EventHorizon

Plataforma de Eventos y Networking Online.
Un espacio virtual para organizar eventos, conferencias y facilitar el networking entre los participantes.

Jose Fernando Chavez Castañeda.
Sofia López Osuna
Froylan Cisneros Alvizo

Competencia:

https://doodle.com/es/

https://whova.com/

https://propartyplanner.com/

https://eventwo.com/es/

https://www.eventtia.com/es/inicio

https://nuboxmkt.mx/

Stack de Tecnologías a utilizar.

Frontend

React.js.

Backend

Node.js con Express.js: Node.js es eficiente para manejar conexiones concurrentes, lo que es crucial para las funcionalidades en tiempo real.

Socket.IO: Para la comunicación en tiempo real, permitiendo chats, videoconferencias, y actualizaciones en vivo de los eventos.

Base de Datos

MongoDB.

Autenticación y Seguridad

Auth0 o Firebase Authentication: Proporcionan autenticación segura y fácil de implementar con servicios de terceros como Google, Facebook, y LinkedIn.

Herramientas Adicionales

WebRTC: Para videoconferencias peer-to-peer en las salas de networking virtual.

Roles de Usuarios:

1. Organizador de Eventos:

- Descripción: Los organizadores de eventos son responsables de planificar, crear y gestionar eventos en la plataforma.
- Tareas Principales:
- Crear nuevo evento.
- Configurar detalles del evento (fecha, hora, descripción, etc.).
- Administrar inscripciones de participantes.
- Interactuar con participantes durante el evento.

2. Participante del Evento:

- Descripción: Los participantes del evento son usuarios que se inscriben y participan en los eventos organizados en la plataforma.
- Tareas Principales:
- Buscar eventos de interés.
- Registrarse para participar en eventos.
- Acceder a salas de conferencias virtuales.
- Interactuar con otros participantes.

3. Administrador del Sistema:

- Descripción: Los administradores del sistema tienen privilegios para gestionar y mantener la plataforma.
- Tareas Principales:
- Gestionar usuarios y roles.
- Monitorizar la actividad del sistema.

Diagramas:

