

TRABAJO FIN DE GRADO

INGENIER´IA EN INFORMÁTICA

**PLATAFORMA WEB DE ORGANIZACIÓN PERSONAL**

DESARROLLO DE UNA WEB EN C#

Autor

Fco Javier Fuentes Barragán

Director

Juan Antonio Holgado Terriza



Escuela Técnica Superior de Ingeniería Informática y de Telecomunicaciones

—

Granada, Septiembre de 2018

**ÍNDICE**

1. [ANÁLISIS DE LOS SISTEMAS DE ORGANIZACIÓN PERSONAL 9](#_Toc523945587)

2. [TECNOLOGÍAS WEB EMPLEADAS 13](#_Toc523945588)

[**2.1 HERRAMIENTAS CLOUD DE MICROSOFT (AZURE)** 14](#_Toc523945589)

3. [GESTIÓN DEL PROYECTO 16](#_Toc523945590)

4. [ESPECIFICACIÓN DE REQUISITOS 18](#_Toc523945591)

[**4.1 ESPECIFICACIÓN INFORMAL** 18](#_Toc523945592)

[**4.2 MODELO FUNCIONAL** 19](#_Toc523945593)

5. [ANÁLISIS 53](#_Toc523945594)

6. [DISEÑO 56](#_Toc523945595)

[**6.1 ARQUITECTURA** 56](#_Toc523945596)

[**6.2 DIAGRAMA ENTIDAD RELACIÓN** 57](#_Toc523945597)

[**6.3 MOCKUPS** 58](#_Toc523945598)

7. [IMPLEMENTACIÓN 62](#_Toc523945599)

[**7.1 Back-End** 63](#_Toc523945600)

[**7.2 Front-End** 69](#_Toc523945601)

[**7.3 Envío de notificaciones** 70](#_Toc523945602)

[**7.4 SEGURIDAD** 71](#_Toc523945603)

[**7.5 USABILIDAD** 72](#_Toc523945604)

[**7.6 PERSISTENCIA** 73](#_Toc523945605)

[**7.7 MANUAL DE USUARIO** 74](#_Toc523945606)

8. [CONCLUSIONES 81](#_Toc523945607)

9. [BIBLIOGRAFÍA 85](#_Toc523945608)

**Ilustraciones**

[Ilustración 1. Notas en IOS. 10](file:///E:\GoogleDrive\Informática\4-CUARTO\TFG\WeCalendarTFG\Memoria\Memoria.docx#_Toc524057435)

[Ilustración 2. Notas en Android. 10](file:///E:\GoogleDrive\Informática\4-CUARTO\TFG\WeCalendarTFG\Memoria\Memoria.docx#_Toc524057436)

[Ilustración 3. Calendario de IOS. 11](file:///E:\GoogleDrive\Informática\4-CUARTO\TFG\WeCalendarTFG\Memoria\Memoria.docx#_Toc524057437)

[Ilustración 4. Calendario de Agenda Escolar. 11](#_Toc524057438)

[Ilustración 5. Pantalla de Google Calendar. 12](#_Toc524057439)

[Ilustración 6. Gestión de Doodle. 12](#_Toc524057440)

[Ilustración 7. Captura del panel de Azure. 15](#_Toc524057441)

[Ilustración 8. Repositorio de GitHub. 18](#_Toc524057442)

[Ilustración 9. Diagrama de casos de uso genérico. 20](#_Toc524057443)

[Ilustración 10. Diagrama de caso de uso de Administración. 21](#_Toc524057444)

[Ilustración 11. Diagrama de casos de uso de Eventos. 32](#_Toc524057445)

[Ilustración 12. Diagrama de casos de uso de Notas. 42](#_Toc524057446)

[Ilustración 13. Diagrama de casos de uso de Chat. 46](file:///E:\GoogleDrive\Informática\4-CUARTO\TFG\WeCalendarTFG\Memoria\Memoria.docx#_Toc524057447)

[Ilustración 14. Diagrama de clases. 54](#_Toc524057448)

[Ilustración 15. Arquitectura de la aplicación. 57](#_Toc524057449)

[Ilustración 16. Diagrama Entidad Relación. 58](#_Toc524057450)

[Ilustración 17. Mockup de registro de usuario. 59](#_Toc524057451)

[Ilustración 18. Mockup de inicio de sesión. 60](#_Toc524057452)

[Ilustración 19. Mockup del calendario. 60](#_Toc524057453)

[Ilustración 20. Mockup de Notas. 61](#_Toc524057454)

[Ilustración 21. Mockup de la sala de chat. 61](#_Toc524057455)

[Ilustración 22. Mockup del buscador de eventos. 62](#_Toc524057456)

[Ilustración 23. Arquitectura en N-Capas con orientación al Dominio, pág. 64. 63](#_Toc524057457)

[Ilustración 24. Ficheros del proyecto en Visual Studio. 64](#_Toc524057458)

[Ilustración 25. Clases en Visual Studio pertenecientes a la base de datos. 64](#_Toc524057459)

[Ilustración 26. Diagrama generado por Visual Studio. 65](#_Toc524057460)

[Ilustración 27. Configuración del Back-End (loaclhost:11111) 70](#_Toc524057461)

[Ilustración 28. Configuración del Front-End (localhost:44444) 70](#_Toc524057462)

[Ilustración 33. Pantalla de registro de usuario. 75](file:///E:\GoogleDrive\Informática\4-CUARTO\TFG\WeCalendarTFG\Memoria\Memoria.docx#_Toc524057463)

[Ilustración 34. Pantalla de inicio de sesión. 75](#_Toc524057464)

[Ilustración 35. Pantalla de Home. 75](#_Toc524057465)

[Ilustración 36. Pantalla de Notas. 76](#_Toc524057466)

[Ilustración 37. Pantalla de Usuario. 77](#_Toc524057467)

[Ilustración 38. Pantalla de Chat. 77](#_Toc524057468)

[Ilustración 39. Pantalla de Eventos. 78](#_Toc524057469)

[Ilustración 40. Ejemplo de buscador. 78](#_Toc524057470)

[Ilustración 41. Notificación de evento pendiente. 79](#_Toc524057471)

[Ilustración 42. Pantalla de eventos pendientes. 79](#_Toc524057472)

[Ilustración 43. Pantalla de Administración de Eventos. 80](#_Toc524057473)

[Ilustración 44. Pantalla de Administración de Usuarios. 81](#_Toc524057474)

**Tablas**

[Tabla 1. Comparativa de funcionalidades de aplicaciones de organización. 7](#_Toc523875268)

[Tabla 2. Comparativa de lenguajes para desarrollo Web 9](#_Toc523875269)

[Tabla 3. Tabla de objetivos. 76](#_Toc523875270)

**INTRODUCCIÓN**

**Palabras clave:** Front-End, Back-End, API Rest, Framework

Este proyecto consistirá en el desarrollo de una aplicación web que se encargará de ayudar a los usuarios a realizar una correcta organización de los eventos personales, tanto propios como comunes entre varias personas. En este trabajo se desarrollará una plataforma web responsiva mediante la cual los usuarios podrán llevar el control de sus actividades y eventos personales. La web permitirá la opción de definir nuevos eventos que podrán sincronizarse entre varios usuarios. La plataforma se encargará de registrar eventos, buscar huecos para asignación de eventos, notificaciones, etc… Entre otras funcionalidades, las cuales estarán centradas en proporcionar a los usuarios mayor comodidad a la hora de organizarse, de manera individual o en grupo.

**Keywords:** Front-End, Back-End, API Rest, Framework

This project will consist in the development of a web application that will be responsible for helping users to correctly organize personal events, both their own and common among several people. In this work we will develop a responsive web platform through which users can take control of their activities and personal events. The web will allow the option to define new events that can be synchronized between several users. The platform will be responsible for recording events, finding gaps to assign events, notifications, etc ... Among other features, which will focus on providing users with greater convenience when organizing, individually or in groups.

**MOTIVACIÓN**

Nuestra sociedad vive inmersa en una vorágine temporal, en el que los humanos de forma frenética tenemos que planificar miles de reuniones, actos y eventos con otros humanos en un tiempo limitado para así poder desarrollar nuestra actividad profesional y en muchos casos incluso el ocio. Hoy en día existen numerosas plataformas que tratan de ayudar a los usuarios a llevar a cabo la organización personal de sus actividades diarias tanto de trabajo como de ocio. En este tipo de aplicaciones las funcionalidades principales se orientan en facilitar la fijación y consulta de actividades. Sin embargo, aparecen ciertos problemas cuando las actividades de varias personas tienen que sincronizarse, por ejemplo, cuando se tiene que organizar un evento, una cita o una reunión.

En estos casos, las herramientas habituales planteadas para la organización personal como las agendas tienen un carácter individual, por lo que la organización de un evento entre varios asistentes obliga a que cada persona realice de forma independiente un registro del evento en su propio sistema. Por otra parte, la programación de recordatorios en dichos sistemas o bien abusan del envío de notificaciones para asegurar que el usuario es consciente del evento, o bien parten de la asunción del evento por parte del usuario. En cualquiera de los casos, el resultado final es que los organizadores de un evento con varios asistentes pueden tener problemas de sincronización que requieran un retraso de última hora o incluso una cancelación del evento.

Este proyecto surge con la idea de investigar por un lado en las soluciones que ofrece el mercado en este sentido, y por otra, tratar de desarrollar una nueva solución integradora que se adapte mucho más a las necesidades del usuario y sea más intuitivo.

la falta de organización por parte de los asistentes a un evento fuerza a los organizadores de los eventos a realizar modificaciones improvisadas de última hora que muchas veces afectan al transcurso normal del evento planeado. Estos errores de los que estoy hablando consisten en su mayoría de errores humanos, despistes o errores en la coordinación de las personas que a menudo suponen muchos contratiempos a la hora de organizarse.

Después de haber visto y vivido personalmente muchas de estas situaciones puedo decir que existen bastantes carencias a día de hoy en las aplicaciones de organización ya que, aunque existen muchas aplicaciones destinadas a esto, no llegan a cubrir todas las necesidades requeridas por los usuarios.

**OBJETIVOS**

El objetivo principal de este proyecto consiste en desarrollar una nueva plataforma que ayude a la organización de eventos y que sea capaz de mitigar el error humano a la hora de organizar eventos, citas, recordatorios... Esta plataforma se encargará por una parte de proporcionar una herramienta para la organización personal de los eventos a los que tiene que asistir un usuario, así como facilitar la difusión y coordinación de las personas que tienen que asistir a dichos eventos.

Para conseguir dicho objetivo se han establecido otros objetivos específicos:

* Estudiar y analizar las herramientas que hay en el mercado para la organización personal de eventos en los que varios participantes tienen que ponerse de acuerdo.
* Desarrollo de una nueva plataforma web enfocada a solventar la sincronización entre varios participantes de un evento de modo colaborativo.
* Aplicar buenas prácticas de diseño gráfico y usabilidad.
* Conocer y utilizar para el desarrollo de la plataforma web soluciones basadas en Microsoft como C# y servicios cloud como Azure como alternativa a otras soluciones basadas en PHP o Java.
* Aplicar buenas prácticas de ingeniería del software para llevar a cabo el proceso de desarrollo del software.
* Desarrollo de una solución multiplataforma.

**ESTRUCTURA**

A lo largo de este documento voy a explicar el proceso de desarrollo que he seguido explicando los análisis previos, diseño, implementación… en los siguientes capítulos.

* **Capítulo 1: Análisis de los sistemas de organización personal.** En este capítulo explicaré el análisis realizado a las demás plataformas de organización que se encuentran en el mercado.
* **Capítulo 2: Tecnologías web empleadas.** Aquí realizaré un análisis de las tecnologías web que estaban a mi disposición para el desarrollo del proyecto y analizaré las ventajas y desventajas para argumentar mi decisión final.
* **Capítulo 3: Gestión del Proyecto.** En este capítulo hablaré de mi organización seguida a lo largo de este proyecto mostrando el ciclo de vida y la metodología empleada.
* **Capítulo 4: Especificación de requisitos.** Aquí analizaremos los requisitos que deberá cumplir la aplicación y su funcionalidad.
* **Capítulo 5: Análisis.** En este apartado hablare de la extracción de clases dentro del proyecto y como deben de ser las relaciones entre ellas para que la aplicación funcione correctamente.
* **Capítulo 6: Diseño.** En este capítulo mostraré como he decidido diseñar la aplicación tanto a nivel estético como a nivel de arquitectura.
* **Capítulo 7: Implementación.** Aquí voy a hablar de las medidas y las pautas llevadas a cabo a la hora de implementar y desarrollar la aplicación.
* **Capítulo 8: Conclusiones.** Finalmente, en este capítulo se hablará de las conclusiones generales del proyecto y el nivel de satisfacción con los logros obtenidos.

