UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de una agenda online en Erlang

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Marcel·li Ruiz Comes

Tutor: Germán Francisco Vidal Oriola

2020 -2021

Resumen

En un mundo en continuo cambio y rápida evolución, es imprescindible tener una buena organización con el fin de ayudarnos en el día a día, por ello el presente proyecto se basa en la creación de una aplicación web que nos sirva para gestionar nuestra agenda. Proporcionará las posibilidades de mantener varios calendarios y asociar eventos a ellos de manera personalizable y al gusto de cada persona, así como compartirlos con otros usuarios, además será fácilmente accesible e intuitivo.

Para ello, se emplearán las metodologías ágiles para desarrollar una aplicación en Elixir, un lenguaje fundado sobre los cimientos de Erlang, con el *framework* Phoenix y PostgreSQL como sistema de gestión para la base de datos. Gracias a estas tecnologías conseguiremos un software concurrente que puede soportar gran cantidad de procesos simultáneos sin llegar a sobrecargarse. Además, se empleará una novedosa librería que nos facilitará la creación de interfaces de usuario y nos proporcionará actualizaciones en tiempo real con el fin de mejorar la experiencia del usuario al hacer uso de la aplicación.

Palabras clave: Calendario, agenda, evento, Elixir, Phoenix, PostgreSQL.

Abstract

In a world of continuous change and fast evolution, it is a need to have a good organization in order to help us day by day. Therefore this project is based on the creation of a web application designed to manage our agenda. It will provide the possibilities of maintaining several calendars and associating events to them in a customizable way to the choice of each person, as well as sharing them with other users, it will also be easily accessible and intuitive.

To that end, agile methodologies will be used to develop an application in Elixir, a language based on the foundations of Erlang, with the Phoenix framework and PostgreSQL as a management system for the database. Thanks to these technologies, we will get a concurrent software that can support a large number of simultaneous processes without becoming overloaded. Furthermore, an innovative library will be used to facilitate the creation of user interfaces and provide us with updates in real time with the aim to improve the user experience when using the application

Keywords: Calendar, agenda, event, Elixir, Phoenix, PostgreSQL.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1. Introducción

El mundo está tomando una dirección hacia digitalizar todo lo posible, y la mayoría de acciones que antaño se realizaban de manera manual con elementos físicos, ahora son tareas más sencillas gracias a la tecnología. Este es el caso de las agendas, llevar una buena planificación es esencial en la actualidad, en la que los desarrollos son muy rápidos y cada vez se hace más uso de dispositivos electrónicos.

Estas nuevas herramientas online nos permiten coordinar nuestras tareas de manera eficiente, así como planificar nuestras rutinas para no olvidar tanto citas, reuniones, como alguna fecha límite de entrega, cumpleaños o un encuentro con amigos. Por ello, su uso en los últimos años ha estado en continuo aumento.

Actualmente existen variedad de aplicaciones de esta índole que nos sirven para mantener nuestra agenda de manera electrónica. Muchas se centran en el entorno móvil, ya que es una herramienta que tenemos a mano en todo momento y por lo tanto es más práctica y accesible para recordar nuestros eventos. Sin embargo, este proyecto se va a centrar en desarrollar una web para escritorio, ya que el uso de estas páginas se encuentra en continuo aumento y son globalmente accesibles en todo momento.

1.1. Motivación

Mi principal motivación a la hora de seleccionar el presente TFG fue la idea de desarrollar una aplicación web de principio a fin, ya que siempre me ha resultado un tema de interés personal, pues considero que es un sector en continua evolución y con una gran demanda.

Por otro lado, me resultó interesante la idea de realizar el desarrollo empleando un lenguaje funcional y concurrente como es Elixir, que está formado en base a Erlang. Considero que aprender a utilizar un lenguaje de programación de un paradigma diferente al habitual puede ayudarme a mejorar como ingeniero y así afrontar los futuros problemas software desde diferentes perspectivas, sin encasillarme en un estilo concreto.

El lenguaje Elixir es un lenguaje relativamente reciente, ya que se creó en 2012, y puede acabar siendo muy relevante para el desarrollo de sistemas web por la escalabilidad y mantenimiento que ofrece este lenguaje funcional.

1.2. Objetivos

Seguidamente se va a proceder a describir los objetivos principales y secundarios, desglosados de los principales, definirlos brevemente y ordenarlos en función de la prioridad en el proyecto.

El objetivo principal del TFG es el desarrollo de una aplicación web cuya funcionalidad principal sea un calendario. De este objetivo principal podemos desglosar los siguientes objetivos secundarios:

- Permitir al usuario añadir elementos en el calendario.
- Ofrecer varias visualizaciones de la agenda, lista, semana, meses y año.
- Permitir al usuario gestionar diferentes calendarios.
- Permitir al usuario compartir sus calendarios con otros usuarios.
- Crear una interfaz actual, intuitiva y atractiva.
- Ser compatible con los principales navegadores modernos.

1.3. Metodología

Para el desarrollo del trabajo se ha optado por intentar aplicar las metodologías ágiles [1], para mejorar la organización del trabajo y en consecuencia mejorar la calidad del producto gracias a una mayor eficiencia del trabajo.

Estas nuevas metodologías se han implantado en el sector del desarrollo software debido a sus múltiples beneficios como la mejora de la calidad del producto, una mayor satisfacción del cliente, aumento de la motivación de las personas que trabajan en el proyecto junto con una mejora de la colaboración entre los integrantes, esto lleva a conseguir unas métricas más realistas si se tiene experiencia con estas metodologías y un mayor control sobre el producto para poder evitar alteraciones grandes en las fechas de entrega o en los costes. Por todo esto también lleva a una reducción de costes.

En este proyecto en concreto no repercutirá a un alto nivel, pero sí que puede ser muy útil para organizar de manera correcta el trabajo y llevar un control del mismo. Es por eso que en este caso se ha optado por una metodología Kanban para tener plasmado todos los desarrollos necesarios para el sistema, para esta tarea se ha utilizado la herramienta Trello.

Trello es un software de administración de proyectos con interfaz web y con cliente para *iOS* y *android* para organizar proyectos, es muy útil y sencillo de utilizar tanto para trabajar en grupo como en proyectos individuales, ya que nos permite visualizar en todo momento el estado del proyecto.

Pero antes de empezar a desarrollar se deberá tener claro lo que vamos a desarrollar, en un proyecto tradicional se reuniría con el cliente y se vería todo lo que necesita, en nuestro caso fue con el tutor.

Después de hablar y debatir sobre cómo abordar el inicio del proyecto, que ideas serían interesantes incluir y la manera más sencilla de tratar con el lenguaje, una vez tenemos esas pautas claras se empezaría a buscar información sobre el tema de estudio, en este caso los calendarios y sus versiones online para poder apreciar los competidores que ya existen en el entorno.

A partir de esto también se pueden ir creando los casos de uso y especificando requisitos, aquí es importante analizar el tiempo que se va a invertir para no tener demasiadas tareas, pero tampoco que se quede corto el proyecto.

En cuanto disponemos de los requisitos para el proyecto, se utilizarán tres columnas para ir situando las diferentes tarjetas que representarán las tareas a realizar. En una primera instancia se transformarán todos los requisitos en tarjetas y se plasmarán en la columna de To Do, esto nos ayudará a saber todo lo que necesitamos, una vez en este punto se deberían ordenar por orden de prioridad, y por comodidad también se puede cambiar el color de la tarjeta para representar diferentes partes del sistema, como en verde las que estén relacionadas con los usuarios, azules con los calendarios, rojas con los eventos y así sucesivamente.

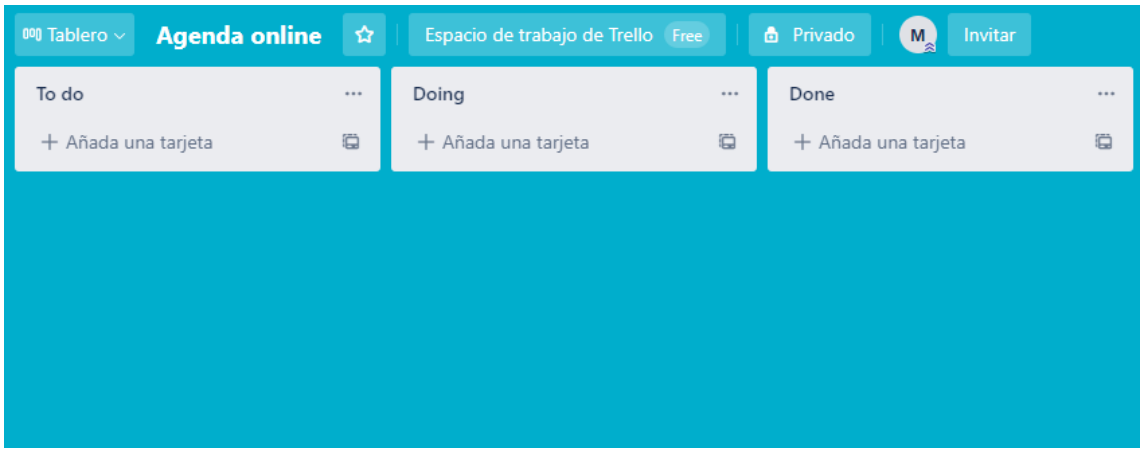

Ilustración 1. Organización Kanban

Lo más recomendable es no tener demasiadas tareas en Doing, ya que disipará la atención y perdería el sentido de organizarlo de esta manera ya que no sabríamos el estado real en el que se encuentra la aplicación, por ello como mucho se tendrán 3 tareas al mismo tiempo ya que puede que haya algunas que dependan de otras o por comodidad personal convenga cambiar lo que estemos trabajando para no atascarnos en un punto.

Finalmente, cuando tenemos los requisitos ordenados por prioridad en la columna de *To Do*, podemos empezar el desarrollo y con él se irán desplazando las diferentes tareas a las columnas.

Pero puede ocurrir que alguna tarea se observa que es demasiado compleja, estas se las conoce como épicas, como es el caso de “Cambiar vista calendario” que se trata de un único requisito funcional, pero debido a la magnitud de esta tarea lo mejor es dividirla en tareas más pequeñas, es por ello pasaremos de una tarjeta que acumula todo ese trabajo a abordarla de manera separada.

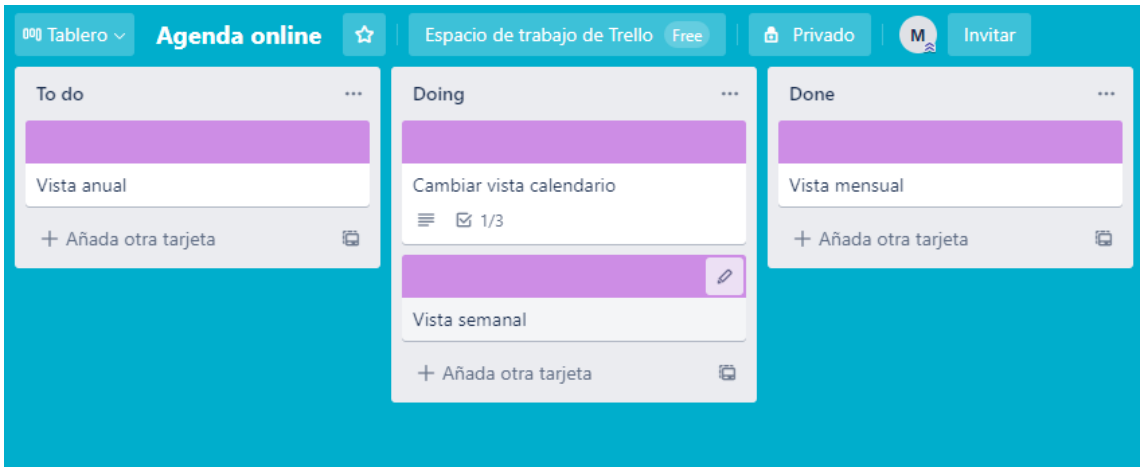

Ilustración 2. Épica dividida

Una vez separada es mucho más sencillo trabajar en sus diferentes componentes y podremos contemplar de manera clara cómo evoluciona esta tarea épica de manera clara sin que una misma tarea nos tenga ocupado demasiado tiempo.

1.4. Estructura de la memoria

Con el fin de organizar de manera óptima la información de este documento, se ha establecido una estructura basada en los siguientes capítulos:

- **Capítulo 1. Introducción.** Introducción al tema principal y motivos que han propiciado su elección. Definición de los objetivos que se pretenden lograr y planteamiento de la metodología proyectual y de la estructura de la tesis.

- **Capítulo 2. Estado del arte.** Análisis del contexto tecnológico actual y de las diversas agendas online y aplicaciones similares ya existentes en el mercado. Tras la investigación, se realiza una propuesta de una agenda online que supere de alguna manera las opciones actuales.

- **Capítulo 3. Análisis del problema.** Se especificarán tanto los requisitos funcionales como los no funcionales del sistema, y los actores que participarán en él representando las distintas funcionalidades de la aplicación.

- **Capítulo 4. Diseño de la solución.** Presentación de la arquitectura que se utilizará en el proyecto, así como la visión detallada de la idea del diseño de la base de datos, los prototipos diseñados a partir de los requisitos y las herramientas de las que se ha hecho uso a lo largo del proyecto.

- **Capítulo 5. Desarrollo de la solución propuesta.** Explicación de la evolución de la idea hasta la obtención del resultado final, exponiendo las dificultades encontradas y las decisiones tomadas para la resolución de estas.

- **Capítulo 6. Pruebas.** Demostración del correcto desarrollo y funcionalidad de la aplicación con el fin de comprobar si satisface las necesidades de los usuarios potenciales.

- **Capítulo 7. Conclusiones.** Valoración general en la que se argumenta el cumplimiento o no de los objetivos planteados al inicio del proyecto, así como el aprendizaje obtenido durante su desarrollo.

- **Capítulo 8. Trabajos futuros.** Lista de posibles ampliaciones o mejoras de eficiencia y funcionalidad, así como posibles flecos que hayan podido quedar al final del desarrollo. También se comentan nuevas ventanas de expansión posibles en base al proyecto realizado.

- **Capítulo 9. Referencias.** Registro de las distintas fuentes consultadas a lo largo de todo el proceso de trabajo.