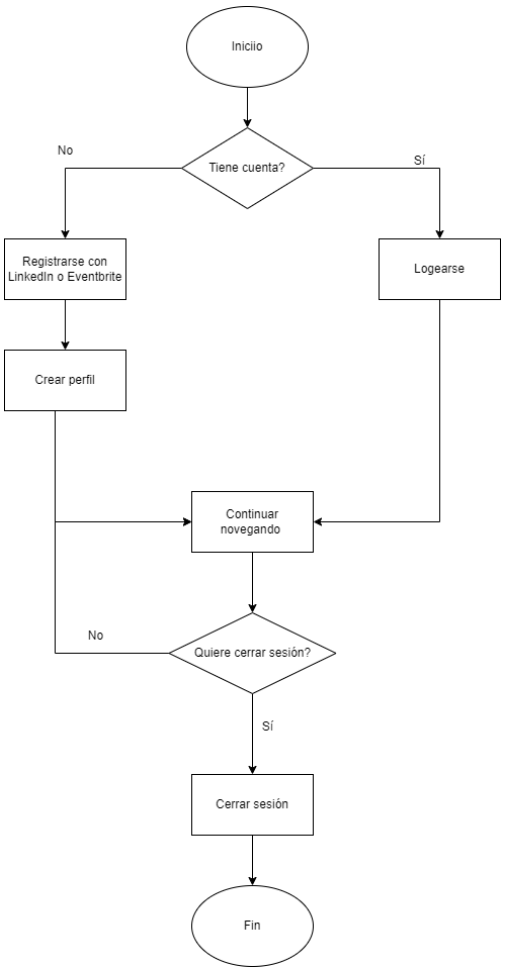


Diagrama para organizador de eventos

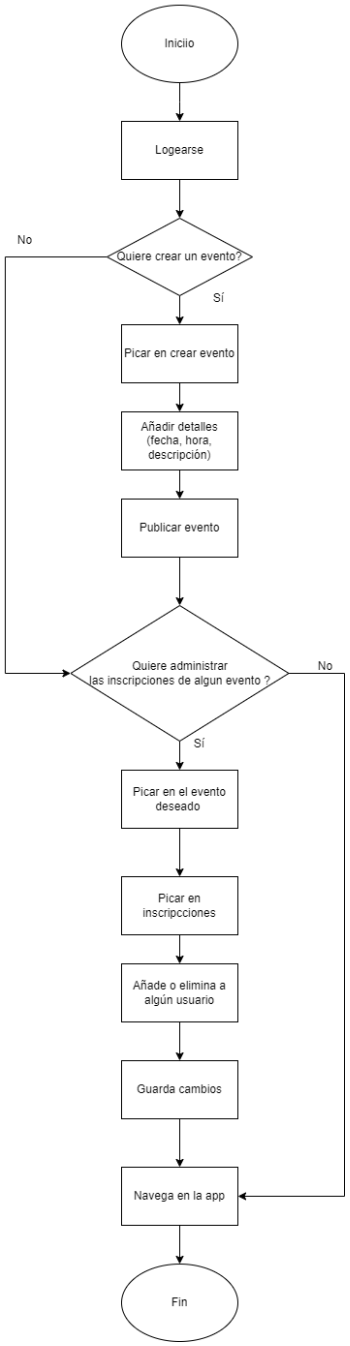