**CAPÍTULO 1**

# **ANÁLISIS DE LOS SISTEMAS DE ORGANIZACIÓN PERSONAL**

Como se ha mencionado anteriormente, sabemos que existen diversas aplicaciones que ya se encargan de ayudar a los usuarios a realizar una organización de sus eventos. Estas aplicaciones pueden dividirse en dos categorías, primero tenemos organizadores personales basados en agenda, los cuales son individuales, cosa que dificulta la colaboración entre usuarios, y, por otra parte, tenemos las aplicaciones basadas en web, que a pesar de tener centralizada la información, en ocasiones puede fallar en la creación de los eventos o en su consulta.

Tenemos a nuestra disposición muchísimas aplicaciones en nuestros teléfonos móviles o en cualquier dispositivo electrónico que tengamos, que se encargan de la organización de notas y recordatorios. Aquí podemos ver el caso de las notas en los dispositivos Android o los recordatorios en IOS. A su vez, tras un breve vistazo por cualquier tienda de aplicaciones podemos ver infinidad de agendas personales. Estos dispositivos se sumarian a la lista de los organizadores personales basados en agenda, los cuales son los encargados de una organización individual.

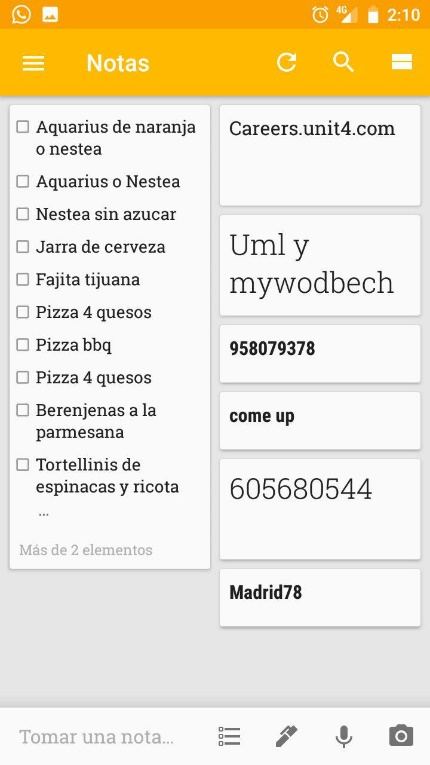
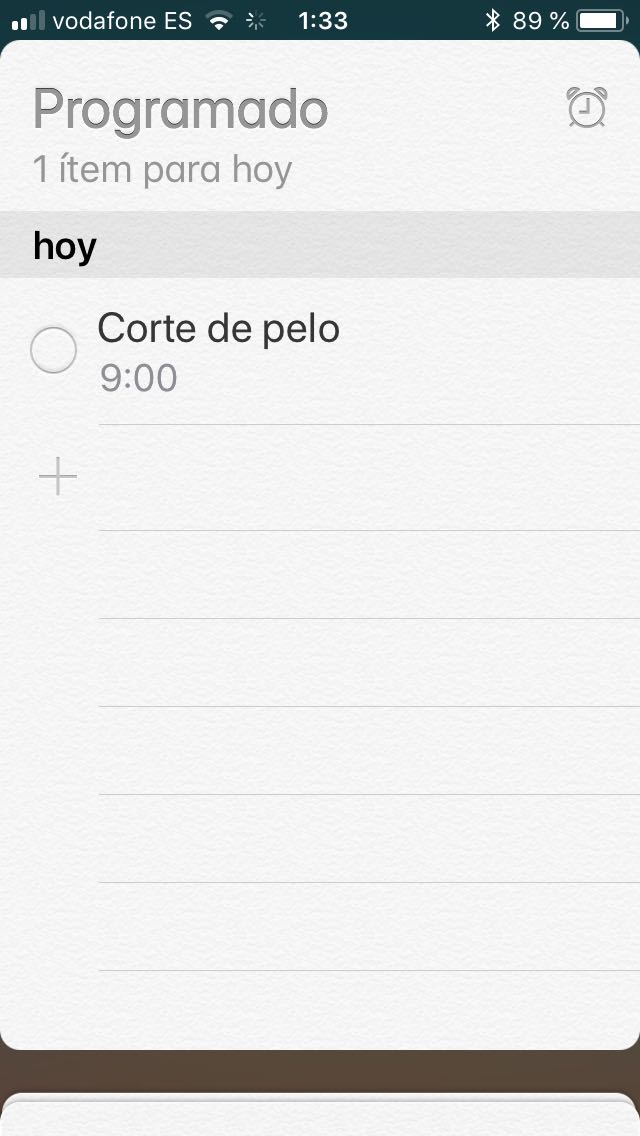


Ilustración 1. Notas en IOS.

Ilustración 2. Notas en Android.

Como se puede apreciar en las dos ilustraciones anteriores podemos ver el funcionamiento de las notas dentro de las aplicaciones nativas del teléfono. En el caso de Android podemos ver que las notas tienen una estética más parecida a las notas adhesivas ya que estas adoptan diferentes tamaños en función del tamaño de la nota que se está tratando. Esto contrasta bastante con las notas en IOS y su estética en modo tabla.

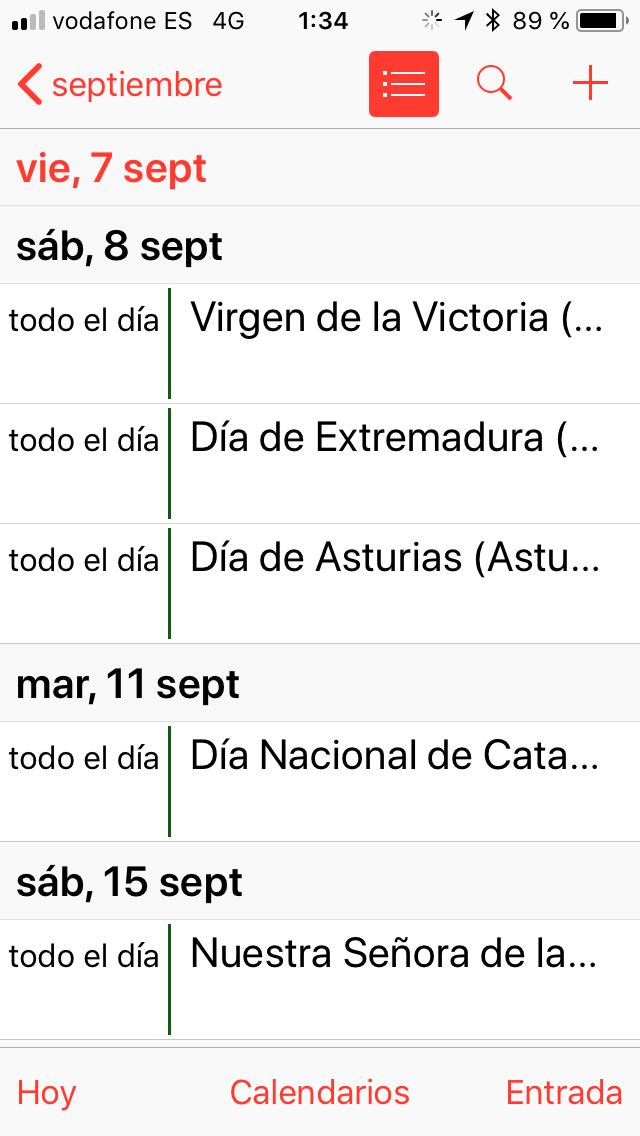


Ilustración 4. Calendario de Agenda Escolar.

Ilustración 3. Calendario de IOS.

Aquí podemos ver las diferencias entre los calendarios en IOS, y en Android. La estética del calendario de Android se basa en una vista general de los eventos mensuales, mostrando debajo una lista de los eventos específicos de cada día. Mientras que en la aplicación de IOS se puede ver una lista ordenada de los eventos que se van a encontrar en los próximos días.

La siguiente tanda de aplicaciones que denominamos basadas en web. De estas aplicaciones podemos destacar dos, primero tenemos Google Calendar, la cual se basa en un calendario personal con la capacidad de compartir eventos entre usuario y Doodle, que consiste en una aplicación capaz de gestionar y coordinar un evento entre varios usuarios para desarrollar correctamente una organización en grupo.

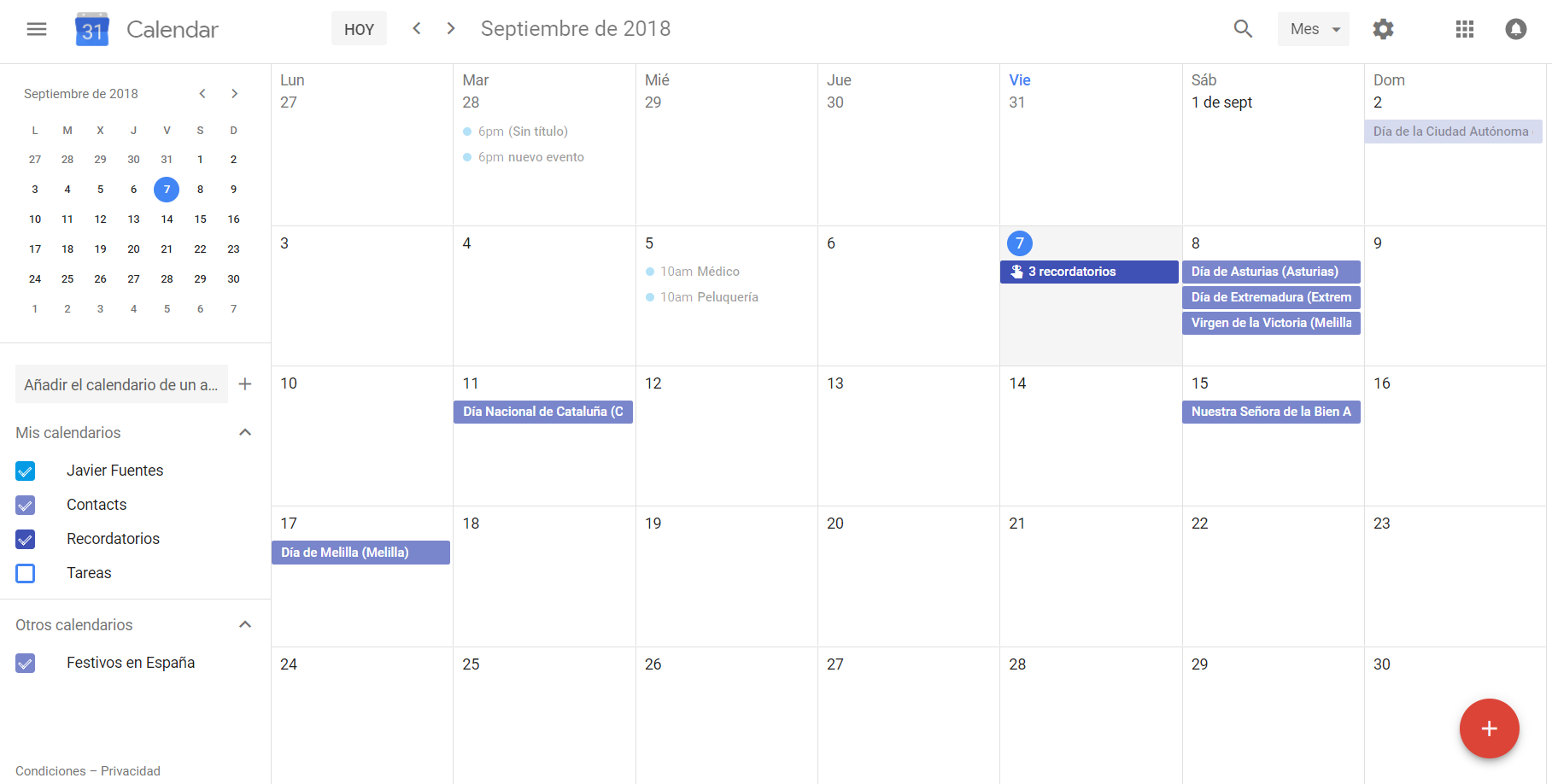


Ilustración 5. Pantalla de Google Calendar.

