FUNDACIÓN SALVADOR DEL MUNDO – FUSALMO

CENTROS JUVENILES SALESIANOS
FUSALMO
COMPARTE • SUEÑA • TRANSFORMA

CURSO:

Desarrollador Backend

NOMBRE DE PROYECTO:

EventFlow

EQUIPO:

N° 6 - Irkana

INTEGRANTES:

Integrantes
Angie Michelle Díaz Urias
Fabio Alejandro Ordoñez Cerritos
José Rodolfo Vargas Blanco

FECHA DE ENTREGA:

Miércoles 3 de julio de 2024

ÍNDICE

Contenido

INTRODUCCIÓN

En el mundo empresarial actual, la coordinación y la colaboración son esenciales para el éxito de cualquier equipo de trabajo. Las agendas ocupadas, las reuniones constantes y los plazos ajustados requieren una herramienta eficaz que permita a los equipos organizarse y comunicarse de manera eficiente. En este contexto, presentamos nuestra aplicación web de calendario colaborativo, diseñada específicamente para satisfacer las necesidades de empresas y equipos de trabajo pequeños.

Nuestra aplicación permite a los usuarios crear, eliminar y editar eventos fácilmente, asegurando que todos los miembros del equipo estén al tanto de las actividades y compromisos importantes. La característica clave de nuestra herramienta es su enfoque en la colaboración: cada evento creado incluye detalles sobre el usuario que lo ha creado, lo que facilita la responsabilidad y la transparencia dentro del equipo.

Al utilizar nuestra aplicación, los equipos pueden:

- Coordinar Eventos: Crear y gestionar eventos de manera sencilla, garantizando que todos los miembros del equipo conozcan las reuniones, plazos y otras actividades importantes.
- Colaborar Eficazmente: Al ser una herramienta colaborativa, permite que múltiples usuarios interactúen con el calendario, añadiendo, modificando o eliminando eventos según sea necesario.
- Optimizar el Tiempo: Minimizar el tiempo perdido en la organización de reuniones y actividades, permitiendo que los equipos se enfoquen en sus tareas principales y sean más productivos.

OBJETIVOS

Objetivo general

Desarrollar una aplicación web de calendario colaborativo, con funcionalidades como la creación, visualización y gestión de eventos, que ofrezca a los usuarios una interfaz intuitiva para una experiencia de usuario óptima.

Objetivos específicos

- Implementar el registro e inicio de sesión de usuario desarrollando endpoints para el registro de nuevos usuarios, asegurando la verificación de datos y la creación de perfiles, e implementar un sistema de inicio de sesión que permita la autenticación de usuarios registrados mediante el uso de JWT (JSON Web Tokens).

- Desarrollar la generación de reportes, crear endpoints que permitan la generación de reportes sobre eventos, que incluyen métricas como la cantidad de eventos creados, eventos por usuario, y eventos por periodo de tiempo, y asegurar que los reportes puedan ser exportados en formatos comunes como PDF o Excel, facilitando su análisis y distribución.

DESCRIPCIÓN DEL PROYECTO

Problemática

Esta aplicación se enfrenta a varias problemáticas que los usuarios pueden experimentar con la gestión de sus calendarios y eventos. Estas problemáticas incluyen:

- **Organización y Gestión del Tiempo:**
    a. Los usuarios a menudo tienen dificultades para organizar y gestionar sus eventos y tareas diarias, lo que puede llevar a olvidos, superposiciones de eventos y una planificación ineficaz.

    b. La falta de una herramienta centralizada para registrar y gestionar eventos puede resultar en la dispersión de información en múltiples plataformas y dispositivos.

- **Accesibilidad y Sincronización de Datos:**
    c. La necesidad de acceder a la información del calendario desde diferentes dispositivos y ubicaciones es crucial. La falta de sincronización en tiempo real puede causar discrepancias y confusiones.

    d. Los usuarios necesitan que sus calendarios estén disponibles en todos sus dispositivos, independientemente de la plataforma que estén utilizando.

- **Privacidad y Seguridad de la Información:**
    e. La seguridad de la información personal y la privacidad de los eventos son preocupaciones importantes. Los usuarios necesitan estar seguros de que sus datos están protegidos contra accesos no autorizados.

f. Las aplicaciones deben cumplir con regulaciones de privacidad como el GDPR, especialmente si se manejan datos personales sensibles.

- **Personalización y Facilidad de Uso:**
  g. Cada usuario tiene diferentes necesidades y preferencias en cuanto a la gestión de su tiempo. La falta de personalización puede hacer que la herramienta no sea adecuada para todos.

  h. La interfaz de usuario debe ser intuitiva y fácil de usar, para que incluso personas con poca experiencia tecnológica puedan manejarla sin dificultades.

Solución propuesta

La solución propuesta es una aplicación de calendario basada en el stack MERN (MongoDB, Express.js, React.js y Node.js) que aborda las problemáticas mencionadas mediante las siguientes características y funcionalidades:

- **Centralización de Eventos y Gestión del Tiempo:**
  a. La aplicación permite a los usuarios crear, editar y eliminar eventos en un calendario personal. Cada usuario tendrá su propio calendario, asegurando que los eventos sean independientes y personalizados.

  b. Se pueden establecer recordatorios y notificaciones para eventos importantes, ayudando a los usuarios a gestionar mejor su tiempo y evitar olvidos.

- **Accesibilidad y Sincronización:**
    c. Utilizando tecnologías web modernas, la aplicación será accesible desde cualquier dispositivo con un navegador web, incluyendo computadoras de escritorio, laptops, tablets y smartphones.

    d. La sincronización en tiempo real se implementará para asegurar que los cambios realizados en un dispositivo se reflejen inmediatamente en todos los dispositivos del usuario.