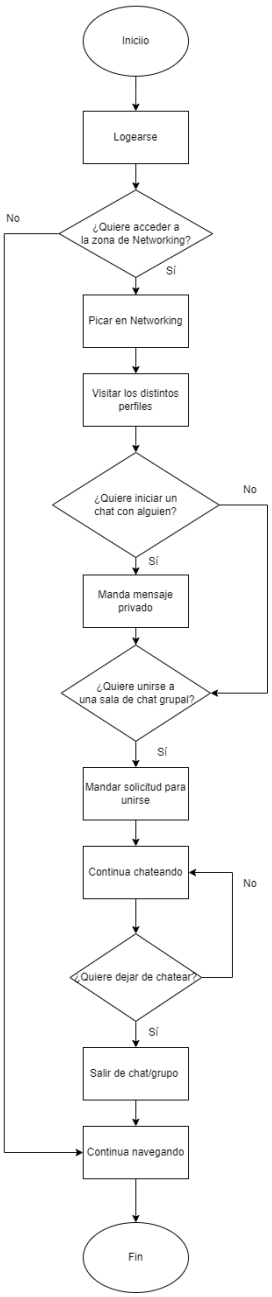
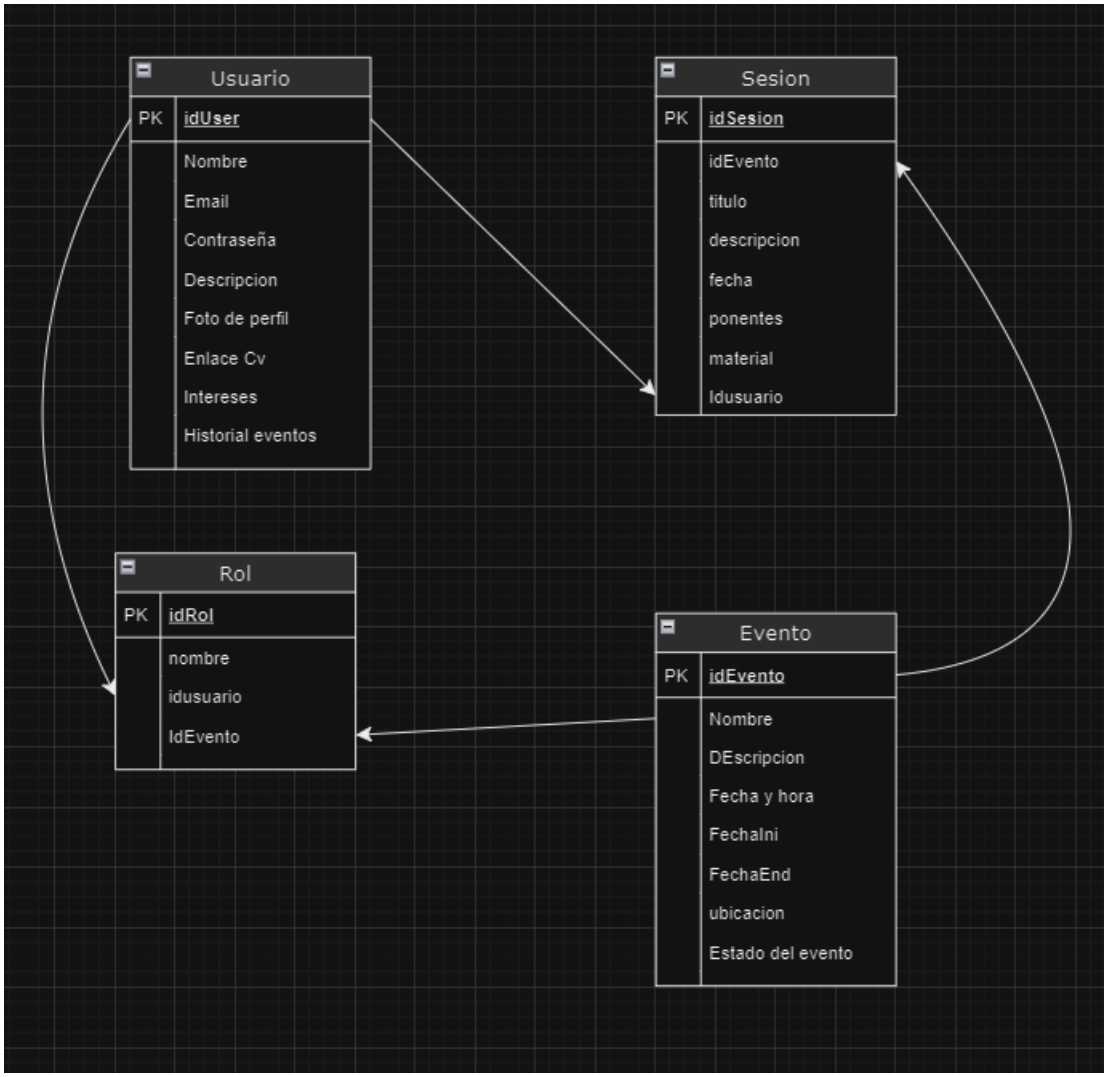


