Facultad
de
Ciencias

APLICACIÓN WEB SOBRE JAVA PARA LA GESTIÓN DEL TIEMPO EN REUNIONES EN EL ENTORNO EMPRESARIAL

(Java-based web application for the management of time in meetings in the business environment)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Jesús Revuelta San Emeterio

Director: Julio Luis Medina Pasaje
Codirector: Guillermo Menguez Álvarez

Febrero – 2024

Agradecimientos

Quiero comenzar expresando mi más profundo agradecimiento a aquellos que han sido mi roca durante todo este trayecto. A mi familia, amigos, novia y compañeros, quienes han sido mi apoyo inquebrantable en cada paso de este viaje.

En segundo lugar, quiero extender mi gratitud a mis tutores, D. Julio Luis Medina Pasaje y D. Guillermo Menguez Álvarez, cuya orientación experta y dedicación han sido fundamentales en el desarrollo de este proyecto.

A la Universidad de Cantabria y a todos los profesores que han compartido sus conocimientos conmigo. Sin su dedicación y enseñanzas, este logro no habría sido posible. Estoy sinceramente agradecido por su invaluable contribución a mi formación académica y profesional.

Y finalmente, deseo dedicar un especial agradecimiento a mi abuela Elena por su amor incondicional, que ha sido una fuente constante de inspiración para mí.

Índice

Índice de tablas

Índice de figuras

Resumen

Este proyecto propone una solución web para la gestión eficiente del tiempo en el entorno laboral, especialmente en la era del teletrabajo. Implementa un sistema analítico que evalúa la distribución del tiempo de los empleados, identificando el balance entre las horas dedicadas a reuniones y las disponibles para otras tareas.

Se estructura en tres niveles fundamentales: Una capa de análisis que se basa en una estructura que permite el procesamiento eficiente de los datos del calendario, evitando redundancias y optimizando el acceso a la información. Esto se logra mediante algoritmos que interpretan y categorizan las actividades registradas en el calendario.
La capa de lógica de negocio se implementa mediante un servicio web que maneja de forma segura y eficiente los datos, garantizando la privacidad al realizar el análisis del tiempo de trabajo. Este nivel se centra en interpretar los datos del calendario para ofrecer una visión precisa de la distribución del tiempo laboral.
La interfaz de usuario es una aplicación web intuitiva, diseñada para que los empleados puedan interactuar fácilmente con el sistema, independientemente de su nivel de experiencia en tecnologías similares. Esta interfaz permite a los usuarios configurar parámetros de análisis y visualizar los resultados de forma sencilla y comprensible.

Se utilizan tecnologías como Java para el desarrollo del back-end, y React para el front-end, e implementa conectores para acceder a los calendarios de Google Calendar. Además, los usuarios pueden personalizar el análisis incluyendo días no laborables, como festivos o vacaciones, para obtener una visión más precisa de su tiempo de trabajo.

En resumen, este sistema web propone una solución integral para la gestión eficiente del tiempo en el ambiente laboral, aprovechando tecnologías avanzadas de análisis de datos y una interfaz de usuario amigable, con el objetivo de maximizar la productividad y minimizar el tiempo perdido en reuniones innecesarias o mal gestionadas.

Palabras clave: Back-end, Java, Spring Boot, React, Google Cloud

Abstract

This project proposes a web solution for efficient time management in the workplace, especially in the era of telecommuting. It implements an analytical system that evaluates employees' time distribution, identifying the balance between hours spent in meetings and those available for other tasks.

It is structured on three fundamental levels: An analysis layer that relies on a structure allowing efficient processing of calendar data, avoiding redundancies, and optimizing access to information. This is achieved through algorithms that interpret and categorize the activities registered in the calendar.
The business logic layer is implemented via a web service that manages data securely and efficiently, ensuring privacy when analyzing working time. This level focuses on interpreting calendar data to offer an accurate view of labor time distribution.
The user interface is an intuitive web application, designed so that employees can easily interact with the system, regardless of their experience level with similar technologies. This interface allows users to set analysis parameters and visualize the results in a simple and understandable way.

Technologies such as Java for backend development, and React for the frontend, are utilized, and it implements connectors to access Google Calendar calendars. Additionally, users can customize the analysis by including non-working days, such as holidays or vacations, to get a more accurate view of their working time.

In summary, this web system proposes an integrated solution for efficient time management in the work environment, leveraging advanced data analysis technologies and a user-friendly interface, with the goal of maximizing productivity and minimizing time lost in unnecessary or poorly managed meetings.

Keywords: Back-end, Java, Spring Boot, React, Google Cloud

1. Introducción

1.1 Contexto

En la era actual, marcada por cambios muy significativos en el ámbito laboral especialmente con el auge del teletrabajo tras la pandemia, las soluciones informáticas han cobrado un gran protagonismo. Las aplicaciones no solo tienen cabida en el ámbito personal, sino que también vienen a mejorar la eficiencia en los entornos laborales.

La necesidad de coordinar equipos situados en distintos puntos geográficos, junto con el objetivo de mantener la productividad en un ámbito en el que no existe control, hace que nazca la necesidad de aplicaciones más sofisticadas. Con el incremento de reuniones virtuales y la gestión del tiempo de trabajo, surge la necesidad de desarrollar herramientas que permitan una organización eficiente del tiempo.

El nivel de dificultad de uso se ha convertido en un factor a tener en cuenta para que cualquier tecnología sea adoptada en la sociedad. Vivimos en un mundo, en el que la curva de aprendizaje para conocer este tipo de herramientas tiene que ser lo más corta posible, por lo que diseñar aplicaciones intuitivas y sencillas de cara al usuario final, es esencial. Esto, junto con la seguridad de los datos y la facilidad de uso para alcanzar una mayor adopción por parte de los usuarios, juegan un papel vital en el éxito de estas herramientas.

1.2 Objetivo

El objetivo de este proyecto es el desarrollo de una aplicación web full stack diseñada para la gestión eficiente del tiempo laboral, teniendo como propósito facilitar a los usuarios la gestión del análisis de su tiempo, empleado en reuniones y otras tareas. Para lograr esto, el sistema será diseñado teniendo en cuenta la funcionalidad como la facilidad de uso, asegurando que la aplicación es accesible y de utilidad, con el fin de que se adapte a distintos entornos laborales, incluyendo aquellos con alto volumen de trabajo remoto.

Para el desarrollo de este trabajo, y siguiendo la nomenclatura que sugiere Google Cloud para acceso a los datos des calendario, se han identificado dos tipos de usuario, que se describen a continuación.

1.2.1 Usuario externo

Los usuarios denominados externos son los usuarios normales de la aplicación, tales como empleados y gerentes, que podrán así acceder a sus calendarios, buscar sus reuniones por palabra clave, y analizar el tiempo dedicado a diversas tareas.

1.2.1 Usuario interno: Administrador

Los usuarios externos o administradores podrán gestionar cuentas de usuario, configurar parámetros del sistema y acceder a informes agregados para una visión global del uso del tiempo.

1.3 Organización de la memoria

Las secciones que componen el resto de esta memoria se organizan como se enumera a continuación.

1. Tecnologías, Herramientas y Lenguajes Utilizados: Descripción detallada de las tecnologías y herramientas utilizadas en el desarrollo del proyecto.

2. Metodología y Planificación: Descripción de la metodología de trabajo adoptada y la planificación del proyecto.

3. Especificación de Requisitos y Casos de Uso: Desarrollo de los requisitos funcionales y no funcionales, así como los casos de uso derivados de estos.

4. Diseño del Sistema: Detalle del diseño del sistema, incluyendo arquitectura y diseño de interfaz.

5. Implementación: Explicación de la implementación técnica del proyecto, abarcando tanto el desarrollo del back-end como del front-end.

6. Pruebas y Resultados: Presentación del proceso de pruebas realizado y los resultados obtenidos para validar la funcionalidad de la aplicación.

7. Conclusiones y Trabajos Futuros: Reflexiones finales sobre el proyecto, su impacto y posibles mejoras o expansiones futuras.

2. Tecnologías, herramientas y lenguajes utilizados

2.1 Tecnologías y herramientas

En este punto se proporciona una breve descripción de las tecnologías y herramientas que se han utilizado para el desarrollo de la aplicación.

- **Eclipse IDE.** Eclipse IDE es un entorno de desarrollo integrado de código abierto que facilita el desarrollo de aplicaciones basadas en Java [1]. En el proyecto se usa como entorno de desarrollo para implementar el back-end.

- **Spring Framework.** Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java [2]. En este caso lo utilizamos para el desarrollo de la API REST.

- **Spring Boot.** Spring Boot permite compilar nuestras aplicaciones Web como un archivo .jar que podemos ejecutar como una aplicación Java normal consiguiéndolo gracias a que integra el servidor de aplicaciones en el propio .jar [3]. En el proyecto se utiliza para el despliegue de la aplicación.

- **Maven.** Maven se utiliza en la gestión y construcción de software. Facilita tareas de procesamiento del código y configuración claramente definidas, como la compilación del código y su empaquetado. Además, gestiona dependencias entre librerías utilizadas e incluidas dentro de la estructura del JAR [4].

- **Visual Studio Code.** Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización

inteligente de código, fragmentos y refactorización de código [5]. En el proyecto se utiliza como entorno de desarrollo para el front-end.

- **React.** React ayuda a crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado de la aplicación. Se encarga de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien [6].

- **Node.js.** Es Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones web escalables [7].

- **Bootstrap.** Es una biblioteca multiplataforma y un conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web [8].

- **Postman.** Postman es una aplicación que nos permite realizar pruebas de una API. Es un cliente HTTP que nos da la posibilidad de testear 'HTTP requests' a través de una interfaz gráfica de usuario, por medio de la cual obtendremos diferentes tipos de respuestas para su validación [9].

- **GitHub.** GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git [10].

2.2 Lenguajes

En este apartado se mencionan los lenguajes que se han utilizado para el desarrollo de la aplicación.

- **Java.** Java es un sistema de programación muy usado para diversos fines. Un lenguaje de programación sencillo, multiplataforma y orientado a objetos.
- **SCSS.** SCSS es una sintaxis de preprocesador de CSS que permite usar características que no existen en CSS puro, como variables, anidación, mixins y herencia. [11]
- **JavaScript.** JavaScript es un lenguaje de scripts dinámico que admite la construcción de objetos basada en prototipos. Intencionalmente, la sintaxis básica es similar a Java y C++ para reducir la cantidad de conceptos nuevos necesarios para aprender el lenguaje [12].

    - **React**: React es una biblioteca de JavaScript desarrollada por Facebook para construir interfaces de usuario. Se usa principalmente para el desarrollo del front-end de aplicaciones web. React permite crear componentes de UI reutilizables y manejar eficientemente el estado y la renderización en aplicaciones web dinámicas.

2.3 Definiciones

- **Back-end.** Parte de la aplicación que procesa el tratamiento de datos y realiza la conexión entre la base datos y el front-end.

- **Front-end.** Parte de la aplicación correspondiente con la interfaz que facilitará el uso de esta al usuario desde un navegador web.

- **API REST.** Definición fundamental de los servicios ofrecidos por el back-end al front-end. Constituye el sustrato medular de la aplicación gracias a un amplio conjunto de funciones accesibles a través del protocolo HTTP.

- **HTTP (Hypertext Transfer Protocol)**. Protocolo que permite hacer las llamadas para la transmisión de datos entre front-end y back-end.

2.4 Elección de tecnologías y lenguajes

Se han elegido las herramientas mencionadas previamente por las siguientes razones:

- Los entornos de desarrollo han sido seleccionados por la familiaridad que tenía el desarrollador con ellos previo al comienzo del proyecto. Son lenguajes que permiten al usuario un desarrollo rápido gracias a la gran cantidad de ayudas y herramientas internas que proporcionan.

- Los frameworks utilizados se han elegido por encima de otros por la gran compatibilidad que tenían con el tipo de desarrollo que se pedía implementar además de que en la gran mayoría de los casos son herramientas que el desarrollador había visto o utilizado previamente. El framework de Spring ha hecho el desarrollo de la API REST más liviano gracias a las facilidades que nos brindan con sus librerías prediseñadas.

- Maven ha sido seleccionado debido a que tiene una gran integración con el entorno de desarrollo con el que se ha realizado el back-end facilitando así la tarea del desarrollador. Al igual que en los anteriores casos, se estaba previamente familiarizado con la compilación y ejecución del código, así como con la gestión que hace de las dependencias.

- React y Node.JS son tecnologías con las que realmente no se estaba familiarizado, pero tras una pequeña investigación se decidió que gracias a las oportunidades que brindan con su facilidad de uso eran las dos mejores herramientas con las que desarrollar el código de la parte front-end frente a otras como podría ser Angular. De esta misma manera se utilizó Bootstrap para basarse en la realización de componentes.

- Para realizar pruebas sobre la API REST se optó por Postman debido a la facilidad de uso de su interfaz gráfica para hacer llamadas HTTP además de que se estaba familiarizado con la herramienta.

- Para el control de versiones se decidió utilizar GitHub por la familiaridad que se tiene con la herramienta además de la gran integración que tiene con los entornos de desarrollo, permitiendo consolidar todos los cambios directamente a través de la IDE o desde una terminal.

- Por último, los lenguajes utilizados se seleccionaron realizando una valoración entre lo que mejor se adecuaba a la aplicación en cuestión y el conocimiento del

desarrollador sobre ellos. Java y Javascript son lenguajes muy utilizados a nivel de back-end y front-end y, a su vez, encajaban a la perfección para la realización de la aplicación. Por otra parte, aunque el lenguaje de JavaScript no era tan conocido por el desarrollador son punteros a nivel mundial a la hora de la programación de front-end lo cual hizo que se tomase la decisión de usarlo.