- **Privacidad y Seguridad:**
    e. La autenticación y autorización se gestionarán utilizando JWT (JSON Web Tokens) para asegurar que solo los usuarios autenticados puedan acceder a sus calendarios y eventos.

    f. Los datos se almacenarán en una base de datos MongoDB, con encriptación en tránsito y en reposo para proteger la información sensible.

    g. La aplicación cumplirá con las regulaciones de privacidad aplicables, asegurando que los datos personales de los usuarios se manejen de manera responsable y segura.

- **Personalización y Experiencia de Usuario:**
    h. La aplicación ofrecerá opciones de personalización, como diferentes vistas de calendario (diaria, semanal, mensual).

    i. Se desarrollará una interfaz de usuario intuitiva utilizando React.js, asegurando que la navegación y la gestión de eventos sean sencillas y

directas. Las librerías y componentes de UI modernos como Material-UI o Bootstrap pueden ser utilizados para mejorar la experiencia de usuario.

ESPECIFICACIONES DE REQUISITOS

Requisitos Funcionales

- Registro de Usuarios
  - El sistema debe permitir a los usuarios registrarse proporcionando nombre, correo electrónico y contraseña, además de colocar nuevamente su contraseña para verificarla.
  - El sistema debe verificar que el correo electrónico no esté registrado previamente.
  - El sistema debe verificar que las contraseñas ingresadas sean la misma.

- Inicio de Sesión
  - El sistema debe permitir a los usuarios iniciar sesión proporcionando correo electrónico y contraseña.
  - El sistema debe autenticar a los usuarios utilizando JWT (JSON Web Tokens).

- Gestión de Eventos
  - Crear eventos: El usuario debe poder crear eventos proporcionando título, fecha de inicio, fecha de fin y notas.
  - Consultar eventos: El usuario debe poder ver la lista de todos los eventos creados por todos los usuarios registrados.
  - Editar eventos: El usuario debe poder editar los eventos que ha creado, y no tener permisos de editar un evento ajeno.
  - Eliminar eventos: El usuario debe poder eliminar los eventos que ha creado, y no tener permisos de eliminar un evento ajeno.

- Diferenciación de Eventos
  - Los eventos creados por el usuario registrado deben mostrarse en un color específico.
  - Los eventos creados por otros usuarios deben mostrarse en un color diferente a los del usuario registrado.

- Generación de Reportes
    - El sistema debe permitir la generación de reportes que incluyan los datos principales de cada evento para diferenciarlos de los demás.
    - Los reportes deben ser exportados en formato PDF.

- Recuperación de Contraseña
    - El sistema debe permitir a los usuarios solicitar la recuperación de contraseña proporcionada por su correo electrónico.
    - El sistema debe enviar un correo electrónico con un enlace para restablecer la contraseña.
    - El enlace para restablecer la contraseña debe expirar después de un período determinar por motivos de seguridad.
    - El sistema debe permitir a los usuarios restablecer su contraseña utilizando el enlace proporcionado.
    -

Requisitos No Funcionales

- Seguridad
    - Las contraseñas de los usuarios deben ser almacenadas de forma segura utilizando técnicas de hashing.
    - La autenticación de usuarios debe realizarse utilizando JWT.
    - El acceso a las funcionalidades del sistema debe estar restringido a usuarios autenticados.
    - El sistema debe enviar correos electrónicos de recuperación de contraseña de forma segura, asegurando que los enlaces para restablecer la contraseña expiren después de un tiempo.

- Rendimiento
    - El sistema debe poder manejar múltiples usuarios y eventos sin afectar el rendimiento.
    - Las operaciones de creación, consulta, edición y eliminación de eventos deben ser rápidas y eficientes.
    - Las solicitudes de recuperación de contraseña deben ser procesadas rápidamente para proporcionar una buena experiencia de usuario.

- Escalabilidad
    - El sistema debe ser escalable para soportar un número creciente de usuarios y eventos sin pérdida de rendimiento.

- El sistema debe ser escalable para adquirir mejoras con el tiempo.

- Usabilidad
  - La interfaz de usuario debe ser intuitiva y fácil de usar.
  - Los eventos deben estar claramente diferenciados por colores según el creador del evento.
  - El proceso de recuperación de contraseña debe ser sencillo y directo para los usuarios.

- Disponibilidad
  - El sistema debe estar disponible 24/7, con un tiempo de inactividad mínimo para mantenimientos planificados.

- Compatibilidad
  - El sistema debe ser compatible con los navegadores web modernos.
  - La aplicación debe ser responsive, es decir, debe funcionar correctamente en dispositivos de diferentes tamaños de pantalla (móviles, tablets, desktops).

- Mantenibilidad
  - El código del sistema debe estar bien documentado para facilitar su mantenimiento y futuras actualizaciones.
  - El sistema debe seguir buenas prácticas de desarrollo para asegurar su mantenibilidad.

DISEÑO DEL SISTEMA

Arquitectura del Sistema

La arquitectura de EventFlow puede describirse como una aplicación web basada en el modelo Cliente-Servidor, utilizando las siguientes tecnologías:

- Cliente (Frontend):
  - Framework: React.
  - Comunicación con el servidor: HTTP a través de una API REST.
  - Responsabilidad: Interfaz de usuario, envío de solicitudes HTTP al servidor, presentación de datos y manejo de interacciones del usuario.

- Servidor (Backend):
  - Framework: Node.js con Express.
  - API REST: Endpoints para autenticación, gestión de usuarios, gestión de eventos y generación de reportes.
  - Base de datos: MongoDB, utilizando Mongo Atlas para la gestión de datos.
  - Responsabilidad: Procesar solicitudes HTTP, interactuar con la base de datos, realizar operaciones CRUD, autenticar usuarios y generar tokens.

