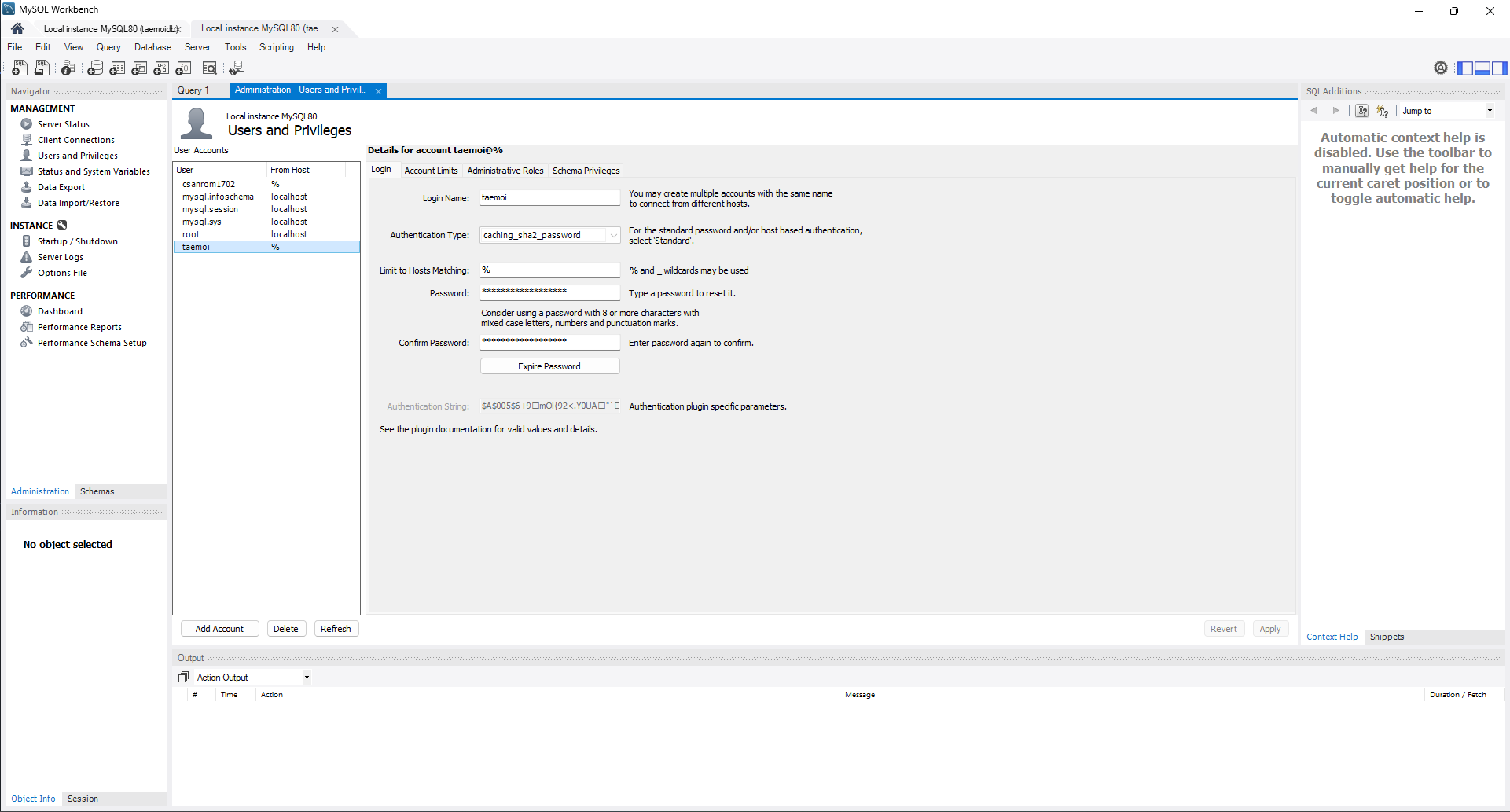
Abrimos la instancia local con root.



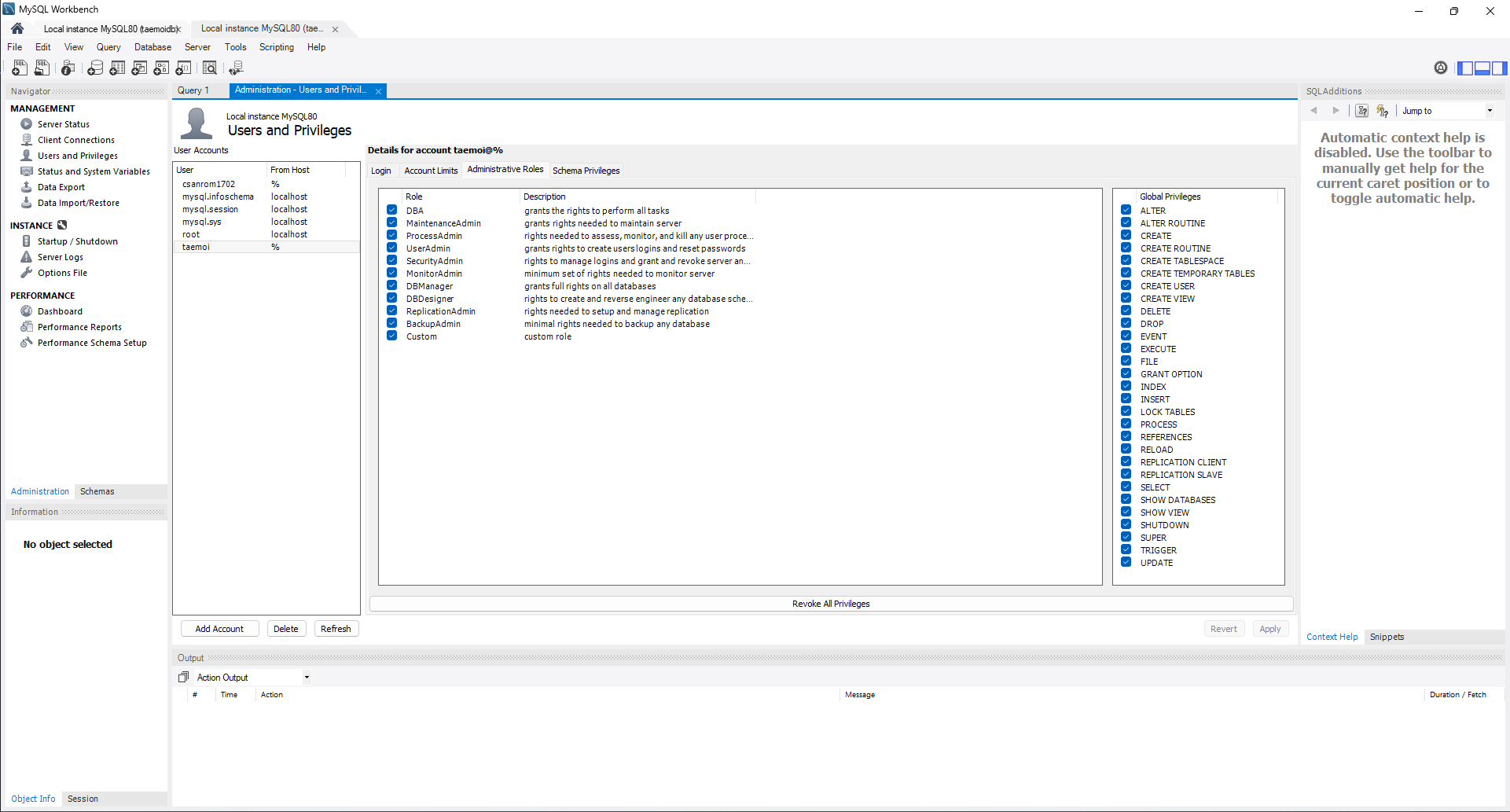
Pulsamos en Administration y le damos a Users and Privileges