2. Estado del arte

Como ya hemos comentado el mundo de los calendarios es muy antiguo y se ha adaptado de manera rápida al mundo digital. Lo más común es ya usar aplicaciones que nos lleven de manera sencilla esta tarea diaria para así facilitarnos la vida.

En este entorno existen numerosas aplicaciones que satisfacen esta necesidad, la más famosa del momento es Google Calendar, como es comprensible dado la magnitud de Google y su gran compatibilidad con todos sus sistemas diferentes. Pero no es la única, es por ello que vamos a analizar algunas de las más importantes y conocidas que tienen objetivos similares a los de nuestro proyecto.

Estas son algunas de las diferentes agendas online que tienen una buena cantidad de usuarios activos, y con características relevantes en el ámbito:

2.1. Zoho calendar

Se trata de un calendario online gratuito orientado al sector empresarial, aunque puede usarse para gestiones y eventos particulares. Es una herramienta completa con muchas opciones para personalizar las actividades y eventos del calendario. Tiene la posibilidad de sincronizarse con otras herramientas de la misma marca lo que ayuda a aumentar las posibilidades del sistema.

Añadir eventos se hace de una manera muy sencilla y rápida y se pueden editar también de manera intuitiva. También se pueden manejar diferentes calendarios y mostrar solo los que desees y editarlos para compartirlos con quien prefieras. Y permite la sincronización de tus calendarios con otros dispositivos.

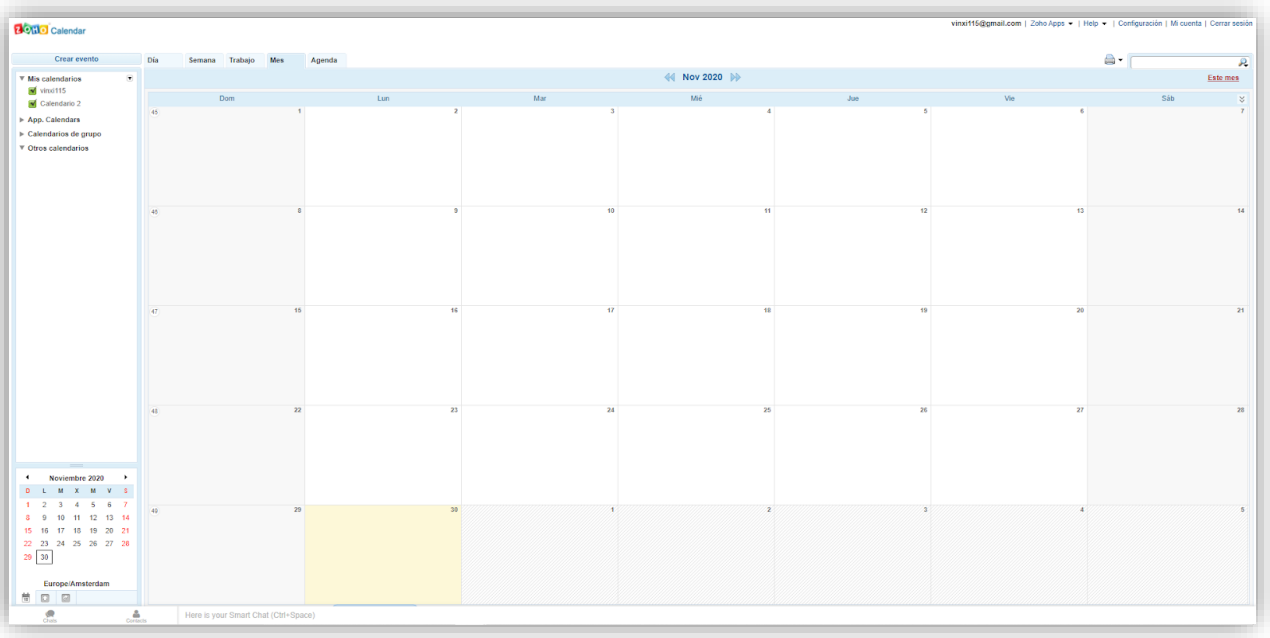

Ilustración 3. Zoho calendar

Las partes negativas de la herramienta son en primer lugar, su aspecto visual que se siente anticuado y poco atractivo, sobre todo en la gestión de calendarios y eventos. Para gestionar la mayoría de las características se accede a un submenú que no es tan intuitivo ya que hay que dar bastantes pasos hasta llegar a nuestro objetivo.

2.2. Shore

No es solamente una agenda online, es más bien una herramienta para facilitar la gestión empresarial con clientes coordinando citas y tareas, pero en ella se puede encontrar una funcionalidad de agenda online para facilitar la organización del resto de actividades. Existen multitud de aplicaciones similares a esta orientadas a todo tipo de equipos desde pequeños grupos a grandes empresas y aunque no se trate de una aplicación del mismo ámbito que este trabajo, se puede considerar que han ayudado al desarrollo de las agendas online dada su importancia en sus herramientas.

La agenda online integrada es bastante potente y permite una sincronización eficiente con el resto de las funcionalidades del programa, además se pueden añadir citas de manera online y registrarlas en la agenda directamente además de poder hacer lo mismo desde Google directamente.

La parte negativa de estas herramientas es que la mayoría sólo permiten utilizarse si formas parte de alguna empresa, ya que están orientadas a este contexto. Otro punto negativo es que la mayoría son de pago mensual, aunque hay planes gratuitos con muy pocas opciones.

2.3. Google Calendar

Encontramos la que probablemente sea el calendario online más famoso actualmente, desarrollado por Google tiene amplia compatibilidad con el resto de servicios de la compañía.

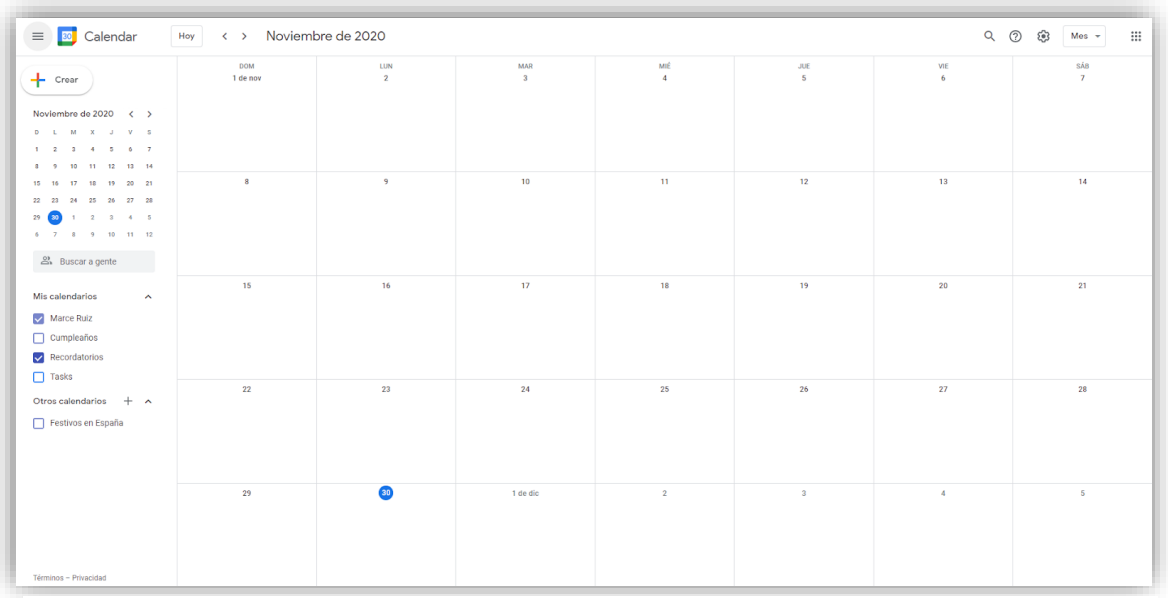

Ilustración 4. Google Calendar

Tiene grandes facilidades para usarse tanto en entorno web como en móvil, así como invitar a otros usuarios a eventos y compartir calendarios a través de Gmail, lo que facilita en gran medida la manera de compartir los calendarios. Su interfaz es realmente atractiva e intuitiva y se puede usar en todos los navegadores. Además, permite integrarse en otras páginas web por lo que es mundialmente conocida y usada. Su creación de eventos es sencilla y rápida y además permito bastante personalización a estas tareas.

Como contrapartida no se pueden destacar muchos defectos de la herramienta, solo el que puede acarrear Google, sus problemas de seguridad y privacidad de datos [2], ya que se puede ver fácilmente cualquier calendario marcado como público y de ahí obtener información de la empresa propietaria del calendario o datos personales, es por ello que algunas compañías y personas pueden llegar a desconfiar de esta opción y optar por mantener sus calendarios privados. Además, que no existe ningún indicador cuando estás creando eventos de que te encuentras en un calendario público o privado.

2.4. Propuesta

Se puede observar que existe una gran variedad de aplicaciones con el mismo propósito que nuestro proyecto, pero muchas de ellas están integradas en sistemas más grandes que requieren de suscripción para poder usarlas, o son aplicaciones muy orientadas a empresas que buscan coordinar un gran número de empleados y clientes y de esta manera planificar sus citas. Mientras que, las que están más orientadas a usuarios comunes, o bien se ha comprobado que pueden tener fallos en su seguridad o son antiguas y están quedando obsoletas.

Debido a estos motivos, vamos a intentar satisfacer la necesidad de ese grupo de usuarios de encontrar una alternativa a Google para poder llevar una agenda de manera rápida, clara y eficiente sin tener que recurrir a las grandes compañías sacrificando su privacidad. De igual forma se intentará crear un estilo más moderno y sencillo con el fin de que sea agradable de usar y rápido de aprender para adaptarse al estilo actual.

3. Análisis del problema

En esta sección se pretenden especificar todos los requisitos funcionales y no funcionales de nuestro sistema, así como indicar qué actor podrá realizar cada uno y la importancia que tendrán en el desarrollo

3.1. Actores

A continuación, procedemos a describir los actores existentes en la aplicación:

Usuario no registrado: Se trata de cualquier persona que no ha iniciado sesión y se encuentra con la página de inicio de la web.

Usuario registrado: Persona que tiene una cuenta en el sistema y ha iniciado sesión y puede gestionar sus calendarios y los que hayan sido compartidos con él por otros usuarios.

3.2. Casos de uso

Los siguientes casos de uso representan las funcionalidades que tendrá la aplicación, se pueden observar uno por cada actor que interviene en el sistema.

El usuario no registrado podrá acceder a la aplicación iniciando sesión si ya tiene una cuenta, o crear una nueva cuenta.

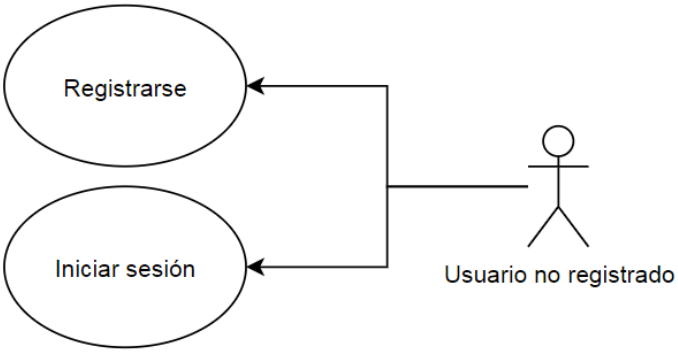

Ilustración 5. Casos de uso usuario no registrado

El usuario registrado podrá realizar las principales acciones del calendario, como gestionar sus calendarios, creando nuevos, editándolos, borrándolos o compartiendo éstos con otros usuarios. También pueden gestionar eventos al igual que los calendarios. De la misma manera pueden editar su perfil de usuario y borrarlo si así lo desean. También pueden cambiar la visualización que quieren para el calendario, en formato mensual, semanal o agenda.

En cuanto a los calendarios compartidos, los usuarios podrán cargarlos y podrán editarlos y crear eventos en ellos dependiendo de sus permisos, ya que pueden ser de lectura, de escritura o de administrador.

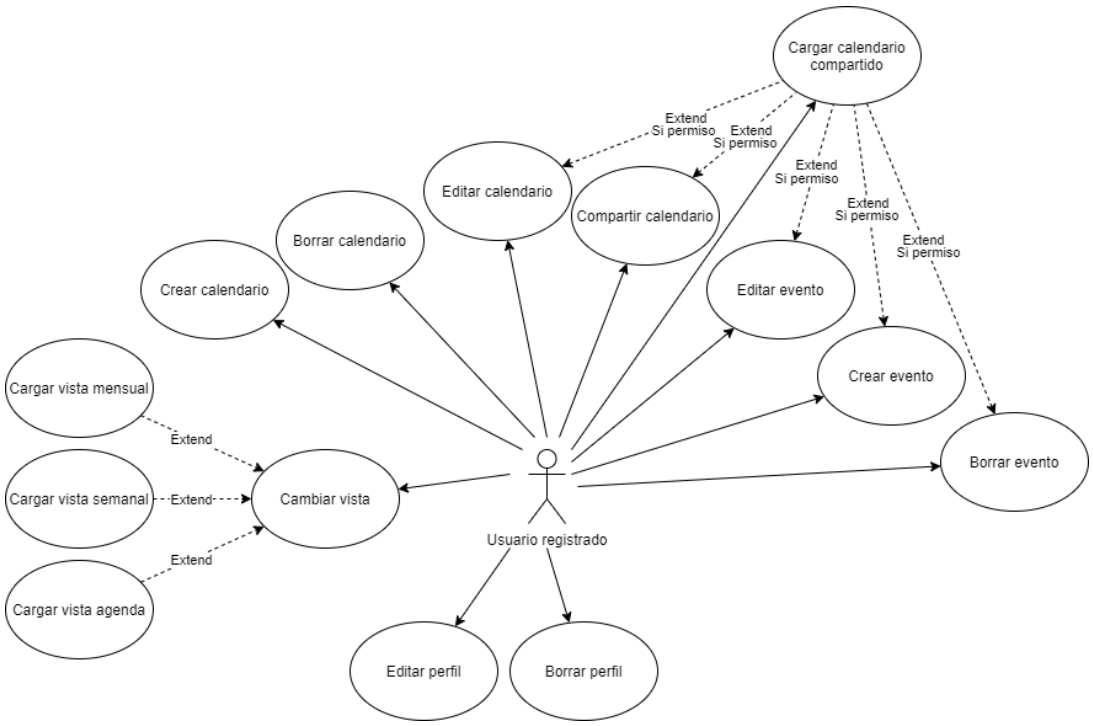

Ilustración 6. Casos de uso usuario registrado

3.3. Requisitos funcionales

En este apartado se van a describir todos los requisitos funcionales que contendrá el sistema. Con el fin de hacerlo más entendible, se dividirán las funcionalidades por actores.