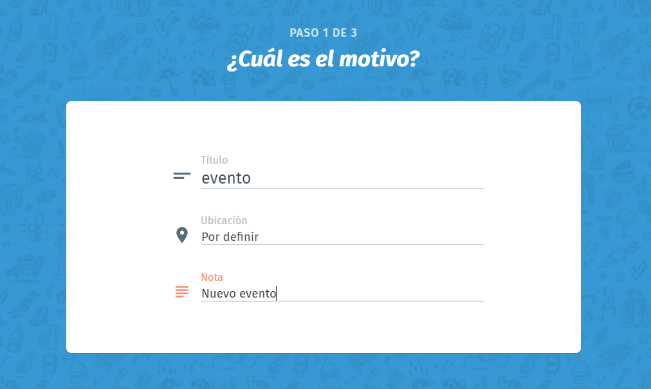


Ilustración 6. Gestión de Doodle.

Como puede verse en las ilustraciones anteriores ambas plataformas disponen de un calendario para la organización de los eventos. La diferencia aquí radica en la persistencia del calendario, ya que Google mantiene una persistencia de los eventos mientras que Doodle se centra en el calendario para el evento que se está tratando de organizar.

Estas aplicaciones a pesar de ser muy conocidas y utilizadas tienen algunas carencias, las cuales vamos a tratar de solventar en este proyecto. Carencias como por ejemplo en la conocida plataforma Google Calendar. Pese a ser una aplicación muy útil y conocida, esta permite la inserción de eventos en el mismo día y hora, pudiendo llevar al usuario final a una confusión a la hora de organizarse. Otro ejemplo, aunque en este caso no se trata de una aplicación de organización, tenemos Telegram. Esta aplicación de mensajería permite anclar un único mensaje en la sala de chat para que el resto de los usuarios puedan verlo permanentemente a modo de recordatorio. El problema viene cuando un usuario desee poner varios recordatorios en el grupo quedando o bien, un mensaje extremadamente largo, o bien dejando que los demás mensajes se pierdan. Teniendo esto en cuenta podemos optimizar nuestra sala de chat para que se evite este problema.

De las aplicaciones que nos podemos encontrar actualmente en el mercado que se encarguen de la organización personal o de grupos podemos destacar Google Calendar (1), Doodle (2) y aplicaciones de agendas de las tiendas de aplicaciones de los dispositivos móviles., en mi caso he analizado Agenda Escolar. Todas ellas son aplicaciones centradas en la organización personal que tratan de coordinar a los usuarios que la emplean para gestionar su día a día. Vamos a ver los servicios que ofrecen dichas aplicaciones y así poder apreciar en que factores podemos mejorarlas u ofrecer servicios que ellas no ofrecen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Características | Google Calendar | Agenda Escolar | Doodle | Específicas del teléfono |
| Notificaciones | Si | Si | Si | Si |
| Multiplataforma | Si | No | Si | No |
| Calendario | Si | Si | Si | Si |
| Persistencia de horario | Si | Si | No | Si |
| Compartir eventos | Si | No | Si | No |
| Notas | No | SI | No | Si |
| Mensajes | No | No | Si | Si |
| Recordatorios | Si | Si | Si | SI |
| Grupos | Si | No | Si | No |
| Notas compartidas | No | No | No | No |

Tabla 1. Comparativa de funcionalidades de aplicaciones de organización.

Como se puede apreciar en la tabla lo que vamos a intentar es realizar una combinación de las funcionalidades de las diversas plataformas y a su vez tratando de suplir las carencias que cada una de estas aplicaciones tiene. Por ejemplo. Una de las carencias comunes es la falta de notas compartidas, carencia que nosotros vamos a tratar de solucionar. También se puede ver que una de las aplicaciones no es multiplataforma, y eso es uno de los mayores errores para este tipo de aplicación. También cogeremos para nuestro proyecto todo lo que estas aplicaciones tienen en común como es el calendario, las notificaciones y los recordatorios. Necesitamos implementar todas estas funcionalidades si deseamos que nuestra aplicación pueda llegar a competir con las aplicaciones del mercado.

**CAPÍTULO 2**

# **TECNOLOGÍAS WEB EMPLEADAS**

Como bien se ha planteado en los objetivos, queremos que esta aplicación se trate de una aplicación multiplataforma. Debido a esto creo que la opción más conveniente para poder lograr este objetivo debe ser una aplicación web responsive para que si sea accesible para el máximo número de usuarios. Existen varios frameworks para el desarrollo web. Por una parte, tenemos el desarrollo en PHP, Python con frameworks como Django y C# que es la herramienta que he decidido utilizar.

La decisión de la tecnología es una de las decisiones más importantes a la hora de realizar un proyecto, y en la actualidad existen muchas posibles frameworks basados en lenguajes como php, java o javascript que proporcionan flexibilidad para el desarrollo web así como facilitar la implementación de aspectos importantes como la seguridad, el acceso concurrente, etc.

Aunque un objetivo que se propuso para este proyecto fue el utilizar concretamente soluciones basadas en Microsoft para conocer y profundizar en las soluciones que tiene disponibles, lo hemos comparado con otros posibles lenguajes para conocer las ventajas y desventajas que ofrecen.

|  |  |  |
| --- | --- | --- |
| Lenguaje | Pros | Contras |
| PHP | * Numerosos frameworks. * Una comunidad muy grande. * Compatible con prácticamente todos los hostings. | * Lento en comparación con el resto. * Más código para realizar la misma tarea que los demás. |
| Python | * Muy rápido. * Fácil de desarrollar. * Gestión interna de paquetes. | * Pocos hostings compatibles. * Necesidad de un framework para desarrollo web. |
| C# | * Mejor procesamiento paralelo * Totalmente orientado a objetos * Uso de Access incluido para el acceso a las bases de datos. | * Más lento que PHP * Compilación necesaria en el lado del servidor |

Tabla 2. Comparativa de lenguajes para desarrollo Web

La elección basada en C# proporciona facilidad para la generación de las clases, el tratamiento de los elementos de la base de datos y la integración de conexiones con los servidores de Microsoft. A parte, gracias al uso de Microsoft Visual Studio puedo emplear una estructura Modelo-Vista-Controlador de una manera mucho más sencilla y natural en el proyecto gracias a esta herramienta.

También hay que resaltar el hecho de que C# se encuentre compilado en el lado del servidor. Aunque esto puede ser una de las desventajas que tiene, ya que ralentiza el trabajo, supone un aumento de la seguridad, ya que lo que se almacena en el servidor es el ejecutable del proyecto.

## **2.1 HERRAMIENTAS CLOUD DE MICROSOFT (AZURE)**

De entre las herramientas que se van a utilizar para este proyecto hay que resaltar la herramienta de Azure. Obviamente, el hecho de desarrollar una aplicación web implica que debemos de tener un sitio en el cual alojar tanto la base de datos como la web en sí. En este caso al trabajar con C# y herramientas de Microsoft debemos tener un servidor con un Sistema Operativo Windows para que se pueda compilar y ejecutar correctamente en este.

Para obtener acceso a los servicios de Azure es necesario crearse una cuenta en el portal de Azure y dentro de este ya podremos crear los recursos necesarios para poder trabajar.



Ilustración 7. Captura del panel de Azure.

Aquí podemos ver una captura de la pantalla que nos encontramos en el panel de Azure. Como se puede observar tenemos varios recursos ya creados, el primero se trata del recurso donde se almacenará la base de datos de SQL con el nombre de *TFGDatabase*. El segundo recurso que observamos es el servidor de SQL, *wecalendar.* En el tercer recurso almacenaremos el Back-End de la web, *WCBackend,* y en el último se encontrará el Front-End, *wecalendar*.

El hecho de trabajar con la herramienta Visual Studio implica que este servicio de Azure ya se encuentra integrado dentro de esta, permitiéndome hacer subidas al servidor directamente desde esta aplicación seleccionando la dirección que desee.

Durante la mayor parte del desarrollo de la aplicación todas las pruebas serán ejecutadas en local, dándole uso a Azure para almacenar la base de datos y poder realizar los accesos. Una vez finalizada la fase de producción tanto el Back-End como el Front-End serán subidas al servidor de Azure.

Existen bastantes tecnologías de almacenamiento que podrían haberse usado como alternativa a Azure, pero en este caso se ha decidido emplear esta por su integración nativa con visual Studio y por el hecho de que uno de los objetivos planteados a priori ha sido el uso de tecnologías de Microsoft.

**CAPÍTULO 3**

# **GESTIÓN DEL PROYECTO**

Para el desarrollo del proyecto se ha seguido una metodología SCRUM con Sprints semanales con un ciclo de vida iterativo e incremental. La base de esta organización ha sido la división del proyecto en tareas sencillas o de corta duración, fechando al final de estas una reunión con el tutor para evaluar el estado o porcentaje de realización de la tarea. Por otra parte para el progreso del proyecto el ciclo de vida iterativo e incremental nos permite corregir e incrementar el trabajo realizado en cada una de las interacciones.

Cada uno de los Sprints deben ir acompañados de la documentación específica generada como consecución de dicha tarea que pueden incluir documentación escrita, modelos, artefactos e incluso código.

Para la organización de los Sprints y las reuniones con el tutor se ha utilizado la herramienta de Redmine con la cual estructuramos los objetivos, aquí llamados *peticiones,* anotando lo que se va a realizar, las horas estimadas de cada sprint y los documentos que se generaran en dicho sprint, así como la fecha tope en la cual se debe de tener terminado. Una vez terminada una petición podemos añadir las horas reales empleadas, y el porcentaje de éxito de esta tarea. También se anotarán las cosas que no han sido terminadas o deben ser corregidas para tenerlas en cuenta para el siguiente Sprint.

Para facilitar la corrección y comunicación con el tutor se me ha proporcionado también un servidor de SVN para poder almacenar ahí mi proyecto. En este servidor almacenare el proyecto dividiendo las carpetas en *Documentación*, *Bibliografía* y *Código* *Fuente*. Esta división de carpetas favorece el trabajo mediante Sprints y ayuda a la posterior realización de la memoria y control de errores, ya que se van guardando aplicaciones independientes de cada interacción.

Finalmente ha sido necesaria una gestión de código por parte del alumno la cual ha requerido de la plataforma GitHub. Este requerimiento viene dado a la facilidad para almacenar el código generado en la herramienta de trabajo unido a la necesidad de una herramienta de control de versiones.

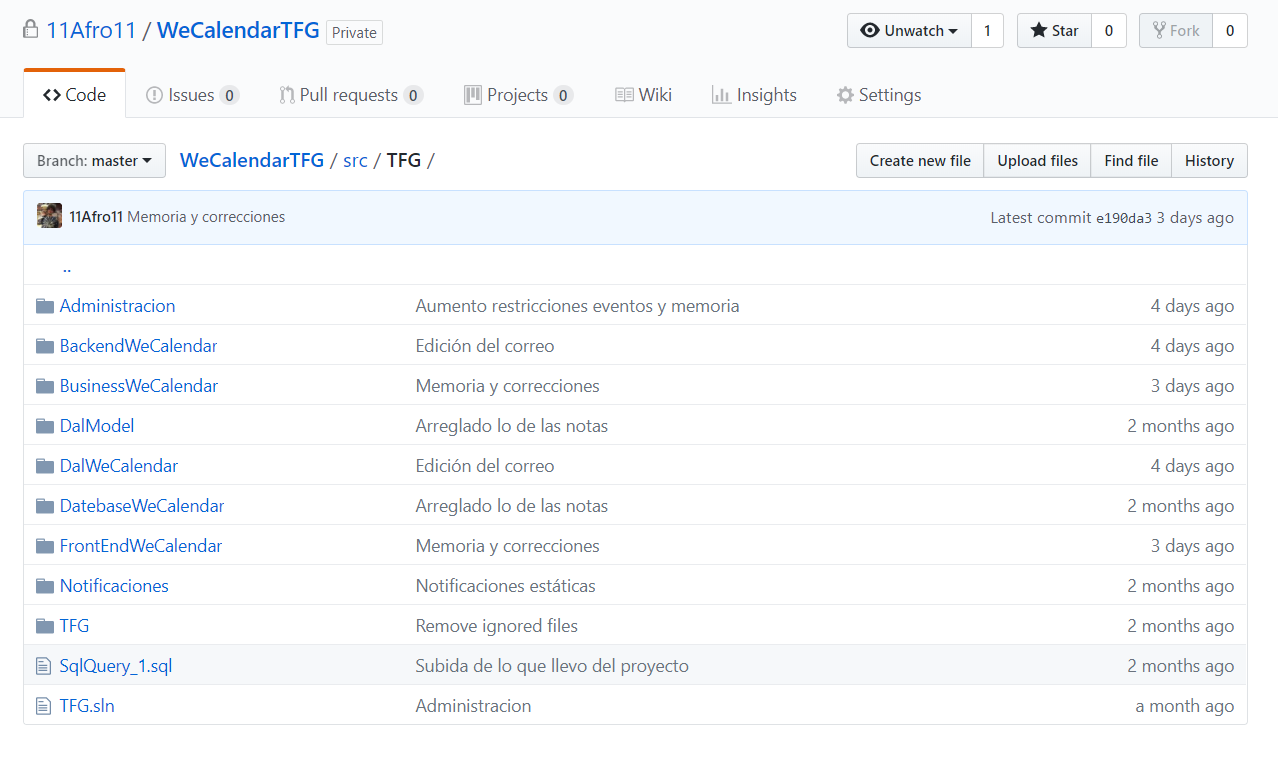


Ilustración 8. Repositorio de GitHub.