3. Metodología y planificación

En esta sección del trabajo, presentamos la metodología utilizada y la planificación seguida, para el desarrollo de nuestra aplicación.

3.1 Metodología

Para el desarrollo de este proyecto se ha empleado una metodología incremental, comúnmente utilizada en los procesos de ingeniería de software. Este tipo de metodología se caracteriza por tener una serie de fases o etapas, en las cuales, al finalizar, se presenta una versión al cliente. Una vez completada la etapa, el proyecto avanza hacia la siguiente fase o versión, en cada una de estas fases sucesivas (tantas como consideremos necesarias), se añaden mejoras o avances a nuestro código, de tal manera que la evolución del proyecto queda controlada, permitiendo la oportunidad de incorporar ideas o *feedback* por parte del cliente, asegurando lograr las expectativas de este.

Como bien hemos comentado previamente, esta metodología está dividida en distintas fases:

- Análisis del problema y de los requisitos a incorporar al incremento.
- Diseño de la solución prevista.
- Desarrollo de código e implementación de la solución.
- Testeo/pruebas del código.

De esta manera, ofrecemos al cliente un seguimiento constante de los avances del proyecto, haciendo más accesible para ambas partes, la comunicación de nuevas necesidades o aclaraciones no detectadas en las fases anteriores.

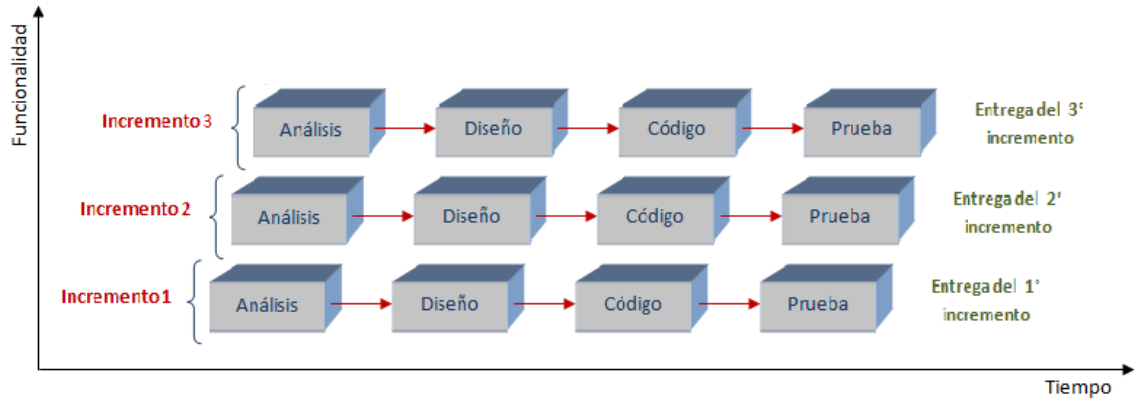


Figura 1. Esquema sintético de una metodología incremental

3.2 Planificación

Para el buen desarrollo del proyecto, se le planifica en diferentes etapas, las cuales quedarán expuestas en detalle en apartados posteriores, siendo estas la toma y especificación de requisitos funcionales y no funcionales, el diseño del sistema, el estudio y formación sobre las herramientas necesarias para el desarrollo de la aplicación, así como el desarrollo de esta y las pruebas necesarias.

3.2.1 Toma de requisitos

En esta primera etapa de nuestra planificación, el foco principal es conocer las necesidades del cliente y conocer el estado del arte en otras aplicaciones de naturaleza similar. El objetivo de esta etapa es tener una recopilación completa de los requisitos funcionales y no funcionales que nuestro sistema necesita para lograr el funcionamiento deseado. Esta parte es fundamental, ya que conocer las necesidades de nuestro usuario final es muy importante, a su vez la investigación de sistemas similares nos proporciona gran cantidad de ideas para perfilar la funcionalidad del sistema.

3.2.2 Investigación y formación

Para poder llevar a cabo el proyecto, se ha empleado un tiempo en investigar diferentes tecnologías y lenguajes de programación, analizando cuáles de estos se adecúan mejor a los requisitos que se detectan en la fase anterior. Una vez seleccionados, es imprescindible conocerlos bien y obtener una formación en los mismos desde diferentes puntos de vista:

- Profundizar en el lenguaje a utilizar, en este caso Java.
- Mejorar la capacidad de desarrollo y conocer mejor React.
- Investigar el uso de las capacidades de Google Cloud para poder acceder a los datos del calendario de Google.

3.2.3 Desarrollo

Como se mencionó anteriormente el desarrollo sigue una metodología incremental, en este caso divida a su vez en 3 fases. Para el desarrollo se crearán dos repositorios en GitHub, uno dedicado a la implementación del back-end y otro para el front-end donde se ha seguido la siguiente misma idea. En la rama *master* se mantendrá el desarrollo funcional de la aplicación. En la rama *develop* se mantendrá una copia de master que será utilizada para realizar las pruebas de integración una vez se ha implementado una nueva funcionalidad con el fin de no llevar a la rama master un código que no esté completamente probado y sea integrable con el resto de la aplicación. Por último, cada vez que se quiere trabajar en una funcionalidad nueva se crea una rama para su desarrollo, una vez el desarrollo está completo se hace el *merge* de dicha rama con *develop* para comprobar su correspondiente integración con el resto de la aplicación. Por último, cuando se corrobora al completo que la nueva funcionalidad es correcta se realiza el paso a *master*.

A continuación, se muestra cómo se han estructurado y qué comprende cada fase:

- **Fase 1**: Esta etapa se centra en la implementación de la API REST y la integración con Firebase. El primer paso es configurar Firebase en el proyecto. Esto implica

crear un nuevo proyecto en la consola de Firebase, lo cual se realiza accediendo a la Consola de Firebase y siguiendo los pasos para crear un nuevo proyecto. Durante este proceso, se otorga un conjunto de credenciales únicas, que incluyen una clave API y otros detalles de configuración necesarios para conectar el proyecto con Firebase. Una vez obtenidas las credenciales, se procede a configurar el back-end desarrollado en Java. Utilizando Spring, se establece la conexión entre el back-end y la API de Google Calendar, aprovechando las funcionalidades que ofrece Firebase, como la autenticación y el almacenamiento de datos. Es importante asegurar que las credenciales de Firebase se manejan de forma segura, generalmente almacenándolas en un archivo de configuración separado o utilizando variables de entorno para proteger la información sensible. En este punto, dado que aún no se ha desarrollado un front-end, se utiliza Postman, una herramienta de prueba de APIs, para verificar la correcta funcionalidad de la API REST desarrollada. Con Postman, podemos simular solicitudes HTTP a un back-end y verificar las respuestas, asegurando así que la integración con Firebase y la API de Google Calendar funciona como se espera.

- **Fase 2**: Esta fase se enfoca en expandir las capacidades del back-end y en el desarrollo del front-end utilizando React. En lugar de implementar JWT de Spring Security, se integrará la seguridad mediante la conexión con la API de Google Calendar. Este enfoque garantizará la seguridad de las llamadas realizadas desde el front-end, aprovechando los protocolos de autenticación y autorización proporcionados por Google. A continuación, se desarrollará la primera versión de la interfaz de usuario, asegurando su correcta integración con el back-end. Finalmente, en esta etapa se completará el desarrollo de los componentes restantes del front-end y se agregarán los servicios necesarios en el back-end para las funcionalidades adicionales solicitadas.

- **Fase 3:** Es fase de asegurar la calidad se lleva a cabo mediante tres tipos principales de pruebas. Inicialmente, se realizan pruebas unitarias para evaluar la funcionalidad de las piezas individuales del código, asegurando que cada componente opere correctamente por sí mismo. Posteriormente, se ejecutan pruebas de integración para verificar la coherencia de las interfases en las interacciones y el correcto funcionamiento conjunto de los diferentes módulos del sistema, identificando problemas en la comunicación entre componentes. Finalmente, se llevan a cabo pruebas de aceptación, las cuales se enfocan en confirmar que el sistema completo cumple con los requisitos específicos y satisface las expectativas del usuario final, garantizando así que el producto está listo para su implementación y uso en entornos reales.

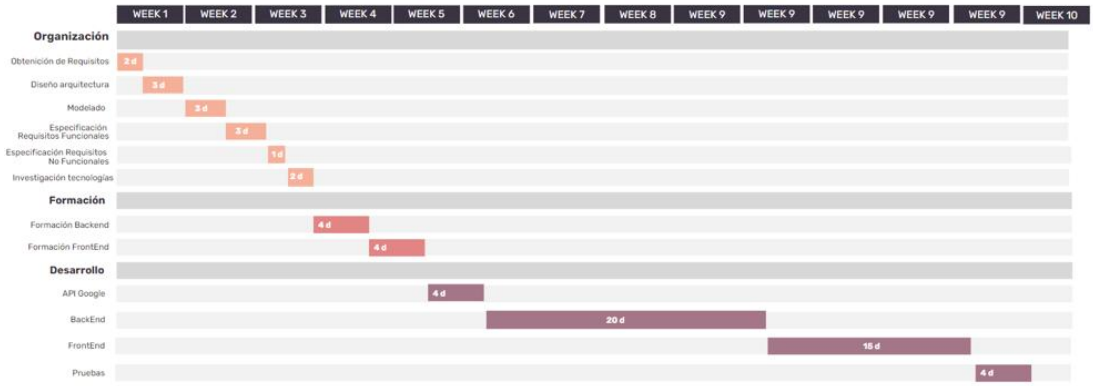


Figura 2. Diagrama de Gantt

4. Especificación de requisitos y casos de uso

Como bien comentábamos en las secciones anteriores, especificar los requisitos a tener en cuenta, es uno de los pasos más importantes para realizar un desarrollo de esta naturaleza.

En este proyecto, el desarrollador realizará la recopilación de los requisitos necesarios para esta aplicación. Una de las primeras fases para esto, es la investigación de aplicaciones para conocer algunos de los requisitos en sistemas de estas características. Del mismo modo, conocer las necesidades y, sobre todo, las funcionalidades requeridas por nuestro cliente son fundamentales para el correcto desarrollo de nuestra aplicación, por lo que detallar los requisitos necesarios para el cliente, sería de gran importancia.

Del mismo modo, conocer las funcionalidades que tiene nuestra aplicación en función del rol del usuario que lo utilice, es muy importante, para ello posteriormente presentaremos diferentes diagramas que muestren las posibilidades de cada usuario dentro de nuestro sistema.

4.1 Actores del sistema

Como bien indica su nombre, los actores del sistema hacen referencia a los diferentes tipos de usuario (según sus características) que utilizarán el sistema. Aunque ya se presentaron en la introducción, se recogen brevemente los diferentes "papeles" y su descripción a continuación:

- **Usuario interno** (Administrador): Tienen la capacidad de gestionar las diferentes cuentas de usuario, configurar parámetros del sistema, acceder a informes agregados para una visión global del uso del tiempo dentro de la organización en la que se encuentren.

- **Usuario externo** (Empleados y gerentes): Tienen permiso para acceder a sus calendarios, buscar sus reuniones y analizar el tiempo dedicado a diferentes tareas.

4.2 Requisitos funcionales

Los requisitos funcionales son los que definen las funcionalidades que va a integrar el sistema.

ID	Descripción
RF01	El usuario deberá poder visualizar el porcentaje o cantidad de su tiempo ocupado en eventos o reuniones.
RF02	El usuario podrá ver el porcentaje o cantidad de tiempo libre que tiene disponible.
RF03	El usuario tendrá la capacidad de ver el número total de eventos o reuniones programadas en su calendario para el mes en curso.

RF04	El usuario deberá poder visualizar una lista o resumen de sus próximos eventos o reuniones.
RF05	El usuario podrá acceder a información sobre la duración promedio de sus eventos o reuniones.
RF06	El usuario deberá poder obtener un análisis detallado de cómo utiliza su tiempo, incluyendo la evaluación de intervalos sin reuniones y tiempo ocupado en reuniones (tanto confirmadas, provisionalmente aceptadas, como superpuestas).
RF07	El usuario podrá seleccionar fechas específicas para el análisis del calendario, eligiendo una fecha de inicio y una de finalización para el período a examinar.
RF08	El usuario tendrá la opción de establecer su horario laboral, indicando la hora de inicio y de finalización de su jornada de trabajo, lo que ayudará a personalizar el análisis del uso del tiempo.
RF09	El usuario podrá marcar ciertos días de la semana como no laborables. Esto permitirá excluir estos días del análisis del calendario para enfocarse en el tiempo de trabajo efectivo.
RF10	Una vez configuradas las preferencias, el usuario podrá iniciar el análisis del calendario al seleccionar la opción "SUBMIT". Este análisis proporcionará una visión detallada de cómo se está utilizando el tiempo, identificando patrones y ofreciendo oportunidades para una mejor gestión del tiempo.
RF11	El usuario deberá poder buscar eventos en su calendario utilizando un sistema de búsqueda avanzada. Esta función le permitirá encontrar eventos específicos que se adapten a sus intereses y necesidades.
RF12	El usuario podrá realizar búsquedas de eventos introduciendo palabras clave. Esto implica la capacidad de filtrar eventos en el calendario basándose en términos relacionados con el tipo de evento, tema, participantes, entre otros criterios relevantes.
RF13	El usuario tendrá la opción de especificar un rango de fechas para la búsqueda de eventos, seleccionando tanto una fecha de inicio como una de finalización. Esto permitirá enfocar la búsqueda en un periodo concreto.
RF14	Tras introducir los criterios de búsqueda, como la palabra clave y el rango de fechas, el usuario podrá ejecutar la búsqueda seleccionando la opción "SUBMIT". Esta acción iniciará el proceso de filtrado y presentación de eventos que coincidan con los criterios establecidos.
RF15	El usuario tendrá la capacidad de utilizar una barra de navegación ubicada en la parte superior de la aplicación para desplazarse eficientemente entre distintas secciones. Al hacer clic en los enlaces o botones de esta barra, el usuario será redirigido instantáneamente a la sección correspondiente de la aplicación.
RF16	El administrador deberá poder dar de alta un usuario.
RF17	El administrador deberá poder dar de baja un usuario.