- Base de datos:
  - Sistema de gestión de base de datos: MongoDB.
  - Modelo de datos: Colecciones para usuarios y eventos.
  - Responsabilidad: Almacenar y gestionar datos de usuarios y eventos.

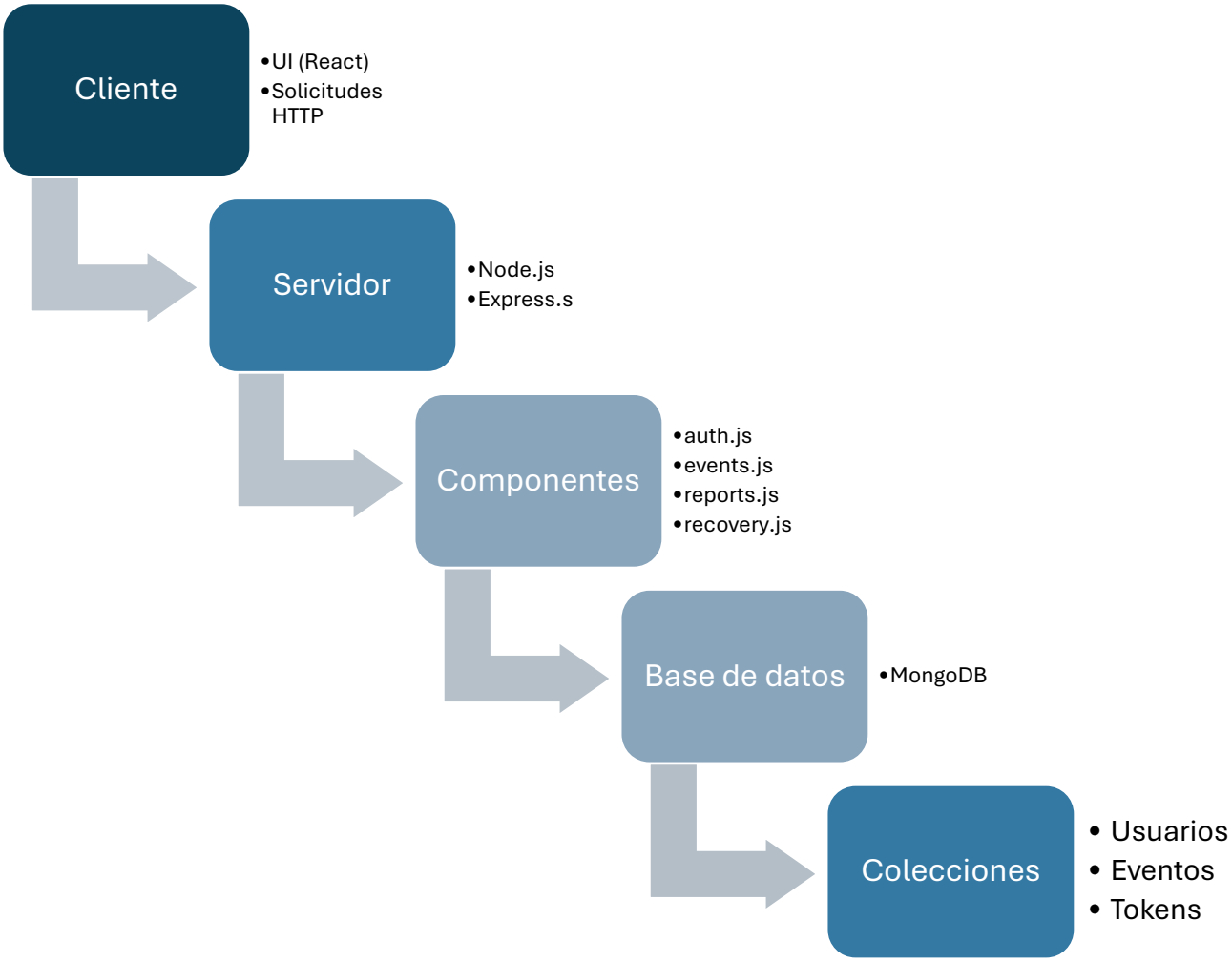
Flujo de datos:

- El cliente envía solicitudes HTTP al servidor para realizar operaciones como registro, inicio de sesión, creación, edición, eliminación de eventos y generación de reportes.

- El servidor procesa las solicitudes, interactúa con la base de datos según sea necesario y devuelve las respuestas al cliente.

- La base de datos almacena y recupera datos según las operaciones solicitadas por el servidor.

Diagrama de Componentes



- **Cliente:**
  - UI (React): La interfaz de usuario que permite a los usuarios interactuar con la aplicación.
  - Solicitudes HTTP: Envío de solicitudes al servidor para realizar diversas operaciones.

- **Servidor:**
  - Node.js: Entorno de ejecución para el backend de JavaScript.
  - Express.js: Framework para manejar rutas y middleware.
  - auth.js: Componente para manejo de autenticación (registro, inicio de sesión, revalidación de tokens).

- events.js: Componente para manejo de eventos (crear, consultar, editar, eliminar eventos).
- reports.js: Componente para la generación de reportes en formato PDF.

- **Base de datos:**
  - MongoDB: Base de datos NoSQL para almacenar información de usuarios y eventos.