3.3.1. Usuario registrado

Los usuarios que se hayan registrado e iniciado sesión podrán realizar las siguientes operaciones:

ID del requisito	RF1
Nombre	Crear calendario.
Descripción	El usuario, siempre que esté registrado, podrá crear un calendario indicando el título.
Prioridad	Alta.
Entrada	Título: String, debe de tener al menos un carácter.
Salida	El resultado de la acción será la creación del nuevo calendario asociado al usuario que lo ha creado.

Tabla 1. Requisito funcional Crear calendario

ID del requisito	RF2
Nombre	Borrar calendario.
Descripción	El usuario podrá borrar un calendario del que sea propietario borrando así todos los eventos existentes en él.
Prioridad	Alta.
Entrada	-
Salida	Si el usuario es el propietario del calendario o es administrador, el calendario se eliminará completamente.

Tabla 2. Requisito funcional borrar calendario

ID del requisito	RF3
Nombre	Modificar calendario.
Descripción	El usuario podrá modificar los parámetros de un calendario existente del que sea propietario.
Prioridad	Alta.
Entrada	Título.
Salida	Si los datos son correctos, se almacenarán los nuevos datos.

Tabla 3. Requisito funcional modificar calendario

ID del requisito	RF4
Nombre	Compartir calendario.
Descripción	El usuario podrá compartir un calendario con otra persona indicando si la otra u otras personas tendrán permisos de escritura, de lectura o de administrador. • Lectura: podrá visualizar los eventos del calendario así como su descripción y sus datos. • Escritura: tendrá la capacidad de añadir eventos al calendario y modificar los ya existentes. • Administrador: el usuario con este permiso podrá realizar las mismas acciones que el creador, modificar el título del calendario, compartirlo o borrarlo, además de ver, crear, editar y borrar los eventos.
Prioridad	Media.
Entrada	Usuario: String, no puede ser nulo. Permiso: String, no puede ser nulo y será Lectura, Escritura o Admin.
Salida	Se mostrará un mensaje del resultado, bien si se ha compartido correctamente o que ha surgido algún error y no se ha podido compartir bien.

Tabla 4. Requisito funcional compartir calendario

ID del requisito	RF5
Nombre	Modificar permisos.
Descripción	El usuario podrá modificar los permisos del resto de personas de un calendario del que sea propietario.
Prioridad	Media.
Entrada	El calendario y el usuario.
Salida	Se mostrará un mensaje del resultado y se guardarán los nuevos permisos.

Tabla 5. Requisito funcional modificar permisos

ID del requisito	RF6
Nombre	Crear evento.
Descripción	El usuario podrá crear un evento en el calendario que desee, indicando el título, descripción, un color asociado a él, la fecha, la hora de inicio, así como la final y el calendario en el que se creará.
Prioridad	Alta.
Entrada	Título: String, no puede ser nulo. Descripción: Texto. Fecha y hora de inicio: Datetime, no puede ser nulo. Fecha y hora de fin: Datetime, no puede ser nulo. Color: String, no puede ser nulo. Calendario: Integer del ID del calendario.
Salida	Con la información proporcionada se creará un evento asociado al calendario donde se ha creado.

Tabla 6. Requisito funcional crear evento

ID del requisito	RF7
Nombre	Borrar evento.
Descripción	El usuario podrá borrar un evento que esté en uno de sus calendarios o tenga permisos de borrado si es compartido.
Prioridad	Alta.
Entrada	-
Salida	Mensaje mostrando que se ha borrado el evento correctamente o se ha producido algún error.

Tabla 7. Requisito funcional borrar evento

ID del requisito	RF8
Nombre	Modificar evento.
Descripción	El usuario podrá modificar los datos de un evento como la fecha de inicio, de fin, el título, la descripción y el color. También podrá modificar el calendario al que pertenece.
Prioridad	Alta.
Entrada	Título, descripción, Fecha y hora de inicio y de fin, color y calendario.
Salida	Si los datos son correctos se guardarán los datos en el evento, si existe algún error no se actualizará y se mostrará la fuente del problema.

Tabla 8. Requisito funcional modificar evento

ID del requisito	RF9
Nombre	Modificar perfil.
Descripción	El usuario puede modificar los datos de su perfil, como el nombre, apellidos, correo, usuario y contraseña.
Prioridad	Alta.
Entrada	Nombre, apellidos, correo, usuario y contraseña.
Salida	Se actualizará la información del perfil si es correcta, mientras que si existe algún error se le mostrará al usuario.

Tabla 9. Requisito funcional modificar perfil

ID del requisito	RF10
Nombre	Borrar perfil.
Descripción	El usuario podrá borrar su perfil, de esta manera se borrarán todos los calendarios de los que sea propietario y los eventos que contengan.
Prioridad	Alta.
Entrada	-
Salida	Se mostrará un mensaje con el resultado, si se ha borrado el perfil se le redirigirá a la página de inicio de la aplicación.

Tabla 10. Requisito funcional borrar perfil

ID del requisito	RF11
Nombre	Ver calendario.
Descripción	El usuario podrá escoger si quiere ver un calendario y de esta manera se le mostrará en pantalla, si elige varios se pueden superponer y se mostrarán los eventos de todos los que haya seleccionado.
Prioridad	Alta.
Entrada	-
Salida	Se cargará el calendario seleccionado y se mostrará en el tipo de visualización que exista.

Tabla 11. Requisito funcional ver calendario

ID del requisito	RF12
Nombre	Cambiar vista calendario.
Descripción	El usuario podrá cambiar el tipo de visualización del calendario y escoger entre la vista mensual, semanal y anual.
Prioridad	Alta.
Entrada	-
Salida	El calendario central se mostrará con la vista que haya elegido el usuario.

Tabla 12. Requisito funcional cambiar vista calendario

ID del requisito	RF13
Nombre	Mostrar lista eventos.
Descripción	El usuario podrá escoger la opción de visualizar los eventos de todos los calendarios que tenga seleccionados en forma de lista.
Prioridad	Media.
Entrada	-
Salida	Se mostrará una lista con los eventos existentes en los calendarios seleccionados.

Tabla 13. Requisito funcional mostrar lista eventos

3.3.2. Usuario no registrado

Los usuarios que no hayan iniciado sesión podrán realizar las siguientes operaciones:

ID del requisito	RF14
Nombre	Registrarse.
Descripción	Una persona podrá registrarse facilitando su nombre, apellidos, correo, día de cumpleaños, un nombre de usuario y su respectiva contraseña. El nombre de usuario será único en la aplicación ya que será el que se usará para identificar a los usuarios.
Prioridad	Alta.
Entrada	Nombre: String. Apellidos: String. Correo: String. Nombre de usuario: String, no puede ser nulo. Contraseña: String, no puede ser nulo.
Salida	Si existe algún error se mostrará al usuario y podrá volver a introducir los datos, mientras que, si está todo correcto, se guardarán los datos y se le redireccionará al programa mostrándose la pantalla principal de la aplicación con la sesión ya iniciada.

Tabla 14. Requisito funcional registrarse

ID del requisito	RF15
Nombre	Iniciar sesión.
Descripción	Una persona podrá iniciar sesión indicando su usuario y su contraseña.
Prioridad	Alta.
Entrada	Usuario y contraseña.
Salida	Si sus credenciales son correctas iniciará sesión y se mostrará la pantalla principal de la aplicación, si existe algún error se le mostrará al usuario.

Tabla 15. Requisito funcional iniciar sesión

3.4. Requisitos no funcionales

ID del requisito	RNF1
Nombre	Ajuste de pantalla.
Descripción	La web debe de redimensionarse si se modifica el tamaño de la ventana.
Prioridad	Alta.

Tabla 16. Requisito no funcional ajuste de pantalla

ID del requisito	RNF2
Nombre	Compatibilidad de navegadores.
Descripción	La web debe funcionar en los navegadores más usados, Google Chrome, Mozilla Firefox y Internet Explorer.
Prioridad	Alta.

Tabla 17. Requisito no funcional compatibilidad de navegadores

ID del requisito	RNF3
Nombre	Legibilidad del texto.
Descripción	Todos los textos de la web deben de ser comprensibles y fáciles de leer por todos por lo que se usará una única y clara tipografía sans-serif que será la “Arial”.
Prioridad	Alta.

Tabla 18. Requisito no funcional legibilidad del texto

4. Diseño de la solución

4.1. Arquitectura del sistema

Nuestro sistema utilizará la conocida estructura de software de Modelo-Vista-Controlador.

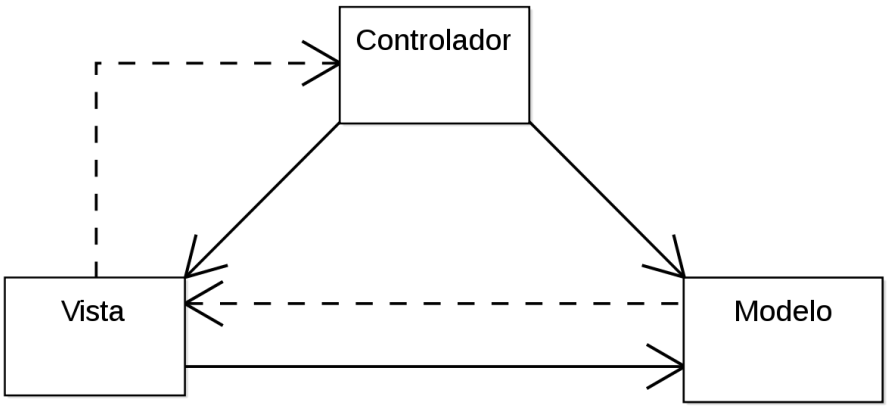

Ilustración 7. Esquema arquitectura

La principal ventaja de esta arquitectura es su gran utilidad para sistemas web, esto se debe a que es muy útil para organizar de manera correcta nuestro código, nos ayuda a separar las responsabilidades y las distintas capas de nuestro sistema. Esto produce que sea más sencillo la reutilización de código.

La contraparte de esta arquitectura es que los controladores pueden llegar a almacenar mucha cantidad de código, pero al ser un sistema sencillo y las partes están claramente diferenciadas, no deberían de llegar a tener mucha carga por lo tanto no debería de convertirse en un problema.

Como se observa el patrón se divide en tres partes, Modelo, Vista y Controlador, a continuación describiremos esas partes:

- **Modelo**: el modelo es la capa donde se trabaja con toda la información relativa a los datos y las ejecuciones de crear, modificar o borrar de la base de datos lo necesario. En esta capa se trabajará con una estructura intermedia para no ejecutar directamente las sentencias SQL. Esto nos proporcionará una persistencia en los datos que servirá para trabajar más cómodamente entre las diferentes partes del sistema.

- **Vista**: la vista, como su nombre indica, es la parte de la aplicación que se mostrará al usuario y por la cual podrá interactuar. Esta capa extraerá los datos del modelo, pero no será un acceso directo a éstos. Principalmente se tratan de códigos *html* sin carga funcional.

- **Controlador**: los controladores son la capa que sirve de enlace entre las demás, es aquí donde se encontrará el código necesario para el funcionamiento del sistema, al servir de enlace entre los otros dos, no debe manejar los datos ni mostrar salidas al usuario, sino más bien preparar todo lo necesario para que las otras dos capas funcionen de manera correcta y no necesitan acumular código que no les corresponde.

4.2. Diseño detallado

4.2.1. Diseño de la base de datos

El diseño de la base de datos para nuestro proyecto no es muy complejo ya que solo requiere una pequeña cantidad de clases.

- **User:** Será la tabla de la base de datos donde se almacene la información de los usuarios, como su nombre, apellidos, correo electrónico, el nombre de usuario y su contraseña. También almacenará la fecha de creación y la fecha de última modificación.

- **Calendar:** Aquí se almacenará la información de los calendarios, su título y el id del usuario que lo haya creado, también guardará la fecha de inserción y su última actualización.

- **Event:** En esta tabla se encontrarán los datos de los eventos, tanto su título y descripción, así como el color que se le quiera asociar, el tiempo de inicio y fin. Aquí también se guardará el campo de *calendar*, que es el que nos permite asociar los distintos eventos a un calendario en concreto, también se guardará la fecha de alta del evento y la de su último cambio.

- **Shared:** Esta será una tabla relación de la unión de *calendar* y *user*. Representará a la compartición de un calendario para que otros usuarios puedan visualizarlo, añadir eventos y modificarlos, esto dependerá de su permiso, por eso almacenará el campo *permission* junto con el id del usuario y del calendario a los que haga referencia esta relación.