Tabla 1. Requisitos funcionales del sistema

4.3 Requisitos no funcionales

Los requisitos no funcionales son aquellos que detallan las necesidades de la aplicación que no tienen que ver específicamente con las funcionalidades que se van a desarrollar.

ID	Descripción	Categoría	Importancia
RNF01	La aplicación deberá utilizar el idioma castellano.	Localización	Alta
RNF02	La aplicación deberá asegurar que el usuario está autentificado para poder utilizarla.	Seguridad	Alta
RNF03	La aplicación deberá contar con una interfaz de uso sencillo.	Usabilidad	Alta
RNF04	La aplicación debe contar con un mecanismo de gestión de datos fácilmente escalable.	Eficiencia	Alta
RNF05	La aplicación debe ser capaz de dar servicio a un gran número de conexiones simultáneamente sin perder rendimiento.	Eficiencia	Media

Tabla 2. Requisitos no funcionales del sistema

4.4 Consideraciones sobre seguridad

Aunque es imposible garantizar una seguridad absoluta en cualquier sistema, somos conscientes de la importancia de proteger la información manejada en nuestra aplicación NoMoreMeetings, especialmente cuando involucra datos de eventos y preferencias de los usuarios. Para minimizar los riesgos y asegurar la integridad y seguridad de los datos, hemos implementado varias medidas de seguridad:

• **Acceso Seguro a Información Personal:** Los usuarios deben autenticarse para acceder a sus datos personales y estadísticas. Esto garantiza que cada usuario solo tiene acceso a su información propia, protegiendo la privacidad y seguridad de los datos.

• **Mantenimiento y Supervisión del Sistema:** El administrador de la aplicación, que en este caso es el creador o el equipo de desarrollo, tiene acceso a métricas de uso general y a la supervisión del sistema. Esto incluye el monitoreo del rendimiento de la API, análisis de tráfico y detección de actividades inusuales, pero sin acceso a datos personales de los usuarios.

• **Respaldo de Datos:** Realizamos copias de seguridad regulares para prevenir la pérdida de datos y mantener la integridad de la información almacenada en la aplicación.

• **Comunicaciones Seguras:** Todas las transacciones de datos se realizan a través de canales seguros, empleando HTTPS para encriptar la comunicación y protegerla contra interceptaciones.

Estas medidas de seguridad están diseñadas para proteger tanto la privacidad como la integridad de los datos de los usuarios, y para proporcionar una plataforma confiable y segura para la gestión eficiente de reuniones y tiempo.

4.5 Diagramas de casos de uso

En este apartado se mostrarán los diagramas correspondientes a los diferentes roles de usuario.

4.5.1 Diagrama para el rol Usuario interno: Administrador

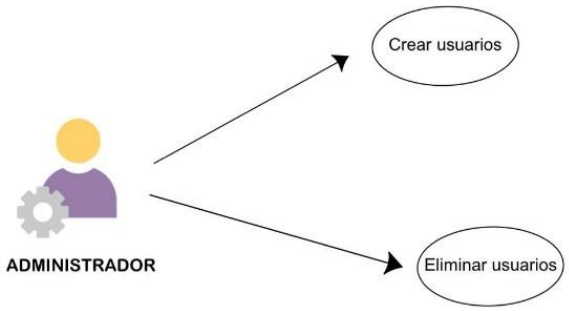


Figura 3. Diagrama Usuario Administrador

4.5.2 Diagrama para el rol Usuario externo

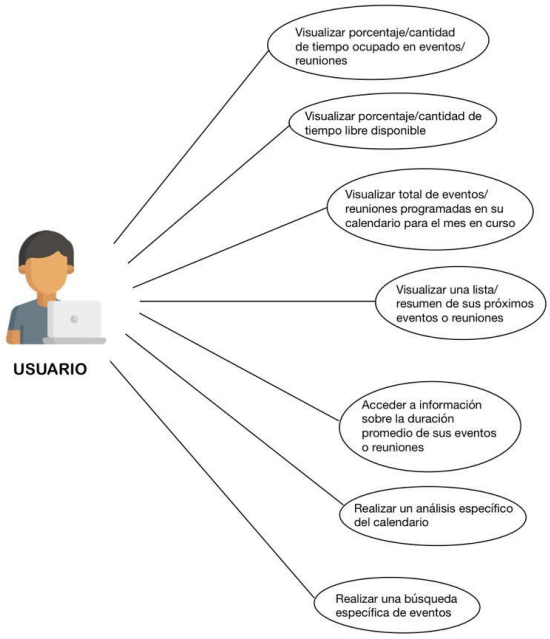


Figura 4. Diagrama Usuario Externo

4.6 Plantillas de casos de uso

ID	**P01**
Caso de uso	Inicio de Sesión
Descripción	El usuario inicia sesión en la aplicación
Actores	Todos
Precondiciones	El usuario deber estar registrado en el sistema
Flujo principal	1. El usuario se registra con Google. 2. El usuario introduce las credenciales en los campos que se muestran en el formulario. 3. EL sistema reconoce al usuario y carga la interfaz.
Flujo alternativo	El sistema no reconoce al usuario y se muestra un mensaje de error.

Tabla 3. Plantilla P01

ID	**P02**
Caso de uso	Visualizar Estadísticas de Tiempo Ocupado
Descripción	El usuario puede ver estadísticas detalladas de su tiempo ocupado de forma general.
Actores	Usuario general
Precondiciones	El usuario deber estar registrado en el sistema
Flujo principal	1. El usuario va a la sección de estadísticas. 2. Mantiene el cursor por encima de la sección de tiempo ocupado.

	3. El sistema muestra un texto con el porcentaje de ocupación.
Flujo alternativo	El sistema no encuentra eventos y muestra un 0.0% de ocupación

Tabla 4. Plantilla P02

ID	**P03**
Caso de uso	Visualizar Estadísticas de Tiempo Libre
Descripción	El usuario puede ver estadísticas detalladas de su tiempo libre de forma general.
Actores	Usuario general
Precondiciones	El usuario deber estar registrado en el sistema
Flujo principal	1. El usuario va a la sección de estadísticas. 2. Mantiene el cursor por encima de la sección de tiempo libre. 3. El sistema muestra un texto con el porcentaje de ocupación.
Flujo alternativo	El sistema no encuentra eventos y muestra un 0.0% de tiempo libre.

Tabla 5. Plantilla P03

ID	**P04**
Caso de uso	Visualizar número total de eventos este mes.
Descripción	El usuario puede ver el número total de eventos este mes.
Actores	Usuario general
Precondiciones	El usuario deber estar registrado en el sistema
Flujo principal	1. El usuario va a la sección de estadísticas. 2. Mantiene el cursor por encima de la sección de número total en el mes actual de eventos. 3. El sistema muestra un texto con el número de eventos de este mes.
Flujo alternativo	El sistema no encuentra eventos y muestra un numero 0 de eventos.

Tabla 6. Plantilla P04

ID	**P05**
Caso de uso	Visualizar próximos eventos.
Descripción	El usuario puede ver cual son los próximos eventos.
Actores	Usuario general
Precondiciones	El usuario deber estar registrado en el sistema

Flujo principal	1. El usuario va a la sección de estadísticas. 2. Mantiene el cursor por encima de la sección de próximos eventos. 3. El sistema muestra un texto con los próximos eventos (máximo de 10) detallados con el título, día y la hora.
Flujo alternativo	El sistema no encuentra eventos y muestra un mensaje de que no hay eventos.

Tabla 7. Plantilla P05

ID	**P06**
Caso de uso	Visualizar Estadísticas de duración promedio de eventos.
Descripción	El usuario puede ver estadísticas de la duración del promedio de la duración de todos sus eventos.
Actores	Usuario general
Precondiciones	El usuario deber estar registrado en el sistema
Flujo principal	1. El usuario va a la sección de estadísticas. 2. Mantiene el cursor por encima de la sección de tiempo ocupado. 3. El sistema muestra un texto con las horas y minutos o solo los minutos de la duración promedio de sus eventos.
Flujo alternativo	El sistema no encuentra eventos y muestra un valor de 0 minutos.

Tabla 8. Plantilla P06

ID	**P07**
Caso de uso	Análisis de Calendario
Descripción	El usuario configura un rango de fechas y horas laborales para analizar su calendario y optimizar la gestión de su tiempo.
Actores	Usuario general
Precondiciones	El usuario deber estar registrado en el sistema
Flujo principal	1. El usuario selecciona "Configurar Análisis de Calendario" desde el menú de opciones. 2. El sistema presenta un formulario para seleccionar las fechas de inicio y fin del análisis. 3. El usuario elige una fecha de inicio y una fecha de fin para el periodo a analizar. 4. El usuario establece la hora de inicio y fin de su jornada laboral. 5. El usuario marca los días de la semana en los que no trabaja. 6. El usuario confirma la configuración y solicita el análisis. 7. El sistema procesa la información y presenta el análisis del uso del tiempo en el rango especificado.
Flujo alternativo	A) Si el usuario deja campos requeridos sin llenar y confirma la configuración, el sistema muestra el mensaje: "Introduce datos válidos". B) Si la fecha de inicio es posterior a la fecha de fin, o la hora de inicio de la jornada es posterior a la hora de fin, o si las fechas son

	nulas, el sistema muestra el mensaje: "La fecha y hora de inicio debe ser anterior a la fecha y hora de fin." C) Si la fecha de inicio y la fecha de fin son iguales, el sistema muestra el mensaje: "Escoge un rango de fechas, no solo un día".

Tabla 9. Plantilla P07

ID	**P08**
Caso de uso	Búsqueda de Eventos por Palabra Clave
Descripción	El usuario busca eventos en su calendario utilizando palabras clave para encontrar aquellos que coincidan con sus intereses y necesidades.
Actores	Usuario general
Precondiciones	El usuario deber estar registrado en el sistema
Flujo principal	1. El usuario selecciona la opción "Buscar Eventos" en la interfaz. 2. El sistema muestra campos para introducir las fechas de inicio y fin del período a buscar y un campo para la palabra clave. 3. El usuario introduce una palabra clave relacionada con el tipo de evento que desea encontrar y selecciona las fechas de inicio y fin. 4. El usuario confirma la búsqueda haciendo clic en "SUBMIT". 5. El sistema procesa la solicitud y muestra una lista de eventos que coinciden con la palabra clave y el rango de fechas especificado.7. El sistema procesa la información y presenta el análisis del uso del tiempo en el rango especificado.
Flujo alternativo	A) Si el usuario no introduce las fechas o la palabra clave y confirma la búsqueda, el sistema muestra el mensaje: "Introduce datos válidos". B) Si la fecha de inicio es posterior a la fecha de fin, o si alguna de las fechas es nula, el sistema muestra el mensaje: "La fecha y hora de inicio debe ser anterior a la fecha y hora de fin". C) Si el campo de la palabra clave está vacío, el sistema muestra el mensaje: "Introduce una palabra clave, no deje el campo vacío".

Tabla 10. Plantilla P08

5. Análisis y Diseño del sistema

Esta sección describe los modelos de análisis y diseño de la aplicación.

5.1 Modelo de análisis

5.1.1 Elección del patrón arquitectónico

Dada la estructura de tu aplicación, que utiliza un front-end en React y un back-end en Java para interactuar con la API de Google Calendar, aplicando un patrón arquitectónico de tres capas, aquí tienes una descripción reestructurada:

La arquitectura de nuestra aplicación está diseñada siguiendo un patrón de tres capas, enfocado en garantizar escalabilidad y robustez en la seguridad. Este patrón divide la aplicación en tres niveles distintos, cada uno con responsabilidades y roles bien definidos, facilitando el manejo independiente de cada capa y asegurando una gestión

eficiente de la seguridad, especialmente en la capa lógica. Las capas de nuestra aplicación son:

- **Capa de Presentación**: Implementada mediante React, esta capa constituye la interfaz de usuario de la aplicación. Se encarga de la interacción directa con el usuario, proporcionando una experiencia de usuario fluida y atractiva. Su función principal es presentar la información de manera clara y recoger los datos del usuario, sirviendo como el punto de entrada para las interacciones del sistema.

- **Capa de Lógica de Negocio:** Desarrollada en Java, esta capa forma el núcleo central de la aplicación. Aquí se procesa la información recogida por la capa de presentación, se aplican los análisis y se gestionan las interacciones con la API de Google Calendar. Esta capa es responsable de ejecutar la lógica de negocio, incluyendo el procesamiento y análisis de los eventos del calendario, y actúa como el intermediario entre la interfaz de usuario y la base de datos.

- **Capa de Datos:** Aunque en este caso no interactuamos directamente con una base de datos tradicional, esta capa está representada por la API de Google Calendar. Aquí se almacena y se gestiona la información de los eventos del calendario, y nuestra aplicación la utiliza para recuperar, procesar y analizar dicha información.