**CAPÍTULO 4**

# **ESPECIFICACIÓN DE REQUISITOS**

## **4.1 ESPECIFICACIÓN INFORMAL**

Fundamentalmente se quiere desarrollar un novedoso sistema de organización personal que facilite la creación, gestión, difusión y consulta de eventos como citas, reuniones, o seminarios en el que se tenga que organizar y sincronizarse un conjunto de personas. Para ello, se toma la base de un calendario en el que se pueden definir y compartir eventos de forma sencilla entre varios usuarios.

Este sistema contará primero con un calendario personal en el cual el usuario podrá anotar sus eventos personales. Dichos eventos podrán ser enviados a otros usuarios del sistema que tengamos agregados como amigos. Cuando un usuario reciba un evento de un amigo tendrá la opción de aceptarlo o rechazarlo quedando, en el primer caso agregado también a su calendario personal. Los eventos generados por los usuarios podrán ser privados o públicos por los que los usuarios también podrán buscar eventos públicos para añadirlos a su calendario si desean asistir.

También se ha dispuesto de un sistema de notas con las cuales el usuario podrá anotar recordatorios que no requieran de una fecha y una hora concreta. Junto a este sistema de notas tendremos también una sala de chat por grupo en la cual los usuarios podrán hablar para acordar más detalles de los eventos. Cada grupo vendrá incorporado con un tablón de notas mediante el cual los usuarios podrán compartir notas a todos los miembros del grupo.

Cada usuario tendrá en su perfil la opción de recibir notificaciones. Estas notificaciones hacen referencia a los eventos que el usuario tendrá hoy. Las notificaciones se aplicarán a los eventos que el usuario anote como prioritarios. Este sistema de notificaciones puede ser desactivado en cualquier momento por el usuario.

Finalmente, como todo sistema web con interacción de usuarios debe de tener una serie de administradores. En este caso los administradores serán los encargados de banear a los usuarios que realicen acciones poco éticas en la página, así como de borrar los eventos peligrosos u ofensivos del sistema.

## **4.2 MODELO FUNCIONAL**

Basándonos en la especificación anterior podemos identificar a 5 actores dentro de este sistema. El **Usuario** identificado con un nombre de usuario y un correo, un **Administrador** el cual se va a encargar de controlar el uso que los usuarios dan a la aplicación, identificado por el DNI y el correo, el **Evento**, que almacenará los datos de hora lugar y descripción, la **Nota**, que se encargará de los recordatorios y el **Grupo,** que se encargará de establecer comunicación entre uno o varios usuarios.

Teniendo estos actores en cuanta, podemos hacer una división de los casos de uso en función de su temática. En este caso la división seria: **Administración** para los perfiles de usuario y administradores, **Eventos,** para la gestión y coordinación de los eventos, **Notas**, para la gestión de los recordatorios y **Chat** para los mensajes y los grupos.



Ilustración 9. Diagrama de casos de uso genérico.

En esta ilustración se pueden ver las relaciones entre los actores y los 4 componentes principales de la página. El usuario al ser al actor principal es el que se encargará de interactuar con todas las componentes.

Ahora vamos a proceder a analizar cada uno de los elementos del modelo de forma independiente para poder ver en ellos los actores que intervienen y los casos de uso individuales de cada uno.

Administración

En esta división nos encontramos todo lo referente a la administración de la aplicación, desde la que sigue el usuario que interactúa con ella al registrarse y ver y editar su perfil, hasta las acciones que toma un administrador cuando revisa los eventos y usuarios del sistema.



Ilustración 10. Diagrama de caso de uso de Administración.

Como se puede ver en el diagrama en estos casos de uso intervienen los dos usuarios que tendremos en el sistema, tanto el usuario registrado como el usuario administrador.

Como se ha mencionado anteriormente estos casos de uso corresponden con la administración, por parte del usuario para los registros modificar datos y activar o desactivar las notificaciones, y por parte de los administradores hacen referencia al sistema de baneos del usuario y la cancelación de los eventos. Esta es una parte muy importante de este sistema ya que es necesario proporcionarles a los usuarios un buen sistema para manejar su cuenta, ya sea para evitar notificaciones y los inconvenientes que puedan surgir de estas como para cambiar el correo de la cuenta en caso de desear recibir las notificaciones en una cuenta de correo distinta a la proporcionada. A continuación, procederé a explicar los casos de uso de este subsistema de forma más detallada.

Casos de Uso de Administración

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Registrar Usuario | CU-01 | | **Actores** | Usuario |  | | **Precondición** |  |  | | **Postcondición** | El usuario queda registrado en el sistema |  | | **Autor** | Javier Fuentes Barragán | V-1.0 |      |  | | --- | | **Propósito** | | Que el usuario que inicia el proceso quede registrado en el sistema |  |  | | --- | | **Resumen** | | Al principio un usuario cuando entre en el sistema se podrá registrar en el para poder tener acceso a las funcionalidades del mismo. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la plataforma para registrarse | | 2 | El sistema ofrece el formulario de registro que contiene:  -nombre de usuario  -nombre.  -apellidos.  -correo.  -contraseña. | | 3 | El usuario rellena la información personal necesaria | | 4 | El sistema comprueba que la información del usuario es correcta y procede con el registro. | | 5 | El sistema notifica el estado de su inscripción. |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4.a | Si la información no es correcta, el sistema muestra un mensaje de error. | | 4.a.1 | El sistema vuelve a 2. | | 4.b | Si muestra el mensaje de error 3 veces cancela el caso de uso. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Añadir amigo | CU-02 | | **Actores** | Usuario |  | | **Precondición** | El usuario debe de estar registrado en el sistema |  | | **Postcondición** | Se creó un enlace entre los dos usuarios |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Establecer un enlace de amistad entre los dos usuarios que se encuentren registrados en el sistema |  |  | | --- | | **Resumen** | | El usuario podrá buscar amigos en el sistema para poder realizar las invitaciones de los eventos. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de añadir amigo e introduce el nombre de usuario | | 2 | El sistema muestra las coincidencias | | 3 | El usuario selecciona la coincidencia que desee | | 4 | El sistema envía la petición |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Eliminar a un amigo | CU-03 | | **Actores** | Usuario |  | | **Precondición** | Ambos usuarios deben de ser amigos |  | | **Postcondición** | Los usuarios dejaron de ser amigos |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Eliminar el enlace de amistad que une a dos usuarios |  |  | | --- | | **Resumen** | | Podemos borrar usuarios de nuestra lista de amigos. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a su lista de amigos y selecciona el usuario que desea eliminar | | 2 | El sistema lo elimina el enlace entre estos dos usuarios |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Ver lista de usuarios | CU-4 | | **Actores** | Administrador |  | | **Precondición** | El administrador debe de estar identificado |  | | **Postcondición** | Se mostró una lista de los usuarios registrados |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar al administrador una lista de los usuarios registrados en el sistema |  |  | | --- | | **Resumen** | | El administrador podrá acceder a una pestaña que le mostrará una lista de los usuarios que se encuentren registrados en el sistema. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El administrador accede a la pestaña de visualización de los usuarios | | 2 | El sistema muestra la lista de los usuarios registrados con las opciones disponibles para el administrador:   * Ver. * Banear. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Ver eventos públicos | CU-5 | | **Actores** | Administradores |  | | **Precondición** | El administrador debe estar registrado en el sistema |  | | **Postcondición** | Se mostró una lista de los eventos públicos |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Cualquier administrador podrá ver la lista de los eventos públicos |  |  | | --- | | **Resumen** | | Los administradores accederán a la pestaña de eventos públicos para ver la disposición de los eventos que se pueden encontrar en el sistema |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El administrador accede a la pestaña de los eventos públicos | | 2 | El sistema muestra los eventos públicos disponibles con las opciones para el administrador:   * Ver. * Suspender. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Banear Usuario | CU-6 | | **Actores** | Administrador |  | | **Precondición** | El administrador deberá estar registrado en el sistema |  | | **Postcondición** | Un usuario quedó eliminado del sistema |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Dar de baja a un usuario del sistema |  |  | | --- | | **Resumen** | | Los administradores podrán borrar a los usuarios que deseen del sistema |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El administrador accede a la pestaña de usuarios | | 2 | El sistema muestra los usuarios al administrador | | 3 | El administrador selecciona el usuario que desee y presiona el botón de eliminar | | 4 | El sistema elimina al usuario seleccionado. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Cancelar un evento | CU-7 | | **Actores** | Administrador |  | | **Precondición** | Debe de existir el evento y el administrador debe de estar registrado |  | | **Postcondición** | El evento dejó de encontrarse en el sistema |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Un administrador puede cancelar un evento si este no reúne las condiciones convenientes |  |  | | --- | | **Resumen** | | Cuando un administrador se dé cuenta de que algún evento no debe ser celebrado o que no reúne las condiciones necesarias, este podrá cancelar dicho evento. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El administrador accede al evento que desea borrar | | 2 | El sistema muestra el evento con la opción de cancelar | | 3 | El administrador confirma la cancelación del evento | | 4 | El sistema elimina el evento. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Buscar usuario | CU-8 | | **Actores** | Administradores |  | | **Precondición** | El Administrador debe de encontrarse registrado en el sistema |  | | **Postcondición** | Se muestra una lista de usuarios |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Encontrar a los usuarios del sistema |  |  | | --- | | **Resumen** | | Los Administradores podrán buscar a otros usuarios dentro del sistema. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El Administrador accede a la pestaña de usuarios | | 2 | El sistema muestra los usuarios junto con una barra de búsqueda | | 3 | El Admministrador introduce el criterio de búsqueda | | 4 | El sistema muestra la lista de los usuarios que cumplen con el criterio |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4.1 | Si no hay coincidencias el sistema mostrará un mensaje indicándolo. | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Modificar datos de usuario | CU-9 | | **Actores** | Usuario |  | | **Precondición** | El usuario debe de estar registrado en el sistema |  | | **Postcondición** | Los datos del usuario se modificaron |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar una herramienta para modificar los datos de usuario |  |  | | --- | | **Resumen** | | El usuario podrá editar su información personal en cualquier momento correo y foto de perfil. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de edición | | 2 | El sistema muestra los campos editables:  -correo.  -foto de perfil. | | 3 | El usuario introduce la información nueva | | 4 | El sistema guarda los cambios del usuario |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4.1 | Los campos introducidos son erróneos | |  | 4.1.2 El sistema vuelve a pedir la información al usuario | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Silenciar notificaciones | CU-10 | | **Actores** | Usuario |  | | **Precondición** | El usuario debe de estar registrado en el sistema |  | | **Postcondición** | Las notificaciones quedaron desactivadas |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Silenciar las notificaciones |  |  | | --- | | **Resumen** | | El usuario podrá desactivar las notificaciones de los recordatorios que tenga diarios. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de notificaciones y marca el icono de silencio | | 2 | El sistema desactiva las notificaciones |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Ver lista de baneados | CU-11 | | **Actores** | Administrador |  | | **Precondición** | El administrador debe de estar registrado en el sistema |  | | **Postcondición** | Se mostró una lista de los usuarios baneados por el administrador |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Mostrar la lista de los usuarios baneados por el administrador |  |  | | --- | | **Resumen** | | El administrador podrá ver una lista de los usuarios que han sido baneados por el. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la sección de usuarios baneados | | 2 | El sistema muestra la lista de los usuarios baneados con la opción de retirar baneo. |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 2.1 | Si no hay usuarios el sistema muestra el mensaje correspondiente. | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Retirar Baneo | CU-12 | | **Actores** | Administrador |  | | **Precondición** | El administrador debe de estar registrado en el sistema y el usuario debe de encontrarse baneado |  | | **Postcondición** | El usuario dejó de estar baneado |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Poder devolverle el acceso al sistema a un usuario |  |  | | --- | | **Resumen** | | El administrador podrá retirar el baneo a un usuario de su lista de baneados |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El administrador presiona el botón de retirar baneo | | 2 | El sistema devuelve el acceso a la plataforma al usuario y envía una notificación. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Ver lista de eventos suspendidos | CU-13 | | **Actores** | Administrador |  | | **Precondición** | El administrador debe de estar registrado en el sistema |  | | **Postcondición** | Se mostró una lista de los eventos suspendidos. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Ver la lista de los eventos suspendidos |  |  | | --- | | **Resumen** | | El administrador podrá ver una lista de los eventos que han sido suspendidos por él. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la sección de eventos suspendidos | | 2 | El sistema muestra la lista de los eventos suspendidos. |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 2.1 | Si no hay usuarios el sistema muestra el mensaje correspondiente. | |  |  | |

**Eventos**

Aquí nos encontramos todas las funcionalidades que hacen referencia a la gestión y el mantenimiento de los eventos, desde su creación hasta su eliminación pasando por la edición y el mecanismo de compartir eventos. También tenemos en este ámbito la organización del calendario el cual en nuestra pantalla se irá llenando de eventos de la manera más cómoda posible para que puedan ser examinados por el usuario.

