Proyecto Back-End - API para el Concurso de Robótica.

Idea general

El proyecto Concurso Robótica API consiste en desarrollar un servidor de node.js que se encarga de administrar un campeonato de robótica que está siendo organizado por la Universidad Valle del Momboy. El objetivo es facilitar la gestión y el acceso a la información del concurso, tanto para los organizadores como para los participantes y el público en general.

El servidor ofrece una API RESTful que permite acceder a los datos de las categorías, equipos, integrantes, modalidades, patrocinadores y usuarios del concurso. La API también permite realizar operaciones de creación, actualización, eliminación y búsqueda de los recursos, según el rol y el permiso de cada usuario.

El programa es capaz de:

- Autenticar a los usuarios mediante tokens JWT
- Validar las peticiones y los datos recibidos
- Conectarse a una base de datos MySQL para almacenar y consultar los datos
- Enviar respuestas en formato JSON con los datos solicitados o los mensajes de error
- Documentar las rutas de la API mediante una colección de Postman.

Participación de los miembros del equipo

El equipo está formado por cuatro estudiantes de la carrera de Ingeniería de Computación de la Universidad Valle del Momboy: Luis-Suarezzz, Samel24, Adolf-2005 y Juanito77j.

Cada uno de ellos ha contribuido al proyecto de la siguiente manera:

Luis-Suarezzz: Se ha encargado de diseñar y crear la base de datos MySQL, así como de importar los datos iniciales del concurso. También ha colaborado en la definición de las rutas de la API y en la implementación de las operaciones de creación y eliminación de los recursos.

Samel24: Se ha ocupado de configurar el servidor de node.js y los módulos necesarios, así como de crear el archivo .env con las variables de entorno. También ha participado en la implementación de las operaciones de actualización y búsqueda de los recursos, así como en la validación de los datos recibidos.

Adolf-2005: Se ha responsabilizado de implementar el sistema de autenticación y autorización de los usuarios mediante tokens JWT y roles. También ha ayudado en la documentación de las rutas de la API mediante una colección de Postman y en la prueba de las mismas.

Juanito77j: Se ha dedicado a implementar el manejo de errores y el envío de respuestas en formato JSON. También ha colaborado en la optimización del código y en la revisión de la calidad y el funcionamiento del servidor.

Justificación de la elección de MySQL

Se ha elegido MySQL como base de datos por las siguientes razones:

1. Es un sistema de gestión de bases de datos relacional, lo que permite modelar los datos del concurso de forma estructurada y normalizada, evitando la redundancia y la inconsistencia de los mismos.
2. Es un sistema de gestión de bases de datos gratuito y de código abierto, lo que reduce los costos y facilita la instalación y el uso del mismo.
3. Es un sistema de gestión de bases de datos ampliamente utilizado y soportado, lo que garantiza la disponibilidad y la seguridad de los datos, así como la compatibilidad con diferentes plataformas y herramientas.
4. Es un sistema de gestión de bases de datos que ofrece un buen rendimiento y una alta escalabilidad, lo que permite manejar grandes volúmenes de datos y usuarios concurrentes sin perder eficiencia ni calidad.

Tabla de los endpoints del servidor

Ruta	Método HTTP	Datos Recibidos	Descripción	Roles
/usuarios/login	POST	email, password	Inicia sesión y devuelve un token de autenticación	Todos
/usuarios/register	POST	email, password, nombre, rol	Registra un nuevo usuario y devuelve un token de autenticación	Todos
/usuarios/:id	GET		Devuelve los datos de un usuario por su id	Administrador, Usuario
/categorias	GET		Devuelve la lista de todas las categorías	Todos
/categorias/:id	GET		Devuelve los datos de una categoría por su id	Todos
/categorias	POST	categoria, idModalidad	Crea una nueva categoría y devuelve sus datos	Administrador
/categorias/:id	PUT	categoria, idModalidad	Actualiza los datos de una categoría por su	Administrador

			id	
/categorias/:id	DELETE		Elimina una categoría por su id	Administrador
/equipos	GET		Devuelve la lista de todos los equipos	Todos
/equipos/:id	GET		Devuelve los datos de un equipo por su id	Todos
/equipos	POST	equipo (nombre)	Crea un nuevo equipo y devuelve sus datos	Administrador, Usuario
/equipos/inscribir	POST	idEquipo, idCategoria	Inscribe un equipo existente en una categoría	Administrador, Usuario
/equipos/:id	PUT	nombre, id	Actualiza los datos de un equipo por su id	Administrador, Usuario
/equipos/:id	DELETE		Elimina un equipo por su id	Administrador, Usuario
/equipos/:idEquipo/:idCategoria	DELETE		Elimina la inscripción de un equipo en una categoría	Administrador, Usuario
/integrantes	GET		Devuelve la lista de todos los integrantes	Todos
/integrantes/:id	GET		Devuelve los datos de un integrante por su id	Todos
/integrantes	POST	integrante (nombre), idEquipo	Agrega un integrante dentro de un equipo	Administrador, Usuario
/integrantes/:id	PUT	integrante (nombre), idEquipo	Actualiza los datos de un integrante dentro de un equipo por su id	Administrador, Usuario

/integrantes/:id	DELETE		Elimina un integrante por su id	Administrador, Usuario
/modalidades	GET		Devuelve la lista de todas las modalidades	Todos
/modalidades/:id	GET		Devuelve los datos de una modalidad por su id	Todos
/modalidades	POST	modalidad (nombre)	Crea una nueva modalidad y devuelve sus datos	Administrador
/patrocinantes	GET		Devuelve la lista de todos los patrocinadores	Todos
/patrocinantes/equiposPatrocinados/:id	GET		Devuelve los datos de un patrocinador por su id	Todos
/patrocinantes	POST	patrocinante (nombre)	Crea un nuevo patrocinador y devuelve sus datos	Administrador
/patrocinar/:id	POST	idPatrocinante, idEquipo	Agrega un patrocinante dentro de un equipo	Administrador