  - **Colecciones:**
    - Usuarios: Almacena datos de los usuarios registrados.
    - Eventos: Almacena los eventos creados por los usuarios.
    - Tokens: Almacena tokens temporales para la recuperación de contraseña de los usuarios

PLAN DE GESTIÓN DEL PROYECTO

Cronograma

N°	Nombre de Actividad	Junio									Julio		
		22	23	24	25	26	27	28	29	30	1	2	3
Fase 1: Definición de propuesta de desarrollo													
1	Definición de objetivos del proyecto												
2	Desarrollo de actividad a entregar en la primera jornada												
3	Análisis pertinente de la idea a desarrollar												
4	Establecer tecnologías y herramientas a utilizar												
5	Diseño de la arquitectura del sistema y realización de esquemas												
Fase 2: Implementación y Desarrollo													
6	Configuración del entorno de desarrollo (Node.js, Express, MongoDB)												
7	Creación de estructura del proyecto y repositorio												
8	Desarrollo de la conexión con la base de datos de MongoDB												
9	Implementar API REST												
10	Desarrollar modelos y esquemas de la base de datos												
11	Creación de endpoints para la gestión de eventos del calendario												
12	Implementar validaciones de usuario												

13	Desarrollar funcionalidad de registro y autenticación de usuarios												
14	Encriptar contraseñas y generar JWT												
15	Implementar establecimiento de contraseñas												
16	Desarrollar funcionalidad de generación de reportes con filtrado												
17	Revisión y ajuste de la implementación inicial												
Fase 3: Pruebas y Ajustes													
18	Escribir pruebas unitarias para componentes clave												
19	Configurar herramientas de prueba y entorno de pruebas												
20	Ejecutar pruebas iniciales y corregir errores												
21	Continuar con pruebas unitarias												
22	Realizar pruebas de integración												
23	Ajustes y correcciones basados en resultados de pruebas												
24	Realizar pruebas de rendimiento y carga												
25	Optimizar código y consultas a la base de datos												
26	Revisar y mejorar la seguridad de la aplicación												
27	Pruebas finales y verificación de todas las funcionalidades												
28	Preparar la documentación técnica y de usuario												
29	Realizar pruebas finales y preparar la versión para presentación												

Fase 4: Presentación y seguimiento														
30	Preparar la presentación del proyecto													
31	Crear diapositivas y material de apoyo													
32	Ensayar la presentación y preparar respuestas a posibles preguntas													
33	Revisión final del proyecto													
34	Asegurarse de que todo esté funcionando correctamente													
35	Últimos ajustes y preparación del entorno para la demostración													
36	Día de presentación a usuario final													

Asignación de Tareas

N°	Nombre de Actividad	Asignación
Fase 1: Definición de propuesta de desarrollo		
1	Definición de objetivos del proyecto	Angie, Fabio y José
2	Desarrollo de actividad a entregar en la primera jornada	Angie, Fabio y José
3	Análisis pertinente de la idea a desarrollar	Angie, Fabio y José
4	Establecer tecnologías y herramientas a utilizar	Angie, Fabio y José
5	Diseño de la arquitectura del sistema y realización de esquemas	Angie, Fabio y José
Fase 2: Implementación y Desarrollo		
6	Configuración del entorno de desarrollo (Node.js, Express, MongoDB)	Angie
7	Creación de estructura del proyecto y repositorio	Angie
8	Desarrollo de la conexión con la base de datos de MongoDB	Angie
9	Implementar API REST	Fabio
10	Desarrollar modelos y esquemas de la base de datos	Fabio
11	Creación de endpoints para la gestión de eventos del calendario	Fabio
12	Implementar validaciones de usuario	José
13	Desarrollar funcionalidad de registro y autenticación de usuarios	José
14	Encriptar contraseñas y generar JWT	José
15	Implementar establecimiento de contraseñas	Fabio
16	Desarrollar funcionalidad de generación de reportes con filtrado	Fabio
17	Revisión y ajuste de la implementación inicial	Fabio
Fase 3: Pruebas y Ajustes		
18	Escribir pruebas unitarias para componentes clave	José
19	Configurar herramientas de prueba y entorno de pruebas	José
20	Ejecutar pruebas iniciales y corregir errores	José
21	Continuar con pruebas unitarias	Fabio
22	Realizar pruebas de integración	Fabio

23	Ajustes y correcciones basados en resultados de pruebas	Fabio
24	Realizar pruebas de rendimiento y carga	Angie
25	Optimizar código y consultas a la base de datos	Angie
26	Revisar y mejorar la seguridad de la aplicación	Angie
27	Pruebas finales y verificación de todas las funcionalidades	José
28	Preparar la documentación técnica y de usuario	José
29	Realizar pruebas finales y preparar la versión para presentación	José
Fase 4: Presentación y seguimiento		
30	Preparar la presentación del proyecto	Angie
31	Crear diapositivas y material de apoyo	Angie
32	Ensayar la presentación y preparar respuestas a posibles preguntas	Angie
33	Revisión final del proyecto	Angie, Fabio y José
34	Asegurarse de que todo esté funcionando correctamente	Angie, Fabio y José
35	Últimos ajustes y preparación del entorno para la demostración	José y Fabio
36	Día de presentación a usuario final	Angie, Fabio y José

PLAN DE PRUEBAS

Estrategia de Pruebas

El objetivo de este apartado es garantizar que la aplicación EventFlow funcione de manera correcta, eficiente y segura mediante la implementación de una estrategia de pruebas integral que aborde todos los aspectos funcionales y no funcionales del sistema.

- **Pruebas funcionales**
    - Pruebas unitarias: verificar que las funciones individuales del código funcionen según lo esperado.
    - Pruebas de integración: verificar que los módulos del sistema interactúen entre sí.
    - Pruebas de sistema: verificar el sistema en su totalidad para asegurar que cumple con los requisitos específicos.
    - Pruebas de regresión: verificar que las nuevas actualizaciones o cambios no hayan introducido errores en el sistema.

