Dilery - Digital Gallery API

Dilery es una API para gestionar una galería digital de arte. Permite a los usuarios administrar exposiciones, proyectos artísticos, estudios y perfiles de usuario. Diseñada para ser intuitiva y eficiente, esta API está construida con **Node.js**, **Express** y utiliza **MongoDB** como base de datos.

Características Principales

- **Gestión de Usuarios:**
  - Registro y autenticación de usuarios.
  - Roles de usuario: `admin` y `user`.
  - Actualización de perfil y eliminación de cuenta.
- **Proyectos Artísticos:**
  - Creación, lectura, actualización y eliminación de proyectos.
  - Relación con estudios y exposiciones.
- **Exposiciones:**
  - Organización de exposiciones de arte con datos como ubicación y organizadores.
  - Relación con proyectos artísticos.
- **Estudios:**
  - Gestión de estudios de arte, incluyendo participantes, dueños y obras.
- **Autenticación Segura:**
  - Autenticación mediante JWT.
  - Rutas protegidas para administradores y usuarios autenticados.

Tecnologías Utilizadas

- **Node.js**: Entorno de ejecución.
- **Express.js**: Framework backend.
- **MongoDB**: Base de datos no relacional.
- **Mongoose**: Modelado de datos para MongoDB.
- **bcrypt**: Cifrado de contraseñas.
- **jsonwebtoken (JWT)**: Autenticación segura.
- **dotenv**: Gestión de variables de entorno.

Estructura del Proyecto

Carpeta/Archivo	Descripción
`src`	Carpeta principal del proyecto.
`src/api`	Contiene los modelos, controladores y rutas de la API.
`src/api/controllers`	Lógica de los endpoints (usuarios, proyectos, etc.).
`src/api/models`	Modelos de datos (Mongoose).
`src/api/routes`	Configuración de rutas para la API.
`src/config`	Configuración global (como la conexión a la BD).
`src/middlewares`	Middlewares para autenticación y roles.
`src/utils`	Utilidades (por ejemplo, JWT).
`src/index.js`	Archivo principal del servidor.
`.env`	Variables de entorno (configuración secreta).
`README.md`	Documentación del proyecto.
`package.json`	Dependencias del proyecto.

Instalación y Uso

1. Clonar el repositorio

https://github.com/MSS1410/Dilery.git

2. Instalar Dependencias `npm install`

3. Configurar variables de entorno Crea un archivo .env en la raíz del proyecto con los siguientes valores

DB_URL="mongodb+srv://admin:2xPK0e2nULveJR8y@cluster003iidilery.my8zb.mongodb.net/?retryWrites=true&w=majority&appName=Cluster003IIDILERY"

JWT_SECRET=de00696a7cca0f27ecf75f5221cc08fbf06f1942b1b74feee87e9ed1463fb6f07bafac72815a057eb8b86cb6e99f33691c826d7554e09a5a940969

PORT=3056

4.Sembrar la base de datos

node src/seeddatabase.js

5. Ejecutar servidor

node src/index.js El servidor estará disponible en http://localhost:

 Endpoints Disponibles

Método	Endpoint	Descripción
Usuarios		
POST	/Dilery/users/register	Registro de usuarios.
POST	/Dilery/users/login	Autenticación de usuarios.
GET	/Dilery/users/:id	Obtener un usuario por ID.
PUT	/Dilery/users/:id	Actualizar perfil de usuario.
DELETE	/Dilery/users/:id	Eliminar usuario (admin requerido).
Proyectos		
POST	/Dilery/projects	Crear un proyecto (autenticado).
GET	/Dilery/projects	Obtener todos los proyectos.
GET	/Dilery/projects/:id	Obtener un proyecto por ID.
PUT	/Dilery/projects/:id	Actualizar un proyecto (autenticado).
DELETE	/Dilery/projects/:id	Eliminar un proyecto (admin requerido).
Estudios		
POST	/Dilery/studios	Crear un estudio (admin requerido).
GET	/Dilery/studios	Obtener todos los estudios.
GET	/Dilery/studios/:id	Obtener un estudio por ID.
PUT	/Dilery/studios/:id	Actualizar un estudio (admin requerido).
DELETE	/Dilery/studios/:id	Eliminar un estudio (admin requerido).
Exposiciones		
POST	/Dilery/exhibitions	Crear una exposición (autenticado).
GET	/Dilery/exhibitions	Obtener todas las exposiciones.
GET	/Dilery/exhibitions/:id	Obtener una exposición por ID.
PUT	/Dilery/exhibitions/:id	Actualizar una exposición (autenticado).
DELETE	/Dilery/exhibitions/:id	Eliminar una exposición (admin requerido).

 Seguridad

Los tokens JWT son requeridos para rutas protegidas. Agrega el token en el header:

Authorization: Bearer