

# Documentación Api Trivia

## 1. Introducción

La API está diseñada para gestionar una plataforma de trivias, incluyendo la creación de usuarios, trivias, preguntas y la participación de usuarios. También permite autenticación mediante JWT.

---

## 2. Instalación y configuración

### Clonar el repositorio

Para comenzar, clona el repositorio en tu máquina local y accede al directorio del proyecto:

```
git clone https://github.com/nrc7/trivia-api.git
cd trivia-api
```

### Configurar las variables de entorno

Crea un archivo `.env` en la raíz del proyecto y define las variables necesarias para la ejecución:

```
env
JWT_SECRET_KEY=unaClaveSecretaSegura
```

### Configurar Docker Compose

Asegúrate de que el archivo `docker-compose.yml` esté configurado correctamente. El contenido debería ser el siguiente:

```
services:
  web:
    build: . # Construye el contenedor desde el Dockerfile en la raíz
    ports:
      - "5000:5000" # Expone el puerto 5000
    environment:
      - JWT_SECRET_KEY=${JWT_SECRET_KEY} # Define la key JWT en archivo .env
    volumes:
      - ./app # Sincroniza el código local con el contenedor
      - ./trivia.db:/app/trivia.db # Persistencia para la base de datos SQLite
    command: flask run --host=0.0.0.0 # Ejecuta la aplicación en el contenedor

    restart: always # Reinicia el contenedor automáticamente en caso de fallo
```

## Construir y ejecutar el proyecto

Ejecuta los siguientes comandos para construir y levantar el servicio con Docker Compose:

```
docker-compose build
```

```
docker-compose up
```

Esto iniciará el servidor Flask en el puerto **5000** y creará automáticamente la base de datos SQLite (**trivia.db**) si no existe.

## Acceso a la API

La API estará disponible en <http://localhost:5000>.

## Probar los endpoints

Los endpoints se pueden probar mediante herramientas como **Postman** o usando comandos **cURL**.

## Notas adicionales

- La base de datos SQLite se inicializa automáticamente al iniciar el contenedor, gracias al método **db.create\_all()** implementado en el archivo de inicialización.

Si necesitas reiniciar el entorno eliminando la base de datos, elimina el archivo **trivia.db** en la raíz del proyecto y reinicia los contenedores:

```
docker-compose down
```

```
docker-compose up --build
```

### 3. Esquema de la base de datos

Los modelos y relaciones principales son:

- **User:** Usuarios del sistema con roles **admin** o **jugador**.
  - **Trivia:** Conjunto de preguntas asignadas a usuarios.
  - **Question:** Preguntas con opciones y dificultad.
  - **Ranking:** Registro de puntajes.
  - Relaciones:
    - Trivia ↔ Question: Muchos a muchos.
    - Trivia ↔ User: Muchos a muchos.
  -
- 

### 4. Autenticación

La autenticación utiliza **JWT**.

- **Inicio de sesión:** Genera un token con duración de 30 minutos.

**Encabezado requerido:**

Authorization: Bearer <token>

---

## 5. Listado de endpoints

### Autenticación

#### Registro de usuarios

**Ruta:** `/register`

**Método:** `POST`

**Descripción:** Permite registrar un nuevo usuario.

**Requiere JWT:** No.

#### Cuerpo de la solicitud:

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "securePassword123",
  "role": "jugador" // Opcional, por defecto "jugador" }
```

#### Ejemplo de respuesta exitosa:

```
{
  "code": "201",
  "message": "Usuario registrado",
  "user": {
    "id": 1,
    "name": "John Doe",
    "role": "jugador"
  }
}
```

## Inicio de sesión

- **Ruta:** `/login`
- **Método:** `POST`
- **Descripción:** Autentica a un usuario y genera un token JWT.
- **Requiere JWT:** No.

## Cuerpo de la solicitud:

```
{  
  "email": "john.doe@example.com",  
  "password": "securePassword123"  
}
```

## Ejemplo de respuesta exitosa:

```
{  
  "code": "200",  
  "message": "Login exitoso",  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."}
```

---

## Usuarios

### Obtener lista de usuarios

- **Ruta:** `/users`
- **Método:** `GET`
- **Descripción:** Recupera una lista de usuarios registrados.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

### Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Usuarios recuperados exitosamente",
  "data": [
    {
      "id": 1,
      "name": "John Doe",
      "email": "john.doe@example.com",
      "role": "jugador"
    }
  ]
}
```