- **Pruebas no funcionales**
    - Pruebas de rendimiento: evaluar el rendimiento del sistema bajo diferentes cargas, como crear eventos de manera simultánea.
    - Pruebas de seguridad: identificar y corregir vulnerabilidades de seguridad.
    - Pruebas de usabilidad: evaluar la facilidad de uso y la experiencia del usuario.
    - Pruebas de compatibilidad: asegurar que el sistema funcione correctamente en diferentes navegadores y dispositivos.

Casos de Prueba

- Casos de prueba para operaciones aritméticas básicas:

N°	Caso de Prueba	Descripción	Entrada	Salida	Resultado esperado
1	Creación de evento	Verificar que un usuario puede ver todos los eventos	Título, fecha de inicio, fecha de fin y nota	Redirección a la página de calendario	Se muestran los eventos creados por el usuario en un color para diferenciar los otros eventos
2	Consulta de eventos	Verificar que un usuario puede ver todos los eventos	Ninguna	Redirección a la página de calendario	Se pueden ver todos los eventos, diferenciados por colores
3	Edición de evento	Verificar que un usuario puede editar su evento	Click en el evento, nuevos datos de título, fecha de inicio, fecha de fin y nota.	Redirección a la página de calendario	El evento es actualizado correctamente mostrando los cambios al momento de realizarlos
4	Eliminación de evento	Verificar que un usuario puede eliminar su evento	Click en el botón de eliminación de evento	Redirección a la página de calendario	El evento es eliminado con correctamente y desaparece de la lista de eventos
5	Generación de reportes	Verificar que el sistema puede generar un reporte de eventos	Ninguna	Archivo PDF con la lista de eventos	Se genera y descarga un archivo PDF con la lista de eventos correctamente
6	Recuperación de contraseña	Verificar que un usuario puede recuperar su contraseña	Correo electrónico válido	Correo electrónico enviado	Se envía un correo electrónico con instrucciones para restablecer la contraseña del usuario
7	Prueba de rendimiento	Evaluar el rendimiento del sistema bajo carga	Simulación de 100 usuarios creando eventos	Respuesta del sistema	Se debe generar una excepción o un error indicando que

			simultáneamente		no se puede dividir por cero
8	Prueba de Seguridad	Verificar la seguridad del sistema de autenticación	Intentos de acceso no autorizado y ataques comunes	Mensajes de error o bloqueo del acceso	El sistema bloquea el acceso no autorizado y mitiga los ataques de seguridad
9	Prueba de usabilidad	Evaluar la facilidad de uso y la experiencia del usuario	Realización de tareas comunes (registro, inicio de sesión, creación de eventos)	Feedback de los usuarios	Los usuarios encuentran la interfaz intuitiva y fácil de usar
10	Prueba de compatibilidad	Asegurar que el sistema funcione correctamente en diferentes navegadores y dispositivos	Pruebas en Chrome, Brave, Edge y dispositivos móviles	Comportamiento consistente en todos los navegadores y dispositivos	La aplicación funciona correctamente y de manera consistente en todos los navegadores y dispositivos probados.

- Casos de prueba para gestión de usuarios:

N°	Caso de prueba	Descripción	Entrada	Salida	Resultado esperado
1	Registro de usuario	Verificar que un usuario puede registrarse con un correo y contraseña	Correo electrónico y contraseña válidos	Mensaje de éxito y redirección a la página de calendario	El usuario es registrado correctamente y se muestra un mensaje de éxito
2	Inicio de sesión	Verificar que un usuario registrado puede iniciar sesión	Nombre, correo electrónico y contraseñas válidos	Redirección a la página de calendario	El usuario es autenticado correctamente y redirigido a la página de calendario

MANUALES DE USUARIO

Guía de Instalación

1. **Introducción:**

   Esta guía proporciona instrucciones paso a paso para la instalación, configuración y ejecución de EventFlow, además de presentar los requisitos, pasos para comprobar que EventFlow se ejecuta correctamente y realizar un despliegue exitoso. Se incluyen además algunas recomendaciones de seguridad para la aplicación.

   EventFlow es una aplicación web que permite de manera colaborativa la gestión de eventos en un calendario, la gestión de eventos comprende las siguientes acciones creación de un evento, edición de un evento, eliminar un evento, y de manera predeterminada la visualización de los eventos.

2. **Requisitos previos:**

   Para instalar y utilizar EventFlow, se necesita lo siguiente:

   - Node.js: https://nodejs.org/en/download/package-manager
     - npm, es parte de la instalación por defecto de Node.js
     - yarn, la instalación y uso de yarn es opcional
   - Editor de código de opcional, puede utilizar Visual Studio Code, Sublime Text, entre otros
   - Es necesario tener una cuenta de GitHub, es necesario para el despliegue de la aplicación
   - Poseer una cuenta en MongoDB Atlas (es necesario para poder hacer el despliegue de la aplicación), en caso de poseer una cuenta debe registrarse y realizar las configuraciones necesarias para la base de datos: https://www.mongodb.com/es/cloud/atlas/register
   - Servicio de alojamiento web, esto es necesario para el despliegue de la aplicación, en esta guía se utilizará Heroku, pero es posible utilizar otros servicios como Netlify o Vercel. https://signup.heroku.com/

3. **Instalación:**

- Clonar el repositorio que contiene el backend del proyecto:

```
git clone
https://github.com/angiediaz202/EventFlow-Irkana.git
```

- Instalación de dependencias, navegue hasta la carpeta del proyecto clonado y ejecute el siguiente comando:

```
cd EventFlow-Irkana
npm install
```

- Clonar el repositorio que contiene el frontend del proyecto:

```
git clone
https://github.com/angiediaz202/EventFlow-FrontEnd-Irkana.git
```