****

Ilustración 11. Diagrama de casos de uso de Eventos.

Como se puede ver, los dos actores que intervienen en este conjunto son el usuario, el cual llevará a cabo la mayoría de las acciones del sistema y los eventos. Este paquete es uno de los más centrales del sistema ya que es el encargado de llevar la organización más exacta e importante, la gestión de eventos, los cuales serán los que posteriormente serán notificados.

Casos de Uso de Eventos

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Crear un evento | CU-14 | | **Actores** | Usuario y Evento |  | | **Precondición** | Usuario registrado en el sistema |  | | **Postcondición** | Un evento nuevo fue añadido al sistema |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar una herramienta para la creación de eventos |  |  | | --- | | **Resumen** | | El usuario podrá crear un evento en una fecha y una hora a definir. El método de creación del evento puede ser mediante un formulario de recogida de datos o bien mediante el calendario. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la opción de creación de un evento, bien por el calendario o por la opción dentro del sistema | | 2 | El sistema muestra el formulario que el usuario deberá rellenar para la correcta creación del evento | | 3 | El usuario rellena los datos y acepta el formulario:  -nombre del evento.  -fecha.  -hora de inicio.  -h  -lugar (Opcional).  -descripción(Opcional). | | 4 | El sistema comprueba los datos y los añade a la base de datos creándose así un nuevo evento | | 5 | El sistema devuelve un mensaje con una confirmación de evento creado |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4.1 | Los datos introducidos contienen errores | |  | El sistema deberá de indicárselo al usuario para que este vuelva a introducir los datos | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Compartir un evento | CU-15 | | **Actores** | Usuario y Evento |  | | **Precondición** | Debe de existir un evento creado |  | | **Postcondición** | Para dicho evento se generó una lista de participantes |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar la herramienta necesaria para enviar los eventos al resto de usuarios |  |  | | --- | | **Resumen** | | Una vez creado el evento este puede compartirse con el resto de usuarios que se encuentren en el sistema. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario selecciona en el evento la opción de invitar | | 2 | El sistema muestra la lista de amigos del usuario | | 3 | El usuario elige de esa lista de amigos a los que desea invitar al evento y finalmente marca la opción enviar. | | 3 | El sistema envía a los usuarios seleccionados una invitación al evento |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Aceptar un evento | CU-16 | | **Actores** | Usuario y Evento |  | | **Precondición** | El usuario debe recibir una invitación |  | | **Postcondición** | El usuario aceptó o rechazó la invitación |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar la opción de ver los eventos a los cuales te han invitado y poder aceptarlos o rechazarlos |  |  | | --- | | **Resumen** | | Cuando un usuario es invitado a un evento se le desplegará su calendario con el evento superpuesto en él, para que así el usuario pueda elegir si lo acepta o lo rechaza |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a las notificaciones y accede también a la invitación | | 2 | El sistema muestra la invitación superpuesta en su calendario personal | | 3 | El usuario verá el calendario y realizará los cambios que desee pudiendo finalmente aceptar o rechazar el evento. Los cambios se realizarán en los eventos que había previamente en el calendario, en el caso de no ser un evento generado por el usuario únicamente podrá eliminar su asistencia a dicho evento. Los cambios supondrán la modificación de las fechas o las horas del evento. | | 4 | El sistema guardará los cambios en su calendario personal y añadirá al usuario a la lista de asistentes del evento |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4.1 | Si el sistema detecta incompatibilidades en el calendario se lo notifica al usuario y vuelve a 3. | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Modificar un evento | CU-17 | | **Actores** | Usuario y Sistema |  | | **Precondición** | Debe de existir un evento creado |  | | **Postcondición** | Se cambiaron algunos elementos del evento |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Suministrar una herramienta que permita a los usuarios editar un evento ya fijado |  |  | | --- | | **Resumen** | | El usuario podrá en cualquier momento modificar cualquier elemento del evento quedándose este almacenado en modificación y en el caso de que el evento tenga participantes estos recibirán una notificación indicando que el evento ha sido modificado. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de edición del evento | | 2 | El sistema muestra la información editable del evento:  -fecha.  -hora de inicio  -hora de fin | | 3 | El usuario cambia lo que vea necesario del evento | | 4 | El sistema guarda los cambios y envía una notificación de modificación del evento a los usuarios |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Borrar un evento | CU-18 | | **Actores** | Usuario y Evento |  | | **Precondición** | Debe de existir el evento |  | | **Postcondición** | El evento quedó borrado del sistema |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar la herramienta para poder eliminar eventos |  |  | | --- | | **Resumen** | | Un usuario podrá eliminar un evento ya creado y el sistema deberá de eliminarlo de su calendario y del de los demás usuario que deseen asistir a él, previa notificación. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede al evento que desea y lo elimina | | 2 | El sistema borra el evento del usuario y del resto de participantes y envía una notificación a los participantes al mismo |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Modificar planificación de eventos | CU-19 | | **Actores** | Usuario y Calendario |  | | **Precondición** | El usuario debe de estar registrado en el sistema |  | | **Postcondición** | El calendario pasó a ser distinto |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar la herramienta que nos permita modificar el calendario |  |  | | --- | | **Resumen** | | Cuando un usuario desee realizar cambio en el calendario ya sea por la invitación a un evento o bien por voluntad propia. El sistema proporcionará las herramientas necesarias y guardará el nuevo calendario. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede al calendario | | 2 | El usuario realiza cambios en el calendario. Los cambios del calendario se refieren a edición de los eventos que tenemos en el mismo. SI el evento es nuestro evento privado podemos moverlo a nuestro gusto, mientras que si el evento es de otra persona podremos borrar nuestra asistencia a él | | 3 | El sistema comprueba los cambios y guarda el nuevo calendario |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 3.1 | El sistema detecta incompatibilidades  El sistema notifica al usuario que la nueva configuración no es adecuada. | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Ver eventos públicos | CU-20 | | **Actores** | Usuario y Evento |  | | **Precondición** | El usuario debe estar registrado en el sistema |  | | **Postcondición** | Se mostró una lista de los eventos públicos |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Cualquier usuario podrá ver la lista de los eventos públicos |  |  | | --- | | **Resumen** | | Los usuarios accederán a la pestaña de eventos públicos para ver la disposición de los eventos que se pueden encontrar en el sistema |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de los eventos públicos | | 2 | El sistema muestra los eventos públicos disponibles con las opciones para el administrador:   * Ver. * Asistir. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Buscar eventos | CU-21 | | **Actores** | Usuarios y Eventos |  | | **Precondición** | Deben de encontrarse registrados en el sistema |  | | **Postcondición** | Se mostró una lista de eventos |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar un buscador de eventos |  |  | | --- | | **Resumen** | | Los usuarios podrán mirar en la lista de evento públicos y realizar búsquedas para encontrar el que deseen. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de eventos. | | 2 | El sistema muestra la barra de búsqueda. | | 3 | El usuario introduce las opciones de búsqueda. | | 4 | El sistema muestra las coincidencias. |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4.1 | Si no hay coincidencias el sistema muestra un mensaje indicándolo | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Ver lista de asistentes | CU-22 | | **Actores** | Usuario, evento y administrador |  | | **Precondición** | Debe de existir el evento y los dos usuarios estar registrados |  | | **Postcondición** | Se muestra una lista de asistentes |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Mostrar los asistentes a un evento |  |  | | --- | | **Resumen** | | Los usuarios del sistema podrá comprobar los usuarios que asisten al evento |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede al evento | | 2 | El sistema muestra la información del evento | | 3 | El usuario despliega la pestaña de asistentes | | 4 | El sistema muestra los asistentes al evento |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4.1 | No hay asistentes | |  | 4.1.1 El sistema muestra el mensaje correspondiente. | |

**Notas**

En esta sección hacemos referencia a todo lo que tenga que ver con los recordatorios, notas adhesivas que se encargaran de recordarnos tareas o eventos que no precisan de una fecha determinada. Aquí vamos a tener las interacciones relacionadas con crear las notas, borrarlas, ordenarlas por fecha y compartirlas, ya que en los chats vamos a tener la opción de compartir las notas que tengamos en nuestro sistema para que sean visibles por el resto de los usuarios dentro del grupo.

Ilustración 12. Diagrama de casos de uso de Notas.

Gracias a este sistema el usuario tendrá una manera mucho más sencilla de realizar sus tareas, y a su vez, la particularidad de permitir compartir las notas con el resto de usuarios abre un abanico más de posibilidades a la hora de realizar una organización entre las personas. Ya que en un mismo grupo varias personas tendrán acceso de la manera mas rápida a los mismos recordatorios pudiendo hacerse cargo de esa tarea la primera persona que se encuentre disponible o haciendo un recordatorio masivo en caso de que sea necesario que todos los miembros del grupo.

**Casos de uso de Notas**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Crear una nota | CU-23 | | **Actores** | Usuario y Nota |  | | **Precondición** | El usuario debe de estar registrado en el sistema. |  | | **Postcondición** | Una nota fue añadida a la pestaña de notas. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar la herramienta de organización de notas y recordatorios los cuales no requieres de fecha hora o lugar específico. |  |  | | --- | | **Resumen** | | Un usuario registrado podrá crear una nota con carácter personal dentro de la pestaña de notas del sistema a modo de recordatorio. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede al editor de notas | | 2 | El sistema suministra la pestaña de edición de las notas | | 3 | El usuario escribe la nota y finalmente pulsa el botón de guardar | | 4 | El sistema guarda la nota en la pestaña específica para las notas |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Borrar una nota | CU-24 | | **Actores** | Usuario y Nota |  | | **Precondición** | Debe de existir una nota en el sistema |  | | **Postcondición** | La nota quedó eliminada |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar una herramienta que nos permita borrar las notas. |  |  | | --- | | **Resumen** | | Los usuarios deben poder eliminar las notas que ya no le sean de utilidad |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la nota que desea borra y presiona el botón de eliminar | | 2 | El sistema borra la nota de la base de datos |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Compartir una nota | CU-25 | | **Actores** | Usuario y Nota |  | | **Precondición** | El usuario debe de estar registrado en el sistema y pertenecer a un grupo. |  | | **Postcondición** | Una nota es añadida al tablero del grupo. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Obtener un tablero de notas compartido para un grupo. |  |  | | --- | | **Resumen** | | Los usuarios pertenecientes a un grupo serán capaces de añadir notas al tablero del grupo. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la nota que desea compartir y presiona el botón para compartirla | | 2 | El sistema añade la nota al tablero de las notas compartidas. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Mostrar notas por orden | CU-26 | | **Actores** | Usuario y notas |  | | **Precondición** | El usuario debe de estar registrado en el sistema y deben de existir notas. |  | | **Postcondición** | Las notas se mostraron por orden de fecha de vencimiento. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Mostrar las notas según una fecha de vencimiento. |  |  | | --- | | **Resumen** | | El usuario podrá disponer de este tipo de orden a la hora de visualizar las notas. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario marca la casilla de ordenar por fecha de vencimiento. | |  |  |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

**Chat**

Aquí vemos toda la funcionalidad referente a la comunicación que el sistema proporcionará a los usuarios. En este sistema tendremos una organización de los grupos de usuarios pudiendo ver la lista de los grupos a los que pertenece el usuario y los mensajes enviados dentro de este grupo. Cada mensaje guardará la información referente al texto del mensaje, el usuario que lo ha mandado y la hora a la que ha sido enviado. En este sistema tendremos también un tablero donde se almacenarán las notas compartidas en ese grupo. Con las mismas opciones que tienen las notas dentro de nuestro sistema de notas.