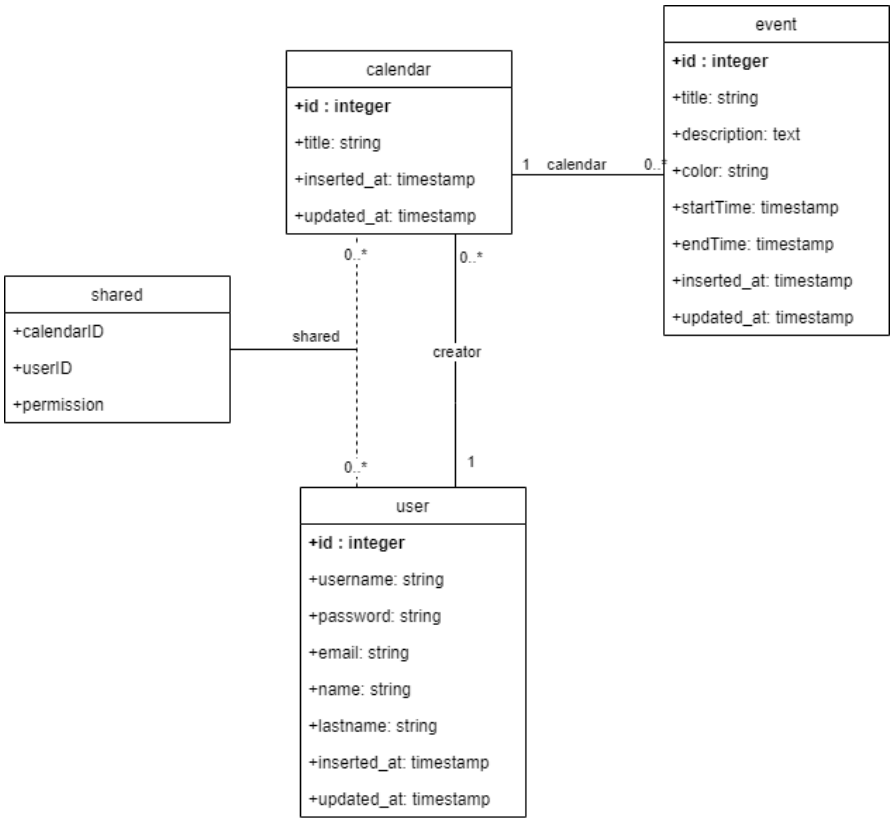

Ilustración 8. Esquema base de datos

4.2.2. Prototipos de la aplicación

Previamente al desarrollo del aplicativo y teniendo en cuenta las diversas características que le otorgaremos al calendario, se ha diseñado una serie de prototipos con el fin de facilitar el desarrollo del mismo.

En primera instancia, encontramos la página de inicio desde la que se accederá a través de un buscador, esta tendrá las sencillas opciones de iniciar sesión con nuestras credenciales, un nombre de usuario y una contraseña, si es que ya disponemos de una cuenta.

Ilustración 9. Prototipo iniciar sesión

En caso de no disponer de una cuenta, podremos crear una con el formulario de "crear nueva cuenta" aportando la información necesaria, como es el nombre de usuario y la contraseña, y la información opcional, como son el nombre, los apellidos y el correo electrónico. Este también servirá para la edición de los datos personales del usuario si desea cambiarlos más tarde.

Ilustración 10. Prototipo datos personales

Una vez iniciada la sesión encontraremos la pantalla principal de la aplicación, cuya apariencia variará dependiendo del modo de vista seleccionado. El modo de vista predeterminado será la vista mensual, cuya apariencia será la siguiente: calendario situado en la zona central y una columna de opciones a la izquierda desde la que podrá crear nuevos calendarios y activarlos o desactivarlos para cargar los eventos asociados a estos mismos. También tendrá la opción de cambiar el tipo de vista con un desplegable y cambiar la información de usuario desde su perfil arriba a la derecha.

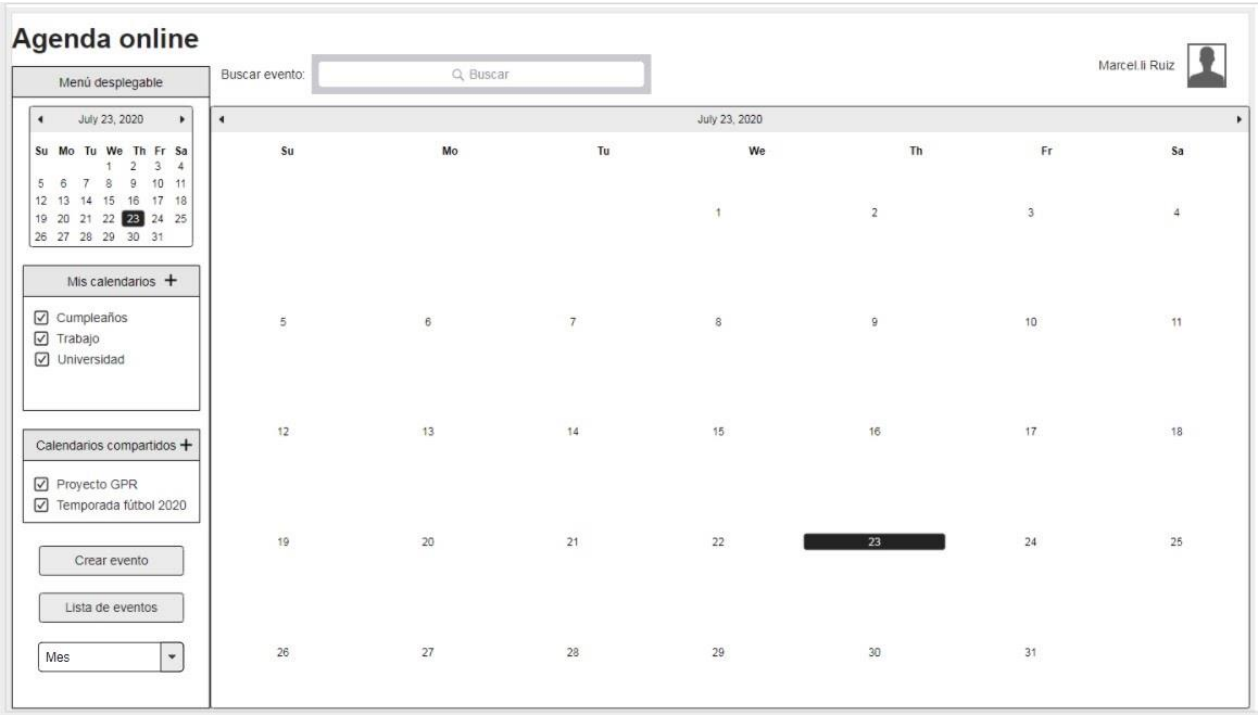

Ilustración 11. Prototipo vista mensual

Otra de las vistas que tendrá será la semanal, que sigue la misma configuración que la mensual, solo que se mostrarán los eventos con sus respectivas horas.

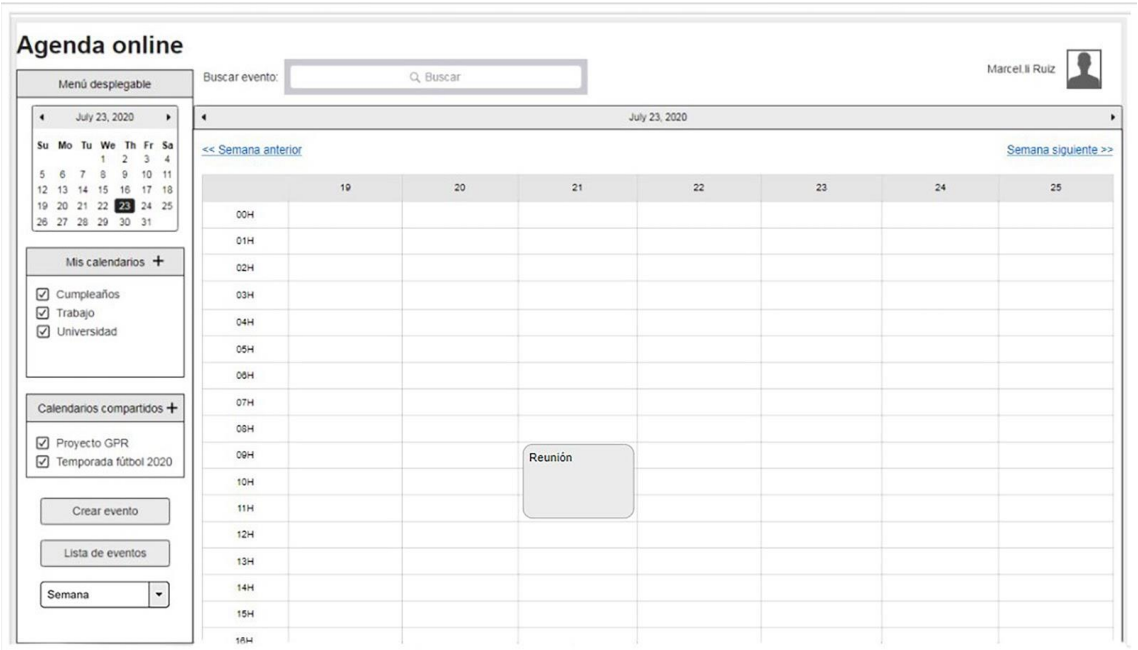

Ilustración 12. Prototipo vista semanal

También tendrá la opción de visualizar una lista de los eventos asociados a los calendarios seleccionados con la misma estructura que las anteriores vistas.

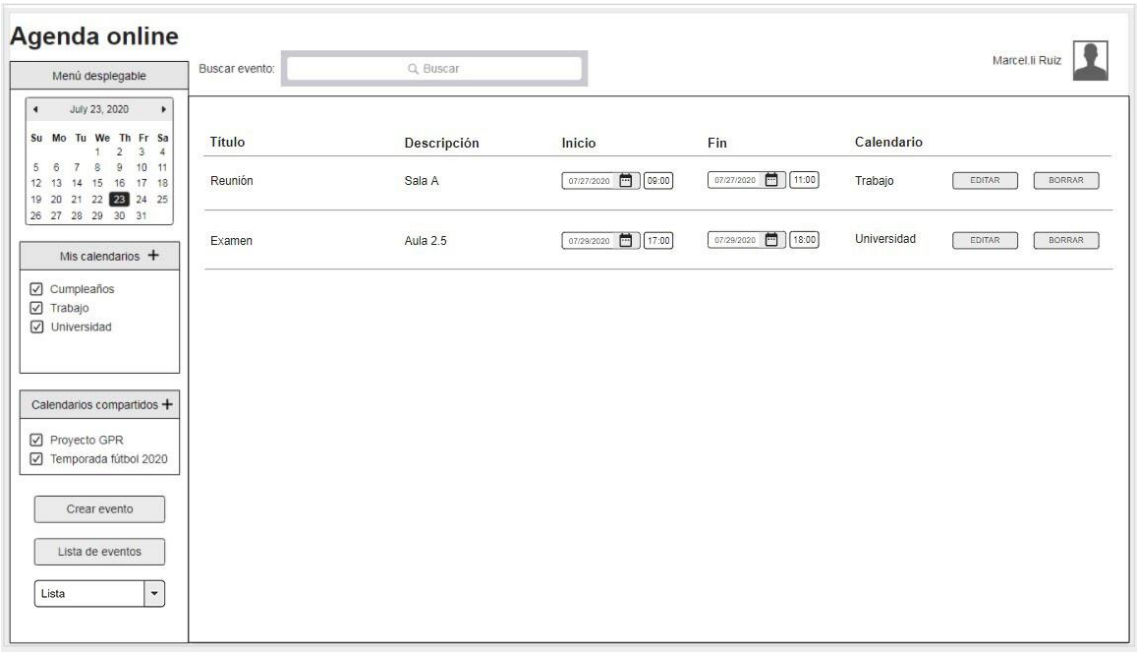

Ilustración 13. Lista de eventos

Desde esta ventana principal podremos acceder a las ventanas de creación de calendarios y eventos, ambas serán ventanas emergentes que se superpondrán a la ventana principal. Para crear un calendario solo será necesario indicarle un nombre o título.

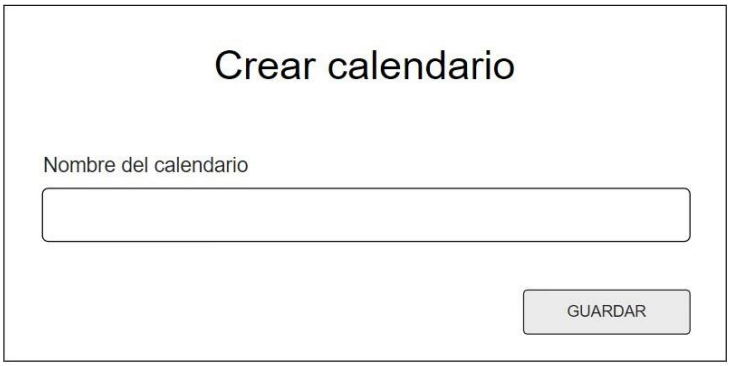

Ilustración 14. Prototipo crear calendario

Por último, para crear un evento será necesaria más información: un título, las fechas y horas de inicio y final, la descripción será opcional.

Ilustración 15. Prototipo crear evento

Con esto tendríamos todos los prototipos principales en los cuales se basará y se intentará aproximar lo máximo posible el desarrollo final, aunque es probable que alguno sufra alguna variación para un mejor funcionamiento de la aplicación o para satisfacer nuevas funcionalidades no previstas surgidas durante el desarrollo.

4.3. Tecnología utilizada

En este apartado vamos a enumerar las tecnologías que se han empleado para desarrollar el proyecto haciendo énfasis en los puntos fuertes que tienen y por qué las hemos elegido.

4.3.1. HTML 5

HTML son las siglas de *HyperText Markup Language*, en español Lenguaje de Marcado de Hipertextos. Es la base del desarrollo de aplicaciones web, sirve para definir la estructura del contenido de las diferentes páginas.

La parte de Hipertexto se refiere a la manera de conectar las diferentes páginas de la web, tanto dentro de un sitio web como con el resto de la red.

La parte de marcado define la estructura básica del lenguaje, que consiste en etiquetas para marcar los diferentes elementos que se mostrarán en el navegador.

4.3.2. CSS

CSS son las siglas de *Cascading Style Sheets*, en español Hojas de Estilo en Cascada. Es el lenguaje de estilos por excelencia para el desarrollo de aplicaciones web, este define cómo debe ser renderizado un elemento del aplicativo cuando se represente en el navegador.

El uso en nuestra aplicación será para definir los diferentes tamaños de los textos y títulos, así como personalizar los diseños de los elementos como los botones, listas, o barras de búsqueda.

Su sistema para aplicar un estilo a una clase concreta de elementos lo hace realmente fácil de utilizar y muy práctico, es por eso que se usa de manera extendida en todas las aplicaciones de la red.

.

4.3.3. Librería Material Icons

Se trata de una librería de iconos muy completa con diseños simples, claros y modernos, que se han creado siguiendo guías de estilo especializadas en crear unos diseños basados en conceptos conocidos mundialmente y de fácil entendimiento. Se puede modificar el tamaño de los iconos para así adaptarlo a la situación de cada elemento, sin modificar su entendimiento y visualización.

