Sistema de Venta de Entradas a Eventos con Autenticación JWT y Notificación por Email

Se requiere el desarrollo de un sistema basado en microservicios para la gestión y venta de entradas a eventos. El sistema deberá estar compuesto por distintos servicios independientes que se comuniquen entre sí mediante APIs REST, y utilizar JWT (JSON Web Token) para la autenticación y autorización de usuarios.

Se debe utilizar por lo menos tres lenguajes de programación entre estos node y el lengujae que investigo con su grupo

Requisitos del sistema:

1. **Autenticación y Autorización (Servicio de Usuarios):**
    - Los usuarios deben registrarse y autenticarse en el sistema para poder comprar entradas.
    - Una vez autenticado, el usuario recibirá un **token JWT** que será requerido para consumir los servicios protegidos del sistema.
    - El token debe incluir información sobre el usuario, como su ID, rol y vigencia del token.
2. **Gestión de Eventos (Servicio de Eventos):**
    - Permite a los administradores crear, actualizar y eliminar eventos disponibles para la venta.
    - Cada evento tiene información como: nombre, fecha, lugar, capacidad y precio por entrada.
3. **Compra de Entradas (Servicio de Compras):**
    - Los usuarios autenticados pueden consultar los eventos disponibles y realizar la compra de entradas.
    - Al comprar, se debe registrar el evento, la cantidad de entradas y el usuario que realiza la operación.
    - Se debe simular la confirmación del pago (por ejemplo, un endpoint /pagar que marca la compra como pagada).
4. **Notificación por Email (Servicio de Notificaciones):**
    - Al confirmarse el pago de una compra, se debe notificar al

usuario vía correo electrónico.

- Este servicio debe funcionar de forma desacoplada, por ejemplo, escuchando una cola de mensajes desde el servicio de compras.

Características técnicas:

- **JWT:** Utilizado para proteger los endpoints sensibles, especialmente los de compra.
- **Microservicios:** Cada servicio (Usuarios, Eventos, Compras, Notificaciones) debe ser independiente y comunicarse vía HTTP (REST) o mensajería (ej: RabbitMQ, si se desea mayor desacoplamiento).
- **Persistencia:** Cada servicio puede tener su propia base de datos para garantizar independencia (patrón base de datos por servicio).

Para el desarrollo del trabajo se genera n nuevos grupos y se realizaran en las tecnologías expuestas por el miembro de tal manera que cada microservicio esta hecho en un diferente lenguaje de programación