Diagrama para zona de Networking



Entidad-Relacion:

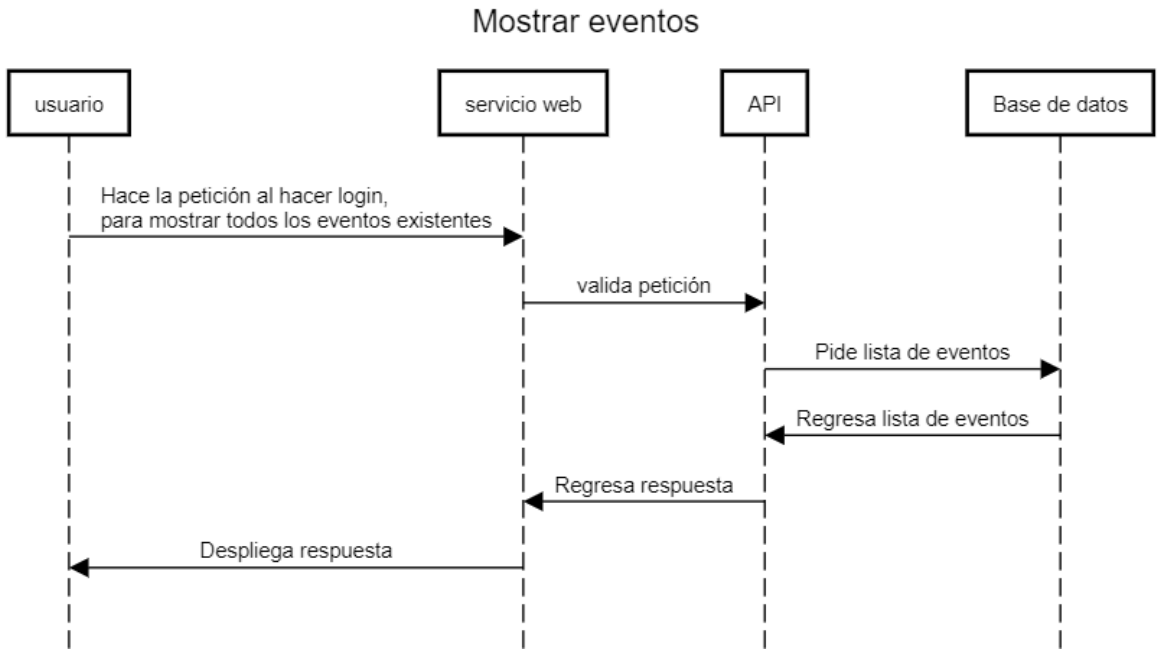


Alcance (temas a cubrir de acuerdo a la guía de aprendizaje) y breve descripción de éste

- MVC y Estructuras de código enfocadas
  al backend: Modelo Vista Controlador y patrones que nos servirán para organizar código de backend
- Frameworks para el Backend: Entornos de desarrollo que facilitarán la creación de el proyecto
- Webpack: Herramienta para empaquetar y minificar archivos JavaScript y otros recursos.
- TypeScript: Lenguaje de programación que añade tipos a JavaScript.
- Handlebars: Plantillas para generar HTML dinámico.

- Paradigma Orientado a Objeto y
  Patrones de diseño: Se utilizará el enfoque POO para poder generar soluciones que se usan de manera común en la industria para resolver problemáticas que se puedan generar.
- Testing e integración continua: Pruebas para asegurar la calidad del código.
- Plataformas para despliegue de
  aplicaciones: Servicios para alojar y ejecutar aplicaciones web

Diagrama de secuencia:



Módulos:

-evento: (lo podrán utilizar usuarios registrados como organizadores de eventos) Definir funciones para la creación de eventos, filtro de eventos, inscripción, etc.

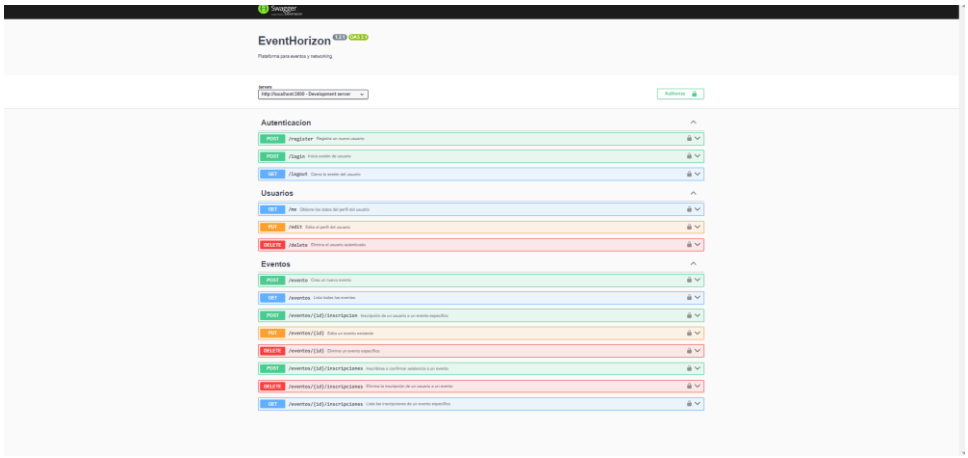
-usuarios: (disponible para todos los usuarios) creación de usuario, log in, actualización de información, solicitar cambio de rol, etc.