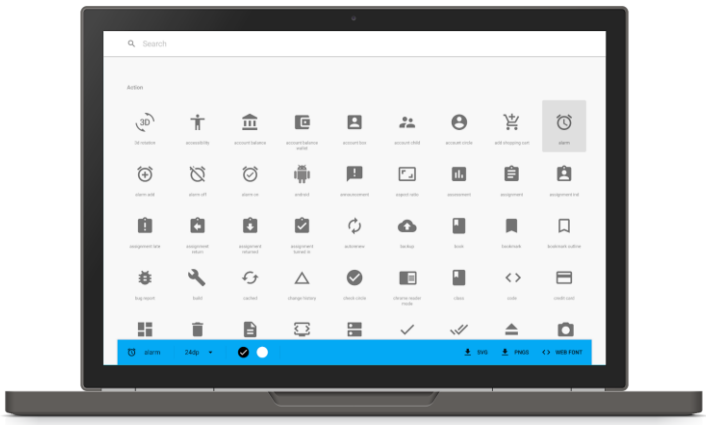

Ilustración 16. Material icons

4.3.4. Lenguaje Elixir

En este apartado se analizará el lenguaje principal sobre el que se ha construido la aplicación, que en nuestro caso es Elixir [3].

Como lenguaje de programación principal se ha utilizado Elixir, un lenguaje dinámico y funcional que fue específicamente diseñado para el desarrollo y mantenimiento de aplicaciones concurrentes y altamente escalables. La razón principal para usar este lenguaje de programación es que Elixir está escrito en Erlang [4], y se ejecuta sobre la máquina virtual del mismo, por ello comparte las mismas funcionalidades principales de este lenguaje. Estas son sus principales características:

- **Escalabilidad**: Todos los códigos son altamente escalables debido al sistema de hilos de ejecución ligeros, conocidos como procesos, que son sobre los que se ejecutan las funciones y se encuentran aislados y no comparten recursos, por lo que se comunican a través de mensajes. Gracias a la naturaleza de estos hilos de ejecución aislados, se puede tener un gran número de procesos ejecutándose simultáneamente y utilizando eficientemente la potencia de la máquina sobre la que se ejecutan.

- **Tolerante a fallos**: Elixir posee unos supervisores que evitan que el sistema quede colapsado cuando se produzcan errores en la aplicación, estas son herramientas que describen cómo reiniciar partes del sistema para así volver a un punto inicial que es seguro para un buen funcionamiento.

- **Programación funcional**: Elixir es un lenguaje funcional, es decir, un paradigma de programación declarativa, principalmente basado en funciones que llevan a cabo todo el funcionamiento del aplicativo.

- **Extensible:** Permite a los desarrolladores extender el lenguaje para dominios concretos con el fin de aumentar la productividad.

La combinación de las características mencionadas, hace al lenguaje Elixir perfecto para el desarrollo de aplicaciones web como es nuestro proyecto, es por ello que se ha seleccionado este lenguaje.

4.3.5. Framework Phoenix

Phoenix es un potente *framework* de desarrollo web escrito en Elixir, que usa la estructura de software de Modelo-Vista-Controlador. Es un *framework* que ofrece una buena velocidad de desarrollo y, al utilizar la máquina virtual de Erlang, brinda las mejores características del lenguaje como el desarrollo de sistemas concurrentes, el buen mecanismo de distribución y una buena tolerancia a fallos.

Por estas principales características sirve perfectamente para el presente proyecto, dado que encaja con la arquitectura del sistema y es un *framework* muy potente para el lenguaje Elixir.

4.3.6. Librería LiveView

LiveView es una novedosa librería de Phoenix que proporciona una experiencia en tiempo real a los usuarios renderizando los HTML desde el servidor. La funcionalidad principal es que los cambios se reciben como mensajes y actualiza el estado de los componentes, para así recargar las partes necesarias del HTML. Una vez realizada la primera carga de la página, que suele ser más costosa, el resto son mucho más rápidas, la contraparte es que usa más memoria en la parte del servidor.

4.3.7. Base de datos PostgreSQL

Se trata de un sistema de gestión de bases relacional orientado a objetos y de código abierto que lleva en continuo desarrollo desde hace más de 30 años. Al tener un modelo relacional orientado a objetos nos ofrece las mejores características de los dos tipos de esquemas, por lo que podríamos destacar que facilita la escalabilidad debido a su sistema de clases. Gracias a estas características, se ha decidido hacer uso de este sistema para el proyecto

5. Desarrollo de la solución

Una vez claras las especificaciones necesarias y la idea del proyecto, procedemos a desarrollarla. Al ser una nueva aplicación no partimos de ningún sistema, por lo que se debe crear desde 0 la aplicación y la base de datos.

Gracias al *framework* Phoenix, tenemos acceso a una variedad de comandos que nos facilitan enormemente la tarea de crear la base de datos, así como sus tablas. Esto se debe a *Ecto*, un conjunto de herramientas que al crear nuestro proyecto nos generan unas tablas asociadas a las clases que hayamos creado.

Ilustración 17. Logo ecto

Para crear el resto de tablas de manera manual, podemos usar *Ecto.Migration*, que nos permite la creación de una serie de *migrations* cuando necesitemos crear, actualizar o borrar algún elemento de nuestra base de datos. Es lo que utilizaremos para crear las tablas necesarias de nuestra aplicación. Por ejemplo, la creación de la tabla de *User* se puede realizar de la siguiente manera:

```
defmodule Agenda.Repo.Migrations.Users do
  use Ecto.Migration

  def change do

    create table(:user) do
      add :name, :string
      add :lastname, :string
      add :email, :string
      add :username, :string
      add :password, :string

      timestamps()
    end

  end
end
```

Ilustración 18. Crear tabla en base de datos

A continuación, se puede añadir la operación de:

```
create index("user", :username, unique: true)
```

Con esto crearemos un índice en la base de datos que levantará un error cuando se intente insertar otras filas con la columna de *username* con el mismo valor, así evitaremos duplicados. Dado que en nuestra aplicación cada usuario tiene que tener un identificativo claro y único, hemos utilizado el nombre de usuario, pero se podría haber usado el correo electrónico de igual forma, esto nos servirá para identificar los calendarios de cada usuario y para compartir los calendarios propios con otras personas desde la aplicación.
Usando esta herramienta crearemos la base de datos con la que almacenar toda la información de la aplicación. Una vez tenemos esa parte, conseguimos tener nuestra aplicación y nuestra base de datos conectada, sin embargo, para realizar cambios o consultas en ella, necesitaremos métodos y un modelo que lo represente, para ello crearemos un *script* para cada clase, *User*, *Calendar*, *Event* y *Shared*. En las que añadiremos una estructura que definirá los componentes del dato con el que trataremos, y también un *Changeset*. Que nos servirá para crear estos datos estructurados dentro de la aplicación, así como validarlos de forma sencilla.

```
schema "user" do
  field :name, :string
  field :lastname, :string
  field :email, :string
  field :username, :string
  field :password, :string

  timestamps()
end

def changeset(user, attrs \\ %{}) do
  user
  |> cast(attrs, [:name, :lastname, :email, :username, :password])
  |> validate_required([:username, :password])
end
```

Ilustración 19. Crear estructura de datos

Esto nos permitirá manejar las estructuras de datos de manera sencilla y de esta manera unificar su formato para poder filtrar o validar sus diferentes campos cuando se introduzcan en un formulario.

La manera de manejar y organizar la base de datos como hemos comentado es a través de *Ecto*, pero dentro de este, encontramos el módulo Repo el cual nos permite hacer las acciones necesarias para manejar los datos y almacenarlos. Proporcionando el nombre de la aplicación y el adaptador, que será Postgres, podemos hacer las consultas gracias a sus métodos.

Algunos ya vienen predefinidos, como son el crear un elemento, el listar toda una tabla o eliminar una línea en concreto. Estas acciones son posibles gracias a que los datos estarán en forma de *changeset*, por lo que se podrán leer los datos de una manera y la base de datos los podrá interpretar de manera correcta. Aunque no siempre será así, la forma de realizar consultas personalizadas es bastante sencilla y nos permite realizar todo tipo de acciones.

El diseño de la parte principal de la aplicación, es decir, el manejo de la ventana central con las diferentes vistas posibles y la columna lateral, en un principio fue pensada para desarrollarse siguiendo la arquitectura MVC, separando sus componentes, y cada vez que fuera necesario se cargaría el componente requerido. Pero finalmente se descartó esa idea y a cambio de ella se ha realizado con la librería *LiveView*, con ella podemos realizar la parte frontend de la aplicación de una manera muy eficiente.

Como se ha comentando anteriormente, la librería LiveView destaca principalmente por su capacidad instantánea de reacción lo que nos proporciona una respuesta en tiempo real de lo que el usuario esté realizando, por ello se ha considerado la mejor opción para el desarrollo de la aplicación, ya que en un calendario en el que lo normal es que se estén creando y viendo eventos de manera continua se deben evitar cargas que frenen y empeoren la experiencia del usuario.

La manera en la que se ha organizado esta ventana principal es la siguiente:

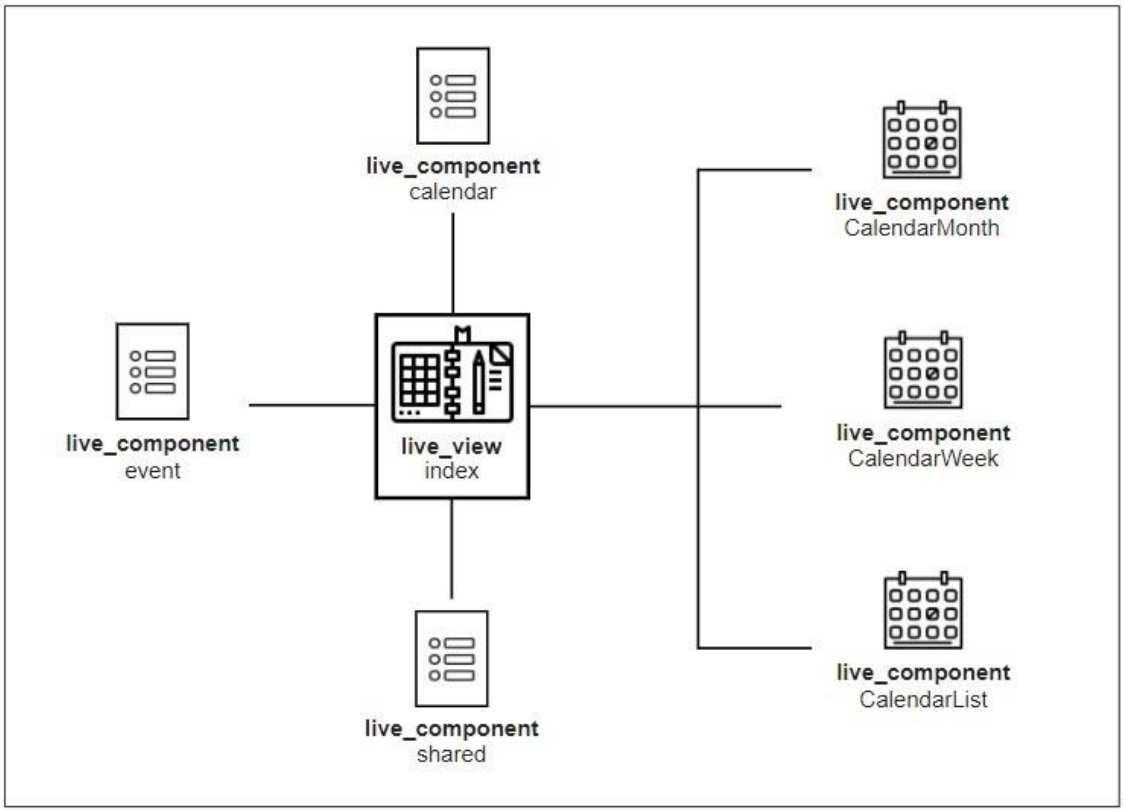

Ilustración 20. Esquema componente live_view

En primer lugar, tenemos el elemento *live_view* principal llamado *Index*, es aquí donde se cargarán todos los demás dependiendo de las selecciones del usuario. En este lugar se encuentran los manejadores que captarán los diferentes eventos que se lancen desde cualquier parte de la ventana.

El elemento *live_view* o los *live_components* que veremos a continuación, renderizan un código html que tienen asociado, pero desde ese *script* se pueden crear métodos que sean necesarios para ese componente en concreto, como pueden ser los manejadores de eventos de los formularios para comprobar la información y crear los datos en la base de datos. Otros, sin embargo, estarán en el componente principal, como serán cuando el usuario seleccione qué calendarios ver, o cuando haga clic en alguna casilla del calendario para crear un evento nuevo.

Como se puede observar, tenemos el *live_view* principal y tres *live_component* que son los diferentes formularios para crear calendarios, eventos o compartir un calendario con otro usuario. Estos tres componentes se renderizan como ventanas emergentes que se muestran por encima de la página principal. También tenemos los componentes de las tres vistas, la mensual, la semanal y la lista o agenda, al contrario que los anteriores, estos componentes se mostrarán en la ventana principal en una posición concreta, dependiendo de la selección del usuario en un desplegable, se cargará una vista u otra.

Para el desarrollo de la vista semanal y mensual, se ha optado por separar y crear un componente por cada casilla de la tabla, con el fin de organizar mejor los diferentes métodos y así obtener los datos de manera clara.

La parte separada de este núcleo principal es la que gestionan los usuarios, esta sí que sigue la estructura tradicional de modelo-vista-controlador con sus componentes. La razón por la que esta parte del sistema está separada y mantiene esa estructura, es porque no supondría una gran diferencia en la eficiencia y en la satisfacción del usuario. Ya que las actualizaciones no se deben de reflejar de manera instantánea. Además facilita la tarea de separar el sistema en función de si el usuario está registrado o no.

```
pipeline :auth do
  plug AgendaWeb.Users.Pipeline
end

# We use ensure_auth to fail if there is no one logged in
pipeline :ensure_auth do
  plug Guardian.Plug.EnsureAuthenticated
  plug(:put_user_token)
end

pipeline :browser do
  plug :accepts, ["html"]
  plug :fetch_session
  plug :fetch_live_flash
  plug :put_root_layout, {AgendaWeb.LayoutView, :root}
  plug :protect_from_forgery
  plug :put_secure_browser_headers
end
```

Ilustración 21. Pipelines de redirección

Para la parte de autenticación de usuarios, se ha empleado otra biblioteca de Phoenix, en este caso se trata de Guardian. A través de esta biblioteca podemos acceder a la clase *Plug*, desde la cual podemos comprobar de manera sencilla si el usuario ha iniciado la sesión con *EnsureAuthenticated* para así redirigirlo por los canales que nosotros queremos.