- Instalar las dependencias, nuevamente es necesario navegar hasta la carpeta del proyecto, verificar que se encuentre en la ruta correcta del proyecto

```
cd EventFlow-FrontEnd-Irkana
npm install
```

- Lo siguiente a realizar es configurar las variables de entorno en el backend, principalmente agregar el string connection para su cuenta de MongoDB Atlas,

```
PORT= 4000
BASE_URL= http://localhost:3000
DB_CNN = mongodb+srv://myUserName:myDbPassword
@cluster0test.dbwyx0d.mongodb.net/db_flowtime?retryWrites=true&w=maj
ority&appName=Cluster0Test
SECRET_JWT_SEED=addYourSecretWord
RECOVERY_JWT_SEED= addYourSecretWordForRecoveryPassword
```

EventFlow realiza el envío de un correo de electrónico para la recuperación de contraseña por lo cual también es necesario modificar el código del envío de correo con el enlace para la recuperación de contraseña en el archivo recovery.js que se encuentra en la siguiente ruta backend/controllers lo siguiente:

```
// Configuración del emisor del correo
const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: 'myEmail@gmail.com',
        pass: 'applicationCode'
    }
});
```

```
// Cuerpo del email de recuperación de contraseña
        const mailOptions = {
            from: 'myEmail@gmail.com',
            to: email,
            subject: 'Recuperación de contraseña',
            text: `Hola ${usuario.name},
                    Para reestablecer tu contraseña de EventFlow,
                    por favor haz click el siguiente enlace:
                    ${resetLink}`
        };
```

4. **Pruebas:**

   Para poner en ejecución EventFlow en un entorno local o de desarrollo es necesario abrir dos ventanas de terminal una para el backend y otra para el frontend.

   - Pasos para ejecutar el backend
     - En una ventana de terminal navegar hasta el proyecto y realizar el siguiente comando:

       ```
       cd EventFlow-Irkana
       nodemon index.js
       ```

     - En otra ventana de terminal navegar hasta el proyecto y realizar el siguiente comando:

       ```
       cd EventFlow-FrontEnd-Irkana
       yarn run dev
       ```

   - Una vez EventFlow está en ejecución, para poder probar la conexión con el backend se puede utilizar herramientas como Postman o Thunder Client de Visual Studio Code. Lo siguiente son los endpoints del backend de eventFlow, se muestran de forma breve ya que en la guía de usuario final se muestra cómo se usan cada uno de los endpoints.

url base: http://localhost:3000 endpoints de autenticación del usuario /api/auth endpoints del crud de eventos /api/events endpoints para la recuperación de la contraseña /api/recovery

5. **Despliegue:**

Como se hizo mención anteriormente se utilizará Heroku para el despliegue de la aplicación.

- En caso de no tener una cuenta de Heroku, debe registrarse en el siguiente enlace: https://signup.heroku.com/, se recomienda utilizar una cuenta asociada a GitHub, se debe configurar y añadir un método de pago.
- Una vez se haya registrado y añadido un método de pago puede configurar su aplicación para el despliegue.
- Acceder a Heroku, en la página principal hacer clic en la opción de crear una aplicación (Create a new app).
- Escoger un nombre para la aplicación, se le indicará que el nombre está disponible, dejar la región por defecto, y dar clic en la opción crear app (Create app)
- En la siguiente página escoger un método de despliegue, entre las opciones se encuentra Heroku Git, GitHub y container Registry, se utilizará GitHub por lo que es necesario que suba el repositorio clonado en un repositorio de GitHub asociado a su cuenta de Heroku.
- Una vez escoja GitHub como método de despliegue, se presenta la opción para ingresar el nombre del repositorio del cual quiere hacer el despliegue.
- Antes de hacer dar clic en la opción de conectar al repositorio es necesario hacer ciertas modificaciones en el apartado de configuraciones de la aplicación en Heroku, específicamente en el apartado de configuración de variables.
- Dar clic en mostrar configuración de variables (Reveal Config Vars), se abrirá un menú en el que se debe ingresar las variables de entorno del backend.
- Finalmente regresar al apartado de Despliegue, conectar con el repositorio y hacer el despliegue, es posible verificar el estado del despliegue desde la consola de comandos de Heroku.

6. **Seguridad:**
   Para mejorar la seguridad de EventFlow puede seguir los siguientes consejos:

   - Permitir el acceso a la base de datos de MongoDB Atlas, solo a direcciones IP de las cuales se quiera permitir el acceso.
   - Modificar cors en el backend para que solo se permitan solicitudes de direcciones IP confiables
   - Modificar las variables de entorno para la generación de los tokens JWT, y no subir el archivo. env a un repositorio.
   - Modificar el código para que se bloqueen peticiones, cuando estás se generan masivamente desde una misma dirección IP, aunque si se hicieron las primeras modificaciones esto puedo agregar una capa extra de seguridad.
   - Modificar el código para que el usuario tenga que registrarse con una contraseña que contenga combinaciones de letras, números y símbolos, esto con el fin de que el usuario tenga una mejor seguridad.
   - Estar al tanto de posibles actualizaciones

7. **Solución de problemas:**
   Si tiene algún problema al instalar o utilizar EventFlow, es necesario consultar la documentación de EventFlow o ponerse en contacto con los desarrolladores.

Guía del Usuario Final

Registro de Usuarios	
Endpoint	POST /api/auth/new
Descripción	Este Endpoint permite a un nuevo usuario registrarse en la aplicación proporcionando un correo electrónico y una contraseña. Si el registro es exitoso, el usuario recibirá un token de autenticación.