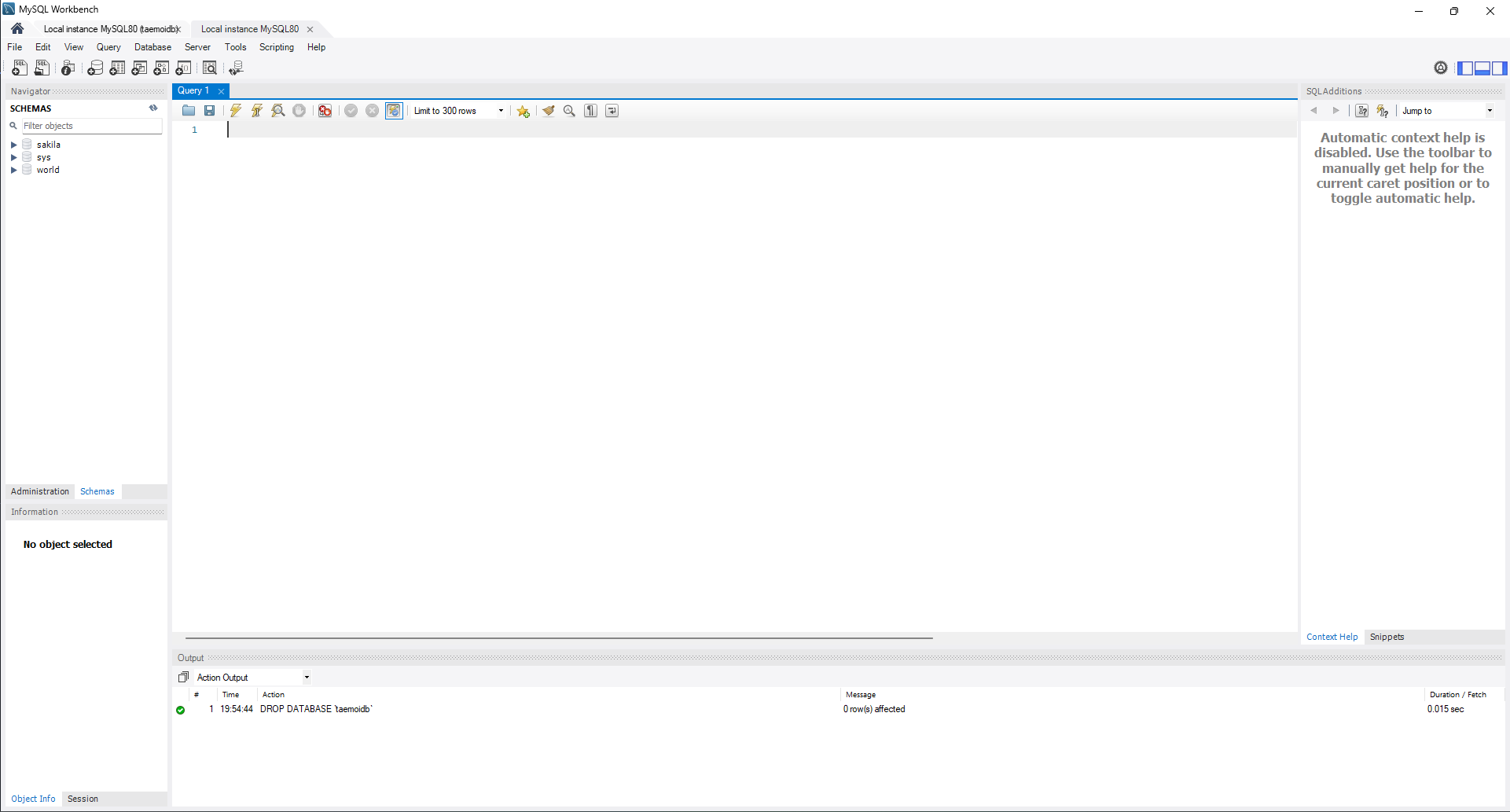
Pulsamos en Add Account y creamos un usuario con una contraseña de mínimo 8 caracteres (**IMPORTANTE PARA EL DESPLIEGUE EN AWS**). En mi caso se va a llamar taemoi.



Le damos a la pestaña “Administrative Roles” y le damos todos los permisos posibles.



Aplicamos y volvemos a “Schemas”.





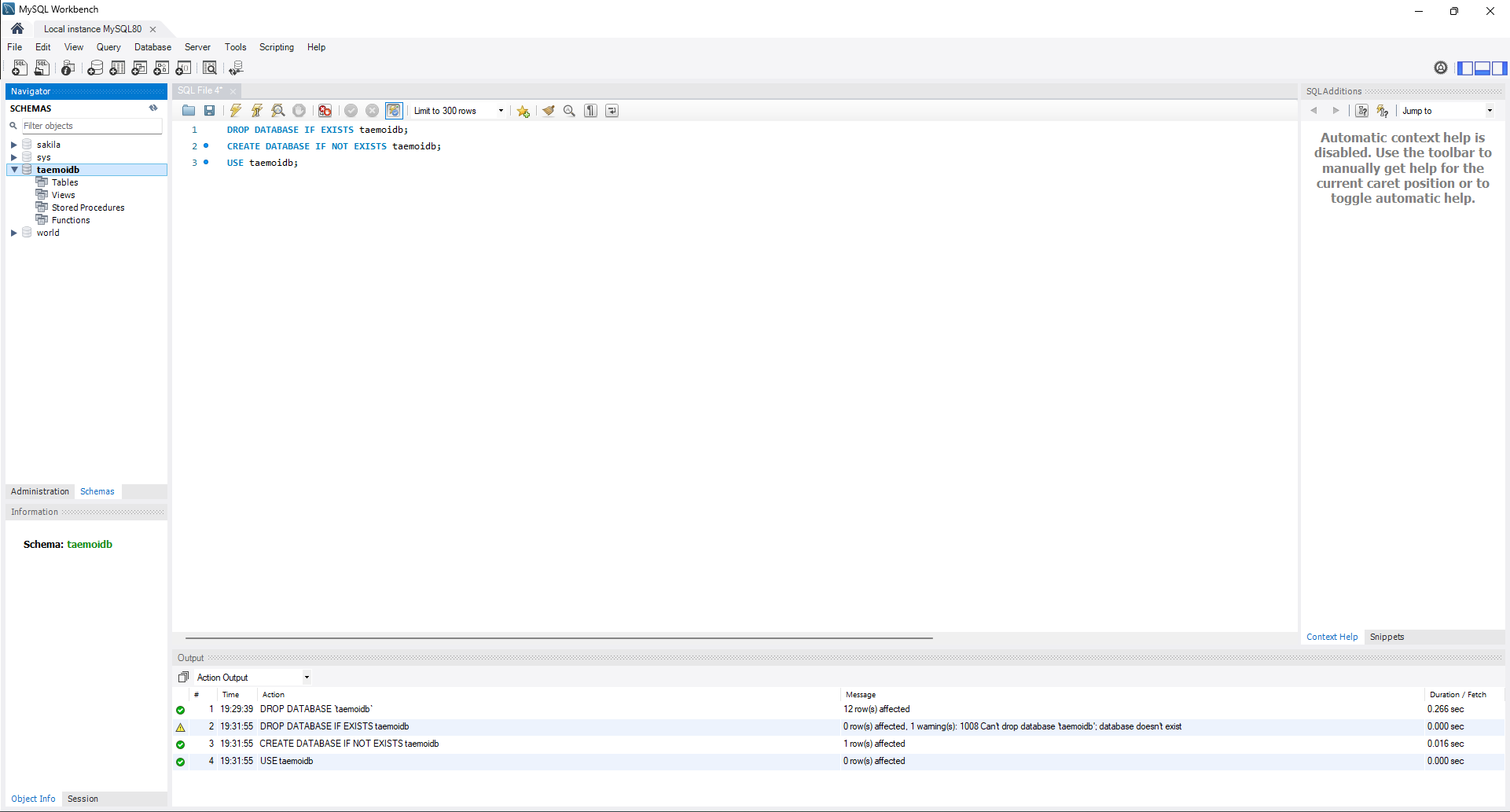
Le damos a nuevo script SQL ( ), escribimos las siguientes sentencias en el fichero y lo ejecutamos:

DROP DATABASE IF EXISTS taemoidb;

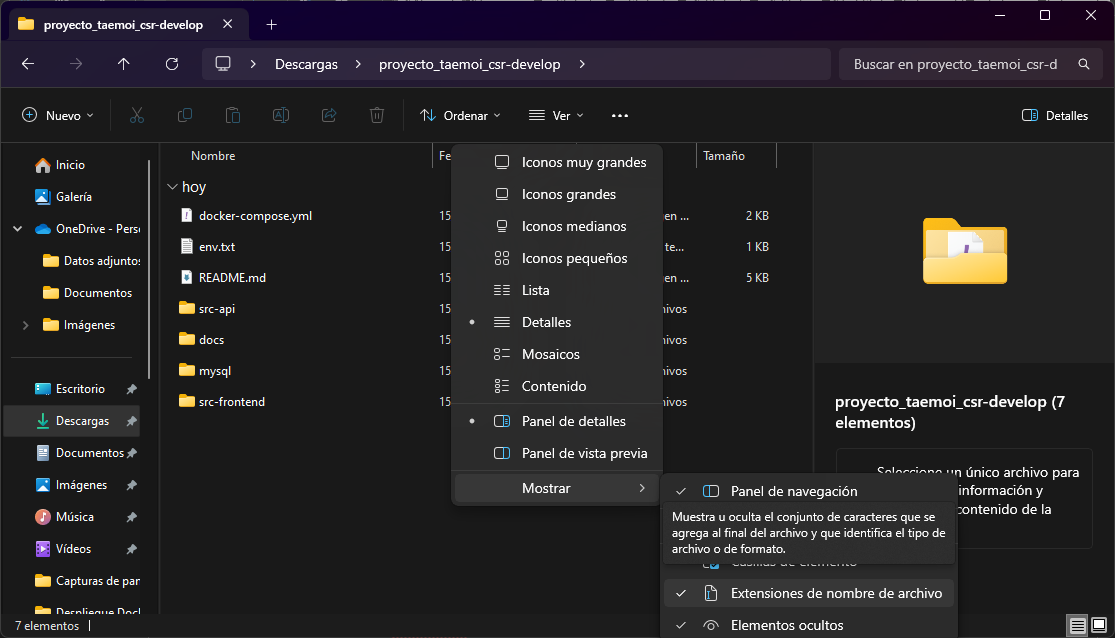
CREATE DATABASE IF NOT EXISTS taemoidb;

USE taemoidb;