### Actualizar usuario

- **Ruta:** `/users/<user_id>`
- **Método:** `PUT`
- **Descripción:** Actualiza los datos de un usuario.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

### Cuerpo de la solicitud:

```
{
  "name": "Jane Doe",
  "email": "jane.doe@example.com",
  "role": "admin"
}
```

### Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Usuario actualizado exitosamente",
  "data": {
    "id": 1,
    "name": "Jane Doe",
    "email": "jane.doe@example.com",
    "role": "admin"
  }
}
```

## Eliminar usuario

- **Ruta:** `/users/<user_id>`
- **Método:** `DELETE`
- **Descripción:** Elimina un usuario por su ID.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

## Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Usuario eliminado exitosamente"
}
```

---

## Preguntas

### Crear pregunta

- **Ruta:** `/questions`
- **Método:** `POST`
- **Descripción:** Permite crear una nueva pregunta.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

### Cuerpo de la solicitud:

```
{
  "question_text": "¿Cuál es la capital de Francia?",
  "correct_option": "París",
  "options": ["París", "Londres", "Berlín"],
  "difficulty": "medio"
}
```

### Ejemplo de respuesta exitosa:

```
{
  "code": "201",
  "message": "Pregunta creada exitosamente",
  "data": {
    "id": 1,
    "question_text": "¿Cuál es la capital de Francia?",
    "difficulty": "medio"
  }
}
```

### Obtener todas las preguntas

- **Ruta:** `/questions`
- **Método:** `GET`
- **Descripción:** Recupera todas las preguntas existentes.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

### Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "length": 1,
  "message": "Preguntas recuperadas exitosamente",
  "data": [
    {
      "id": 1,
      "question_text": "¿Cuál es la capital de Francia?",
      "options": {
        "option_1": "París",
        "option_2": "Londres",
        "option_3": "Berlín"
      },
      "correct_option": "París",
      "difficulty": "medio"
    }
  ]
}
```



## Actualizar pregunta

- **Ruta:** `/questions/<question_id>`
- **Método:** `PUT`
- **Descripción:** Actualiza una pregunta específica.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

## Cuerpo de la solicitud:

```
{
  "question_text": "¿Cuál es la capital de Italia?",
  "correct_option": "Roma",
  "options": ["Roma", "Madrid", "Lisboa"],
  "difficulty": "fácil"
}
```

## Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Pregunta actualizada exitosamente",
  "data": {
    "id": 1,
    "question_text": "¿Cuál es la capital de Italia?",
    "difficulty": "fácil"
  }
}
```

## Eliminar pregunta

- **Ruta:** `/questions/<question_id>`
- **Método:** `DELETE`
- **Descripción:** Elimina una pregunta específica.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

## Ejemplo de respuesta exitosa:

```
{  
  "code": "200",  
  "message": "Pregunta eliminada exitosamente"  
}
```

---

## Trivias

### Crear trivia

- **Ruta:** `/trivias`
- **Método:** `POST`
- **Descripción:** Permite crear una trivia con preguntas y usuarios asociados.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

### Cuerpo de la solicitud:

```
{  
  "name": "Trivia General",  
  "description": "Trivia de cultura general",  
  "user_ids": [1, 2],  
  "question_ids": [1, 2, 3] }
```

### Ejemplo de respuesta exitosa:

```
{
  "code": "201",
  "message": "Trivia creada exitosamente",
  "data": {
    "id": 1,
    "name": "Trivia General",
    "description": "Trivia de cultura general",
    "questions": [1, 2, 3],
    "users": [
      {"id": 1, "name": "John Doe"},
      {"id": 2, "name": "Jane Doe"}
    ]
  }
}
```

### Obtener todas las trivias

- **Ruta:** `/trivias`
- **Método:** `GET`
- **Descripción:** Recupera todas las trivias existentes con su información asociada.
- **Requiere JWT:** Sí.
- **Permisos:** Cualquier rol.

### Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "length": 1,
  "message": "Trivias obtenidas exitosamente",
  "data": [
    {
      "id": 1,
      "name": "Trivia General",
      "description": "Trivia de cultura general",
      "questions": [
        {"id": 1, "question_text": "¿Cuál es la capital de Francia?"},
        {"id": 2, "question_text": "¿Cuál es la capital de Italia?"}
      ],
      "users": [
        {"id": 1, "name": "John Doe"},
        {"id": 2, "name": "Jane Doe"}
      ]
    }
  ]
}
```

## Obtener trivia por ID

- **Ruta:** `/trivias/<trivia_id>`
- **Método:** `GET`
- **Descripción:** Recupera los detalles de una trivia específica por su ID.
- **Requiere JWT:** Sí.
- **Permisos:** Cualquier rol.

## Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Trivia recuperada exitosamente",
  "data": {
    "trivia": {
      "id": 1,
      "name": "Trivia General",
      "description": "Trivia de cultura general",
      "questions": [
        {"id": 1, "question_text": "¿Cuál es la capital de Francia?"},
        {"id": 2, "question_text": "¿Cuál es la capital de Italia?"}
      ],
      "users": [
        {"id": 1, "name": "John Doe"},
        {"id": 2, "name": "Jane Doe"}
      ]
    }
  }
}
```

## Actualizar trivia

- **Ruta:** `/trivias/<trivia_id>`
- **Método:** `PUT`
- **Descripción:** Actualiza la información de una trivia específica.
- **Requiere JWT:** Sí.
- **Permisos:** Rol `admin`.

## Cuerpo de la solicitud:

```
{
  "name": "Trivia Geografía",
  "description": "Trivia sobre geografía",
  "user_ids": [1, 3],
  "question_ids": [1, 4]
}
```

## Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Trivia actualizada exitosamente",
  "data": {
    "id": 1,
    "name": "Trivia Geografía",
    "description": "Trivia sobre geografía"
  }
}
```

## Eliminar trivia

**Ruta:** `/trivias/<trivia_id>`

**Método:** `DELETE`

**Descripción:** Elimina una trivia específica.

**Requiere JWT:** Sí.

**Permisos:** Rol `admin`.

## Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Trivia eliminada exitosamente"
}
```

## Obtener trivias de un usuario

- **Ruta:** `/users/<user_id>/trivias`
- **Método:** `GET`
- **Descripción:** Recupera las trivias asociadas a un usuario específico.
- **Requiere JWT:** No.

### Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Trivias de John Doe obtenidas exitosamente",
  "data": [
    {
      "id": 1,
      "name": "Trivia General",
      "description": "Trivia de cultura general",
      "questions": [
        {"id": 1, "question_text": "¿Cuál es la capital de Francia?"}
      ]
    }
  ],
  "len": 1,
  "user": {
    "id": 1,
    "user_name": "John Doe"
  }
}
```

---

## Participación

### Participar en una trivia

- **Ruta:** `/participate/<trivia_id>`
- **Método:** `POST`
- **Descripción:** Permite que un usuario participe en una trivia y registre sus respuestas.
- **Requiere JWT:** Sí.
- **Permisos:** Cualquier rol.

### Cuerpo de la solicitud:

```
{  
  "trivia_id": 1,  
  "user_name": "John Doe",  
  "answers": {  
    "1": "París",  
    "2": "Roma"  
  }  
}
```



### Ejemplo de respuesta exitosa:

```
{
  "code": "201",
  "message": "Participación registrada",
  "data": {
    "score": 5,
    "correct_answers": {
      "1": {
        "correct_answer": "París",
        "difficulty": "medio",
        "is_correct": "correcta"
      },
      "2": {
        "correct_answer": "Roma",
        "difficulty": "fácil",
        "is_correct": "correcta"
      }
    },
    "trivia": {
      "id": 1,
      "name": "Trivia General"
    },
    "user": {
      "name": "John Doe"
    }
  }
}
```

---

## Rankings

### Obtener ranking de una trivia

- **Ruta:** `/ranking/<trivia_id>`
- **Método:** `GET`
- **Descripción:** Recupera el ranking de puntajes para una trivia específica.
- **Requiere JWT:** Sí.
- **Permisos:** Cualquier rol.

### Ejemplo de respuesta exitosa:

```
{
  "code": "200",
  "message": "Ranking recuperado exitosamente",
  "data": {
    "trivia": {
      "id": 1,
      "name": "Trivia General",
      "descripcion": "Trivia de cultura general"
    },
    "ranking": [
      {
        "user_name": "John Doe",
        "score": 5
      },
      {
        "user_name": "Jane Doe",
        "score": 3
      }
    ]
  }
}
```