Ilustración 13. Diagrama de casos de uso de Chat.

Gracias a este sistema los usuarios podrán estar en contacto para organizar mejor los eventos o simplemente establecer una línea de comunicación entre cierto sector de los usuarios. El proceso de las notas puede sustituir al anclado de mensajes que ya posee Telegram o WhatsApp, pero de una manera más eficiente y sin entorpecer la vista de los mensajes dentro del chat.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Enviar Mensaje | CU-27 | | **Actores** | Usuario y Grupo |  | | **Precondición** | El usuario debe de estar registrado en el sistema. |  | | **Postcondición** | Un mensaje es enviado de un usuario a otro. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Tener un servicio de mensajería interno |  |  | | --- | | **Resumen** | | En este caso vamos a proporcionar un sistema para poder enviar y recibir mensajes dentro de nuestra aplicación |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de edición del mensaje | | 2 | El sistema muestra el formulario en el que habrá:  -título.  -mensaje. | | 3 | El usuario rellena el mensaje y pulsa enviar. | | 4 | El sistema envía el mensaje. |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 4 | Si el mensaje no está completo el sistema se lo notificara al usuario y volverá a 3. | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Recibir Mensaje | CU-28 | | **Actores** | Usuario y Grupo |  | | **Precondición** | El usuario debe estar registrado en el sistema |  | | **Postcondición** |  |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Sistema de notificaciones de los mensajes |  |  | | --- | | **Resumen** | | Notificación de los mensajes internos del sistema |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El sistema muestra la notificación del mensaje. | |  |  |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Leer Mensaje | CU-29 | | **Actores** | Usuario y Grupo |  | | **Precondición** | El usuario debe de estar registrado en el sistema y debe de existir un mensaje. |  | | **Postcondición** | La notificación del mensaje ha desaparecido |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Mostrar al usuario el contenido del mensaje |  |  | | --- | | **Resumen** | | El sistema mostrará una pestaña con el contenido del mensaje |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede al mensaje | | 2 | El sistema muestra el contenido del mensaje | | 3 | El usuario sale de la pestaña de vista del mensaje | | 4 | El sistema borra la notificación del mensaje. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Crear grupo | CU-30 | | **Actores** | Usuario y Grupo |  | | **Precondición** | El usuario debe de estar registrado en el sistema. |  | | **Postcondición** | Un grupo es añadido al sistema con el usuario que lo ha generado como miembro. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Proporcionar una herramienta de creación de grupos |  |  | | --- | | **Resumen** | | Los usuarios podrán crear un grupo nuevo y añadir a sus amigos a él. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de generación de grupo. | | 2 | El sistema muestra el formulario de creación de grupo el cual constará de:  -Nombre.  -Descripción (opcional).  -Imagen(opcional). | | 3 | El usuario rellena los datos. | | 4 | El sistema muestra la lista de los amigos del usuario para añadirlos al grupo (mínimo debe de haber 2 usuarios contando el creador). | | 5 | El usuario rellena a los amigos y pulsa confirmar. | | 6 | El sistema crea el grupo e introduce a los usuarios a dicho grupo enviándoles una notificación. |  |  |  | | --- | --- | |  | **Cursos Alternos** | | 3.1 | SI algún campo obligatorio no ha sido rellenado el sistema lo muestra con un mensaje y no avanza hasta que se complete. | | 5.1 | Si no se cumple el mínimo de usuarios añadidos el sistema no permitirá avanzar. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Eliminar grupo | CU-31 | | **Actores** | Usuario y Grupo |  | | **Precondición** | El usuario debe de estar registrado en el sistema y debe de existir un grupo. |  | | **Postcondición** | El grupo quedó eliminado del sistema. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Poder borrar los grupos creados |  |  | | --- | | **Resumen** | | El usuario que creo el grupo podrá borrar este del sistema. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de eliminar grupo. | | 2 | El sistema muestra un mensaje de confirmación. | | 3 | El usuario marca la opción que desee. | | 4 | El sistema elimina el grupo. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Salir del grupo | CU-32 | | **Actores** | Usuario y Grupo |  | | **Precondición** | El usuario debe de encontrarse registrado en el sistema y estar dentro de un grupo. |  | | **Postcondición** | El usuario deja de estar en el grupo. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Salir de un grupo en el que el usuario estaba previamente. |  |  | | --- | | **Resumen** | | Debemos proporcionar una herramienta para que los usuarios puedan salir de grupos en los que no desean estar. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario accede a la pestaña de salir del grupo | | 2 | El sistema muestra un mensaje de confirmación. | | 3 | El usuario marca la opción que desee. | | 4 | El sistema elimina al usuario del grupo. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Añadir usuario a grupo | CU-33 | | **Actores** | Usuario y Grupo |  | | **Precondición** | El usuario debe de estar registrado y encontrarse dentro del grupo |  | | **Postcondición** | Otro usuario se añadió al grupo |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Añadir a más usuarios a un grupo. |  |  | | --- | | **Resumen** | | El usuario creador del grupo podrá añadir a más usuarios al grupo. |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El usuario entra al grupo y selecciona añadir a más. | | 2 | El sistema muestra la lista de usuarios amigos. | | 3 | El usuario selecciona a la persona a añadir. | | 4 | El sistema la añade. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

Finalmente hace falta mencionar otro caso de uso que no precisa de ningún actor del sistema, este caso de uso es el de enviar una notificación. Como se ha explicado anteriormente en esta aplicación contaremos con un sistema de notificaciones el cual será un proceso regulado por un temporizador que se encargará de enviar un correo notificando a los usuarios con notificaciones activadas y con eventos de prioridad alta en ese día.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Caso de uso** | Envío de recordatorios | CU-34 | | **Actores** | temporizador |  | | **Precondición** | Deben de existir usuarios registrados en el sistema con eventos. |  | | **Postcondición** | El sistema envió recordatorios de los eventos a los usuarios. |  | | **Autor** | Javier Fuentes | v-1.0 |      |  | | --- | | **Propósito** | | Notificar a los usuarios de los eventos que tengan ese mismo día |  |  | | --- | | **Resumen** | | El día del evento el usuario recibirá una notificación de los eventos que tiene programado para ese mismo día |  |  |  | | --- | --- | |  | **Curso Normal** | | 1 | El sistema envía la notificación a los usuarios en dicha franja horaria. |  |  |  | | --- | --- | |  | **Cursos Alternos** | |  |  | |  |  | |

Este proceso será la vía principal con la cual el sistema se comunicará con los usuarios fuera de la aplicación. Este sistema de notificaciones será activado diariamente a la 1 de la madrugada para asegurar que las molestias en los usuarios sean mínimas y a su vez garantizar que los usuarios reciban diariamente los recordatorios de las actividades que tengan programadas para cada día.

**CAPÍTULO 5**

# **ANÁLISIS**

Una vez que hemos visto los actores dentro del sistema debemos de extraer el modelo con estos datos. A partir de lo que ya tenemos vamos a proceder a diseñar las clases y las relaciones dentro del sistema para poder generar correctamente la base de datos definitiva del sistema. Las principales clases del sistema van a ser la clase **Usuario**, **Evento**, **Administrador**, **Nota**, **Grupo** y **Mensaje**.

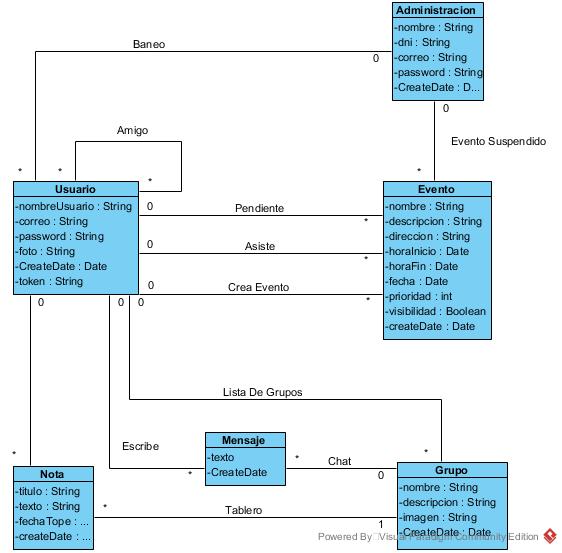


Ilustración 14. Diagrama de clases.

Para empezar, tenemos la clase **Usuario**, obviamente como se ha podido ver en los casos de uso, el usuario es el más central dentro de la aplicación. Dentro del usuario guardaremos los datos básicos tales como el nombre, correo, contraseña, la foto, la opción de si desea recibir o no notificaciones, la fecha en la que se registró y un token. El token será un atributo muy importante en cuanto a la seguridad del usuario, de la cual hablaremos posteriormente. Esta clase se relacionará con las demás clases del sistema de diversas formas:

* **Amistad (Usuario – Usuario):**  Dentro de este sistema encontraremos esta relación que nos informará de que dos usuarios del sistema tienen una relación de amistad, lo que les permitirá compartir eventos y añadirse a grupos.
* **Creado (Usuario –** **Evento):** Esta relación entre el usuario y el evento indica que usuario ha sido el responsable de crear el evento, para así conocer la procedencia del mismo.
* **Asistencia (Usuario –** **Evento):** En esta relación representaremos que usuarios asistirán a los eventos del sistema, llevando así una lista de los asistentes a dicho evento.
* **Pendiente (Usuario –** **Evento):** Aquí encontramos una lista de los eventos que han sido enviados como invitación a los demás usuarios y aún no han sido aceptados o rechazados por el usuario.
* **Crear Nota (Usuario –** **Nota):** Esta relación nos servirá para saber la pertenencia de una nota a un usuario.

La clase **Evento** se va a encargar de registrar datos del tipo fecha y hora. Los eventos tendrán almacenadas información referente al nombre, la descripción y la dirección del evento. Obviamente debemos guardar la fecha y horas de inicio y fin del evento, así como la prioridad que tiene el evento, para saber si ese evento deberá ser o no notificado y la visibilidad para que los demás usuarios puedan ver y buscar eventos públicos. Las relaciones de los eventos son las descritas anteriormente.

La clase **Nota** será la encargada de los recordatorios sin tiempo establecido, así que esta clase deberá de almacenar un título y un texto de nota y la fecha de creación. Las relaciones que se pueden encontrar en esta clase, a parte de las mencionadas anteriormente son:

* **Tablero (Grupo – Nota):** Esta relación se encargará de registrar las notas que está compartidas y en que grupos han sido compartidas.

La clase **Grupo** es la encargada de almacenar la relación que pueden tener los usuarios a la hora de entablar comunicaciones y compartir notas. En esta clase guardaremos un nombre una descripción, una imagen de perfil, y una fecha de creación. Las relaciones que se encuentran dentro de esta clase son:

* **Chat (Grupo – Mensaje):** Gracias a esta relación podemos almacenar todos los mensajes compartidos en un chat en concreto y distinguirlos del resto de los mensajes del sistema.
* **Lista de Grupos (Grupo –** **Mensaje):** Gracias a esto sabremos todos los grupos creados por un usuario o grupos en los que un usuario participa para mostrarlos y permitirles el acceso.

La clase **Mensaje** se trata de la clase más simple almacenada en el sistema, esta clase solamente almacenará un texto, y la fecha de envío. Las relaciones del mensaje son:

* **Enviado (Usuario – Mensaje):** Con esta relación almacenamos el usuario que se ha encargado de enviar el mensaje.

La clase **Administración** será la encargada de controlar las interacciones de los usuarios. De los administradores debemos almacenar el nombre, el DNI, el correo la contraseña y la fecha de registro. Las relaciones que se encuentran en esta clase son:

* **Baneo (Usuario –Administrador):**  Los administradores dentro del sistema tendrán la posibilidad de realizar baneos a los usuarios registrados en este, dicho baneo quedará registrado en el sistema impidiendo al usuario acceder a este.
* **Evento Suspendido (Administrador – Evento):** Los administradores del sistema serán capaces de suspender aquellos eventos que crean oportunos, así que en esta relación se almacenarán los eventos suspendidos por cada administrador para almacenar un histórico.