-administradores: (disponible solo para administradores) dar de baja a un usuario, dar de alta a un usuario, inscribir o desinscribir a un usuario de un evento, eliminar evento, agregar evento, modificar información del evento, etc.

-autenticación: (disponible solo para administradores) funcionalidad para autenticar a los usuarios,

Swagger

http://localhost:3000/api-docs/



Explicación

Se definen rutas y operaciones utilizando formato YAML, los comentarios están ubicados encima de cada ruta y describen la operación, sus parámetros, respuestas y otros detalles

Se importa Swager UI y Swagger Docs para poder configurar la interfaz de Swagger y proporcionar la documentación de la API en formato JSON

Finalmente se genera la documentación usando los comentarios

Autenticación y permisos

Se tienen dos Middlewares de autenticación, en verificarToken se evalua si se proporciona un token de autorización en la solicitud, si sí fue proporcionado entonces se intenta decodificar el token usando una clave secreta. Si la verificación es exitosa entoces se agrega el usuario al Request y se llama a next() para pasar a la siguiente función. Si no se da un token o para que la verificación no sea exitosa se envía un código 403 de error, o 401 seguido de un mensaje indicando el error.

En establecerContextoAutentucación obtenemos el token de autorización del encabezado o de las cookies, Si se encuentra un token, intenta verificar y decodificar el token. Si es exitoso, establece unas variables en Response. De manera contraria, si la verificación falla, se establece userLoggedIn en false y se llama a next()

Ejemplo:

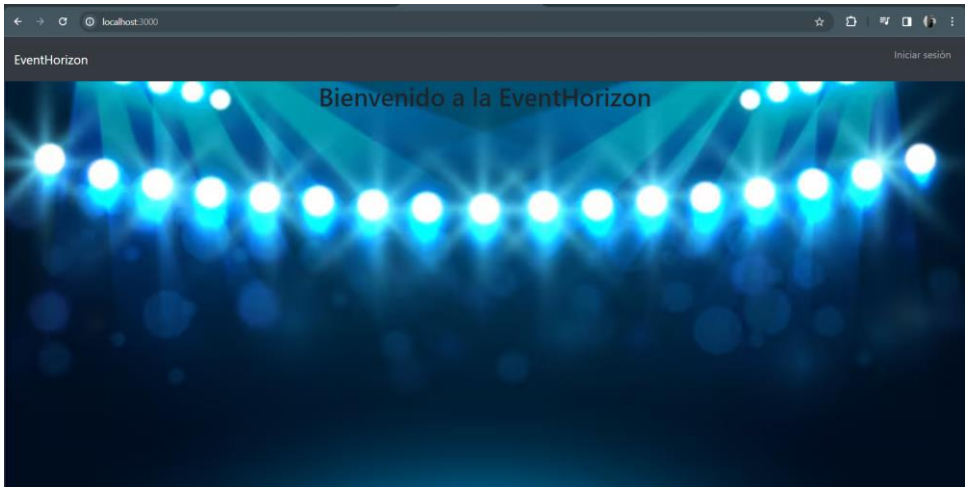
En al caso de nuestro proyecto, cada vez que se quieran ver los eventos disponibles el usuario debe estár autenticado forzosamente, se hace llamado a esta ruta: router.get('/eventos', verificarToken, listarEventos);

Si el usuario si está autenticado entonces sin problemas se le muestran los diferentes eventos futuros que hay en la plataforma, de otro modo si el usuario no está autenticado entonces no se le podrán