Cuando aseguramos que es un usuario autentificado pasamos el *token* por las *cookies* de *session* para mantener la sesión iniciada aunque cambie la ventana y así redireccionarlo. Una vez tenga la sesión iniciada puede acceder al calendario principal.

Para crear un nuevo usuario y cumplir con la seguridad es necesario encriptar la contraseña, para ello usaremos *bcrypt_elixir*, que usará una función para encriptar la contraseña cuando se cree el usuario y guardarla en la base de datos de esa forma. Cuando una persona inicie sesión se requerirá otra función que nos desencripte esa contraseña que tenemos almacenada en la base de datos y compararla con la que hayan introducido en el formulario, de esta manera si son iguales le permitirá el acceso al usuario.

6. Pruebas

Una vez completado el desarrollo completo de la aplicación es necesario validarlo para comprobar que no tiene vulnerabilidades y se cumplen las especificaciones. Para ello vamos a crear un par de casos de prueba y con ellos comprobar la eficacia del sistema con diferentes entradas.

Nombre: Pruebas de usuarios		**Identificador:** CP-001		
Propósito:				
Verificar la parte del sistema encargada de los datos del usuario como que inicien sesión, se creen una cuenta.				
Acciones y salidas				
ID	**Acciones**	**Entradas**	**Salida esperada**	**Salida obtenida**
01	Iniciar sesión en el sistema sin una cuenta.	Usuario: marcel Contraseña: contraseña	Mensaje que las credenciales no son correctas.	No nos inicia sesión indicando que no existe el usuario.
02	Crear usuario nuevo con un nombre de usuario ya existente.	Nombre: Marcel Apellidos: Ruiz Correo: marcel@gmail.com Usuario: usuario Contraseña: contraseña	Mensaje de error diciendo que el nombre de usuario ya existe.	No nos crea el usuario y nos muestra la ilustración 22.
03	Crear usuario nuevo con datos correctos.	Nombre: Marcel Apellidos: Ruiz Correo: marcel@gmail.com Usuario: marcel Contraseña: contraseña	Nos inicia sesión y nos carga la ventana principal de la aplicación.	Nos redirige a la ventana principal con la sesión iniciada en nuestro perfil recién creado.
04	Modificar el nombre de usuario.	Nombre: Marcel Apellidos: Ruiz Correo: marcel@gmail.com Usuario: marcelruiz Contraseña: contraseña	Nos devuelve a la ventana principal con los datos cambiados.	Nos carga la pantalla principal y comprobamos que se han actualizado los datos.
05	Cerrar sesión.	-	Nos devuelve a la página de inicio de sesión.	Nos redirige a la página del *login*.
06	Iniciar sesión con credenciales correctos.	Usuario: marcelruiz Contraseña: contraseña	Nos inicia la sesión correctamente y nos carga la ventana principal de la aplicación.	Nos inicia la sesión correctamente y nos carga la ventana principal de la aplicación.

Tabla 19. Caso de prueba 1

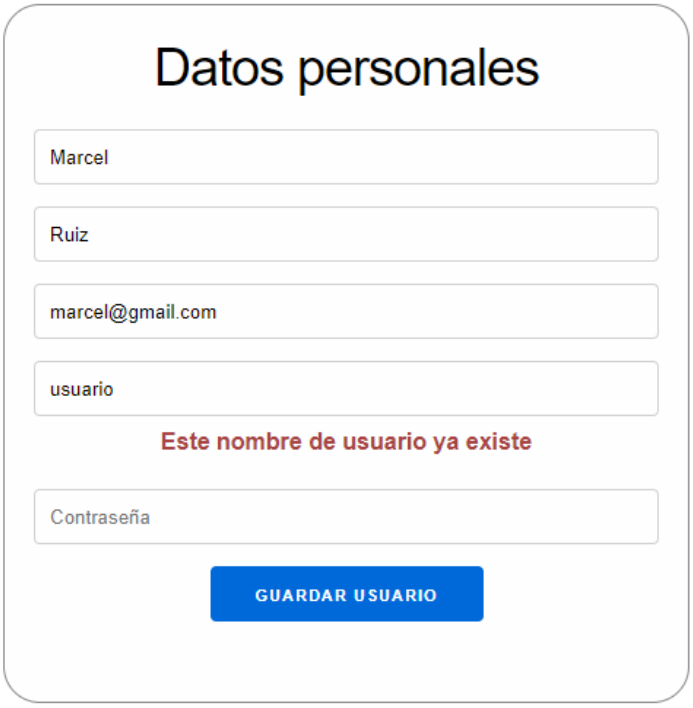

Ilustración 22. Error datos crear usuario

Como podemos observar el caso de prueba se ha cumplido satisfactoriamente por lo que podemos concluir que los requisitos funcionales RF9, RF14 y RF15 están bien realizados y no parecen tener fallos.

Nombre: Pruebas de eventos y calendarios		**Identificador:** CP-002		
Propósito:				
Comprobar la creación de eventos y calendarios.				
Acciones y salidas				
ID	**Acciones**	**Entradas**	**Salida esperada**	**Salida obtenida**
01	Crear un evento sin ningún calendario creado.	Título: Reunión Descripción: Reunión con el departamento de informática Inicio: 2020/12/04 10:00 Fin: 2020/12/04 11:00 Calendario: - Color: Rojo	Mensaje indicando que es necesario indicar un calendario.	Podemos observar que nos muestra un mensaje de error en la ilustración 23.
02	Crear un calendario	Título: Reuniones	Se crea el calendario y se muestra en la columna lateral.	Se crea el calendario con el título pasado y aparece en la columna lateral.
03	Crear un evento nuevo	Título: Reunión Descripción: Reunión con el departamento de informática Inicio: 2020/12/04 10:00 Fin: 2020/12/04 11:00 Calendario: Reuniones Color: Rojo	Se crea el evento y nos devuelve a la página principal	Nos carga la ventana principal de la aplicación.
04	Seleccionamos un calendario para cargarlo en el calendario	Calendario seleccionado	Nos carga los eventos del calendario en la visualización y veremos el evento Reunión	Se visualiza el evento en el calendario con el título “Reunión” y de color rojo. Lo podemos observar en la ilustración 24.
05	Modificar el título de un evento	Título: Reunión informática Descripción: Reunión con el departamento de informática Inicio: 2020/12/04 10:00 Fin: 2020/12/04 11:00 Calendario: Reuniones Color: Rojo	Nos devuelve a la ventana principal y podemos ver el nombre del evento Reunión cambiado a Reunión informática	Carga la ventana principal y comprobamos que ha cambiado el título.
06	Modificar el calendario en el que se encuentra un evento	Título: Reunión informática Descripción: Reunión con el departamento de informática Inicio: 2020/12/04 10:00 Fin: 2020/12/04 11:00 Calendario: Trabajo Color: Rojo	Nos devuelve a la ventana principal y ya no se observa el evento Reunión informática ya que se encuentra en otro calendario	Recarga la ventana principal de la aplicación y se ha cambiado el evento de calendario como se puede observar en las ilustraciones 25, 26 y 27.
07	Borrar evento	Evento a borrar	Nos devuelve a la ventana principal y ya no existe el evento	Se ha borrado el evento y ya no aparece en el calendario

Tabla 20. Caso de prueba 2

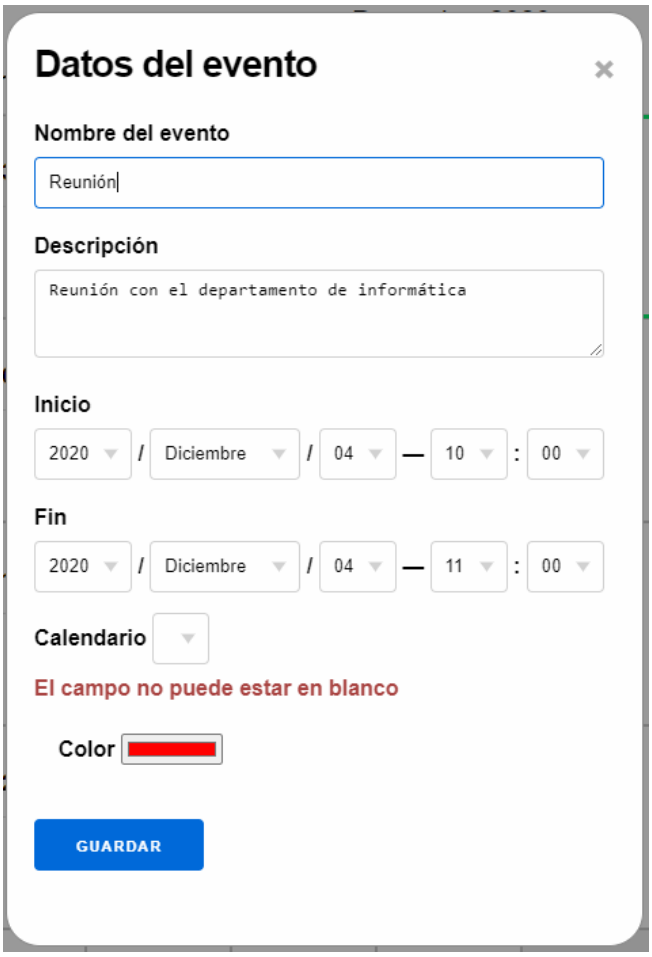

Ilustración 23. Error crear evento

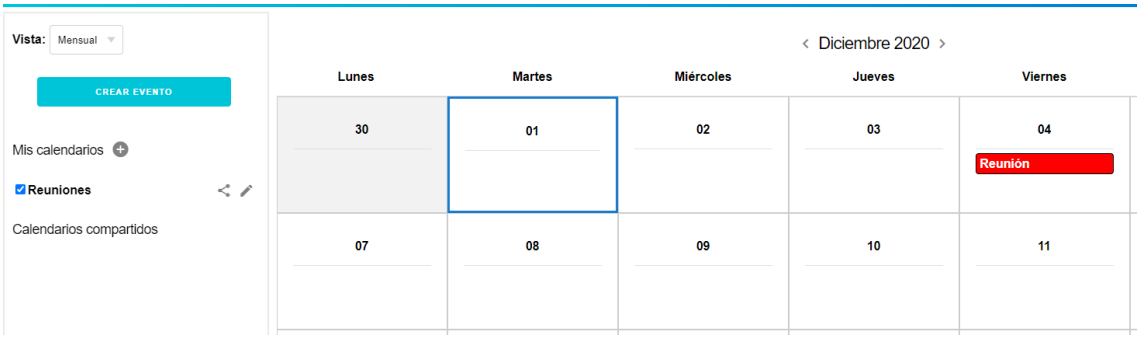

Ilustración 24. Evento creado

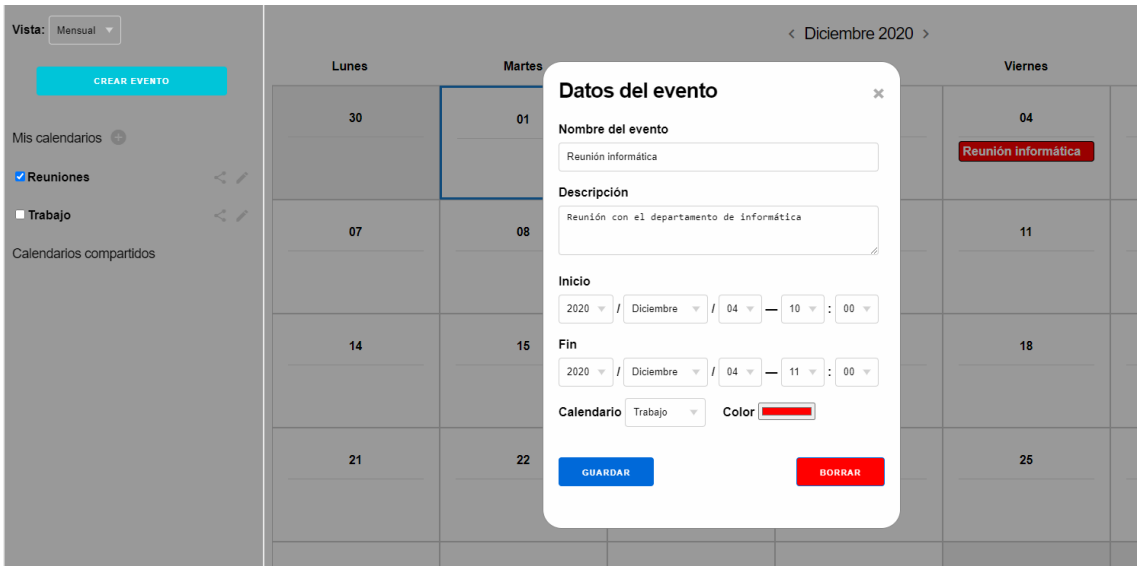

Ilustración 25. Cambiar calendario del evento

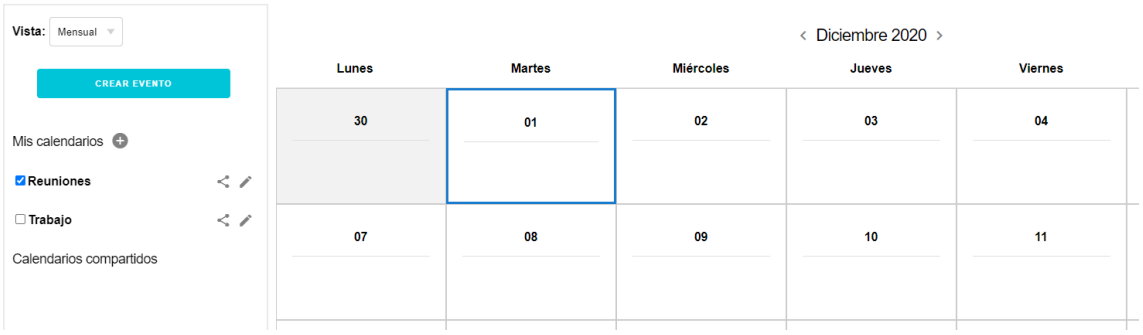

Ilustración 26. Calendario vacío

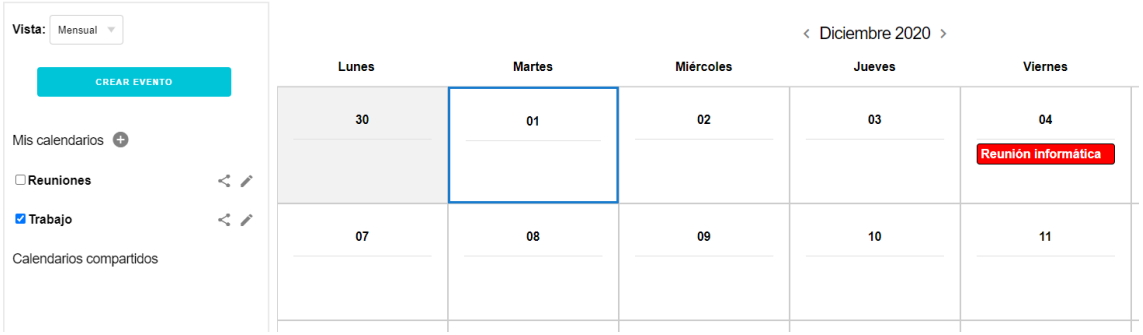

Ilustración 27. Evento en el nuevo calendario

7. Conclusiones

7.1. Resultado final

Una vez finalizado el trabajo, podemos observar cómo ha sido el resultado final y si ha cumplido con los requisitos y prototipos previamente diseñados.

