Historia de Usuario

Título: Plataforma de gestión de eventos comunitarios

Como organizador de eventos
Quiero una aplicación web que permita a los usuarios registrarse, autenticarse, crear eventos, inscribirse a eventos y administrar la asistencia
Para practicar desarrollo backend y frontend con validaciones, seguridad y consultas avanzadas

Criterios de Aceptación

- **Autenticación y autorización**
  - Los usuarios pueden registrarse y loguearse con email y contraseña.
  - Las contraseñas deben almacenarse encriptadas con **bcrypt**.
  - El login devuelve un **JWT** válido que se usará en las peticiones posteriores.
  - Algunas rutas son públicas (ej: listar eventos), otras privadas (crear evento, inscribirse, gestionar asistencia).
- **Modelado de datos (Postgres con Sequelize)**
  - Usuario: id, nombre, email, password, rol (admin, participante).
  - Evento: id, título, descripción, fecha, ubicación, capacidad, organizadorId (relación con Usuario).
  - Inscripcion: id, estado (pendiente, confirmada, cancelada), usuarioId, eventoId.

MongoDB con Mongoose:

  - Log: acción, usuarioId, recurso, fecha (ej: "Usuario X creó evento Y").
- **Rutas y operaciones**
  - CRUD básico de usuarios (solo admin), eventos (organizador) e inscripciones (participantes).

    - **Consultas avanzadas con Sequelize**:
        - Buscar eventos por ubicación o nombre con Op.like.
        - Filtrar eventos cuya fecha sea mayor a la fecha actual (Op.gt).
        - Consultar usuarios cuyo rol esté dentro de una lista (Op.in).
- **Middlewares**
    - Validar los req.body con **Zod** (registro, login, crear evento, inscribirse).
    - Middleware de verificación de **JWT** para rutas protegidas.
- **Frontend mínimo**
    - Pantalla de registro/login (guarda JWT en localStorage).
    - Listar eventos (público).
    - Crear evento (solo organizador logueado).
    - Inscribirse a un evento (si está autenticado).
    - Ver lista de inscripciones del usuario.
- **Configuración de seguridad**
    - Configurar **CORS** para permitir peticiones desde el frontend.

Tasks

Backend

- Inicializar proyecto con Express + TypeScript.
- Configurar Sequelize con Postgres y definir modelos Usuario, Evento, Inscripcion + relaciones.
- Configurar Mongoose con MongoDB y definir modelo Log.
- Implementar autenticación con JWT y bcrypt.
- Validar entradas con Zod.
- Crear middlewares para validar JWT.
- Implementar rutas REST: usuarios, eventos, inscripciones.
- Implementar consultas avanzadas con Sequelize (Op.gt, Op.like, Op.in).

- Registrar logs en MongoDB en cada acción importante (crear evento, inscribirse, etc).

Frontend

- Pantalla de login/registro.
- Listar eventos (público, con filtros por ubicación/nombre).
- Crear evento (si es organizador).
- Inscribirse a un evento (si está autenticado).
- Listar inscripciones propias.

Configuración

- Configurar **CORS** en Express para permitir requests desde el frontend.
- Definir variables de entorno en .env (DBs, JWT_SECRET, puertos).

Ejemplos de pruebas de aceptación

- **Dado** un usuario registrado, **cuando** haga login con credenciales correctas, **entonces** debe recibir un JWT válido.
- **Dado** un usuario con rol participante, **cuando** intente crear un evento, **entonces** debe recibir un error 403.
- **Dado** que existan eventos en Medellín, **cuando** consulte /eventos?ubicacion=Medellín, **entonces** debe devolver solo esos eventos.
- **Dado** que haya eventos futuros, **cuando** consulte /eventos/futuros, **entonces** debe devolver solo los que tienen fecha mayor a hoy.
- **Dado** un JWT inválido, **cuando** intente inscribirse a un evento, **entonces** debe recibir un error 401.