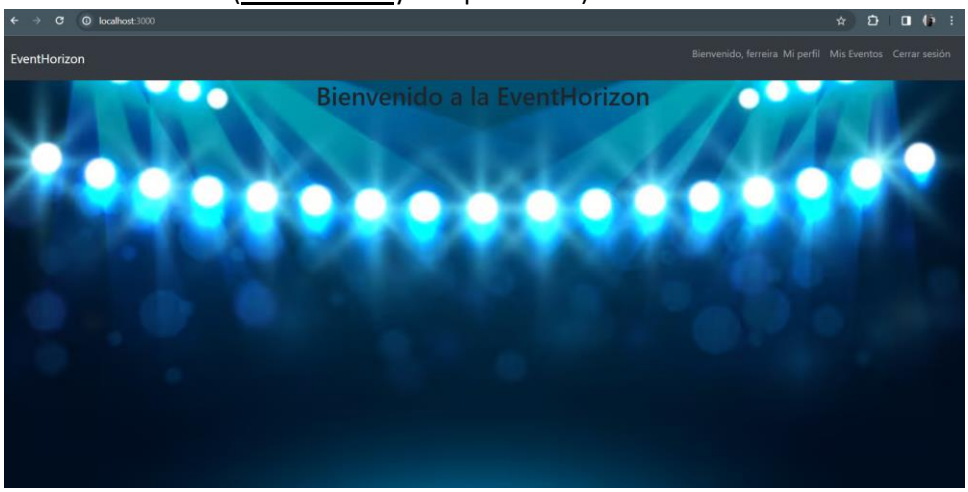
mostrar los eventos y se le muestra un mensaje de token no válido.

Capturas de pantalla de distintas respuestas:

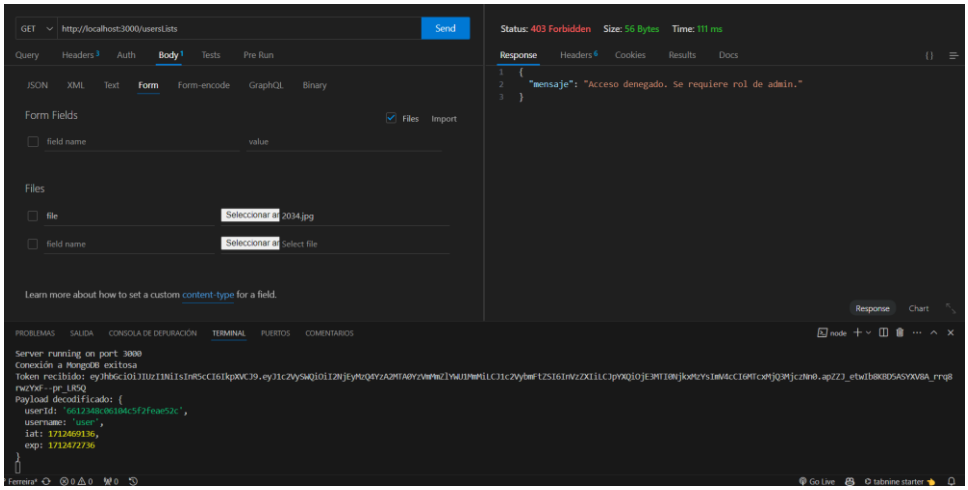
Home sin autenticar(No autenticado):



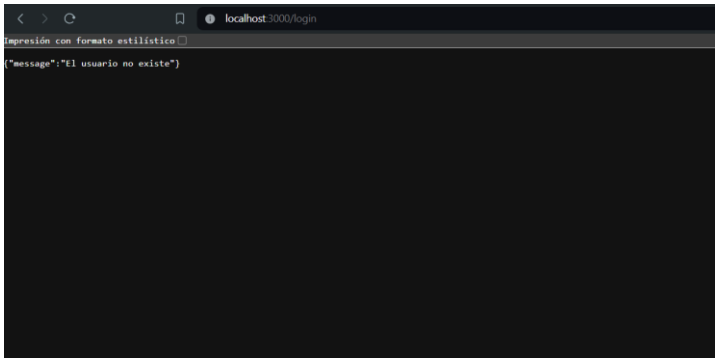
Home autenticado (Autenticado y con permisos:):



Autenticado sin permisos:



Mensaje de error al intentar autentificarse con un perfil no autorizado (front en proseso)


localhost:3000/login
Impresión con formato estilístico
{"message":"El usuario no existe"}