Cada una de estas capas está diseñada para funcionar de manera independiente, permitiendo una escalabilidad eficaz y facilitando la implementación de medidas de seguridad específicas, especialmente en la capa de lógica de negocio, donde se procesan y se gestionan los datos críticos de la aplicación.

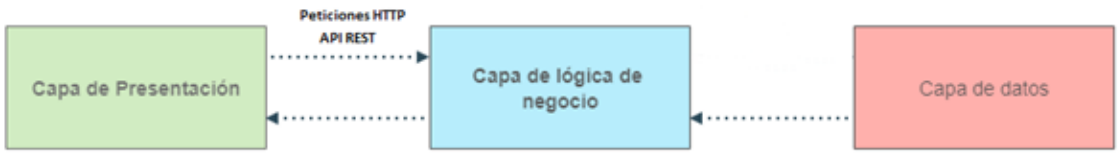


Figura 5. Esquema de Capas

5.2 Diseño arquitectónico

Como se ha explicado en el punto anterior se utiliza un modelo de tres capas para el desarrollo de este proyecto. Donde cada capa irá enlaza con las correspondientes como se puede observar en el siguiente diseño.

- **Capa de datos:** En lugar de una tradicional capa de datos o persistencia, nuestra aplicación se basa en una "Capa de Integración de Datos" que interactúa directamente con la API de Google Calendar. Esta capa se encarga de obtener información en tiempo real de los eventos almacenados en los calendarios de Google. Al no utilizar una base de datos propia para almacenar y modificar información, nuestra arquitectura simplifica el flujo de datos y se centra en el análisis y la extracción de información relevante directamente desde la fuente.

- **Capa de lógica de negocio:** es el núcleo de la aplicación y se basa en una API REST implementada con Java y Spring Boot. Esta capa recibe las llamadas HTTP de la capa

de presentación, gestiona la información y realiza las peticiones pertinentes a la capa de datos.

- **Capa de presentación**: es la más visual de todas ya que desarrolla la interfaz de usuario que se implementa con React. Esta se encarga de recibir las peticiones del usuario y transmitirlas a la capa de negocio.

A continuación, se muestran los métodos de la clase RestController que implementa la funcionalidad del back-end para ser invocadad desde el fornt-end.

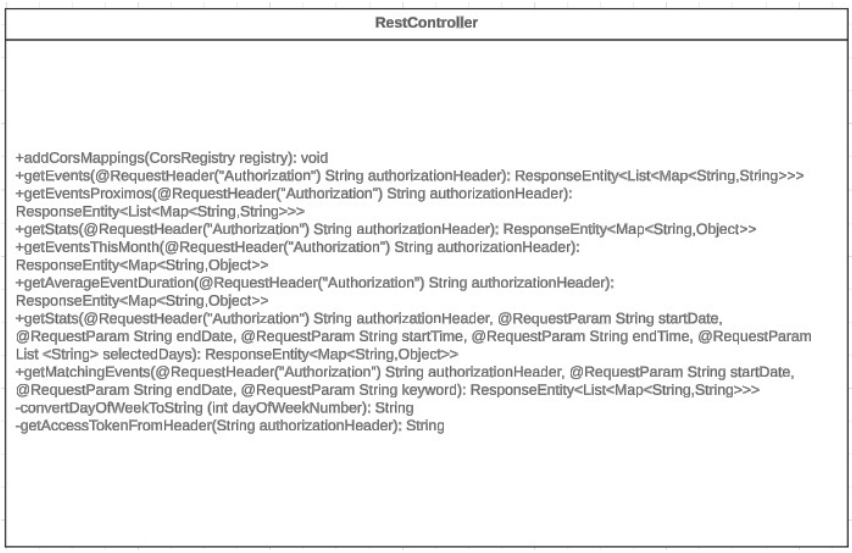


Figura 6. Diagrama del diseño arquitectural del sistema

5.2.1 Diseño de la capa de datos

Como se explica en la fase de desarrollo (3.2.3), en la fase 1 del proyecto se consideró detenidamente la estructura de manejo de datos. Tras evaluar diversas opciones, se decidió prescindir de una base de datos relacional, como MySQL, dada la naturaleza y objetivos de nuestra aplicación. En su lugar, optamos por una integración directa y eficiente con la API de Google Calendar. Esta elección se basó en la necesidad de acceder y analizar datos en tiempo real, priorizando una arquitectura simplificada que se centra en el procesamiento y análisis dinámico de los eventos del calendario, evitando así la complejidad y la gestión de una capa de datos tradicional.

Las características principales de esta capa son:

- Integración Directa con Google Calendar:

  La capa de integración se conecta directamente con la API de Google Calendar, extrayendo datos de eventos en tiempo real.
  Esto permite un análisis actualizado y dinámico de los datos sin la necesidad de almacenarlos en una base de datos separada.

- Procesamiento y Análisis en Tiempo Real:

Al trabajar directamente con los datos de Google Calendar, nuestra aplicación puede procesar y analizar la información al momento, proporcionando insights y análisis basados en los datos más recientes.
Esta aproximación asegura que los usuarios tengan acceso a información actualizada y relevante sin retrasos por procesos de almacenamiento o sincronización de datos.

- Simplificación y Eficiencia:

  Al evitar una capa de datos tradicional, reducimos la complejidad del sistema y minimizamos los requisitos de mantenimiento y gestión de infraestructura de base de datos.
  Esta estructura ofrece una solución más ligera y directa para las necesidades de análisis de datos de calendario, optimizando tanto el rendimiento como la escalabilidad de la aplicación.

Cada evento en Google Calendar puede contener información específica relevante, como puede ser el tipo de evento directamente en su descripción o en campos personalizados. En nuestra aplicación, nos enfocamos exclusivamente en extraer y analizar información de reuniones programadas en Google Calendar. No gestionamos ni controlamos directamente los horarios de estas reuniones, ya que estas tareas son manejadas automáticamente por Google Calendar. Nuestro papel es analizar los datos existentes de las reuniones, como su frecuencia, duración, y participantes, para obtener información valiosa. Esto se hace sin la necesidad de intervenir en la programación directa de las reuniones o en la gestión de horarios disponibles, aprovechando la eficiencia de Google Calendar en el manejo de estos aspectos.

5.2.2 Diseño de la capa de lógica de negocio.

La capa de lógica de negocio se ofrece a través de una API REST implementada con Java y Spring Boot. La capa de lógica de negocio en nuestra aplicación se gestiona a través de una API, la cual interactúa directamente con la API de Google Calendar para obtener y analizar datos de reuniones. Aunque nuestra aplicación no sigue estrictamente el patrón de "modelo, vista, controlador" debido a su naturaleza especializada, podemos entender sus componentes de la siguiente manera:

- Modelo: En lugar de una representación de datos de una capa de persistencia, nuestro "modelo" es la información obtenida directamente de Google Calendar. Aquí, el modelo consiste en los datos de eventos y reuniones, como fechas, horarios y participantes.

- Vista: No manejamos una 'vista' en el sentido tradicional de interfaces de usuario, ya que nuestra aplicación se centra en el procesamiento de datos. En cambio, la 'vista' puede considerarse como la presentación de análisis y resultados obtenidos de los datos de Google Calendar.

- Controlador: El controlador es el núcleo de nuestra aplicación, gestionando la interacción con la API de Google Calendar y procesando los datos de eventos para análisis. Es el intermediario entre la fuente de datos (Google Calendar) y el resultado del análisis.

La API de nuestra aplicación gestiona las solicitudes para extraer y analizar datos de eventos de Google Calendar. Aunque no manejamos una base de datos propia, utilizamos códigos de estado HTTP para indicar el resultado de las solicitudes, tales como:

- 200 OK: La solicitud de extracción de datos fue exitosa.
- 204 No Content: La solicitud fue exitosa pero no hay nuevos datos para mostrar.
- 400 Bad Request: La solicitud contiene un error, posiblemente en los parámetros de la consulta.
- 404 Not Found: No se encontraron eventos o datos correspondientes a la consulta.

El desarrollo de nuestra aplicación se estructura en fases, donde inicialmente se enfoca en establecer la conexión y comunicación eficiente con la API de Google Calendar, seguido por la implementación de análisis de datos avanzados en fases posteriores.
Como se explicó en el apartado de planificación el desarrollo se divide en tres fases.

5.2.3 Diseño de la capa web

Un aspecto destacado de NoMoreMeetings es su sistema de inicio de sesión simplificado. Los usuarios pueden acceder a la aplicación mediante un inicio de sesión con Google, lo que facilita un proceso de autenticación rápido y seguro. Al ingresar, se redirige a los usuarios a la página principal, donde pueden comenzar a explorar todas las características que ofrece la aplicación.

La interfaz de NoMoreMeetings se caracteriza por su facilidad de navegación, gracias a una barra de navegación intuitiva. Esta barra incluye enlaces a distintas secciones vitales como el home, beneficios de organizar tu tiempo, unos análisis generales, un análisis más detallado y un buscador de eventos. Además, el diseño responsivo garantiza una experiencia de usuario coherente en dispositivos de diferentes tamaños.

Una vez el usuario ha iniciado sesión nos encontramos una aplicación con múltiples secciones:

- **Home**: La página de inicio de NoMoreMeetings presenta un diseño claro y profesional con un fuerte mensaje de productividad. Con un llamativo titular y un lema que invita a liberar el tiempo del usuario, establece inmediatamente el propósito de la aplicación. La sección incluye una explicación concisa de los beneficios y botones prominentes para "Comenzar ahora" o "Aprender más", facilitando la navegación y acción inmediata. La imagen de fondo muestra a un equipo trabajando, lo que refuerza visualmente el enfoque colaborativo de la herramienta y la importancia de organizarse.

Figura 7. Mockup Página de Inicio

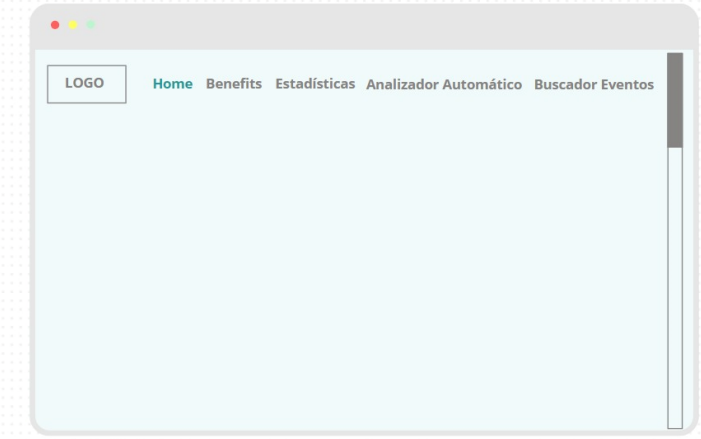


Figura 8. Mockup Barra de Inicio

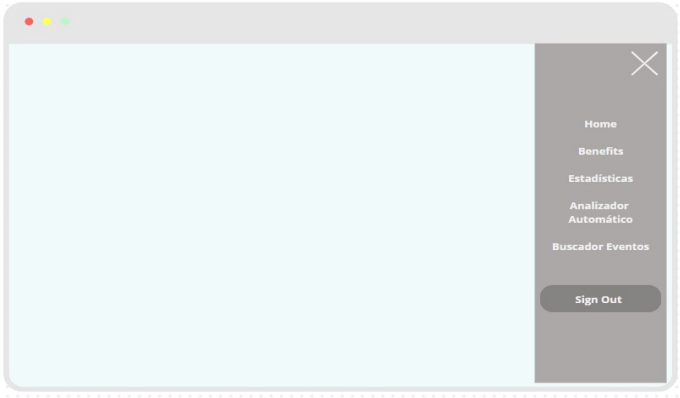


Figura 9. Mockup Barra Lateral

- **Benefits**: En esta sección se destaca cinco ventajas principales: "Mayor Productividad", enfocando en la eficiencia en la gestión del tiempo; "Reducción del Estrés", ofreciendo una planificación clara para evitar la sobrecarga de tareas; "Mejor Toma de Decisiones", gracias a un manejo de tiempo que permite reflexionar con más claridad; "Mejora Calidad de Trabajo", al dedicar el tiempo necesario a cada tarea; y "Crecimiento a nivel Profesional", potenciando las oportunidades de avance en la carrera gracias a una mejor organización del tiempo.

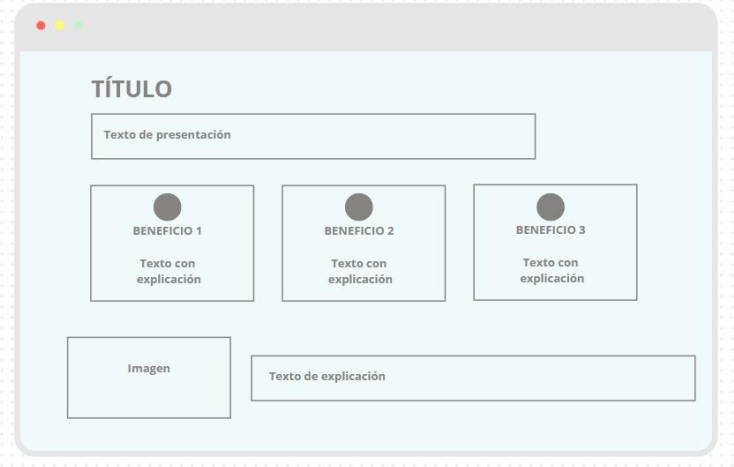


Figura 10. Mockup Página Benefits

- **Estadísticas**: Esta sección ofrece a los usuarios una vista detallada visual e interactiva que ofrece a los usuarios una visión global de su tiempo, enfatizando la organización personal y profesional. Al pasar el cursor sobre las imágenes de calendarios, se despliegan datos clave como el porcentaje de tiempo libre y ocupado, el número total de eventos del mes, los próximos eventos programados y la duración promedio de estos. Esta función interactiva no solo proporciona información relevante de un vistazo sino que también facilita la toma de decisiones informadas para una planificación futura más efectiva.