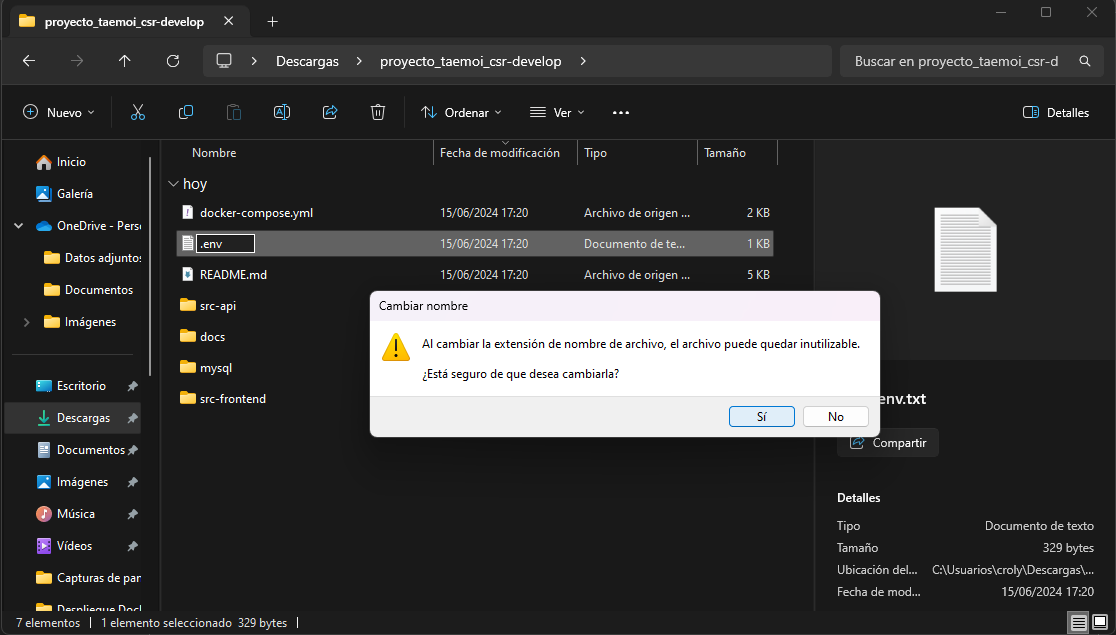
(Estas sentencias sirven para eliminar la base de datos si existe y crearla nueva si no existe).

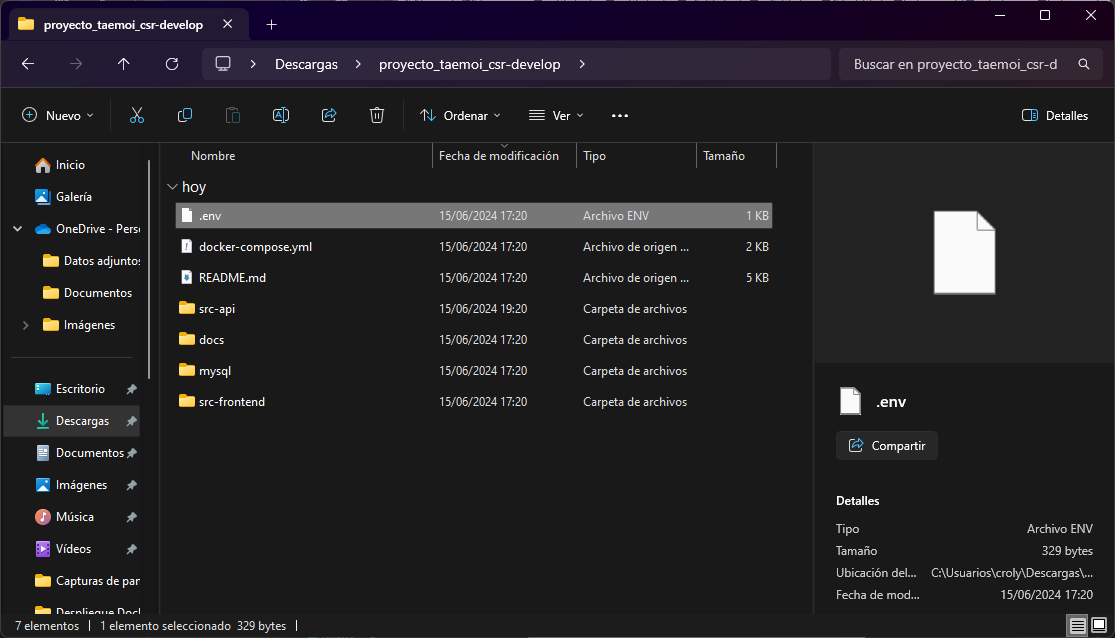


Activamos en el explorador de archivos ver las extensiones de nombre de archivo.



Cambiamos el fichero env.txt por .env.



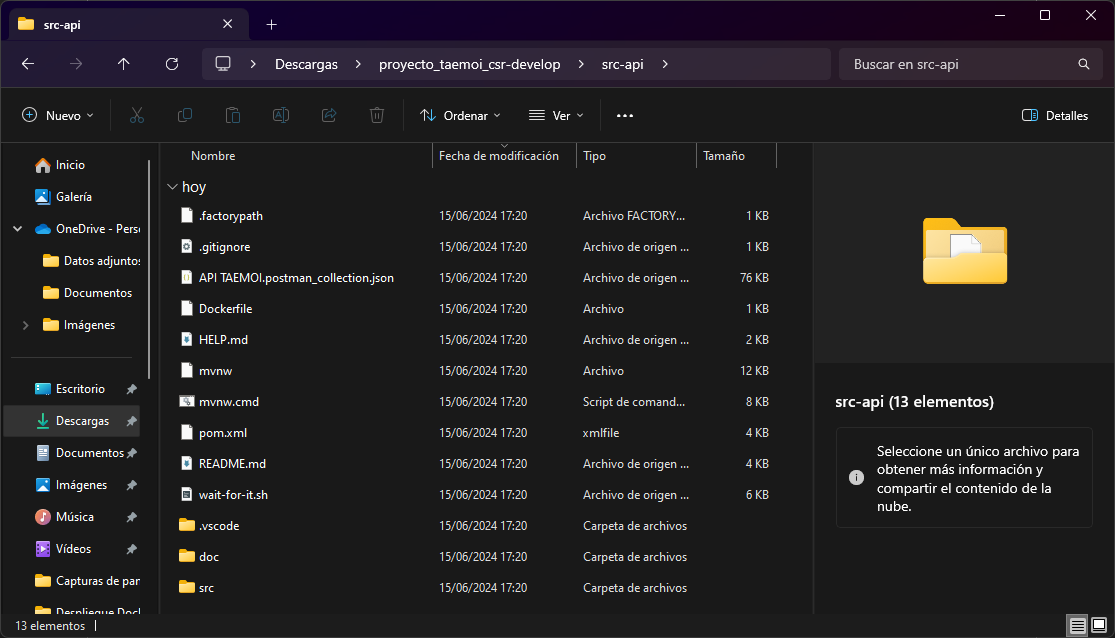


Lo abrimos con el bloc de notas por ejemplo y rellenamos las variables de entornos por las nuestras:

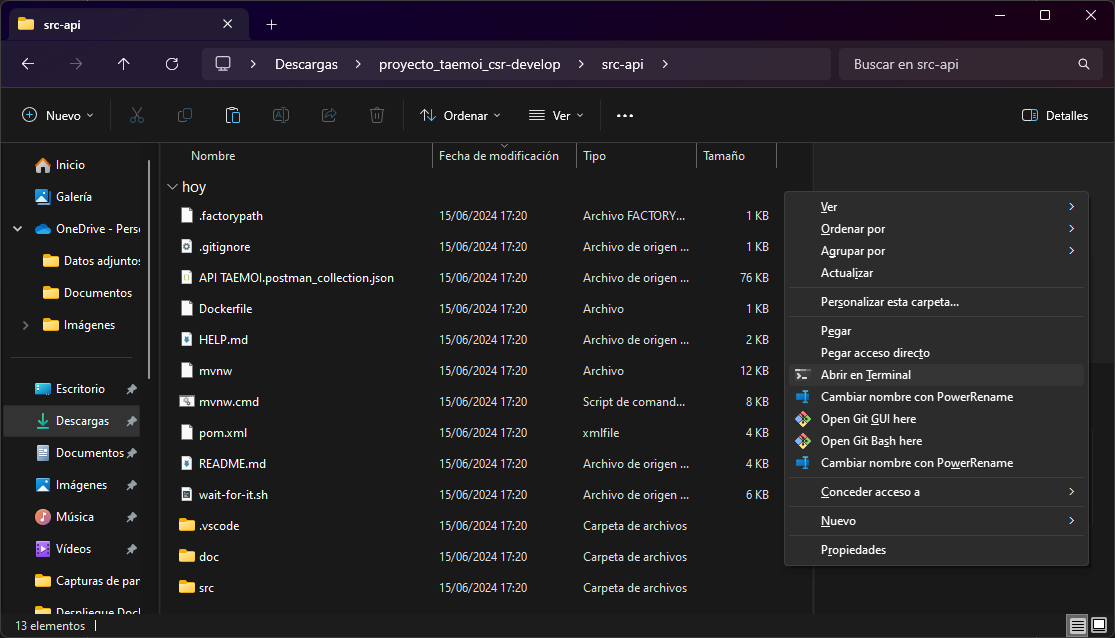
* MYSQL\_ROOT\_PASSWORD: La contraseña de root de mysql. Por defecto es root.
* MYSQL\_DATABASE: El nombre de la base de datos. En nuestro caso pondremos taemoidb, ya que es la que creamos anteriormente.
* MYSQL\_USER: El usuario que vamos a usar para la base de datos. En mi caso uso taemoi, ya que es el usuario que creé en pasos anteriores.
* MYSQL\_PASSWORD: La contraseña de ese usuario. La misma que le pusimos al usuario.
* MYSQL\_HOST: El dominio que va a usar la base de datos. Para local usamos localhost.
* MYSQL\_PORT: El puerto en el que se va a lanzar MySQL. Usaremos el 3306 en este caso.
* CORS\_ALLOWED\_ORIGIN: La url del front para que se comuniquen correctamente back y front. Usaremos <http://localhost:4200>, ya que es el puerto en el que se lanza Angular.
* SPRING\_DATASOURCE\_URL\_LOCAL: La url del controlador de MySQL del back para local. Usaremos jdbc:mysql://localhost:3306/taemoidb, que es el host, el puerto y la base de datos que vamos a usar.
* SPRING\_DATASOURCE\_URL\_DOCKER: La url del controlador de MySQL del back para docker. Usaremos jdbc:mysql://database:3306/taemoidb, aunque de momento no lo necesitamos.
* SPRING\_DATASOURCE\_USERNAME: El nombre de usuario de MySQL para Docker. Pondremos el mismo que en MYSQL\_USER.
* SPRING\_DATASOURCE\_PASSWORD: La contraseña de MySQL para Docker. Pondremos el mismo que en MYSQL\_PASSWORD.
* SPRING\_MAIL\_USERNAME= El correo electrónico a utilizar para el servidor de correo de la api. Yo pongo mi Gmail.
* SPRING\_MAIL\_PASSWORD= La contraseña para el servidor de correo. Puedes poner tu contraseña del correo electrónico, pero te recomiendo crear una contraseña de aplicación para esto, ya que es más seguro. Aquí adjunto el enlace para hacerlo: <https://myaccount.google.com/apppasswords>

