Proyecto: Migración de Autenticación y Eventos de REST a GraphQL con NestJS

Este proyecto implementa un sistema de autenticación de usuarios y gestión de eventos, migrando de una arquitectura REST tradicional a GraphQL utilizando NestJS.

Módulos

1. Auth (Autenticación y Autorización)

El módulo `Auth` gestiona el registro, inicio de sesión, autorización basada en roles, actualización y eliminación de usuarios.

Funcionalidades:

- **Registro de Usuario (register)**: Permite a cualquier usuario registrarse.
- **Registro de Event Manager (registerEventManager)**: Permite al admin registrar usuarios con rol `event-manager`.
- **Inicio de sesión (login)**: Devuelve un token JWT en una cookie segura.
- **Obtener todos los usuarios (users)**: Solo accesible por administradores.
- **Obtener un usuario por ID (user)**: El usuario puede verse a sí mismo, o si es admin, a cualquiera.
- **Actualizar información del usuario (updateUser)**: Solo se puede actualizar a sí mismo.
- **Eliminar usuario (deleteUser)**: Solo se puede eliminar a sí mismo.
- **Actualizar roles (updateUserRoles)**: Solo accesible por administradores.

Decoradores personalizados:

- `@Auth(...)`: Protege resolvers mediante validación de roles.
- `@GetUser()`: Extrae el usuario autenticado desde el contexto.

---

2. Events (Gestión de eventos)

El módulo `Event` permite la creación, consulta, actualización y eliminación de eventos, con autorización basada en roles.

Funcionalidades:

- **Crear evento (createEvent)**: Requiere rol `event-manager`. El evento se asocia al usuario autenticado.
- **Consultar eventos públicos (findAllEvents)**: Lista todos los eventos con paginación.
- **Consultar eventos propios (findEventsByUser)**: Solo accesible por `admin` o `event-manager`, lista eventos creados por el usuario.
- **Buscar evento sin restricciones (findEventUnrestricted)**: Solo accesible por `admin` o `event-manager`.
- **Buscar evento (findEvent)**: Público.
- **Actualizar evento (updateEvent)**: Solo para `event-manager`, que sea creador del evento.
- **Eliminar evento (removeEvent)**: Accesible por `admin` o `event-manager`.

Tecnologías Usadas

- **NestJS** con **@nestjs/graphql**
- **GraphQL** (via Apollo Server)
- **Autenticación JWT**
- **Autorización por roles personalizados**

Objetivo de la Migración

El propósito de esta migración fue aprovechar la flexibilidad de GraphQL, permitiendo:

- Reducir el overfetching y underfetching de datos.
- Un esquema estrictamente tipado para validación.
- Mejor integración con clientes modernos como Apollo Client o Relay.

Notas Adicionales

- El token JWT es gestionado mediante cookies HTTP-only para mayor seguridad.
- Se aplican controles estrictos para evitar que usuarios no autorizados accedan o modifiquen recursos de otros.
- El proyecto puede ser extendido para otros roles o entidades según sea necesario.

Para ejecutar el proyecto asegúrese de:

1. Tener configurado un backend NestJS con soporte para GraphQL.
2. Configurar las variables de entorno necesarias para JWT, base de datos, etc.
3. Ejecutar:

```
npm install
npm run start:dev
```

Este proyecto ofrece una arquitectura base sólida para sistemas que requieren control de acceso basado en roles y gestión de recursos personalizados como eventos.