Entrada	{
	"name": "Angie Michelle Diaz",
	"email": "angiediaz@gmail.com",
	"password": "123456"
	}

Salida	{ "ok": true, "uid": "6684b218be0f975867b189a6",

	"name": "Angie Michelle Diaz", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0YjIxOGJlMGY5NzU4NjdiMTg5YTYiLCJuYW1lIjoiQW5naWUgTWljaGVsbGUgRGlheiIsImlhdCI6MTcxOTk3MjM3NywiZXhwIjoxNzE5OTc5NTc3fQ.GyNhS2pKf5oJghY2dMXMkP6ka0gO2Z7Fr9yqe3lBC_0" }

Inicio de Sesión	
Endpoint	POST /api/auth
Descripción	Este Endpoint permite a un usuario registrado iniciar sesión proporcionando su correo electrónico y contraseña. Si las credenciales son correctas, el usuario recibirá un token de autenticación.

Entrada	{
	"email": "angiediaz@gmail.com",
	"password": "123456"
	}

Salida	{
	"ok": true,
	"uid": "6684b218be0f975867b189a6",
	"name": "Angie Michelle Diaz",
	"token":
	"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0YjIxOGJlMGY5NzU4NjdiMTg5YTYiLCJuYW1lIjoiQW5naWUgTWljaGVsbGUgRGlheiIsImlhdCI6MTcxOTk3MjQ2OCwiZXhwIjoxNzE5OTc5NjY4fQ.F_rFqgqFhWxnkT_awDO6PHwqSNCSuiLCDa6zkHIj4hU"
	}

Revalidación de JWT (JSON Web Tokens)	
Endpoint	GET /api/auth/renew
Descripción	Este Endpoint permite revalidar el token de autenticación de un usuario. Si el token es válido, se genera un nuevo token.

Entrada	{
	"email": "angiediaz@gmail.com",
	"password": "123456"
	}

	HEADERS: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0YjIxOGJlMGY5NzU4NjdiMTg5YTYiLCJuYW1lIjoiQW5naWUgTWljaGVsbGUgRGlheiIsImlhdCI6MTcxOTk3MjM3NywiZXhwIjoxNzE5OTc5NTc3fQ.GyNhS2pKf5oJghY2dMXMkP6ka0gO2Z7Fr9yqe3lBC_0

Salida	{
	"ok": true,
	"uid": "6684b218be0f975867b189a6",
	"name": "Angie Michelle Diaz",
	"token":
	"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0YjIxOGJlMGY5NzU4NjdiMTg5YTYiLCJuYW1lIjoiQW5naWUgTWljaGVsbGUgRGlheiIsImlhdCI6MTcxOTk3MzEwOCwiZXhwIjoxNzE5OTgwMzA4fQ.lkaYo5cMaJi-kkeCyPwcbgEH7YjhOJWBXnf5P9XWlOQ"
	}

Consulta de Eventos	
Endpoint	GET /api/events
Descripción	Este Endpoint permite obtener todos los eventos del calendario. Los eventos propios del usuario y los de los demás usuarios.
Entrada	HEADERS: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0YjIxOGJlMGY5NzU4NjdiMTg5YTYiLCJuYW1lIjoiQW5naWUgTWljaGVsbGUgRGlheiIsImlhdCI6MTcxOTk3MjM3NywiZXhwIjoxNzE5OTc5NTc3fQ.GyNhS2pKf5oJghY2dMXMkP6ka0gO2Z7Fr9yqe3lBC_0

Salida	{
	"ok": true,
	"eventos": [
	{
	"title": "Título del Evento",
	"notes": "Notas del Evento",
	"start": "2024-07-01T10:00:00Z",
	"end": "2024-07-01T11:00:00Z",
	"user": {
	"name": "Nombre del Usuario"
	},

	"id": "0987654321" }] }

Creación de Eventos	
Endpoint	POST /api/events
Descripción	Este endpoint permite al usuario crear un nuevo evento en el calendario.

Entrada	{
	"title": "Título del Evento",
	"notes": "Notas del Evento",
	"start": "2024-07-01T10:00:00Z",
	"end": "2024-07-01T11:00:00Z"
	}

Salida	{
	"ok": true,
	"evento": {
	"title": "Título del Evento",
	"notes": "Notas del Evento",
	"start": "2024-07-01T10:00:00Z",
	"end": "2024-07-01T11:00:00Z",
	"user": "1234567890",
	"id": "0987654321"
	}
	}

Edición de Eventos	
Endpoint	PUT /api/events/:id
Descripción	Este Endpoint permite a un usuario editar un evento existente en el calendario.

Entrada	{
	"title": "Nuevo Título del Evento",
	"notes": "Nuevas Notas del Evento",
	"start": "2024-07-01T12:00:00Z",
	"end": "2024-07-01T13:00:00Z"
	}

Salida	{ "ok": true,

	"evento": {
	"title": "Nuevo Título del Evento",
	"notes": "Nuevas Notas del Evento",
	"start": "2024-07-01T12:00:00Z",
	"end": "2024-07-01T13:00:00Z",
	"user": "1234567890",
	"id": "0987654321"
	}
	}

Eliminación de Eventos	
Endpoint	DELETE /api/events/:id
Descripción	Este Endpoint permite a un usuario eliminar un evento existente en el calendario, pero sólo puede eliminar sus eventos creados, no los de los demás usuarios.
Entrada	HEADERS: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0YjlxOGJlMGY5NzU4NjdiMTg5YTYiLCJuYW1lIjoiQW5naWUgTWljaGVsbGUgRGlheiIsImlhdCI6MTcxOTk3MjM3NywiZXhwIjoxNzE5OTc5NTc3fQ.GyNhS2pKf5oJghY2dMXMkP6ka0gO2Z7Fr9yqe3lBC_0

Salida	{
	"ok": true
	}

Generación de Reportes	
Endpoint	GET /api/reports/generate/all/all
Descripción	Este Endpoint permite generar un reporte en formato PDF de todos los eventos del calendario.
Salida	Un archivo PDF con los detalles de todos los eventos

Restablecimiento de Contraseña	
Endpoint	POST /api/recovery

Descripción	Este Endpoint verificar que el usuario que intenta restablecer su contraseña exista en la base de datos.