Ahora entramos a la carpeta src-api y abrimos una terminal del sistema:

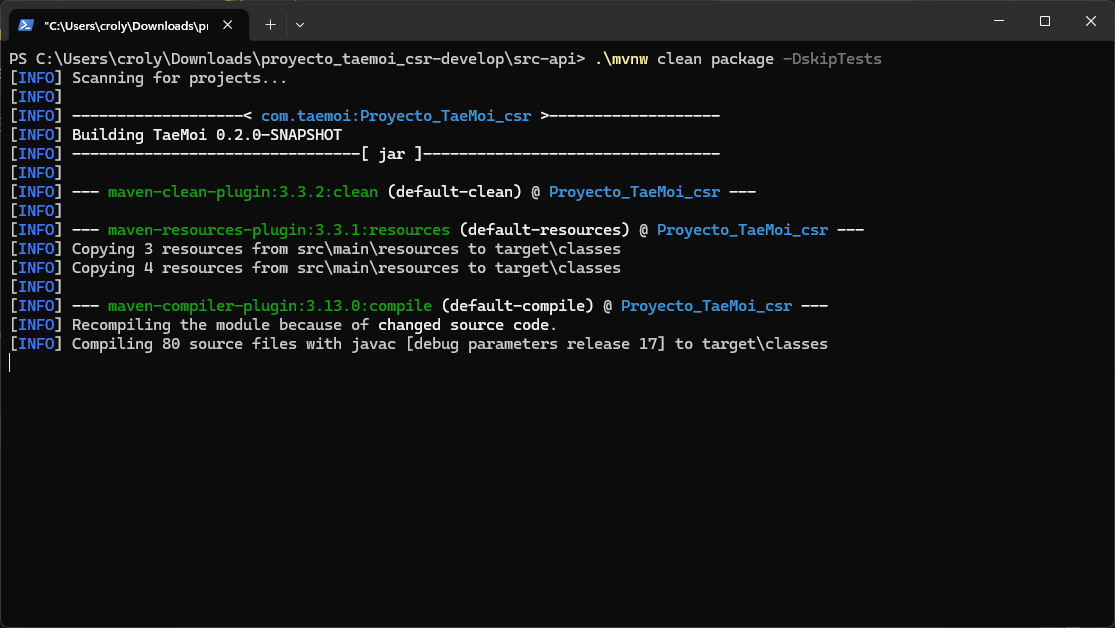
* Primero entramos al directorio.



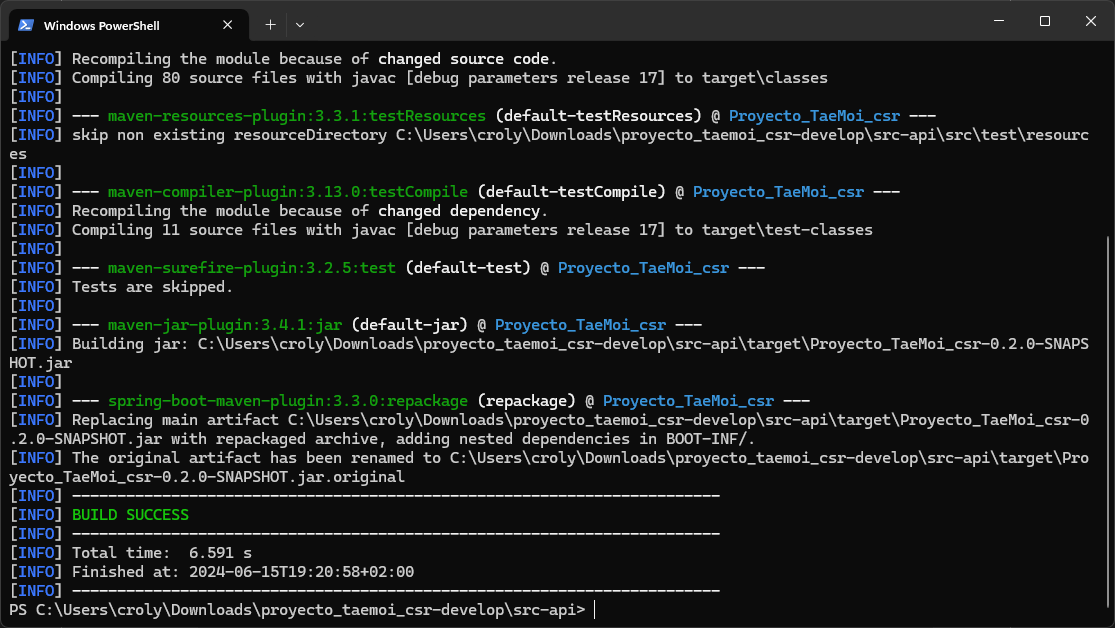
* Y abrimos la terminal.



Una vez con la terminal abierta, vamos a ejecutar el comando ./mvnw clean package -DskipTests (Limpia la carpeta target, compila el proyecto y se salta los tests para evitar problemas en el compilado). Para Linux habría que instalar mvn y ejecutar mvn clean package -DskipTests.



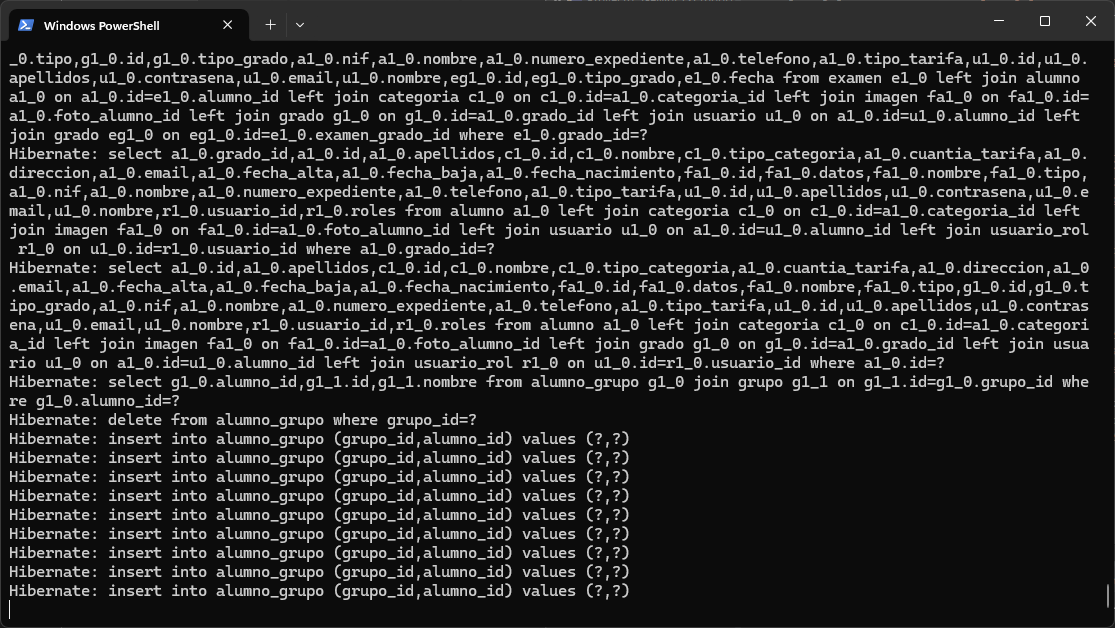
(Sabremos que ha terminado cuando aparezca el mensaje en verde de “BUILD SUCCESS”)



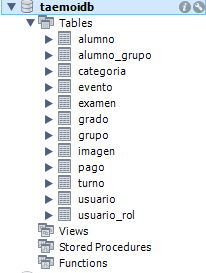
Ahora nos movemos a la carpeta target del proyecto (comando cd .\target\ en Windows o cd ./target en Linux).