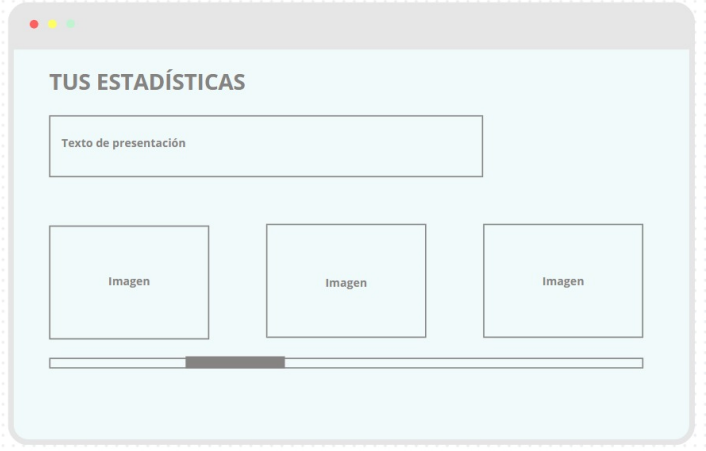


Figura 11. Mockup Página Estadísticas

- **Analizador Automático**: Aquí, se invita a los usuarios a ingresar las fechas de inicio y fin para el análisis, así como a especificar sus horas laborales y seleccionar los días no laborables. Esta funcionalidad permite una personalización del análisis de calendario para reflejar el tiempo ocupado en reuniones, tiempo sin reuniones, y otros compromisos, excluyendo los días no laborables. Tras hacer clic en "SUBMIT", la página revela estadísticas detalladas de tiempo libre y ocupado, visualizadas gráficamente en la parte derecha de la pantalla encima de la foto, proporcionando una herramienta interactiva para una gestión del tiempo más efectiva y consciente.

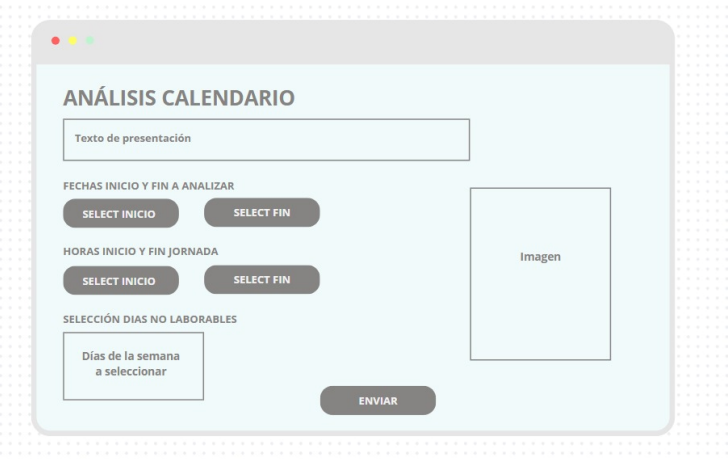


Figura 12. Mockup Página Analizador Automático

- **Buscador de Eventos:** Aquí, se invita a los usuarios a ingresar las fechas de inicio y fin para el análisis, así como a especificar sus horas laborales y seleccionar los días no laborables. Esta funcionalidad permite una personalización del análisis de calendario para reflejar el tiempo ocupado en reuniones, tiempo sin reuniones, y otros compromisos, excluyendo los días no laborables. Tras hacer clic en "SUBMIT", la página revela estadísticas detalladas de tiempo libre y ocupado, visualizadas gráficamente en la parte derecha de la pantalla, proporcionando una herramienta interactiva para una gestión del tiempo más efectiva y consciente.

Figura 13. Mockup Página Búsqueda de Eventos

La paleta de colores de NoMoreMeetings, compuesta por tonos neutros y acentos cálidos, crea un ambiente acogedor y enfocado, mientras que la tipografía clara y contemporánea asegura facilidad de lectura, ambas reflejando profesionalismo y

accesibilidad. Las imágenes cuidadosamente seleccionadas, como calendarios y espacios de trabajo, complementan visualmente el enfoque de la aplicación en la gestión del tiempo y la organización.

La interactividad es un componente clave de NoMoreMeetings. Los formularios de búsqueda y análisis permiten a los usuarios ingresar datos específicos, como fechas y palabras clave, para personalizar los resultados de eventos y estadísticas de tiempo, lo que facilita una planificación más eficiente y una visión clara de su agenda. Los botones de acción como "Start Now" y "Learn More" están diseñados para redirigirte de manera rápida a la sección principal de la app, proporcionando un acceso inmediato a las funcionalidades clave y fomentando una experiencia de usuario fluida y sin interrupciones.

El footer de nuestra página web incluye una breve descripción de los servicios que ofrece NoMoreMeetings, enfatizando su rol en la optimización de reuniones laborales. Además, muestra los derechos de autor, asegurando la propiedad intelectual de los contenidos con la frase "© UC All Rights Reserved". Para mayor interacción y engagement, el footer proporciona enlaces directos a las redes sociales de la plataforma y ofrece medios de contacto, como direcciones de correo electrónico y un número de teléfono, ofreciendo múltiples canales de comunicación con la empresa.

Figura 14. Mockup Footer

5.3 Modelo de despliegue

En este apartado se describe el despliegue de la aplicación durante el tiempo de desarrollo.

Durante el desarrollo se despliega la aplicación de la siguiente manera:

- Interacción con la API de Google Calendar: Nuestra aplicación se conecta directamente con la API de Google Calendar para obtener y procesar datos en tiempo real.
- La API REST se despliega en un servidor tomcat proporcionado por Maven y springboot. Se accede a ella a través de la url: “http://localhost:8080/”
- La interfaz web se despliega en el servidor de desarrollo en el puerto 5173, es decir a través de la url: “http://localhost: 5173/”.

6. Implementación

Esta sección describe la implementación realizada. El código se ofrece de forma abierta para su utilización en el repositorio de GitHub: https://github.com/JesRevu/NoMoreMeetings

6.1 Implementación back-end

El back-end de nuestra aplicación está diseñado para interactuar eficientemente con la API de Google Calendar, desarrollado en Java utilizando Spring Boot.

6.1.1. API REST

Nuestra API REST, definida en la clase App, incluye varios endpoints, cada uno diseñado para una funcionalidad específica:

- @GetMapping("/events"):
    - Propósito: Este endpoint recupera una lista de los próximos eventos del calendario.
    - Funcionamiento: Utiliza la API de Google Calendar para obtener los próximos 10 eventos del calendario principal del usuario. Devuelve una lista de mapas, cada uno representando un evento, incluyendo detalles como el resumen y la hora de inicio.
- @GetMapping("/eventsProximos"):
    - Propósito: Similar al endpoint /events, pero parece estar enfocado en eventos futuros a partir de la fecha y hora actual.
    - Funcionamiento: Obtiene eventos que ocurren después del momento actual, devolviendo detalles como el resumen, la hora de inicio y finalización.
- @GetMapping("/stats"):
    - Propósito: Calcula estadísticas de ocupación del calendario para el mes actual.
    - Funcionamiento: Determina el porcentaje de tiempo ocupado y libre basado en los eventos del mes actual. Calcula la duración total de los eventos y la compara con el tiempo total disponible en el mes.
- @GetMapping("/eventsThisMonth"):
    - Propósito: Obtiene el número total de eventos en el calendario para el mes actual.
    - Funcionamiento: Recupera todos los eventos del mes actual y devuelve su cantidad.
- @GetMapping("/averageEventDuration"):

    - Propósito: Calcula la duración media de los eventos en el calendario.
    - Funcionamiento: Obtiene todos los eventos del calendario, calcula la duración de cada uno y luego determina la duración media.
- @GetMapping("/calendar"):

    - Propósito: Proporciona estadísticas detalladas de ocupación para un rango de fechas y horas específico, junto con días seleccionados.
    - Funcionamiento: Este endpoint es más complejo, permite a los usuarios especificar un rango de fechas, horas y días específicos para analizar. Calcula el porcentaje de tiempo ocupado durante ese período.

- @GetMapping("/calendarString"):

    - Propósito: Busca eventos que coincidan con una palabra clave específica en un rango de fechas.
    - Funcionamiento: Los usuarios pueden proporcionar una palabra clave, una fecha de inicio y una fecha de fin, y el sistema devuelve eventos que contengan esa palabra clave en su título o descripción durante el período especificado.

Las anotaciones son fundamentales para el funcionamiento de la aplicación Spring Boot, ya que definen cómo se manejan las rutas, los métodos HTTP, la seguridad y el flujo de datos en la API.

- @RestController: Esta anotación se usa para definir un controlador en Spring Boot. Indica que la clase se utiliza para manejar solicitudes web. Un RestController es una combinación de @Controller y @ResponseBody que indica que los datos devueltos por cada método son escritos directamente en el cuerpo de la respuesta, en lugar de ser enviados a una plantilla de vista.

- @RequestMapping("/app"): Esta anotación se utiliza para mapear solicitudes web a métodos en los controladores. En tu caso, define que todas las solicitudes que comiencen con "/app" serán manejadas por la clase App.

- @CrossOrigin(origins = "http://localhost:5173"): Esta anotación se usa para permitir solicitudes de origen cruzado (CORS) desde el origen especificado. Esto es útil durante el desarrollo para permitir que tu front-end, que se ejecuta en un servidor diferente, acceda a tu back-end.

- @GetMapping: Especifica que un método debería ser llamado cuando se realiza una solicitud GET a la URL definida. Por ejemplo, @GetMapping("/events") indica que el método asociado se ejecutará cuando se acceda a la ruta "/app/events" con una solicitud GET.

- @RequestHeader("Authorization"): Se utiliza para extraer el valor de un encabezado específico de la solicitud HTTP. En tu caso, obtienes el valor del encabezado "Authorization" que se utiliza para la autenticación.

- @RequestParam: Se usa para extraer parámetros de la URL de la solicitud. Por ejemplo, si tu URL es algo como "/app/calendar?startDate=2021-01-01", puedes usar @RequestParam para obtener el valor de startDate.

- @Autowired: Esta anotación se utiliza para la inyección automática de dependencias en Spring. Aunque no la veo en tu código, se usaría típicamente para inyectar beans como servicios o repositorios en tus controladores.

- @ResponseBody: Aunque no se utiliza explícitamente en tu código, está implícita en @RestController. Indica que el resultado de los métodos del controlador se serializa automáticamente a JSON y se devuelve en el cuerpo de la respuesta HTTP.

6.1.2. Herramientas de Desarrollo

- Maven: Se utiliza para la gestión de dependencias y construcción del proyecto.
- Spring Boot: Facilita la configuración y despliegue de la aplicación.

6.1.3. Seguridad y el Token de Acceso

- Uso del Token de Acceso:

  Cada solicitud a la API de Google Calendar requiere un token de acceso. Este token es un elemento clave en la seguridad de la API.

  El token de acceso es una cadena de texto que actúa como una credencial temporal. Es proporcionado por Google una vez que un usuario se autentica y autoriza tu aplicación para acceder a su calendario.

  Cuando un usuario realiza una solicitud a tu back-end, debe incluir este token en el encabezado Authorization de su solicitud HTTP.

- Validación del Token:

  Se utiliza el método getAccessTokenFromHeader para extraer y validar este token de las solicitudes entrantes.
  Este proceso asegura que solo los usuarios que se han autenticado correctamente y han concedido permisos a tu aplicación puedan acceder a sus datos de Google Calendar.

  Si el token no está presente o es inválido, la solicitud se rechaza, lo que previene el acceso no autorizado.

- Cross-Origin Resource Sharing (CORS):

  - Necesidad de CORS:

CORS es un mecanismo de seguridad en los navegadores web que restringe cómo los recursos de una página web pueden ser solicitados desde otro dominio.
Durante el desarrollo, es común tener el front-end y el back-end corriendo en diferentes puertos o dominios (por ejemplo, tu back-end en localhost:8080 y tu front-end en localhost:5173), lo que puede causar problemas de CORS.

- Configuración de CORS:

  En el back-end, se usa la anotación @CrossOrigin(origins = "http://localhost:5173") para permitir explícitamente las solicitudes cruzadas desde el dominio de tu front-end.
  Esto le dice al navegador que está bien aceptar solicitudes a tu back-end provenientes de tu front-end, aun cuando estén en diferentes dominios o puertos.
  Además, en el método addCorsMappings, se ha configurado políticas CORS más detalladas, como los métodos HTTP permitidos y los encabezados permitidos.

6.1.4. Detalles de Implementación y Configuración

Nuestros controladores no solo gestionan solicitudes HTTP sino que también incorporan la lógica de negocio y el procesamiento de datos. Por ejemplo, el controlador /stats, no solo recupera datos sino que realiza cálculos para proporcionar estadísticas relevantes del calendario y el controlador /averageEventDuration calcula la duración promedio de eventos en el calendario, ofreciendo insights importantes sobre el uso del calendario.
Este enfoque de controladores integrados simplifica la arquitectura de la aplicación, mejorando la eficiencia y la mantenibilidad, especialmente útil para aplicaciones con un enfoque y alcance claros como la nuestra.