**CAPÍTULO 6**

# **DISEÑO**

## **6.1 ARQUITECTURA**

En este proyecto vamos a tener una división muy fuerte entre el cliente que accede a los recursos que proporciona una plataforma web. Por eso, se ha seguido una arquitectura de dos niveles en el que distinguimos un Front-End y el Back-End de la aplicación. En este caso como Front-End tendremos la aplicación cliente que se encargará de conectarse a la plataforma y tener acceso a los recursos que proporciona el Back-End. En el Back-End nos encontraremos todo lo relacionado con las consultas y el procesado de los datos en este caso nos lo encontramos dividido en varias capas, por un lado, la API REST, luego una capa de negocio, encargada de realizar la lógica de las peticiones y finalmente una capa de datos que se encargará de hacer consultas simples a la base de datos.



Ilustración 15. Arquitectura de la aplicación.

Gracias a esta arquitectura podemos realizar una abstracción mucho más eficiente de los datos, manteniendo una seguridad y permitiendo el acceso a los datos desde diversos dispositivos gracias a la API.

## **6.2 DIAGRAMA ENTIDAD RELACIÓN**

En la capa de datos es necesario almacenar de forma persistente la información que se gestiona con la plataforma. Por este motivo, es necesario elaborar un diagrama entidad-relación que nos permita identificar las entidades y relaciones que se tienen que almacenar en una base de datos. Dicho diagrama se puede obtener a partir del modelo conceptual obtenido en la fase de análisis.



Ilustración 16. Diagrama Entidad Relación.

En este diagrama vamos a dejar estipuladas cuales van a ser las clases y su relación dentro de ellas. A su vez no solo especificamos cuales van a ser los tipos de los atributos sino que también marcaremos su relevancia dentro del sistema identificándolos como claves, o si pueden o no ser únicos.

## **6.3 MOCKUPS**

Otro aspecto importante es detallar la interfaz de cliente que va a tener el usuario para interaccionar con la plataforma. Dicha interfaz debe proporcionar facilidad al usuario para ejecutar cada uno de los casos de uso identificados en la especificación de requisitos. La Parte del Front-End de esta aplicación Es posiblemente la parte más importante de la aplicación, ya que esta está plenamente orientada al usuario final. A continuación, vamos a ver los diseños previos de las pantallas que poseerá la aplicación.

**Inicio y registro**

En esta aplicación va a ser necesario disponer de unas pestañas de inicio y registro de sesión. Las cuales constarán de un formulario para que el usuario rellene los campos necesarios.

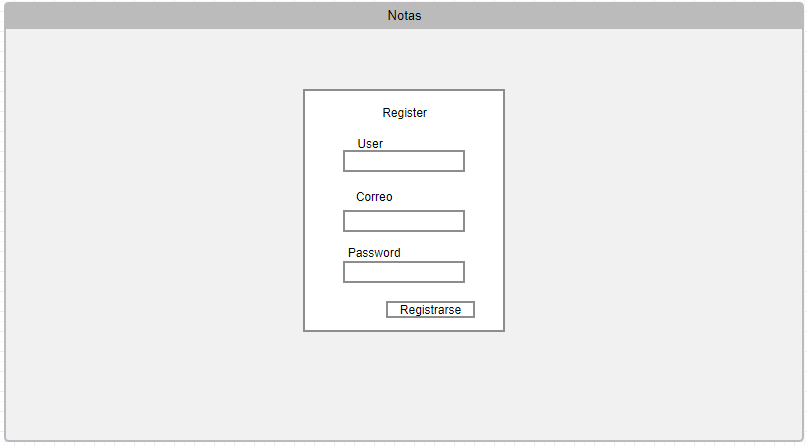


Ilustración 17. Mockup de registro de usuario.

El formulario del registro constará de 3 campos para rellenar con la información referente al nombre de usuario, el correo a insertar y la contraseña.

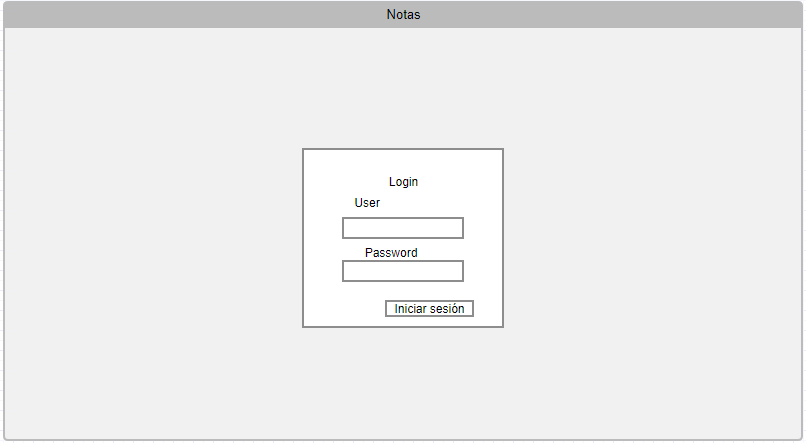


Ilustración 18. Mockup de inicio de sesión.

En la pestaña referente al inicio de sesión nos encontraremos con otro formulario el cual se debe rellenar con los datos de nombre de usuario y contraseña.

**Calendario**

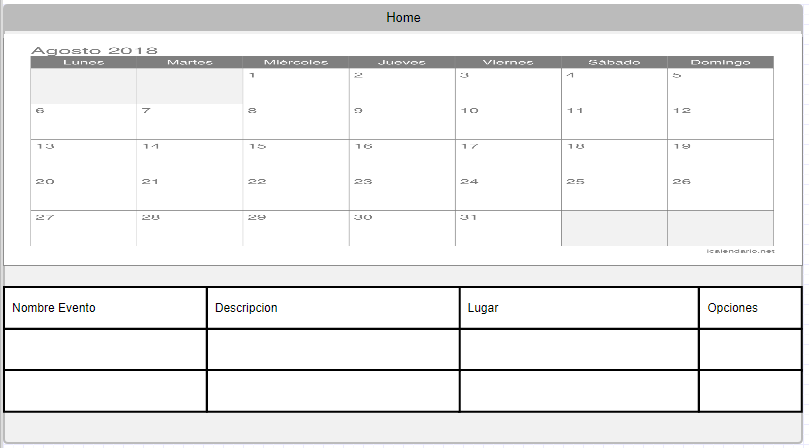
****

Ilustración 19. Mockup del calendario.

La página principal de la aplicación consistirá en un calendario, del cual, se desplegarán unas tablas con los eventos programados para el día que tengamos seleccionado. Dentro de esta tabla tendremos opciones en los eventos para visualizarlos, editarlos u eliminarlos.

**Notas**

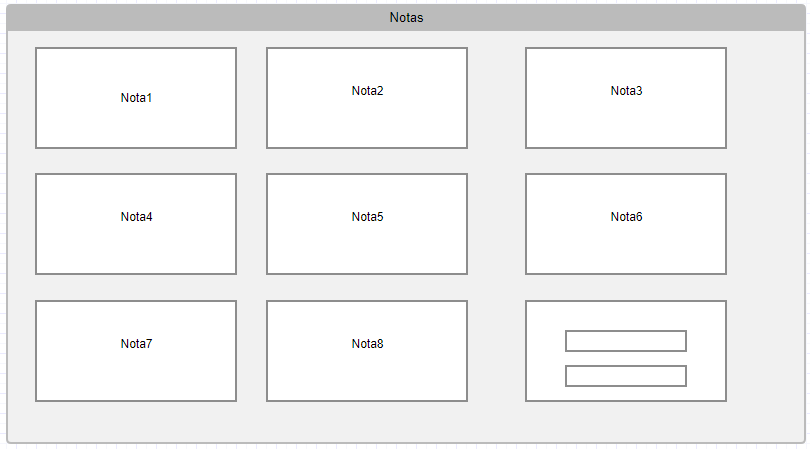
****

Ilustración 20. Mockup de Notas.

Como se ha mencionado anteriormente el sistema deberá contar con una serie de notas las cuales tendrán un formato de nota adhesiva para que consiga una estética más cercana.

**Chat**

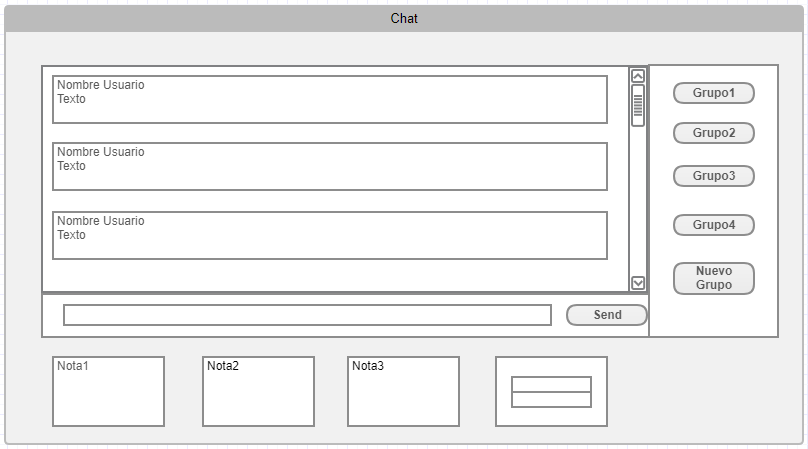
****

Ilustración 21. Mockup de la sala de chat.

Aquí se puede ver la estética que tendrá la sala de chat. Dispondremos de una pantalla central en la que se podrán observar los mensajes. A la derecha podemos ver la lista de los grupos a los que pertenece el usuario. Justo debajo de los mensajes, encontramos el formulario de texto para escribir un nuevo mensaje. Finalmente, tenemos debajo de la página un tablero con las notas que son compartidas en cada grupo.

**Buscador de eventos**

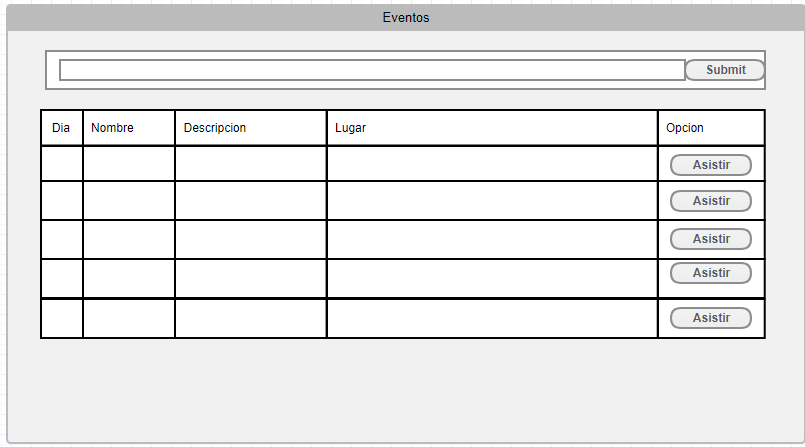


Ilustración 22. Mockup del buscador de eventos.

En esta pantalla mostraremos el motor de búsqueda que facilitará a los usuarios filtrar y seleccionar los eventos a los que desea asistir.

**CAPÍTULO 7**

# **IMPLEMENTACIÓN**

Ahora vamos a pasar a la fase de implementación del proyecto. En esta fase van a intervenir varias herramientas, las cuales son: Visual Studio (como se ha mencionado anteriormente), SQL Manager y Postman. Gracias a SQL Manager puede visualizar el estado de las tablas de la base de datos y mediante Postman puedo comprobar la viabilidad de las llamadas y peticiones a la API.

Para la implementación se ha dividido el proyecto en dos partes: la parte de Front-End y la parte de Back-End. Cada una de estas partes tiene un lenguaje determinado. La parte del Back-End estará escrita en C# y para la parte del Front-End, se utilizará el Framework React, el cual está basado en Type-Script, que es un lenguaje similar a Java Script, pero con la imposición de tipos en las variables. La arquitectura que se ha seguido para el desarrollo de la aplicación se ve en (3 pág. 64) para desarrollar una arquitectura en N capas.