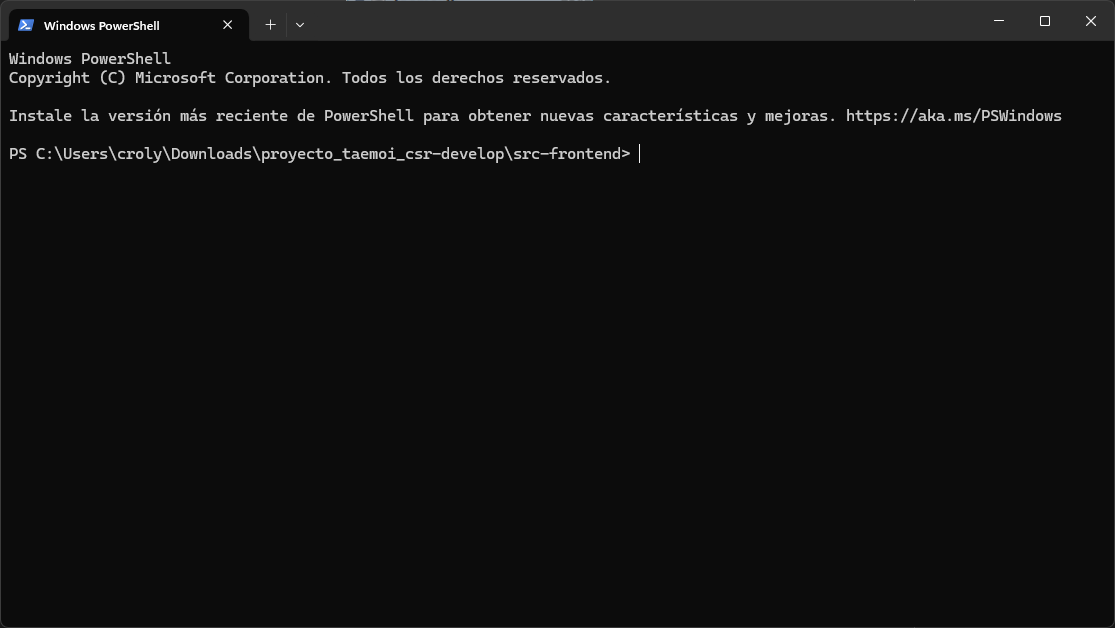
Y lanzamos el comando java -jar “Nombre del proyecto compilado” --spring.profiles.active=local para ejecutar nuestra api con el perfil local y cargará automáticamente las variables de entorno del fichero .env que configuramos antes.



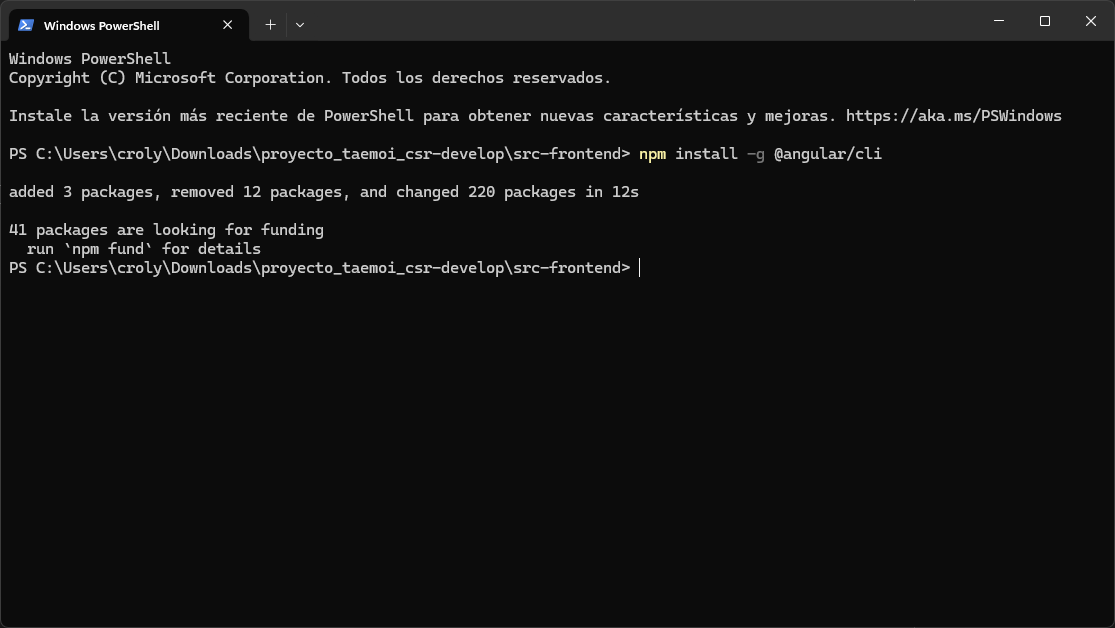
Vamos a MySQL Workbench y comprobamos que se ha creado correctamente la base de datos.



Ahora vamos con la parte del front. Primero, abrimos una nueva terminal y nos movemos a la carpeta src-frontend.

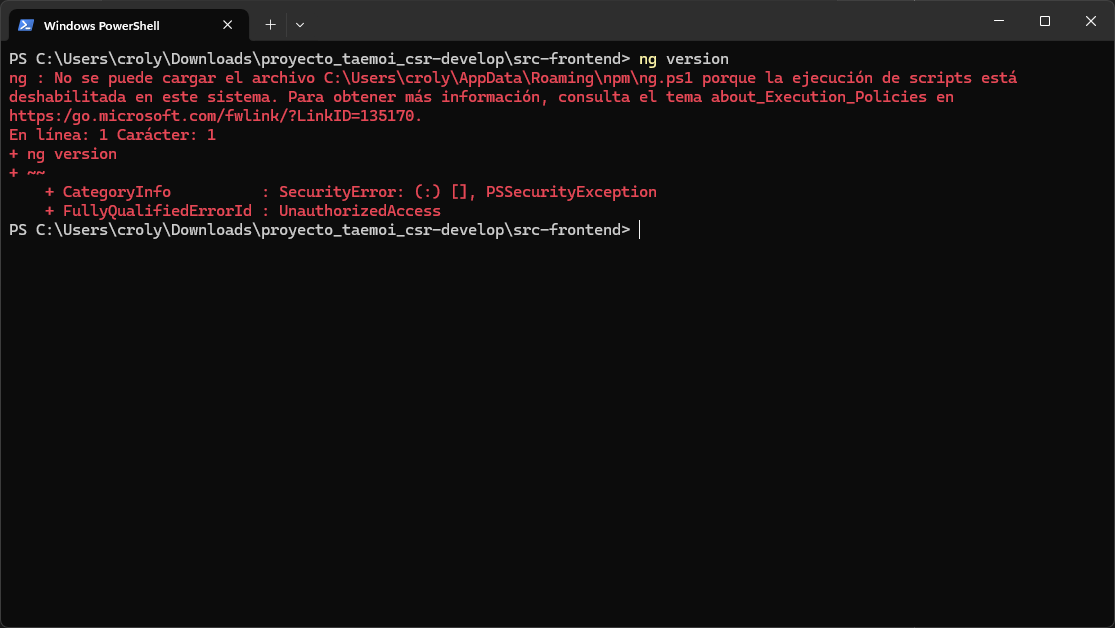


Instalamos el cliente de Angular de forma global usando el comando npm install -g @angular/cli.



Comprobamos que esté correctamente instalado usando el comando ng version.

**IMPORTANTE:** En caso de que el comando te dé este error:

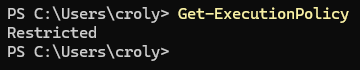


La solución es la siguiente:

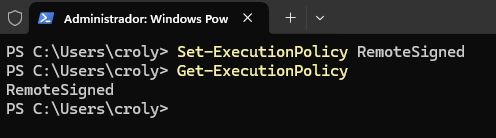
* Abrimos una terminal como administrador.



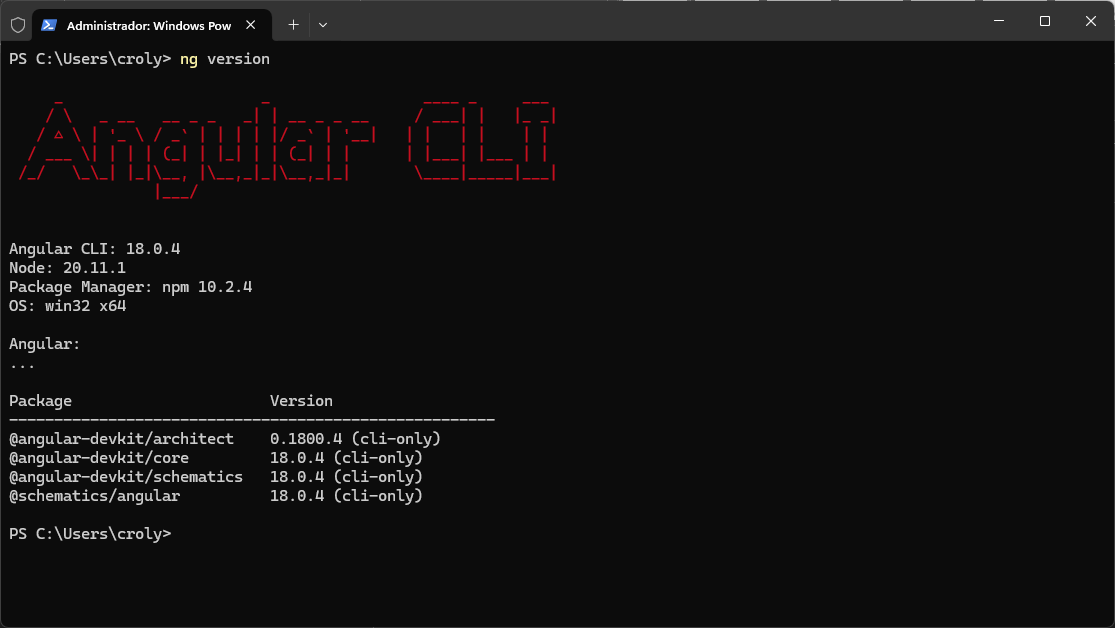
* Miramos la política de ejecución de comandos actual con el comando Get-ExecutionPolicy.