Empezamos con una página de inicio con unas opciones sencillas de iniciar sesión o registrarse, si es que no se posee una cuenta.

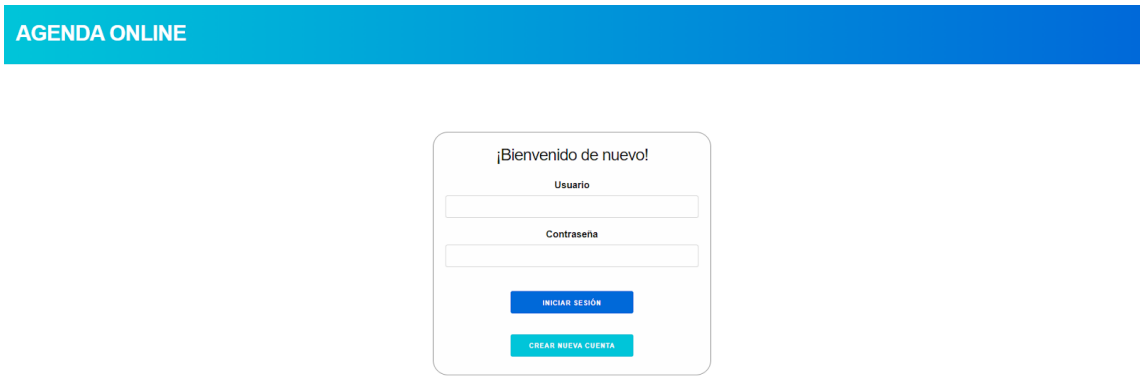

Ilustración 28. Ventana inicio

Cuenta con un diseño simple basado en dos tonalidades de azul que resultan agradables y armónicos visualmente. Si accedemos a crear cuenta se nos redireccionará al siguiente formulario, donde introduciendo nuestra información se nos registrará en el sistema y accederemos directamente a la ventana principal de la aplicación, al igual que pasaría si iniciamos sesión con nuestras credenciales.

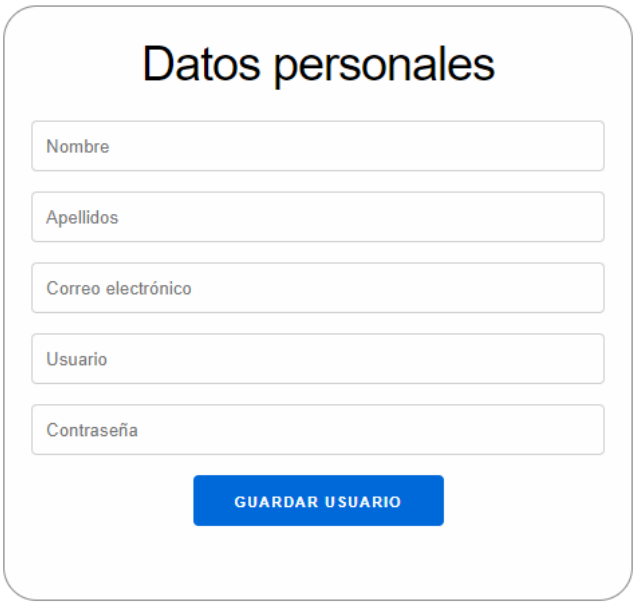

Ilustración 29. Formulario registrarse

Esta sería la ventana principal de la aplicación, en la que encontramos una columna desde la que manejar los calendarios y decidir cuales seleccionar para cargar sus eventos en el calendario central, también podremos crear nuevos calendarios, eventos y compartir los calendarios con otros usuarios.

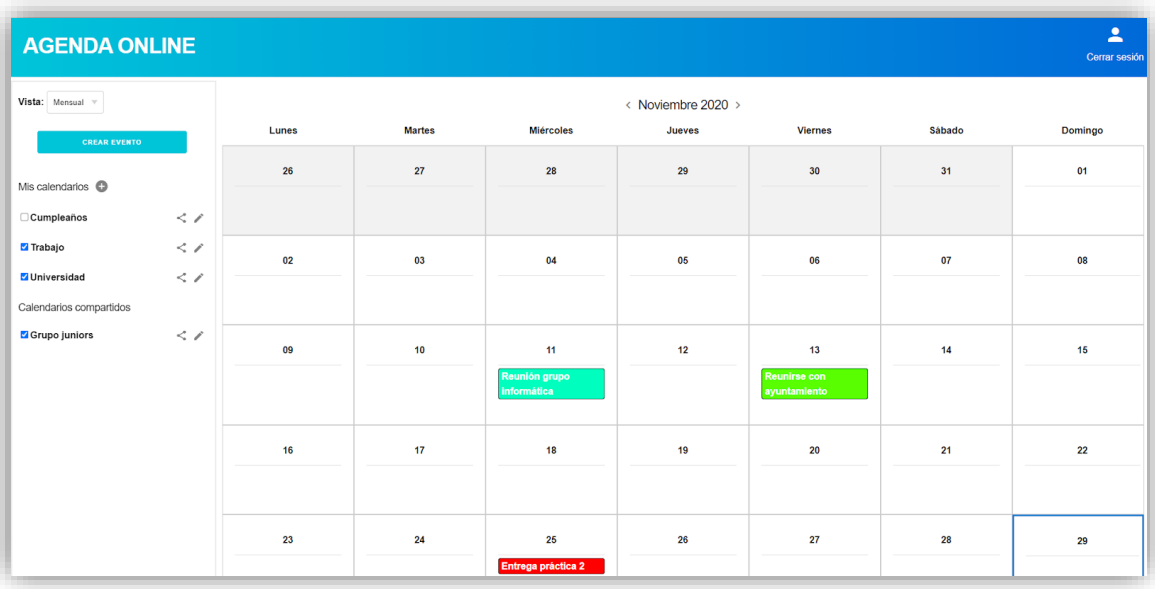

Ilustración 30.Ventana principal vista mensual

Las otras dos vistas, la semanal y la lista de los eventos, tienen la misma estructura que la mensual, con el calendario en la zona central derecha.

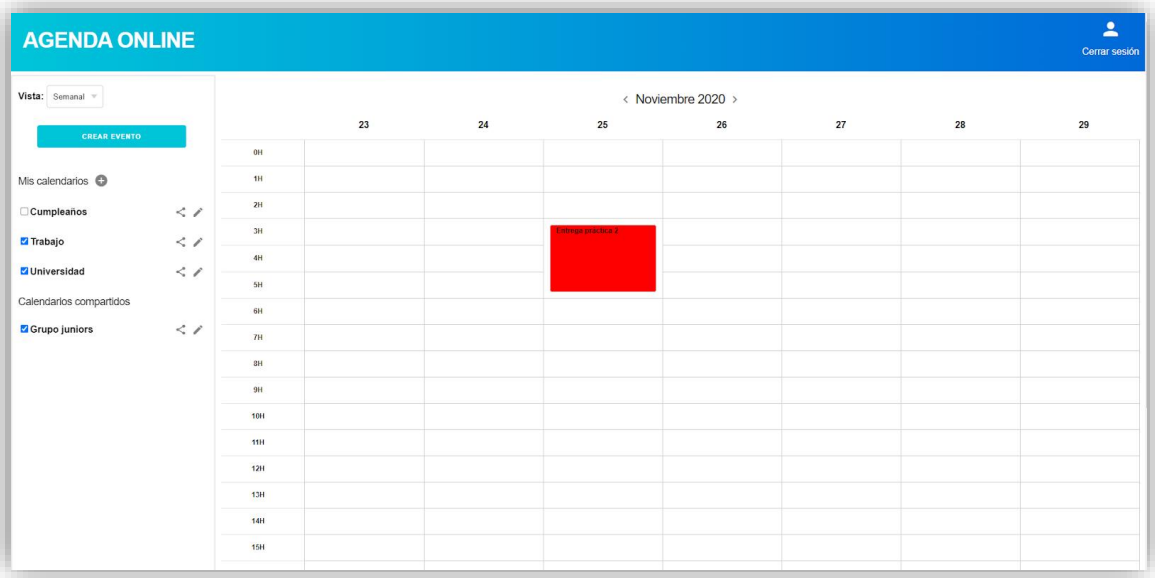

Ilustración 31. Vista semanal

Tanto la vista semanal como la mensual, permiten que al hacer clic en el calendario se les abra el formulario para crear un evento, estableciendo de manera predeterminada la fecha que hayan seleccionado; también permiten avanzar o retroceder en el tiempo desde la parte superior, avanzando un mes en la vista mensual y una semana en la semanal.

Los calendarios seleccionados se mantienen, aunque se cambie de vista, esto permite mayor versatilidad en el uso del mismo, por ello cuando cargamos la vista en forma de lista se mantienen los mostrados en el calendario, pero también aquellos que no podemos ver por el rango de tiempo. En este tipo de visualización podemos observar todos los eventos de los calendarios seleccionados en orden cronológico y podemos editarlos y borrarlos desde aquí.

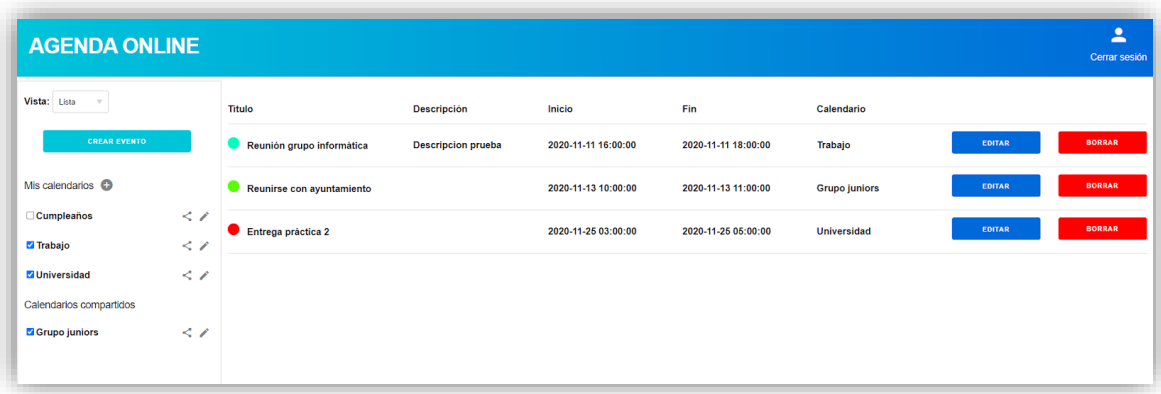

Ilustración 32. Lista de eventos

Como hemos comentado anteriormente, desde la columna lateral podemos crear y editar los calendarios, por lo que al pulsar el botón de crear (símbolo de más que aparece junto al título de "Mis calendarios"), se nos abrirá el formulario para crear un calendario, mientras que cuando pulsemos al lápiz que aparece junto a los diferentes calendarios, podremos editar este mismo título o borrar el calendario.

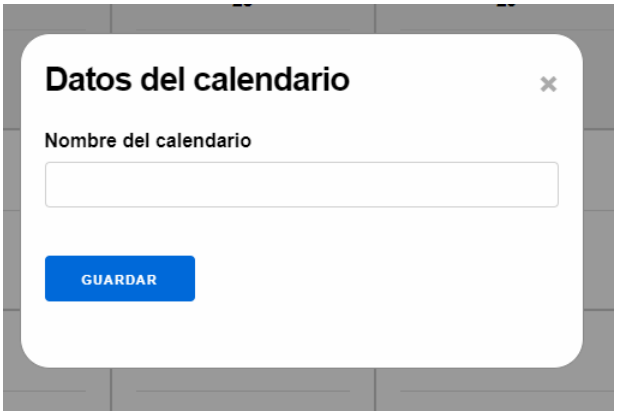

Ilustración 34. Crear calendario

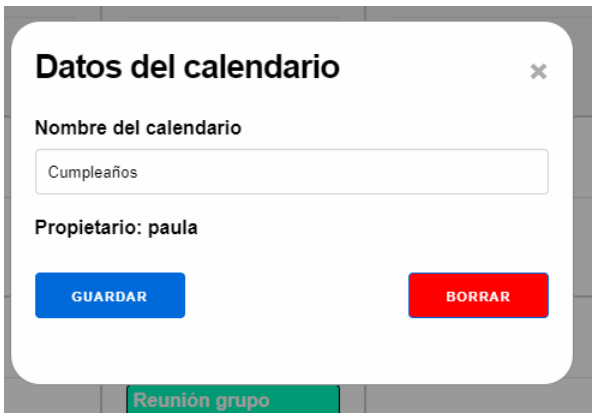

Ilustración 33. Editar calendario

También podemos observar que, al lado del icono para editar los calendarios, se encuentra el símbolo de compartir. Con este icono accederemos a la vista para compartir el calendario, además de visualizar la lista de usuarios con los que está compartido en ese momento y sus permisos.