6.2 Implementación del front-end

La implementación del front-end, de una aplicación moderna es un componente crucial para garantizar una experiencia de usuario fluida y atractiva. En el desarrollo de NoMoreMeetings, se ha elegido React, una biblioteca de JavaScript para construir interfaces de usuario, como el eje central del front-end, aprovechando su capacidad de crear aplicaciones web dinámicas y de una sola página.

Este proyecto basado en un template [13], utiliza TypeScript, una extensión de JavaScript que añade tipado estático, para mejorar la calidad del código y facilitar el mantenimiento y la escalabilidad. Cada archivo .tsx representa un componente o una "escena" dentro de la aplicación, lo que permite una separación clara de la lógica y la presentación.

Visual Studio Code, un editor de código fuente ligero pero poderoso, se ha utilizado como entorno de desarrollo, aprovechando sus numerosas extensiones y herramientas integradas que mejoran la eficiencia del desarrollo. La arquitectura de la aplicación está diseñada para ser modular y reutilizable, haciendo uso de componentes de React para encapsular distintas áreas de funcionalidad y presentación.

A continuación, se detallará la estructura y el código específico de app.tsx, así como las diferentes "escenas" que componen la interfaz de usuario de NoMoreMeetings, proporcionando una visión de cómo se configuran y se relacionan entre sí para crear una experiencia de usuario cohesiva y funcional.

Para comenzar con la descripción detallada del front-end, veamos el código de app.tsx, el cual actúa como el corazón de la aplicación NoMoreMeetings. Este archivo es responsable de coordinar los componentes principales, gestionar la navegación y mantener el estado relevante de la aplicación.

En app.tsx, se importan los componentes esenciales de la aplicación, como la barra de navegación (Navbar), la página de inicio (Home), la página de análisis general (GeneralAnalysis), la página de beneficios (Benefits), el análisis de eventos (EventAnalysis), la herramienta de búsqueda de eventos (FindEvent), y el pie de página (Footer). También se importa Login, que es el componente encargado de manejar la autenticación de los usuarios.

```tsx
function App() {
  const [selectedPage, setSelectedPage] = useState<SelectedPage>(
    SelectedPage.Home
  );
  const [isTopOfPage, setIsTopOfPage] = useState<boolean>(true);
  const [user, setUser] = useState< User | null>(null); // Agrega una variable de estado para el usuario

  useEffect(() => {
    const handleScroll = () => {
      if (window.scrollY === 0) {
        setIsTopOfPage(true);
        setSelectedPage(SelectedPage.Home);
      }
      if (window.scrollY !== 0) setIsTopOfPage(false);
    };
    window.addEventListener("scroll", handleScroll);
    return () => window.removeEventListener("scroll", handleScroll);
  }, []);

  return (
    <div className="flex-center">
      {user ? ( // Si el usuario ha iniciado sesión, muestra el contenido normal
        <>
          <div className="app bg-gray-20">
            <Navbar
              isTopOfPage={isTopOfPage}
              selectedPage={selectedPage}
              setSelectedPage={setSelectedPage}
              setUser={setUser} // Pasa la función setUser al componente Navbar
            />
            <Home setSelectedPage={setSelectedPage} />
            <Benefits setSelectedPage={setSelectedPage} />
            <GeneralAnalysis setSelectedPage={setSelectedPage} />
            <EventAnalysis setSelectedPage={setSelectedPage} />
            <FindEvent setSelectedPage={setSelectedPage} />
            <Footer />
          </div>
        </>
      ) : ( // Si el usuario no ha iniciado sesión, muestra la página de inicio de sesión
          <Login onLogin={(userData) => setUser(userData)} />
      )}
    </div>
  );
}
export default App;
```

Figura 15. App.tsx

El estado de la aplicación se maneja mediante hooks de React. Se utiliza useState para mantener la página seleccionada, la posición de desplazamiento y los datos del usuario. Estos estados son fundamentales para la lógica de navegación y la personalización de la interfaz de usuario en función del estado de la sesión del usuario.

El hook useEffect se implementa para escuchar los eventos de desplazamiento (scroll) de la ventana, lo cual es una técnica común para cambiar el estado de la UI basándose en la interacción del usuario con la página. Por ejemplo, la aplicación puede destacar la barra

de navegación o cambiar la página seleccionada automáticamente cuando el usuario se desplaza.

Dependiendo del estado de autenticación del usuario (user), app.tsx decide si mostrar la página de inicio de sesión (Login) o el contenido principal de la aplicación. Esto es un ejemplo de renderizado condicional en React, lo que permite proteger las rutas y asegurar que sólo los usuarios autenticados puedan acceder a ciertas partes de la aplicación.

El uso de TypeScript se evidencia en las anotaciones de tipo, como SelectedPage y User, que mejoran la detección de errores en tiempo de compilación y proporcionan autocompletado y referencias más claras para los desarrolladores.

El código también muestra una preocupación por la estética y la usabilidad, ya que incluye la importación de un archivo SASS para estilos, lo que sugiere una atención a la personalización y coherencia visual de la aplicación.

La estructura modular y el uso de hooks y TypeScript ilustran una aplicación bien organizada y mantenible, reflejando las mejores prácticas modernas en el desarrollo del front-end.
La página de inicio de sesión, SignIn, es un componente clave en la arquitectura de NoMoreMeetings, que maneja la autenticación de los usuarios. Se utiliza React junto con la biblioteca de íconos phosphor-react y el servicio de autenticación de Firebase para implementar el inicio de sesión con Google.

El componente SignIn define una prop onLogin que se utiliza para pasar los datos del usuario una vez que se ha autenticado exitosamente. Se gestiona un estado local user para almacenar la información del usuario de Firebase. La función signInWithGoogle crea una instancia de GoogleAuthProvider, añade un ámbito para permitir el acceso de solo lectura al calendario de Google, y luego maneja el proceso de inicio de sesión utilizando signInWithPopup. Si el inicio de sesión es exitoso, se guarda la información del usuario en el estado local y en localStorage, y se invoca onLogin con los datos del usuario.

Visualmente, la página presenta una interfaz limpia y centrada. Utiliza un esquema de colores suave, con un gradiente de fondo que va de un tono claro a un rosa más cálido, creando un ambiente acogedor y profesional. El estilo del botón de inicio de sesión se mantiene simple, con un borde que incorpora un gradiente y un icono de Google, lo que lo hace instantáneamente reconocible. Los estilos aplican una transición en el hover, mejorando la retroalimentación visual para el usuario.

El archivo styles.scss configura los estilos globales y específicos del componente SignIn. La configuración del contenedor asegura que el formulario de inicio de sesión se centre en la pantalla, y los estilos del botón se diseñan para ser amplios y cómodos para la interacción del usuario. Además, se sigue una convención de nomenclatura clara y se utilizan variables CSS para mantener la coherencia y facilitar los cambios de tema.

```tsx
export function SignIn({ onLogin }: SignInProps) {
  const [user, setUser] = useState<User>({} as User);

  function signInWithGoogle() {
    const provider = new GoogleAuthProvider();

    provider.addScope('https://www.googleapis.com/auth/calendar.readonly');

    signInWithPopup(auth, provider)
      .then((result) => {
        console.log(result);
        setUser(result.user);
        onLogin(result.user);
        localStorage.setItem("googleUser", JSON.stringify(result));

      }).catch((error) => {
        console.log(error);
      });
  }

  const googleUserString = localStorage.getItem("googleUser");
  if (googleUserString) {
    const googleUser = JSON.parse(googleUserString);
    onLogin(googleUser.user);
  }
```

Figura 16. Sign In

```css
body {
  margin: 0;
  padding: 0;
  font-family: Arial, Helvetica, sans-serif;
  background: linear-gradient(to bottom, #F0F4C3, #e98989);;
}

.container {
  width: 100%;
  height: 100vh;
  display: flex;
  flex: 1;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  text-align: center;
  padding: 2rem;

  span {
    color: var(--caption);
  }
}

.button {
  height: 56px;
  width: 650px;

  color: var(--text);
  background: none;
  font-size: 18px;

  border-width: 2px;
  border-style: solid;
  border-image: linear-gradient(to right, var(--primary), var(--gray)) 1;

  margin-top: 32px;

  display: flex;
  align-items: center;
  justify-content: center;

  >svg {
    font-size: 24px;
    margin-right: 7px;
  }
```

Figura 17. Estilo de la página Sign In

La sección Home de la aplicación NoMoreMeetings es una combinación bien estructurada de diseño visual y funcionalidad interactiva. Este componente, escrito en React y utilizando TypeScript, sirve como la página principal de la aplicación y está diseñado para captar la atención del usuario desde el primer momento.

El componente comienza con la importación de varios recursos y herramientas, incluyendo imágenes para el contenido y patrocinadores, un hook personalizado

useMediaQuery para la responsividad, y animaciones de framer-motion para dar vida a los elementos de la UI.

La estructura principal del componente Home se define dentro de una etiqueta <section>, que contiene dos partes principales: el encabezado y la imagen principal, y la sección de patrocinadores.

Encabezado e Imagen Principal:

Se utiliza motion.div de Framer Motion para animar la entrada de los elementos.
Dentro del primer motion.div, hay un encabezado con texto explicativo sobre la aplicación y un llamado a la acción, junto con un enlace "Learn More". Estas animaciones se activan al entrar en el viewport.

Además, hay una imagen que ilustra el propósito de la aplicación, probablemente mostrando una escena relacionada con reuniones o gestión del tiempo.
Sección de Patrocinadores:

Utiliza el hook personalizado useMediaQuery para mostrar esta sección solo en pantallas medianas o más grandes.

```
type Props = {
  setSelectedPage: (value: SelectedPage) => void;
};

const Home = ({ setSelectedPage }: Props) => {
  const isAboveMediumScreens = useMediaQuery("(min-width:1060px)");

  return (
```

Figura 18. Home

El componente Benefits de NoMoreMeetings, dividido en dos archivos, es un ejemplo perfecto de cómo React y sus ecosistemas pueden ser utilizados para crear una interfaz de usuario interactiva y atractiva.

Archivo Principal de Benefits:
En el primer archivo, se importan varios componentes y recursos clave para construir la sección "Benefits". Se utiliza motion de Framer Motion para animar los elementos de la página, mejorando la experiencia visual del usuario. Los beneficios específicos de la aplicación se definen en un array benefits, cada uno con un ícono, título y descripción. Estos beneficios abarcan aspectos como la productividad, la reducción del estrés y la mejora de la calidad del trabajo, destacando las ventajas de utilizar la aplicación.

La estructura de la página se define en un contenedor <section>, donde cada beneficio se presenta a través del componente Benefit. También incluye una parte gráfica (BenefitsPageGraphic) y una descripción textual que resalta la utilidad y el valor de la aplicación para sus usuarios. Un botón ActionButton se coloca al final para fomentar la interacción activa del usuario.

Componente Benefit:
El segundo archivo define el componente Benefit, que es responsable de mostrar cada beneficio individual. Utiliza animaciones para dar vida a los íconos y textos, lo que hace

que la presentación de cada beneficio sea más dinámica e interesante. El componente Benefit se diseñó pensando en la modularidad y la reutilización, permitiendo que cada beneficio se presente de manera consistente y atractiva.

En conjunto, estos dos archivos trabajan armoniosamente para crear una sección "Benefits" que no solo informa a los usuarios sobre las ventajas de la aplicación, sino que también mejora la experiencia general del usuario a través de un diseño visualmente atractivo y una interactividad fluida.

```
const container = {
  hidden: {},
  visible: {
    transition: { staggerChildren: 0.2 },
  },
};

type Props = {
  setSelectedPage: (value: SelectedPage) => void;
};

const Benefits = ({ setSelectedPage }: Props) => {
  return (
```

Figura 19. Animaciones dinámicas y manejo de estado de Benefits

La sección "Estadísticas" en NoMoreMeetings, desarrollada a través de dos archivos de

```
const benefits: Array<BenefitType> = [
  {
    icon: <HomeModernIcon className="h-6 w-6" />,
    title: "Mayor Productividad",
    description:
      "La gestión del tiempo eficiente te permite priorizar tareas y enfocarte en lo más importante.",
  },
  {
    icon: <HeartIcon className="h-6 w-6" />,
    title: "Reducción del Estrés",
    description:
      "Al tener un plan claro y organizado, puedes evitar la sensación de estar abrumado por múltiples tareas y plazos.",
  },
  {
    icon: <CalendarIcon className="h-6 w-6" />,
    title: "Mejor Toma de Decisiones",
    description:
      "Al tener un manejo adecuado del tiempo, tienes la oportunidad de tomar decisiones más informadas y pensadas.",
  },
  {
    icon: <BuildingOfficeIcon className="h-6 w-6" />,
    title: "Mejora Calidad de Trabajo",
    description:
      "Al poder dedicar más tiempo y atención a cada tarea, la calidad de tu trabajo tiende a mejorar, lo que puede tener un impacto positivo.",
  },
  {
    icon: <AcademicCapIcon className="h-6 w-6" />,
    title: "Crecimiento a nivel Profesional",
    description:
      "Una gestión eficiente del tiempo puede hacer que seas más eficiente y efectivo en tu trabajo, por lo que lleva a oportunidades de crecimiento y desarrollo dentro de tu
  },
];
```

Figura 20. Benefits

código, ofrece una visión integral y dinámica de las estadísticas personales del usuario. Esta parte de la aplicación se enfoca en presentar datos clave sobre el uso del tiempo, como el porcentaje de ocupación, tiempo libre, número de eventos y duración promedio de las reuniones.

El archivo principal establece la estructura y lógica del componente. Utiliza hooks de React, como useState y useEffect, para manejar el estado y recuperar datos del back-end. Estas solicitudes proporcionan información en tiempo real sobre las actividades y el calendario del usuario. La visualización de los datos se realiza a través del componente Class, donde cada estadística se muestra en una tarjeta con texto e imagen correspondiente.