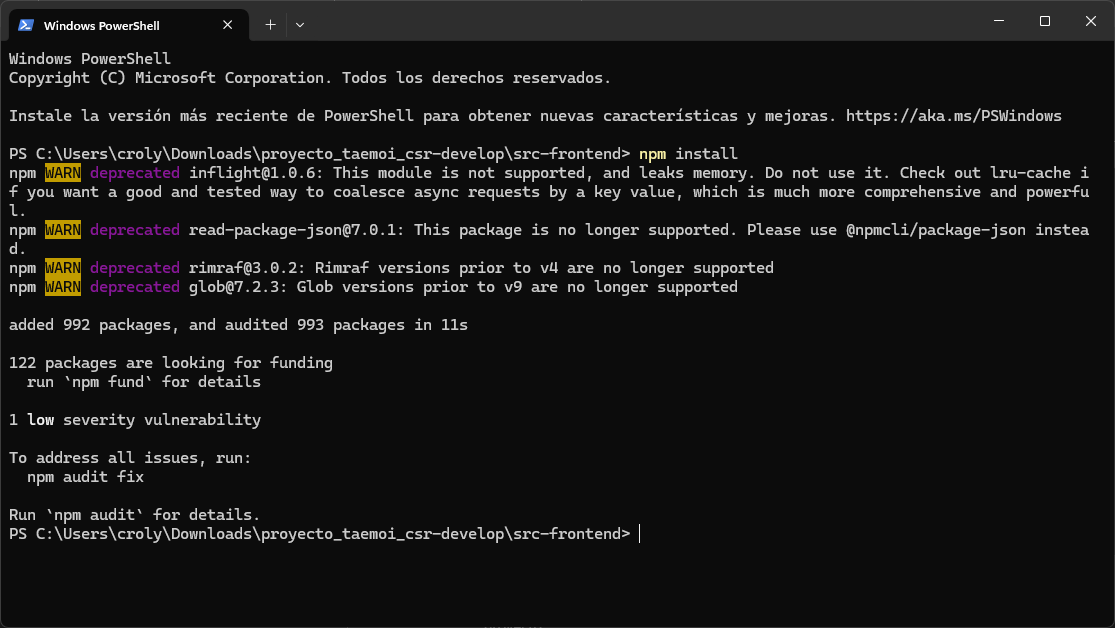
* En caso de ser Restricted, usamos este comando (Set-ExecutionPolicy RemoteSigned) para cambiarlo de Restricted a RemoteSigned y volvemos a lanzar el anterior para probarlo.



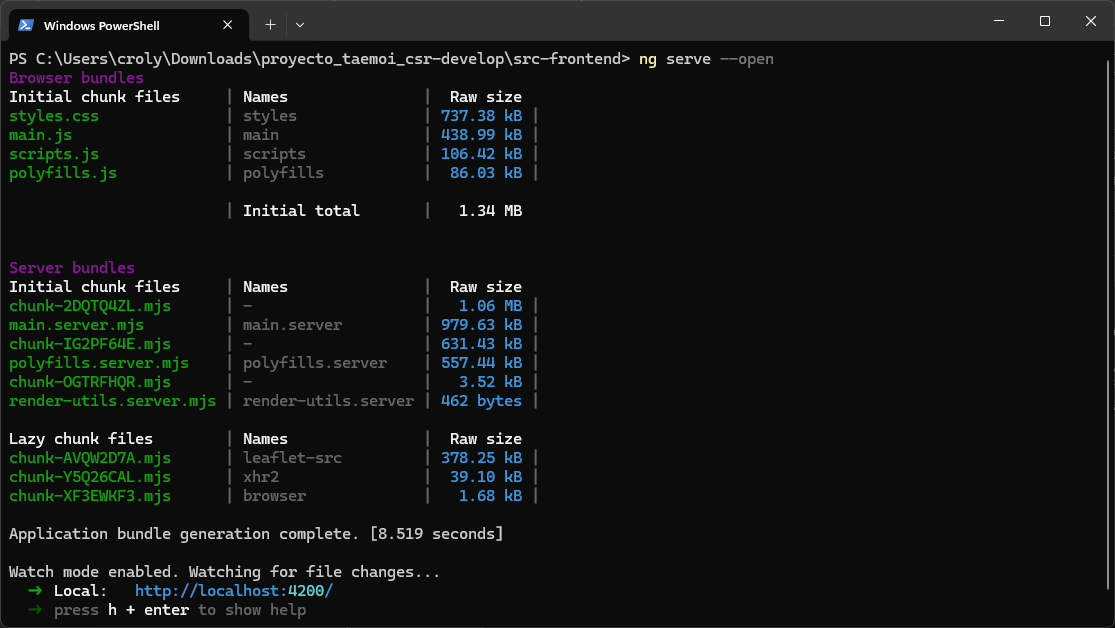
* Volvemos a lanzar ng version para ver si ahora podemos.



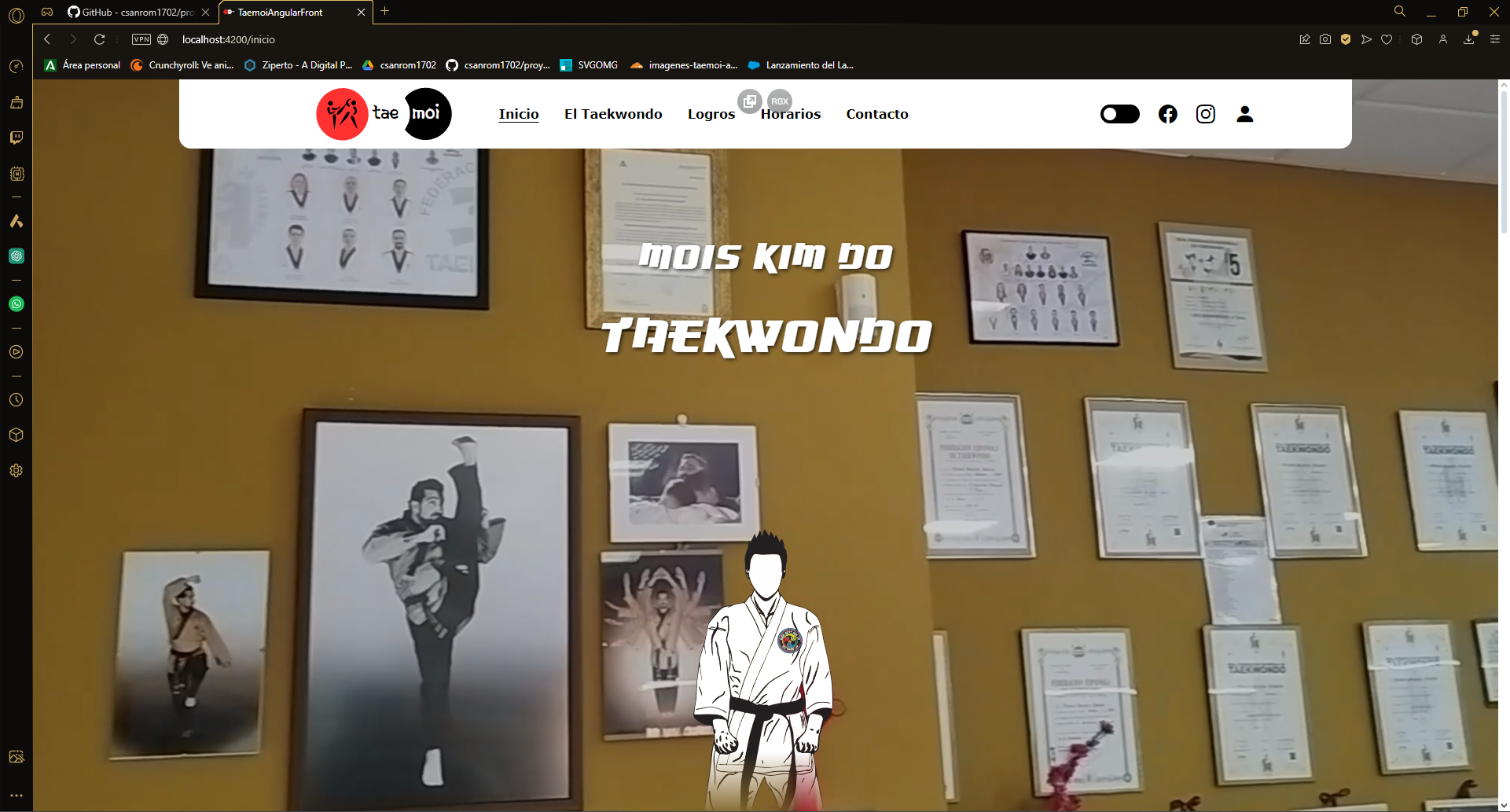
Ahora volvemos a la carpeta src-frontend y ejecutamos npm install para instalar todas las dependencias del proyecto.



Y ahora lo lanzamos usando ng serve --open.