Entrada	{
	"email": "faboordo@gmail.com"
	}

Salida	{
	"ok": true,
	"msg": "Correo de recuperación enviado",
	"name": "fabio Alejandro",
	"tokenRecovery":
	"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0NmNkZDM0ZjU3YjM4MmY2M2JlNDEiLCJuYW1lIjoiZmFiaW8gQWxlamFuZHJvIiwiaWF0IjoxNzE5OTg1ODMwLCJleHAiOjE3MTk5ODk0MzB9.zQkNOGJIEDD1QzjN-GfyDTA6VMZV33JX_b2otPuBJGs"
	}

Eliminación de Contraseña 1	
Endpoint	GET /api/recovery/reset-password
Entrada	Query Parameters Token = eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0NmNkZDM0ZjU3YjM4MmY2M2JlNDEiLCJuYW1lIjoiZmFiaW8gQWxlamFuZHJvIiwiaWF0IjoxNzE5OTU0Njk2LCJleHAiOjE3MTk5NTgyOTZ9.lRz-nFXNPKDTeEVG07sgSWYK0XWw0SxAakdlajj9LWQ
Descripción	Este Endpoint comprueba la existencia del token en la base de datos.

Salida	{
	"ok": true,
	"msg": "Token válido",
	"recoveryToken":
	"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0NmNkZDM0ZjU3YjM4MmY2M2JlNDEiLCJuYW1lIjoiZmFiaW8gQWxlamFuZHJvIiwiaWF0IjoxNzE5OTg1ODMwLCJleHAiOjE3MTk5ODk0MzB9.zQkNOGJIEDD1QzjN-GfyDTA6VMZV33JX_b2otPuBJGs",
	"user": "66846cdd34f57b382f63be41"
	}

Eliminación de Contraseña 2	
Endpoint	POST /api/recovery/reset-password

Entrada	Query Parameters
	Token =
	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOiI2Njg0NmNkZDM0ZjU3YjM4MmY2M2JlNDEiLCJuYW1lIjoiZmFiaW8gQWxlamFuZHJvIiwiaWF0IjoxNzE5OTU0Njk2LCJleHAiOjE3MTk5NTgyOTZ9.lRz-nFXNPKDTeEVG07sgSWYK0XWw0SxAakdlajj9LWQ
	
	Body =
	{
	“newPassword”: “123456”
	}

Descripción	Este Endpoint verifica que el token exista y obtiene la nueva contraseña del usuario.

Salida	{
	"ok": true,
	"msg": "Contraseña actualizada correctamente"
	}

DOCUMENTACIÓN TÉCNICA

Estructura del Código

Consideraciones previas:

- La tecnología backend elegida es Express.js para el servidor
- La tecnología frontend elegida es React.js para la interfaz del usuario
- Se eligió MongoDB Atlas para alojar la base de datos.

Estructura de carpetas

```
EventFlow
|__Backend
|       |__index.js
|       |__routes
|       |__controllers
|       |__models
|       |__helpers
|       |__middlewares
|       |__db
|       |__.env
|
|__Frontend
|       |__src
|       |       |__api
|       |       |__auth
|       |       |__calendar
|       |       |__helpers
|       |       |__hooks
|       |       |__store
|       |       |--db
|       |       |__CalendarApp.jsx
|       |__.env
```

Backend está carpeta contiene el código del servidor de la aplicación.

- **index.js:** Es el punto de entrada principal del servidor. Aquí se inicializa el servidor, se configuran las rutas y la conexión a la base de datos
- **routes:** Archivos que definen las rutas de la API del servidor.
- **controllers:** Almacena los controladores de la API. Los controladores manejan las peticiones, interactúan con los modelos y devuelven las respuestas.
- **models:** Contiene las definiciones de los modelos de datos de la aplicación. Estos modelos representan la estructura de los datos que se manejan en la base de datos.
- **helpers:** Funciones reutilizables que ayudan en la lógica del backend.
- middlewares: Alberga funciones middleware que se ejecutan antes de llegar a los controladores.
- **db:** Contiene el archivo de conexión a la base de datos.
- **.env:** Este archivo contiene las variables de entorno puerto, palabra de generación de tokens JWT y credenciales de la base de datos.

Frontend está carpeta contiene el código de la interfaz de usuario de la aplicación.

- **src:** Contiene el código fuente de la aplicación React (o similar).
- **api:** Almacena el código relacionado con la interacción con la API del backend.
- **auth:** Esta carpeta contiene los componentes y lógica para el login, registro y gestión de sesiones.
- **calendar:** Contiene los componentes y lógica específica para la funcionalidad del calendario.
- **helpers:** Funciones reutilizables
- **hooks:** Esta carpeta contiene los hooks personalizados para manejar estados o efectos secundarios.
- **CalendarApp.jsx:** Componente principal de la aplicación del calendario. Este componente renderiza el resto de los componentes y gestiona el flujo general de la aplicación.

Comentarios en el Código

Debe contener la siguiente clase de comentarios:

- Comentarios explicativos: Describir el propósito general de la sección de código.
- Comentarios de documentación: Documentar los parámetros, valores de retorno y excepciones de una función o método.
- Comentarios de ejemplo: Proporcionar ejemplos de cómo usar una función o método.
- Comentarios de depuración: Depurar el código y se deben eliminar una vez se soluciona el problema.