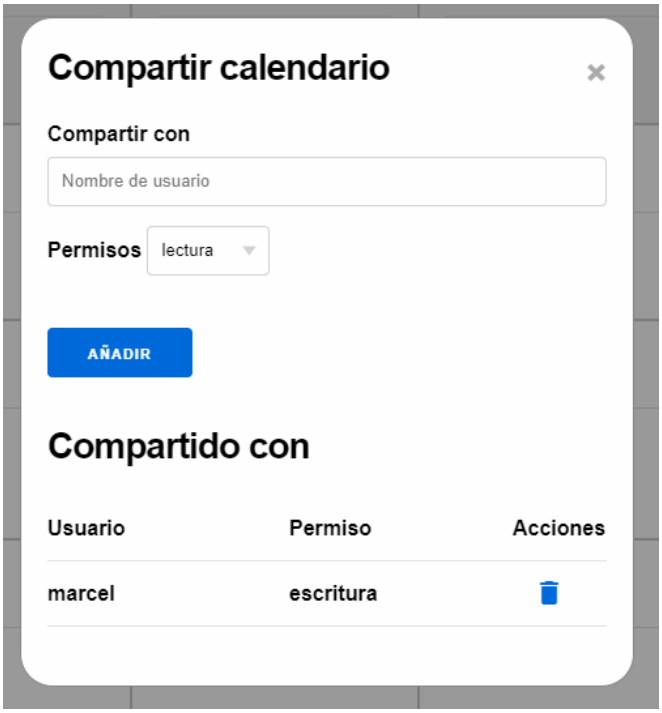

Ilustración 35. Compartir calendario

Como se puede observar, tendremos que introducir el nombre del usuario con el que deseemos compartir el calendario y seleccionar el tipo de permisos que le queramos ofrecer: de lectura, de escritura o de administrador. También podremos eliminar las relaciones que existan dejando así de compartir el calendario con los usuarios.

Por último, queda el formulario para crear un evento, al que se puede acceder como hemos comentado desde las vistas semanal y mensual al pulsar en alguna casilla o desde el botón de "crear evento" del menú lateral.

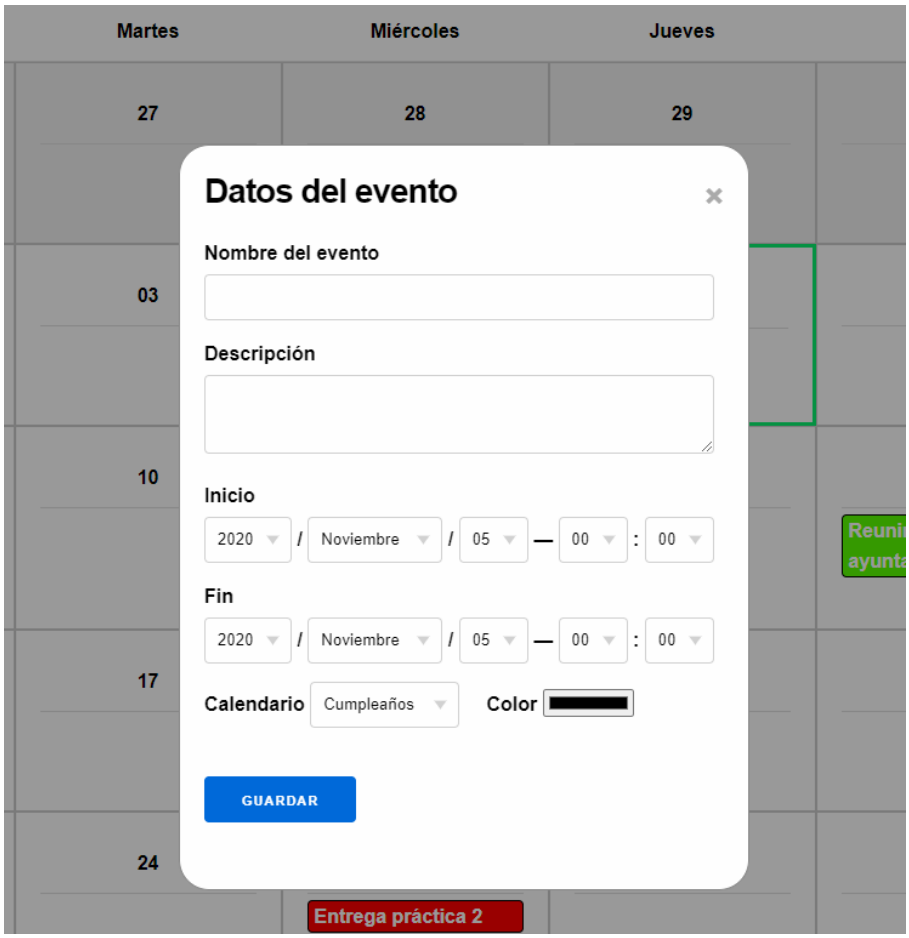

Ilustración 36. Crear evento

Se trata de un formulario sencillo en el que introduciremos un título o nombre de evento, una descripción opcional, luego las fechas y horas de inicio y fin, el calendario en el que queremos que se cree y también tendremos la posibilidad de seleccionar el color con el cual queramos que se cree para luego visualizarlo en el calendario, de esta manera podremos organizar por colores los distintos tipos de eventos que tengamos, como reuniones, entregas, aniversarios etc.

Con esto tenemos la totalidad de la aplicación. Algunas ventanas como la de edición de eventos o la de edición del perfil de usuario son iguales que sus respectivas al crearlos, pero con la información ya cargada y con la posibilidad de borrar si tenemos los permisos necesarios.

7.2. Conclusiones

En este apartado vamos a analizar las conclusiones finales que se han extraído de la realización del trabajo, además analizaremos el cumplimiento de los objetivos planteados al inicio de la memoria y en qué medida se han podido lograr. Para finalizar, también se expondrán las conclusiones personales que se han obtenido al completar el desarrollo de la aplicación.

En relación a los objetivos que se plantearon al inicio del desarrollo, consideramos que se han cumplido en su totalidad satisfactoriamente:

- Para empezar, tenemos la posibilidad de que los usuarios añadan elementos en un calendario, esto se refleja en que se pueden crear eventos desde las diferentes vistas rellenando un simple formulario.

- Ofrecer varias visualizaciones de la agenda está cumplido de igual forma, ya que se puede cargar el calendario en formato de semana, mes y lista de eventos, aunque aquí podría faltar una vista anual por lo que no se podría dar por completado el objetivo totalmente.

- También existe la posibilidad de gestionar diferentes calendarios de manera simultánea y cargar los que el usuario desee, por lo que también se puede dar por superado este objetivo.

- El sistema para compartir los calendarios se puede dar por completado, ya que nos permite elegir un calendario y compartirlo con cualquier usuario que esté registrado en el sistema, indicando además el tipo de permisos de los que dispondrá para ver los eventos, poder crear o editarlos, o en su defecto ser administrador y poder compartir el calendario y modificar sus parámetros.

- El objetivo de crear una interfaz actual, intuitiva y atractiva se puede dar por superado, ya que la interfaz mantiene un estilo simple, moderno y fácilmente entendible, lo que podría provocar satisfacción en los usuarios y por tanto crear seguidores constantes en la web.

- Por último, el objetivo de compatibilidad con los navegadores consideramos que se cumple, ya que la aplicación puede ejecutarse en los principales navegadores actuales, como Chrome, Edge y Mozilla.

En resumen, concluimos que los objetivos se han cumplido de manera satisfactoria y que el resultado del proyecto es bastante positivo. Esto es un buen resultado de manera personal, ya que en un principio yo no estaba familiarizado con el uso de lenguajes funcionales en proyectos grandes, y en concreto no había utilizado Erlang o Elixir nunca, por lo que he tenido que dedicar bastante tiempo de aprendizaje a estas tecnologías.

Además, ha sido muy interesante buscar información y aprender sobre Elixir y el *framework* Phoenix, ya que tiene una gran variedad de bibliotecas y módulos muy útiles, así como una comunidad y unos foros muy activos que proporcionan mucha ayuda. Aunque existen partes que han sido difíciles de realizar debido a la poca información que había, ya que se trataba de una nueva biblioteca como es la de

LiveView, es por esto que el tiempo para el desarrollo se vio incrementado, además que en un principio no iba a usar esta tecnología, pero al observar sus grandes características me decidí por esta librería a pesar de que aún se encuentre en desarrollo.

Aun así, considero que ha sido una experiencia muy satisfactoria y enriquecedora, ya que ha ayudado a potenciar uno de mis gustos personales como es el desarrollo web, y me ha abierto un abanico de nuevas posibilidades en base a lo aprendido, por lo que, a pesar del esfuerzo, considero que ha merecido la pena.

7.3. Relación del trabajo desarrollado con los estudios cursados

Al completar un proyecto de esta envergadura, se puede llegar a observar el impacto que pueden tener algunas asignaturas del grado en el momento de realizar un trabajo así, considero muy útiles todas las asignaturas que son impartidas, sin embargo, tengo que destacar algunas de ellas debido a las capacidades, métodos y formas de trabajar que me han enseñado y que me han ayudado a realizar el presente TFG.

Alguna de las más destacad podrían ser las de Proceso de software y Proyecto de ingeniería de software, que han sido de gran ayuda a evolucionar como persona y como ingeniero, ya que me han servido para organizarme de la mejor manera posible a abordar el proyecto de manera eficiente y que además mejore en consecuencia la calidad del producto final.

Otro apartado importante es el de la base de datos, es por ello que considero que la asignatura de Bases de datos ha sido fundamental para saber trabajar con estos sistemas, y para crear una estructura adecuada que hace más sencillo la tarea de almacenar datos.

Por otro lado, considero la especificación de requisitos otro campo indispensable para desarrollar productos software, es gracias a la asignatura de Análisis y especificación de requisitos que se ha podido empezar este proyecto, ya que, sin una base firme y unos requisitos claros, el desarrollo se puede ver afectado debido a continuos cambios que van surgiendo debido a una mala especificación.

Diseño de software es otra asignatura cursada que ha sido muy relevante, tanto para desarrollarme como informático como de manera personal debido a las diferentes maneras de abordar problemas que podemos aprender, considero muy importante esta asignatura para mantener un entorno de programación limpio y bueno, que ayuda tanto a entender nuestro código a otras personas como a nosotros mismos. Además, es imprescindible si se quiere facilitar la tarea de desarrollar a gran escala.

La comprensión de la mentalidad de los usuarios y crear interfaces atractivas que al mismo tiempo sean fácilmente entendibles por cualquiera se trata de algo esencial en el desarrollo de aplicaciones web, por este motivo las asignaturas de este índole me han ayudado a crear en la medida de lo posible las ventanas de la aplicación para que sean comprensibles e intuitivas, esto nos llevaría a un incremento de las visitas a nuestra web y a un aumento de usuarios, lo que marca la diferencia en el mundo de la web.

Para finalizar también ha sido relevante la realización de pruebas y validación de software, que se transcribe como un producto de calidad. Es importante ayudarnos de estos conocimientos para detectar errores y comprobar de lo que es capaz o en lo que flaquea nuestra aplicación para buscar una manera de solucionarlo y mejorar el producto antes de que llegue al usuario final, ya que un software con multitud de errores no resulta atractivo para el público.

8. Trabajos futuros

Una vez completado el proyecto, se puede observar que existen varias mejoras posibles que han surgido durante el desarrollo, algunas de estas posibles mejoras son:

- **Calendarios predeterminados:** Una lista de calendarios predeterminados con los días festivos en función de la residencia del usuario que se muestran en el calendario.

- **Opción de marcar como repetible el evento:** Al añadir un evento nuevo, proporcionar la posibilidad de si el evento es repetible semanalmente, mensualmente o anualmente.

- **Opción de recibir un correo:** Habilitar una opción para recibir un correo electrónico cuando se aproxime un evento importante.

Además de las mejoras en la aplicación ya desarrollada, se nos abren diferentes caminos desde los que poder expandir el proyecto, ya que, en un mundo tan conectado por la red, es fácil que la aplicación desarrollada pueda adaptarse a otros entornos como pueden ser:

- **Versión móvil:** El desarrollo de una versión para dispositivos móviles que esté conectada con la versión web para poder usar con la misma cuenta desde diferentes dispositivos, se consideraría realmente útil ya que la mayoría de usuarios hacen uso de su dispositivo móvil para el día a día.

- **Incorporación a otra herramienta de gestión:** Como se ha visto en el estudio del arte, existen diferentes herramientas que hacen uso de calendarios para ayudar a gestionar diferentes tareas, desde coordinar equipos con reuniones a programar citas para los clientes, por lo que podría ser una vía interesante de desarrollo seguir con una herramienta de esta índole, dado que cada vez son más utilizadas por las empresas.

En definitiva, se puede observar que la aplicación permite seguir desarrollándose por una variedad de caminos igual de eficientes, además se debe de tener en cuenta la gran escalabilidad y concurrencia de la aplicación debido al lenguaje en el que está escrita, gracias a esto podría llegar a evolucionar de manera sencilla admitiendo gran cantidad de usuarios y adaptarse al aumento que requiera la aplicación.

9. Referencias

[1] Andrew Stellman, Jennifer Greene. Learning Agile: Understanding Scrum, XP, Lean, and Kanban. O'Reilly, 2014.

[2] https://www.bankinfosecurity.com/google-calendar-privacy-concerns-raised-a-13133#:~:text=A%20misconfiguration%20in%20a%20Google,corporate%20details%2C%20a%20researcher%20reports.

[3] Ulisses Almeida. Learn Functional Programming with Elixir: New Foundations for a New World. The Pragmatic Programmers 2018.

[4] Fred Hebert. Learn You Some Erlang for Great Good! A Beginner's Guide. 2013