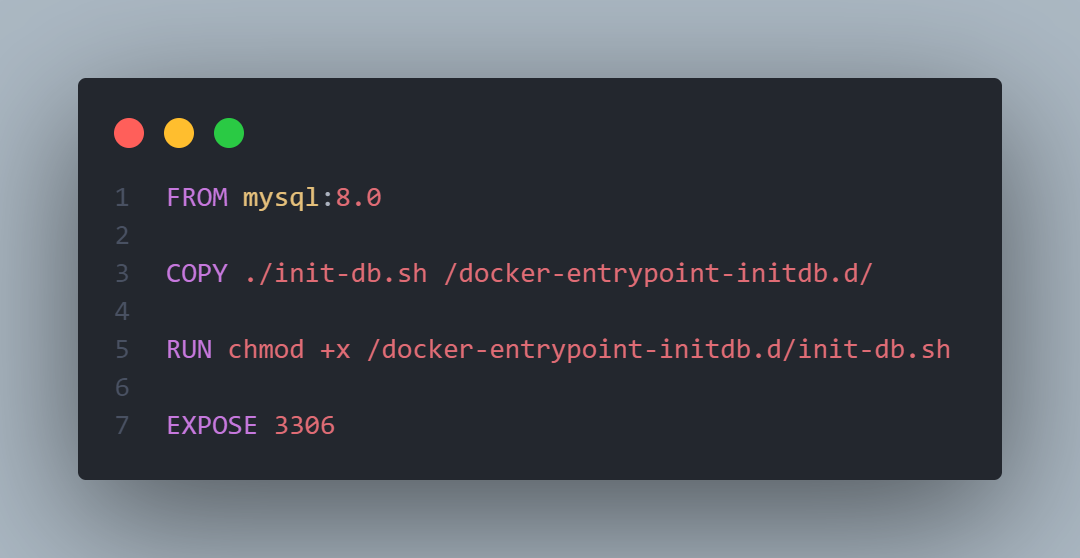
Y listo, aplicación perfectamente lanzada en local y funcionando.



## Despliegue en Docker.

Para el despliegue en Docker, el fichero .env no se tiene que modificar, ya que está configurado en el apartado Local. El despliegue en Docker es automático, por lo que voy a explicar los ficheros involucrados en el despliegue:

* **Dockerfile de MySQL:**



* + - **FROM mysql:8.0**

Esta línea especifica la imagen base que se usará para construir la nueva imagen de Docker. En este caso, está utilizando la imagen oficial de MySQL en su versión 8.0. Esto significa que cualquier cosa definida en el Dockerfile se construirá sobre esta imagen base de MySQL.

* + - **COPY ./init-db.sh /docker-entrypoint-initdb.d/**

Esta línea copia un archivo llamado init-db.sh desde el directorio actual en tu sistema host al directorio /docker-entrypoint-initdb.d/ dentro de la imagen de Docker. Este directorio específico es utilizado por la imagen de MySQL para ejecutar scripts de inicialización cuando el contenedor se inicia por primera vez.

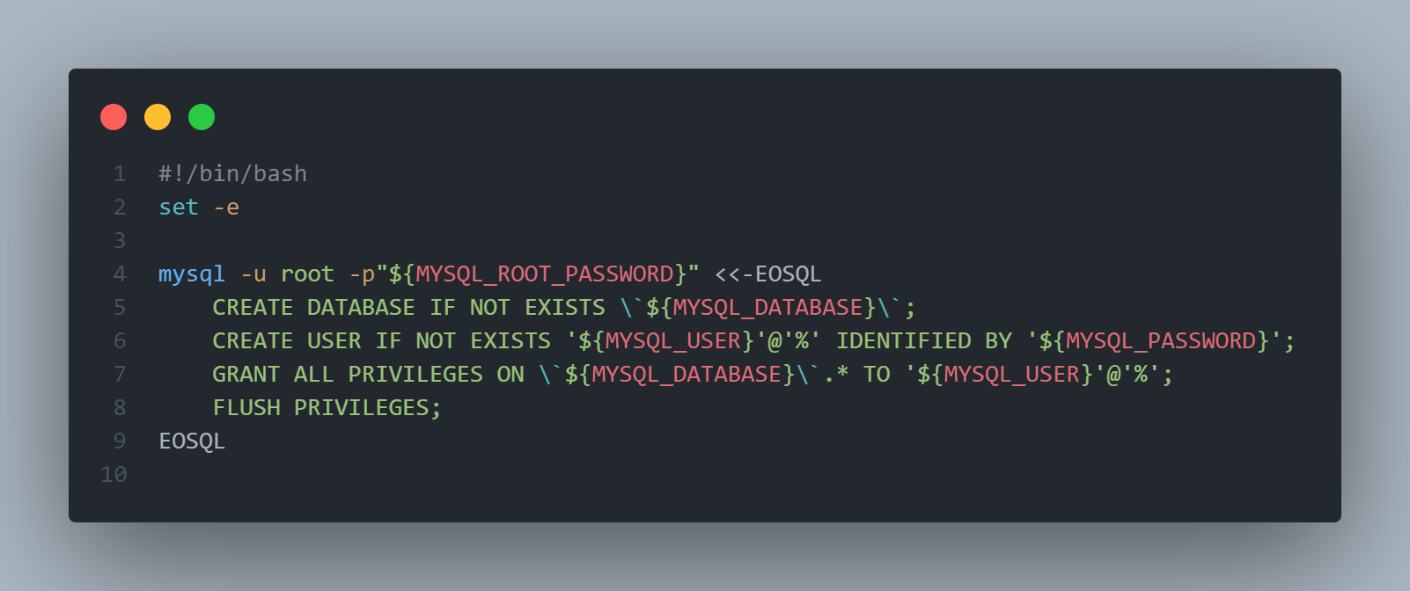
* + - **RUN chmod +x /docker-entrypoint-initdb.d/init-db.sh**

Aquí se otorgan permisos de ejecución al archivo init-db.sh que fue copiado anteriormente. chmod +x hace que el script sea ejecutable, lo que es necesario para que el script se pueda ejecutar automáticamente cuando el contenedor se inicie.

* + - **EXPOSE 3306**

Esta línea informa a Docker que el contenedor escucha en el puerto 3306, que es el puerto predeterminado para el servicio MySQL. Aunque EXPOSE no publica el puerto por sí mismo, es una forma de documentar y configurar el puerto en el que la aplicación en el contenedor estará escuchando. Para hacer que este puerto esté accesible desde fuera del contenedor, usualmente se utiliza la opción -p al iniciar el contenedor con docker run.

* **init-db.sh:**



* + - **#!/bin/bash**

Esta línea indica que el script debe ser ejecutado utilizando el intérprete de comandos bash (el por defecto de linux).

* + - **set -e**

Esta línea asegura que el script se detenga inmediatamente si algún comando devuelve un valor diferente de cero (lo que indica un error).

* + - **mysql -u root -p"${MYSQL\_ROOT\_PASSWORD}" <<-EOSQL**

Este comando ejecuta el cliente de MySQL usando el usuario root y la contraseña almacenada en la variable de entorno MYSQL\_ROOT\_PASSWORD. La sintaxis <<-EOSQL se utiliza para crear un bloque de texto que será pasado al comando mysql. Este bloque contiene una serie de comandos SQL que se ejecutarán en el servidor MySQL.

* + - **Contenido del bloque SQL:**
* **CREATE DATABASE IF NOT EXISTS \${MYSQL\_DATABASE}`:** Crea una base de datos con el nombre almacenado en la variable de entorno MYSQL\_DATABASE` si esta no existe.
* **CREATE USER IF NOT EXISTS '${MYSQL\_USER}'@'%' IDENTIFIED BY '${MYSQL\_PASSWORD}':** Crea un usuario en MySQL con el nombre almacenado en la variable de entorno MYSQL\_USER y la contraseña almacenada en la variable de entorno MYSQL\_PASSWORD, si este usuario no existe ya. El @'%' permite que el usuario se conecte desde cualquier host.
* **GRANT ALL PRIVILEGES ON \${MYSQL\_DATABASE}`. TO '${MYSQL\_USER}'@'%':** Otorga todos los privilegios en la base de datos especificada (MYSQL\_DATABASE) al usuario especificado (MYSQL\_USER`).
* **FLUSH PRIVILEGES:** Recarga las tablas de privilegios en MySQL, aplicando los cambios realizados.
* **Dockerfile del Backend:**



* + - **FROM maven:3.8.5-openjdk-17 AS build**

Esta línea especifica la imagen base que se utilizará para la etapa de construcción. Aquí se usa una imagen de Maven con OpenJDK 17. El alias build se utiliza para referenciar esta etapa más adelante.

* + - **WORKDIR /app**

Establece el directorio de trabajo en /app dentro del contenedor. Todos los comandos siguientes se ejecutarán en este directorio.

* + - **COPY pom.xml .**

Copia el archivo pom.xml desde el directorio actual en el host al directorio de trabajo en el contenedor.

* + - **COPY src ./src**

Copia el directorio src desde el host al contenedor, manteniendo la estructura del directorio.

* + - **RUN mvn clean package -DskipTests**

Ejecuta el comando Maven para limpiar y empaquetar la aplicación, omitiendo las pruebas (-DskipTests). Esto generará un archivo JAR en el directorio target.

* + - **FROM openjdk:17-jdk-slim**

Esta línea especifica la imagen base para la etapa de ejecución. Se utiliza una imagen más ligera de OpenJDK 17 (slim).

* + - **WORKDIR /app**

Establece el directorio de trabajo en /app para la etapa de ejecución.

* + - **COPY --from=build /app/target/Proyecto\_TaeMoi\_csr-0.2.0-SNAPSHOT.jar app.jar**

Copia el archivo JAR generado en la etapa de construcción desde /app/target/Proyecto\_TaeMoi\_csr-0.2.0-SNAPSHOT.jar en el contenedor de construcción al archivo app.jar en el contenedor actual.

* + - **COPY wait-for-it.sh .**

Copia el script wait-for-it.sh desde el host al directorio de trabajo en el contenedor. Este script generalmente se utiliza para esperar a que otros servicios (como bases de datos) estén listos antes de iniciar la aplicación.

* + - **RUN chmod +x wait-for-it.sh**

Otorga permisos de ejecución al script wait-for-it.sh.