El componente Class, incorpora un diseño interactivo donde, al pasar el ratón sobre cada tarjeta, se revela más información, aumentando la interactividad y el atractivo visual. Este diseño asegura que la información sea no solo accesible sino también atractiva, mejorando la experiencia general del usuario.

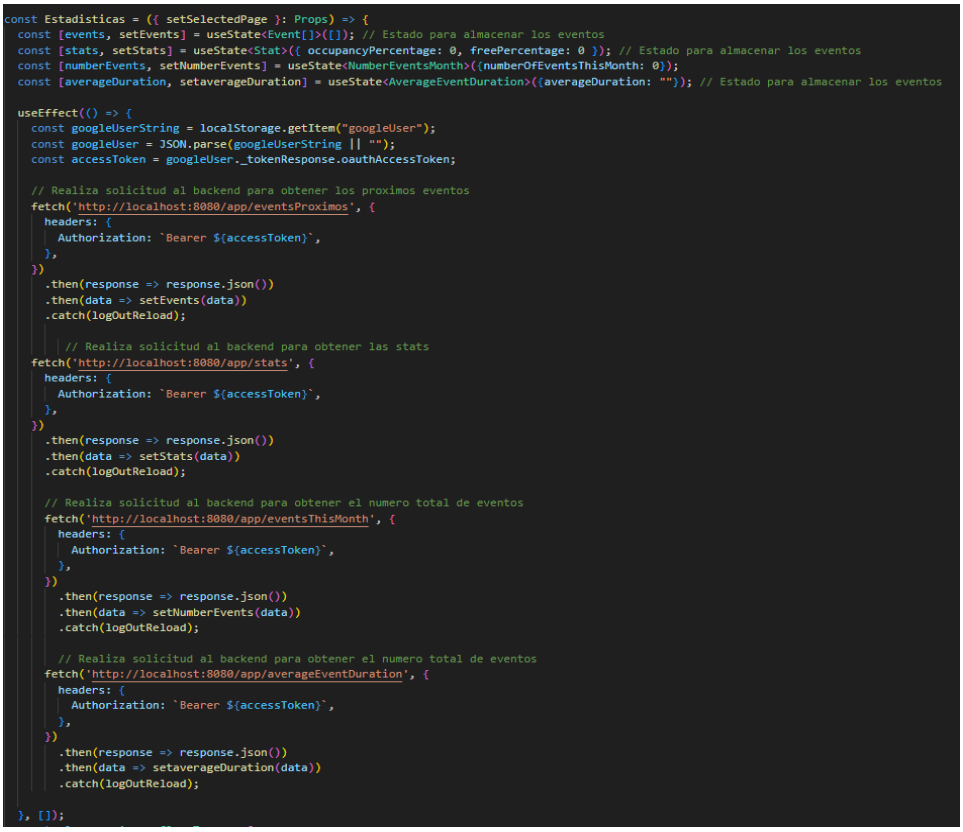
```
type Props = {
  setSelectedPage: (value: SelectedPage) => void;
};
type Event = {
  summary: string; //datos del evento
  start: Date;
  end: Date;
};
type Stat = {
  occupancyPercentage: number; //datos de las stats de eventos
  freePercentage: number;
};
type NumberEventsMonth = {
  numberOfEventsThisMonth: number; //datos de numeros de eventos este mes
};
type AverageEventDuration = {
  averageDuration: string; //duracion promedia de eventos
};
```

Figura 21. Lógica de eventos y estadísticas

La sección "Analizador Automático" en NoMoreMeetings, desarrollada en React, es una herramienta interactiva diseñada para proporcionar un análisis profundo del uso del tiempo del usuario. Esta parte de la aplicación permite a los usuarios ingresar fechas y horas específicas para analizar su calendario, seleccionando días laborales y no laborales para un análisis personalizado.

Al enviar el formulario, la aplicación realiza solicitudes al back-end para recopilar y analizar datos del calendario del usuario. Los resultados del análisis, que incluyen el porcentaje de tiempo ocupado y tiempo libre, se muestran en un popup, proporcionando información detallada y útil de manera clara y concisa.

El diseño de la interfaz de usuario combina elementos gráficos y animaciones para crear una experiencia visualmente atractiva y fácil de usar. La validación del formulario asegura la precisión de los datos antes de su análisis, y en caso de errores en la entrada de datos, se muestra un modal con mensajes de error para guiar al usuario y mejorar la experiencia de interacción.



```
const Estadisticas = ({ setSelectedPage }: Props) => {
  const [events, setEvents] = useState<Event[]>([]); // Estado para almacenar los eventos
  const [stats, setStats] = useState<Stat>({ occupancyPercentage: 0, freePercentage: 0 }); // Estado para almacenar los eventos
  const [numberEvents, setNumberEvents] = useState<NumberEventsMonth>({numberOfEventsThisMonth: 0});
  const [averageDuration, setaverageDuration] = useState<AverageEventDuration>({averageDuration: ""}); // Estado para almacenar los eventos

  useEffect(() => {
    const googleUserString = localStorage.getItem("googleUser");
    const googleUser = JSON.parse(googleUserString || "");
    const accessToken = googleUser._tokenResponse.oauthAccessToken;

    // Realiza solicitud al backend para obtener los proximos eventos
    fetch('http://localhost:8080/app/eventsProximos', {
      headers: {
        Authorization: `Bearer ${accessToken}`,
      },
    })
      .then(response => response.json())
      .then(data => setEvents(data))
      .catch(logOutReload);

        // Realiza solicitud al backend para obtener las stats
    fetch('http://localhost:8080/app/stats', {
      headers: {
        Authorization: `Bearer ${accessToken}`,
      },
    })
      .then(response => response.json())
      .then(data => setStats(data))
      .catch(logOutReload);

      // Realiza solicitud al backend para obtener el numero total de eventos
      fetch('http://localhost:8080/app/eventsThisMonth', {
        headers: {
          Authorization: `Bearer ${accessToken}`,
        },
      })
        .then(response => response.json())
        .then(data => setNumberEvents(data))
        .catch(logOutReload);

        // Realiza solicitud al backend para obtener el numero total de eventos
      fetch('http://localhost:8080/app/averageEventDuration', {
        headers: {
          Authorization: `Bearer ${accessToken}`,
        },
      })
        .then(response => response.json())
        .then(data => setaverageDuration(data))
        .catch(logOutReload);

  }, []);
```

Figura 22. Orquestación de Peticiones a la API

En conjunto, el "Analizador Automático" es una herramienta esencial dentro de NoMoreMeetings que no solo ofrece funcionalidades de análisis de datos avanzadas sino que también presta especial atención a la experiencia del usuario, asegurando que la herramienta sea tanto funcional como atractiva visualmente.

La sección "Buscador de Eventos" en NoMoreMeetings es una herramienta sofisticada y fácil de usar, diseñada para ayudar a los usuarios a encontrar eventos específicos en su calendario. Con una interfaz intuitiva, los usuarios pueden especificar un rango de fechas y una palabra clave, facilitando la búsqueda de eventos que coincidan con sus criterios.

```
  const googleUserString = localStorage.getItem("googleUser");
  const googleUser = JSON.parse(googleUserString || "");
  const accessToken = googleUser._tokenResponse.oauthAccessToken;

    // Construir la URL con parámetros de consulta
const queryParams = new URLSearchParams({
  startDate,
  endDate,
  startTime: startOfWork,
  endTime: endOfWork,
  selectedDays: selectedDays.join(','), // Unir los días seleccionados en una cadena
}).toString();

const url = `http://localhost:8080/app/calendar?${queryParams}`;

try {
  const response = await fetch(url, {
    method: "GET",
    headers: {
      'Authorization': `Bearer ${accessToken}`,
    },
  });

    if (!response.ok) {
      // Manejar errores de respuesta aquí
      throw new Error("Error en la solicitud");
    }

    const data = await response.json();
    setStats(data);
    setPopupVisible(true); // Mostrar el popup

  } catch (error) {
    console.error("Error al realizar la solicitud:", error);
  }

};
```

Figura 23.Gestión de Consultas del Analizador de Calendario, utilizando autenticación y manejo de errores

Al enviar el formulario, se realizan solicitudes al back-end para recuperar eventos relevantes. Los resultados se muestran en un modal interactivo, proporcionando una visión clara de los eventos encontrados, incluyendo detalles como el título y la fecha. Esta funcionalidad no solo es útil para la organización personal, sino que también mejora la eficiencia al permitir a los usuarios acceder rápidamente a la información que necesitan. En caso de errores en la entrada de datos, se muestra un modal con un mensaje de error, ofreciendo una retroalimentación útil y evitando posibles confusiones.

```
const handleSubmit = async (e: { preventDefault: () => void; }) => {
  e.preventDefault();

  //validar valores
  if (startDate == "" || endDate == "" || startOfWork == "" || endOfWork == "" ) {
    setErrorMessage("Introduce datos validos");
    setModalIsOpen(true);
    return;
  }

    // Validar que la fecha de inicio sea menor que la fecha de fin
  if (startDate > endDate || startOfWork > endOfWork || startDate == null || endDate == null) {
    setErrorMessage("La fecha y hora de inicio debe ser anterior a la fecha y hora de fin.");
    setModalIsOpen(true);
    return;
  }

    // Validar que la fecha de inicio sea menor que la fecha de fin
    if (startDate == endDate ) {
      setErrorMessage("Escoge un rango de fechas, no solo un dia");
      setModalIsOpen(true);
      return;
    }
```

Figura 24. Validaciones del Buscador de Eventos

```
const googleUserString = localStorage.getItem("googleUser");
const googleUser = JSON.parse(googleUserString || "");
const accessToken = googleUser._tokenResponse.oauthAccessToken;

// Construir la URL con parámetros de consulta
const queryParams = new URLSearchParams({
  startDate,
  endDate,
  keyword: searchKeyword,
}).toString();

const url = `http://localhost:8080/app/calendarString?${queryParams}`;


try {
  const response = await fetch(url, {
    method: "GET",
    headers: {
      'Authorization': `Bearer ${accessToken}`,
    },
  });

  if (!response.ok) {
    // Manejar errores de respuesta aquí
    throw new Error("Error en la solicitud");
  }

  const data = await response.json();
  setStats(data);
  setPopupVisible(true); // Mostrar el popup

} catch (error) {
  console.error("Error al realizar la solicitud:", error);
}

};
```

Figura 25. Interacción con API de Calendario para obtener eventos

La barra de navegación de NoMoreMeetings, implementada en React, es un elemento crucial para la navegación eficiente dentro de la aplicación. Su diseño responsivo se adapta perfectamente a diferentes tamaños de pantalla, mostrando un menú completo en dispositivos de escritorio y un menú modal en dispositivos móviles. Este enfoque garantiza una experiencia de usuario coherente y accesible en todas las plataformas.

```
const Navbar = ({ isTopOfPage, selectedPage, setSelectedPage, setUser, ...props }: Props) => {
  const flexBetween = "flex items-center justify-between";
  const [isMenuToggled, setIsMenuToggled] = useState<boolean>(false);
  const isAboveMediumScreens = useMediaQuery("(min-width: 1060px)");
  const navbarBackground = isTopOfPage ? "" : "bg-primary-100 drop-shadow";

  const handleSignOut = () => {
    localStorage.removeItem("googleUser");
    signOut(auth)
      .then(() => {
        console.log('Usuario desconectado');
        setUser(null); // Actualiza el estado del usuario después de cerrar sesión
      })
      .catch((error) => {
        console.error('Error al desconectar usuario:', error);
      });
  };
  return (
```

Figura 26. Barra de Navegación Responsiva y Cierre de Sesión

En la barra de navegación, se incluyen enlaces a las secciones principales de la aplicación, como 'Home', 'Benefits', 'Estadísticas', 'Analizador Automático' y 'Buscador de Eventos'. Cada enlace utiliza AnchorLink para un desplazamiento suave y actualiza el estado de la página seleccionada, lo que facilita la navegación y mejora la interacción del usuario con la aplicación. Esta funcionalidad no solo es conveniente, sino que también mejora la organización y la accesibilidad del contenido. Además, la barra de navegación incorpora un botón de 'SignOut', permitiendo a los usuarios cerrar sesión de manera segura y eficiente. Este botón es esencial para la gestión de usuarios, asegurando que los datos personales y las sesiones de los usuarios estén bien protegidos. En el aspecto técnico, se emplea useState para controlar el estado del menú en dispositivos móviles y useMediaQuery para determinar el tamaño de la pantalla y ajustar la visualización de la barra de navegación.

```tsx
import { SelectedPage } from "@/shared/types";
import AnchorLink from "react-anchor-link-smooth-scroll";

type Props = {
  page: string;
  selectedPage: SelectedPage;
  setSelectedPage: (value: SelectedPage) => void;
};

const Link = ({ page, selectedPage, setSelectedPage }: Props) => {
  const lowerCasePage = page.toLowerCase().replace(/ /g, "") as SelectedPage;

  return (
    <AnchorLink
      className={`${selectedPage === lowerCasePage ? "text-primary-500" : ""}
        transition duration-500 hover:text-primary-300
      `}
      href={`#${lowerCasePage}`}
      onClick={() => setSelectedPage(lowerCasePage)}
    >
      {page}
    </AnchorLink>
  );
};

export default Link;
```

Figura 27. Componente de Enlace React que facilita la navegación

7. Pruebas y resultados

7.1. Pruebas Unitarias

En nuestras pruebas unitarias, enfrentamos un desafío particular al intentar hacer mock de la clase Event de la API de Google Calendar. Esta dificultad se debe a varias razones:

- Clase Final: La clase Event en la API de Google Calendar es una clase final. En Java, una clase final no puede ser heredada, lo que también impide que frameworks de mocking, como Mockito, extiendan o modifiquen su comportamiento para fines de prueba.