* + - **EXPOSE 8080**

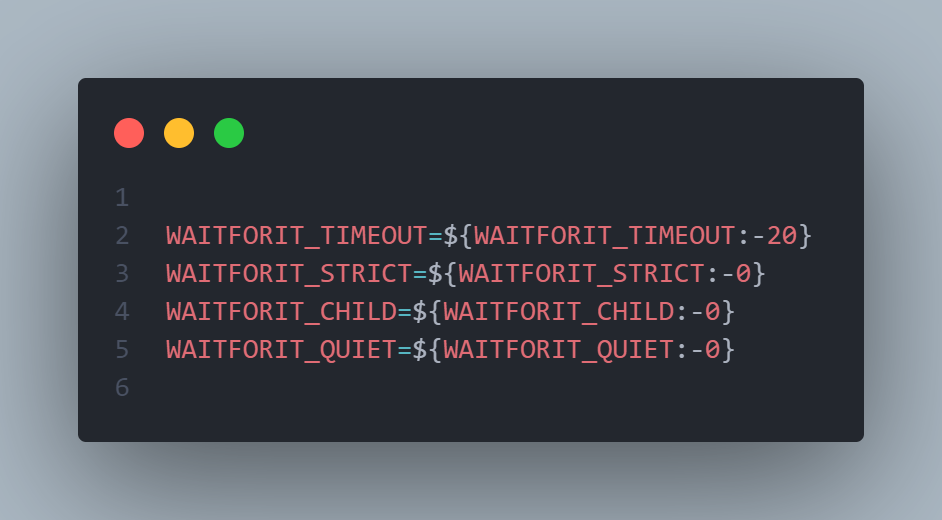
Informa a Docker que el contenedor escuchará en el puerto 8080. Esto es utilizado para documentar y configurar el puerto en el que la aplicación se ejecutará.

* + - **ENTRYPOINT ["./wait-for-it.sh", "database:3306", "--", "java", "-jar", "app.jar"]**

Define el comando que se ejecutará cuando se inicie el contenedor. Aquí se utiliza el script wait-for-it.sh para esperar a que la base de datos en el host database y el puerto 3306 estén disponibles antes de ejecutar el comando java -jar app.jar para iniciar la aplicación.

* **wait-for-it.sh:**

De este fichero sólo nos interesan estas 4 líneas.



Este fichero sirve para que spring se lance después de cargarse MySQL. Cuando el contenedor de MySQL se lanza, mete un timeout de 20 segundos para esperar a que se inicialice correctamente.

Para cambiar el tiempo de espera, sólo tienes que modificar WAITFORIT\_TIMEOUT:-20 por los segundos que necesites.

* **Dockerfile de Angular:**



* + - **FROM node:20 AS build**

Utiliza la imagen oficial de Node.js versión 20 como base para la etapa de construcción de la aplicación Angular. La palabra clave AS build etiqueta esta etapa como "build".

* + - **WORKDIR /app**

Establece el directorio de trabajo dentro del contenedor en /app. Todas las siguientes instrucciones se ejecutarán en este directorio.

* + - **COPY package.json ./\***

Copia los archivos package.json y package-lock.json (si existen) desde el directorio local al directorio de trabajo en el contenedor. Estos archivos son necesarios para instalar las dependencias de Node.js.

* + - **RUN npm install**

Ejecuta npm install para instalar todas las dependencias especificadas en los archivos package.json.

* + - **COPY . .**

Copia todos los archivos y directorios desde el directorio local al directorio de trabajo en el contenedor.

* + - **RUN npm run build**

Ejecuta el script build:local definido en package.json para construir la aplicación Angular. El resultado de la construcción se almacenará en el directorio dist/taemoi-angular-front/browser.

* + - **FROM nginx**

Utiliza la imagen oficial de Nginx en su versión estable como base para la etapa de despliegue.

* + - **COPY --from=build /app/dist/taemoi-angular-front/browser /usr/share/nginx/html**

Copia los archivos construidos desde la etapa de construcción (build) al directorio donde Nginx sirve los archivos HTML (/usr/share/nginx/html).

* + - **COPY ./nginx.conf /etc/nginx/nginx.conf**

Copia un archivo de configuración personalizado de Nginx (nginx.conf) al directorio de configuración de Nginx en el contenedor.

* + - **COPY ./default.conf /etc/nginx/conf.d/default.conf**

Copia un archivo de configuración adicional (default.conf) al directorio de configuración de Nginx.

* + - **EXPOSE 80**

Expone el puerto 80, que es el puerto predeterminado en el que Nginx escucha las solicitudes HTTP.

* + - **CMD ["nginx", "-g", "daemon off;"]**

Establece el comando por defecto que se ejecutará cuando se inicie el contenedor, que en este caso es iniciar Nginx en primer plano (modo no daemonizado).

* **nginx.conf.**



Usuario y Procesos de Trabajo:

user nginx;: Define el usuario bajo el cual correrá el servicio Nginx.

worker\_processes auto;: Especifica el número de procesos de trabajo que se ejecutarán. La configuración "auto" permite que Nginx determine automáticamente el número óptimo basado en el hardware.

Registro de Errores y PID:

error\_log /var/log/nginx/error.log warn;: Configura el archivo de registro de errores y el nivel de log (en este caso, "warn" para advertencias).

pid /var/run/nginx.pid;: Especifica la ubicación del archivo PID que contiene el ID de proceso del servidor Nginx.

Eventos:

events { worker\_connections 1024; }: Dentro del bloque events, se define que cada proceso de trabajo puede manejar un máximo de 1024 conexiones simultáneas.

Configuraciones HTTP:

http { ... }: Este bloque contiene diversas configuraciones relacionadas con el manejo de las peticiones HTTP.

include /etc/nginx/mime.types;: Incluye el archivo de tipos MIME que define los tipos de contenido.

default\_type application/octet-stream;: Establece el tipo de contenido predeterminado para las respuestas.

log\_format main '...': Define el formato del registro de accesos, incluyendo información como la dirección IP del cliente, el usuario remoto, la solicitud, el estado de la respuesta, bytes enviados, el referer, el agente de usuario y cualquier encabezado X-Forwarded-For.

access\_log /var/log/nginx/access.log main;: Especifica la ubicación del archivo de registro de accesos y el uso del formato definido previamente.

sendfile on;: Activa el uso de sendfile, una llamada del sistema para enviar archivos de forma eficiente.

#tcp\_nopush on;: Comentado, podría activarse para optimizar el envío de grandes volúmenes de datos.

keepalive\_timeout 65;: Establece el tiempo de espera para mantener las conexiones activas en 65 segundos.

#gzip on;: Comentado, podría activarse para habilitar la compresión gzip.

include /etc/nginx/conf.d/\*.conf;: Incluye cualquier archivo de configuración adicional ubicado en el directorio /etc/nginx/conf.d/.

# Conclusiones y Postmortem.

Conclusiones

El desarrollo de la aplicación TaeMoi ha sido un proceso integral que ha permitido la creación de una plataforma robusta y eficiente para la gestión de la escuela de taekwondo "MOISKIMDO TAEKWONDO". Los objetivos planteados al inicio del proyecto se han cumplido satisfactoriamente, proporcionando una solución que mejora significativamente la administración de los alumnos, eventos y horarios. La integración de funcionalidades tanto para usuarios anónimos como para alumnos y gestores ha permitido cubrir todas las necesidades identificadas en la fase de análisis.

La elección del stack tecnológico ha resultado acertada, permitiendo un desarrollo ágil y efectivo tanto en el backend como en el frontend. La aplicación es escalable y segura, lo que garantiza su utilidad a largo plazo para la escuela.

Postmortem

El proyecto TaeMoi ha atravesado varias fases de desarrollo y ha enfrentado diversos desafíos, que se han gestionado de la siguiente manera:

Planificación y diseño: Durante esta fase, se identificaron claramente los requisitos y se definió un roadmap detallado. El uso de metodologías ágiles permitió ajustar y refinar los objetivos a medida que avanzaba el desarrollo.

Desarrollo: La implementación se dividió en sprints, con entregas incrementales que facilitaron la integración y el testing continuo. Se utilizaron herramientas como JIRA para la gestión de tareas y Git para el control de versiones, lo que mejoró la colaboración y el seguimiento del progreso.

Testing y calidad: Se realizaron pruebas unitarias y de integración exhaustivas para asegurar la funcionalidad y estabilidad de la aplicación. Se establecieron entornos de pruebas para simular escenarios reales y detectar posibles errores antes del despliegue.

Desafíos enfrentados:

Integración de tecnologías: Uno de los mayores retos fue integrar diferentes tecnologías (Spring Boot, Angular, MySQL) de manera cohesiva. Se superó mediante una arquitectura bien diseñada y el uso de patrones de diseño adecuados.

Gestión de datos sensibles: Asegurar la privacidad y seguridad de los datos de los usuarios fue prioritario. Se implementaron medidas de seguridad como encriptación y autenticación robusta.

Adaptación a cambios: Durante el desarrollo, surgieron cambios en los requisitos y nuevas necesidades. La flexibilidad del equipo y el uso de metodologías ágiles permitieron adaptarse rápidamente a estos cambios.

Lecciones aprendidas:

Importancia de la comunicación: Una comunicación efectiva entre los miembros del equipo y con los stakeholders es crucial para el éxito del proyecto.

Testing continuo: Incluir pruebas continuas en el proceso de desarrollo ayuda a identificar y solucionar problemas de manera temprana.

Documentación: Mantener una documentación clara y actualizada facilita el mantenimiento y la escalabilidad del proyecto en el futuro.