- Complejidad de la API: La API de Google Calendar es una interfaz compleja y robusta que incluye numerosas dependencias y comportamientos que son difíciles de replicar o simular con precisión en un entorno de prueba.

- Interacción con Servicios Externos: La API de Google Calendar interactúa con servicios externos para manejar eventos del calendario, lo que añade una capa adicional de complejidad al proceso de mock, ya que estas interacciones a menudo implican autenticación y comunicación de red.

Dada la dificultad de hacer mock de la clase Event, optamos por un enfoque alternativo para nuestras pruebas unitarias:

- Uso de Datos Reales: En lugar de simular respuestas de la API de Google Calendar, realizamos nuestras pruebas unitarias con datos reales, utilizando un token de

acceso legítimo y haciendo llamadas reales a la API. Esto nos permite verificar el comportamiento de nuestra aplicación en condiciones reales de uso.

- Verificación de Lógica Propia: Nos centramos en probar la lógica propia de nuestra aplicación, como el procesamiento y análisis de los datos obtenidos de Google Calendar, más que en simular la API externa en sí.

Es importante destacar que, aunque estamos utilizando datos reales de la API de Google Calendar, estas pruebas siguen siendo unitarias en su naturaleza:

- Enfoque en Componentes Individuales: Nuestras pruebas se centran en verificar el funcionamiento correcto de métodos individuales en el controlador, en lugar de probar la interacción entre varios componentes o sistemas, que sería el enfoque de las pruebas de integración.

- Independencia de las Pruebas: Cada prueba se diseña para ser independiente, evaluando funciones específicas sin dependencia de otros componentes del sistema.

Prueba	Escenario	Valores Necesarios	Comprobaciones
getEventsTest	Verificar la obtención exitosa de eventos	Token de acceso real	No nulo, estado HTTP OK, verificaciones adicionales según respuesta real.
getEventsProximosTest	Obtener eventos próximos exitosamente	Token de acceso real	No nulo, estado HTTP OK, cuerpo no vacío, verificación de contenido y estructura de eventos.
getStatsTest	Obtener estadísticas de ocupación y tiempo libre	Token de acceso real	No nulo, estado HTTP OK, cuerpo no vacío, porcentajes de tiempo ocupado y libre según lo esperado.
getEventsThisMonthTest	Obtener el número total de eventos de un mes	Token de acceso real	No nulo, estado HTTP OK, cuerpo no vacío, cantidad de eventos según lo esperado.
getAverageEventDurationTest	Obtener duración promedio de eventos	Token de acceso real	No nulo, estado HTTP OK, cuerpo no vacío, cantidad de eventos según lo esperado.
getStatsCalendarTest	Sin solapamiento	Fechas, horas, días seleccionados	No nulo, estado HTTP OK, porcentajes de

			ocupación y tiempo libre según lo esperado.
	Solapamiento días laborables	Fechas, horas, días seleccionados	No nulo, estado HTTP OK, porcentajes de ocupación y tiempo libre según lo esperado.
	Solapamiento con días no laborables	Fechas, horas, días seleccionados	No nulo, estado HTTP OK, porcentajes de ocupación y tiempo libre según lo esperado.
	Solapamiento con el horario laboral al inicio de la jornada	Fechas, horas, días seleccionados	No nulo, estado HTTP OK, porcentajes de ocupación y tiempo libre según lo esperado.
	Solapamiento con el horario laboral al final de la jornada	Fechas, horas, días seleccionados	No nulo, estado HTTP OK, porcentajes de ocupación y tiempo libre según lo esperado.
	Sin solapamiento, todos los días no laborables	Fechas, horas, días seleccionados	No nulo, estado HTTP OK, porcentajes de ocupación y tiempo libre según lo esperado.
	Sin solapamiento, ocupado siempre	Fechas, horas, días seleccionados	No nulo, estado HTTP OK, porcentajes de ocupación y tiempo libre según lo esperado.
getMatchingEventsTest	Búsqueda básica por palabra clave	Fecha de inicio, fecha de fin, palabra clave.	No nulo, estado HTTP OK, lista de eventos según lo esperado.
	Búsqueda con rango de fechas específico	Fecha de inicio, fecha de fin, palabra clave vacía	No nulo, estado HTTP OK, lista de eventos según lo esperado.

	Palabra que no se encuentra	Fecha de inicio, fecha de fin, palabra clave inexistente	No nulo, estado HTTP OK, lista de eventos vacía.
	Búsqueda con fechas invertidas	Fecha de fin antes que fecha de inicio, palabra clave	No nulo, estado HTTP OK, lista de eventos vacía o error.

Tabla 11. Pruebas unitarias

7.2 Pruebas de Integración

Las pruebas de integración son aquellas que una vez comprobados los módulos de los controladores de manera independiente nos van a permitir comprobar el correcto funcionamiento de dichos módulos, pero, esta vez, conectados con la capa de persistencia.
Estas pruebas se han realizado de manera manual, utilizando la herramienta Postman.
Postman nos permite realizar llamadas HTTP a nuestra API REST sin necesidad de una interfaz web. Las pruebas de integración se realizaron de abajo a arriba (bottom-up) integrando primero la capa de negocio con la capa de persistencia y, posteriormente, el front-end sobre el back-end. Para cada servicio implementado en los controladores se han realizado las correspondientes pruebas con el fin de verificar la viabilidad en el funcionamiento de ambas capas juntas. Para ello se realizaba una petición válida a cada servicio y se comprobaba que la respuesta era la esperada, teniendo en cuenta tanto los estatus correctos como los incorrectos.

7.3 Pruebas de aceptación

Las pruebas de aceptación corresponden a la última etapa de pruebas relacionadas con la aplicación. En ellas se comprueba que la aplicación contiene y realiza de manera correcta las funcionalidades que se han propuesto para el desarrollo del proyecto.

En este caso, para realizar dichas pruebas se ha comprobado el correcto funcionamiento de todas las funcionalidades implementadas en el front-end de manera manual en el servidor local. Las siguientes imágenes muestran las pruebas realizadas para la gestión del calendario.

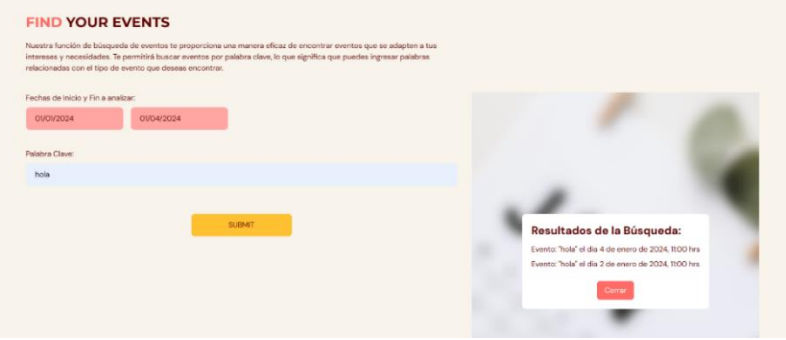


Figura 28. Buscador de Eventos

Figura 29. Analizador Automático

Figura 30. Estadísticas Generales

8. Conclusiones y trabajos futuros

Esta sección esboza algunas conclusiones, conocimientos adquiridos y apunta algunas funcionalidades que podrían enriquecer la aplicación en sus siguientes versiones.

8.1 Conclusiones

Al concluir este proyecto, puedo afirmar con confianza que se han alcanzado los objetivos básicos planteados. La aplicación full stack se ha desarrollado en un período aproximado de seis meses, logrando implementar con éxito todas las funcionalidades.

- Logros y Aprendizajes
    - Inmersión en Procesos de Ingeniería de Software:
      Este proyecto ha sido una excelente oportunidad para aplicar y profundizar en los conocimientos adquiridos durante el grado en ingeniería informática, especialmente en procesos de ingeniería de software. Desde el análisis de requisitos hasta la implementación, cada etapa del desarrollo ha reforzado mi comprensión de las metodologías utilizadas en el desarrollo profesional de software.

    - Mejora en la Estimación de Tiempos:

La planificación y gestión del tiempo han sido aspectos cruciales en este proyecto. A pesar de que algunas funcionalidades requirieron más tiempo del esperado, otras se completaron antes, lo que muestra una mejora significativa en mis habilidades para estimar tiempos de desarrollo.

- Exploración de Nuevas Tecnologías y Frameworks:
  El desarrollo del back-end en este proyecto ha sido una valiosa oportunidad para profundizar en mis conocimientos de Java, especialmente en el uso de Spring Boot para la creación de la API REST. Aunque ya había adquirido una base sólida en estas tecnologías durante mis estudios, la aplicación práctica y los desafíos específicos del proyecto me permitieron explorar aspectos más avanzados y sutilezas que no había experimentado en el aula.
  En el front-end, este proyecto ha sido una excelente oportunidad para sumergirme en el mundo de JavaScript, HTML y CSS, aprovechando las potencialidades de React. Esta experiencia ha ampliado significativamente mi repertorio de habilidades en desarrollo web.

- Desafíos Técnicos y Superación
  - Integración con la API de Google Calendar:
    Uno de los desafíos más interesantes fue la integración con la API de Google Calendar. Este aspecto del proyecto me permitió aplicar conceptos avanzados de interacción con APIs externas y manejo de datos en tiempo real, reforzando mis habilidades en integración de sistemas.

  - Adopción de Prácticas de Ingeniería Ágil:
    La adopción de metodologías ágiles y el enfoque iterativo en el desarrollo permitieron una adaptación fluida a los cambios y una mejora continua del proyecto. La capacidad de responder ágilmente a los desafíos imprevistos fue crucial para el éxito del desarrollo.

- Reflexión Personal y Crecimiento Profesional
  Este proyecto ha sido más que una tarea académica; ha sido un viaje de crecimiento profesional y personal. Me ha brindado la oportunidad de aplicar teorías y conceptos en un entorno práctico, enfrentarme a desafíos reales del desarrollo de software y, lo más importante, ha fortalecido mi confianza como ingeniero informático.

El código realizado se ofrece de forma abierta para su utilización por la comunidad en el repositorio de GitHub: https://github.com/JesRevu/NoMoreMeetings

8.2 Trabajos futuros

Tras la implementación inicial de esta aplicación, hemos identificado oportunidades de expansión y mejoras que podrían realzar su funcionalidad y valor para los usuarios:

- Integración de una Base de Datos:
  Implementar una base de datos para almacenar información detallada sobre los eventos analizados, preferencias de los usuarios y resultados de análisis. Esto facilitaría un análisis más profundo y la personalización de la experiencia del usuario.

- Funcionalidad para Añadir Eventos al Calendario:
  Permitir que los usuarios añadan o modifiquen eventos directamente desde la aplicación. Esto no solo facilitaría la gestión de calendarios, sino que también permitiría un análisis más dinámico basado en cambios en tiempo real.

- Creación de Perfiles de Usuario con Funciones Diferenciadas:
  Desarrollar diferentes roles de usuario, como administradores que pueden gestionar configuraciones de la aplicación y usuarios regulares con acceso a análisis personalizados de sus calendarios.
  Esto podría incluir la creación de perfiles para analistas de datos, que tendrían acceso a herramientas avanzadas para el análisis de patrones de eventos.

- Alertas de Conflicto de Horario:
  Implementa una alerta simple que notifique a los usuarios cuando programen una reunión que se solape con otro evento existente en su calendario.

- Contador de Reuniones Diarias:
  Crea una función que cuente el número de reuniones en un día y lo muestre en la aplicación. Esto puede ayudar a los usuarios a tener una visión rápida de su día.

- Filtro de Reuniones por Tipo:
  Añade una funcionalidad que permita a los usuarios filtrar sus reuniones en el calendario por tipo (por ejemplo, reuniones internas, llamadas con clientes, revisiones de proyectos, etc.).

- Integración con Herramientas de Gestión de Proyectos:
  Permitir la sincronización de eventos con herramientas de gestión de proyectos como Trello, para que los usuarios puedan ver sus tareas y plazos junto con sus reuniones en un único lugar.

9.Bibliografía

[1] Entorno de desarrollo integrado. Eclipse IDE para Java. http://www.edu4java.com/es/java/eclipse-ide-entorno-de-desarrollo.html

[2] Spring Framework. https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework

[3] ¿Qué es Spring Boot? https://blog.codmind.com/que-es-spring-boot/#:~:text=Spring%20Boot%20permite%20compilar%20nuestras,de%20aplicaciones%20en%20el%20propio%20.

[4] ¿Qué es Maven y para qué se utiliza?. http://panamahitek.com/que-es-maven-y-para-que-se-utiliza/

[5] Qué es Visual Studio Code. https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

[6] React. https://es.reactjs.org/

[7] Acerca de Node.js. https://nodejs.org/es/about/

[8] Bootstrap: Bootstrap · The most popular HTML, CSS, and JS library in the world. (getbootstrap.com)

[9] Cómo realizar pruebas automatizadas con Postman. https://www.encora.com/es/blog/como-realizar-pruebas-automatizadas-con-postman

[10] GitHub. https://es.wikipedia.org/wiki/GitHub

[11] SCSS, Sass: Syntax (sass-lang.com)

[12] JavaScript. https://es.wikipedia.org/wiki/JavaScript

[13] Template React. Croma - Acquisition 30 sec version A (youtube.com)