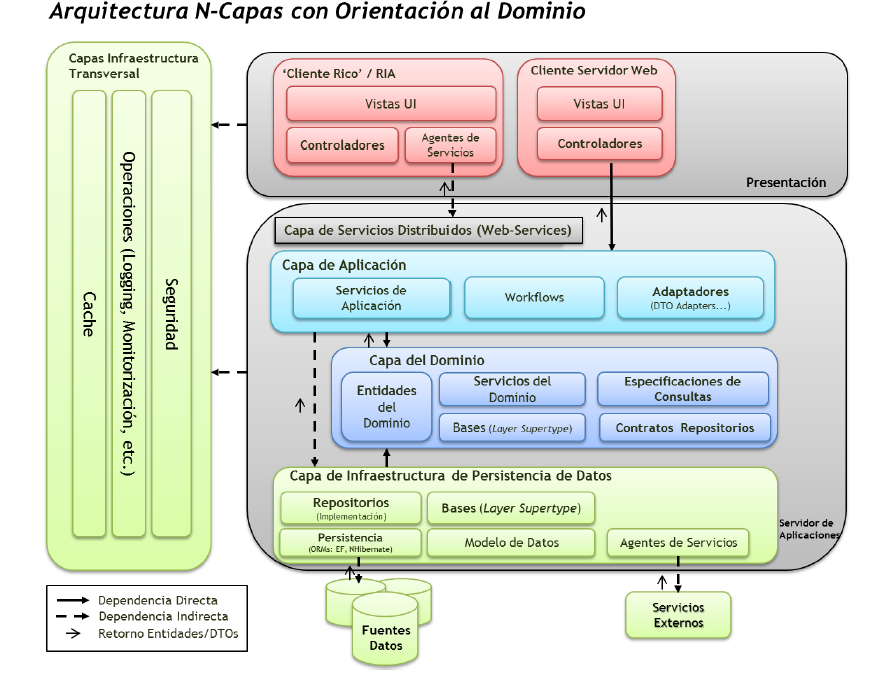


Ilustración 23. Arquitectura en N-Capas con orientación al Dominio, pág. 64.

Este esquema una vez traducido a nuestra arquitectura dentro de Visual Studio quedaría tal y como se ve en la siguiente ilustración.

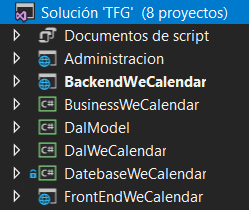


Ilustración 24. Ficheros del proyecto en Visual Studio.

Quedando las capas “Administración” y “Front-End” como clientes. La parte “Back-End” como capa de aplicación, “Business” como la capa de negocio y “DAL” como la capa de acceso a datos. Finalmente tenemos la “DalModel” en la que encontraremos las clases en C# las clases asociadas a la base de datos.

## **7.1 Back-End**

En este caso, nuestro Back-End constará de una capa DAL (Data Access Layer), la cual se encargará exclusivamente de realizar las consultas en la base de datos. Una capa Business, la cual se encargará de llevar a cabo la lógica en el servidor. Finalmente, tendremos una API Rest, la cual se comunicará con la capa de negocio para realizar los cambios en la base de datos. A esta API se accederá mediante las URLs desde las distintas pestañas del Front-End.

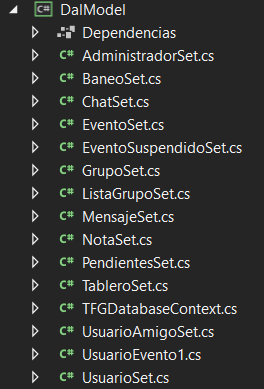
Anteriormente se ha mostrado un diagrama de UML de la que será la base de datos del sistema. Este diagrama puede transcribirse dentro de Visual Studio para que posteriormente sea la aplicación la encargada de generar las tablas de la base de datos partiendo de este diagrama pudiendo editarse visualmente. A partir de la base de datos Visual Studio creará una estructura de clases asociada.

Ilustración 25. Clases en Visual Studio pertenecientes a la base de datos.



Ilustración 26. Diagrama generado por Visual Studio.

Aquí se pueden apreciar las tablas y sus relaciones que posteriormente serán traducidas por la misma herramienta a las sentencias de SQL necesarias para generar la base de datos completa. Una vez que se ha generado la base de datos debemos de actualizar el modelo que tenemos en la capa “DalModel” del proyecto para que la estructura de las clases quede acorde con la estructura de la base de datos. Una vez actualizamos el modelo en la aplicación se genera la estructura vista en la *ilustración 18.*

API

En la capa del Back-End encontramos la API. Gracias a esto podemos comunicar las funciones del Back-End con el Front-End. En este caso tendremos la API dividida según su funcionalidad. Siendo estas divisiones denominadas “Controller”. En este caso veremos un “UsersController” para las operaciones de los usuarios, “EventsController” para las funciones y llamadas procedentes de los eventos, “NotasController” para las funciones derivadas de la administración de las notas y “ChatController” para todas las funciones referentes a los mensajes y a la sala de chat.

Ahora vamos a ver como se realizan las llamadas de las funciones más relevantes dentro de cada sección de la API.

**UsersController**

|  |  |  |
| --- | --- | --- |
| Operación | Argumentos | Respuesta |
| get | nombre | Datos referentes a usuario con el nombre especificado. |
| amigos | identificador | Devuelve la lista de amigos del usuario identificado. |
| listaBaneados |  | Devuelve la lista de los usuarios que se encuentren baneados en el sistema. |
| login | nombreUsuario, contraseña | Valida las credenciales de usuario y devuelve un booleano con la comprobación de los credenciales. |

**Ejemplo:**

llamada:

[**http://localhost:11111/api/Users/Afro**](http://localhost:11111/api/Users/Afro)

Respuesta:

{

"id": 1,

"nombreUsuario": "Afro",

"correo": "javier.fuentes78@gmail.com",

"notificacion": "YES",

"foto": "foto",

"createDate": "2018-07-03T10:02:32.503",

"token": "deea659a-c912-4f64-9f93-3ca477443b30",

}

**EventsController**

|  |  |  |
| --- | --- | --- |
| Operación | Argumentos | Respuesta |
| get | identificadorUsuario | Lista de eventos creados por el usuario identificado. |
| Invitación | identificadorUsuario | Lista de los eventos a los que ha sido invitado el usuario. |
| share | evento, usuario | Comparte el evento que se le ha suministrado como argumento al usuario que también se ha suministrado, devuelve un booleano en función de si ha podido o no compartir. |
| aceptarInvitacion | Usuario, evento | Crea el enlace entre el usuario y el evento, devuelve el booleano en función de si ha podido o no crearse el enlace. |

**Ejemplo:**

llamada:

**http://localhost:11111/api/events/1**

Respuesta:

[

{

"id": 99,

"nombre": "Conferencia de programación",

"descripcion": "Charla sobre desarrollo en C#",

"direccion": "ETSIIT",

"horaInicio": "13:30:00",

"horaFin": "14:00:00",

"fecha": "2018-08-04T",

"prioridad": 1,

"visibilidad": false,

"createDate": "2018-08-04T09:13:12.06",

"usuarioId": 1,

}, …

]

**NotasController**

|  |  |  |
| --- | --- | --- |
| Operación | Argumentos | Respuesta |
| get | identificador | Devuelve las notas asociadas al identificador del usuario suministrado como argumento. |
| grupo | grupo | Devuelve las notas asociadas al grupo suministrado como argumento. |
| post | título, cuerpo, fechaTope | Se crea una nueva nota con el contenido suministrado como parámetro y devuelve un booleano en que refleja la realización de la tarea. |

**Ejemplo:**

llamada:

**http://localhost:11111/api/Notas/grupo/1**

Respuesta:

[

{

"id": 22,

"titulo": "Demo",

"texto": "texto de prueba",

"fechaTope": "2018-08-03T15:58:12.407",

"createDate": "2018-08-03T15:58:12.407",

}, …

]

**ChatController**

|  |  |  |
| --- | --- | --- |
| Operación | Argumentos | Respuesta |
| get | usuario | Devuelve la lista de mensajes que han sido redactados por el usuario suministrado. |
| put | usuario, grupo, texto, hora | Añade un mensaje a la sala de chat del grupo especificado identificado con el usuario, con el texto y la hora suministrados. Devuelve un booleano que representa el éxito de la operación. |
| newUserGrup | usuario, grupo | Añade a un usuario al grupo suministrado por parámetro, devuelve un booleano que representa el éxito de la operación. |
| newGrup | nombre, usuario | Genera un nuevo grupo con el nombre especificado y dejando como creador al usuario suministrado por parámetro, devuelve un booleano con el éxito de la operación. |

**Ejemplo:**

llamada:

**http://localhost:11111/api/Chat/1**

Respuesta:

[

{

"id": 1,

"texto": "Hola",

"grupoId": 1,

"createDate": "26/08/2018 11:18:17",

"usuarioId": 1,

"chatId": 1,

}, …

]

## **7.2 Front-End**

Ahora vamos a abordar la parte del Front-End. En esta parte, como he mencionado anteriormente, utilizaremos el Framework React. He elegido este Framework, en primer lugar, por su integración con Visual Studio, y en segundo lugar por su tratamiento con Ajax, ya que, en este caso, React integra a Ajax de forma nativa. Con esto y el uso de las variables de estado de la plantilla, se consigue un efecto de página estática en la que se están produciendo llamadas a la API y continuas modificaciones en los datos sin que el usuario perciba que la página se recarga. Para las plantillas de estilo de la página se ha optado por la opción de Boostrap, debido a su sencillez visual y a su característica responsiva. Es muy importante esta característica porque necesitamos que esta página sea accesible por todos los dispositivos posibles (ordenadores, tabletas, teléfonos móviles…) y es necesario que la página se redimensione correctamente en función del dispositivo que la esté utilizando para garantizar la mayor accesibilidad posible.

Uno de los inconvenientes que se han encontrado a la hora de implementar es la necesidad de depuración en local de la aplicación. Gracias a la herramienta Visual Studio, no necesitamos hacer subidas constantes de la aplicación para probar las nuevas funcionalidades implementadas, a pesar de tener la base de datos en el servidor. Para ayudar a la depuración se han establecido puertos fijos a la ejecución de Back-End y el Front-End.

|  |  |
| --- | --- |
| Ilustración 27. Configuración del Back-End (loaclhost:11111) | Ilustración 28. Configuración del Front-End (localhost:44444) |

Gracias a esto podemos establecer las direcciones de la API en un fichero a parte y cambiar entre local host y la dirección de Azure una vez queramos depurar o hacer un despliegue.

En la depuración es donde encontramos un problema a la hora de hacer peticiones a la API. El problema es el control de acceso HTTP CORS (5), en este caso es necesario añadir una cabecera específica a nuestras rutas de la API ([EnableCors("AllowSpecificOrigin")]) para permitir que se realicen llamadas desde la aplicación en depuración. A parte de esta cabecera debemos de agregar las direcciones desde las que se accederá a la API al fichero *Startup.cs* que se encuentra dentro de esta.

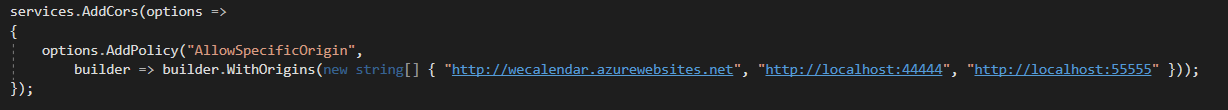


Ilustración 29. Código para permitir acceso en depuración a la API.

También es necesario agregar al navegador (Chrome en este caso ya que es el navegador que utiliza Visual Studio para la depuración) un complemento denominado Allow-Control-Allow-Origin para que el propio navegador también permita este acceso.

Para realizar las peticiones desde el Front-End a la API vamos a ayudarnos de la librería de Axios (6), librería compatible con los paquetes de React que nos ayudará a realizar peticiones a la API y recibir las respuestas con ayuda de las “Promesas”. Gracias a las promesas en las peticiones podemos especificar el formato en el que deseamos que se nos proporcione la respuesta. Esto hace que esa petición no acepte un formato distinto al que le proporcionemos, mitigando los errores en la transferencia de datos.

## **7.3 Envío de notificaciones**

También, en este apartado tenemos una funcionalidad que no depende de ninguna llamada por parte del Front-End. Esta funcionalidad se trata del envío de notificaciones, tal y como se indica en el caso de uso **CU-34**, todos los días deberá de enviarse una notificación a todos los usuarios que tengan las notificaciones activadas y que celebren un evento de prioridad alta ese mismo día. Esto se consigue gracias a la implantación de un demonio en la capa de negocio que, una vez lanzado repetirá periódicamente una vez al día las tareas de consulta de los usuarios mediante los criterios citados anteriormente y el envío de mensajes mediante correo electrónico gracias a una API externa de carácter comercial llamada Send Grid (7). La decisión de utilizar una API externa para el envío de mensajes viene de las limitaciones de los servidores de Azure de correos semanales, todo ello con la finalidad de evitar el SPAM.

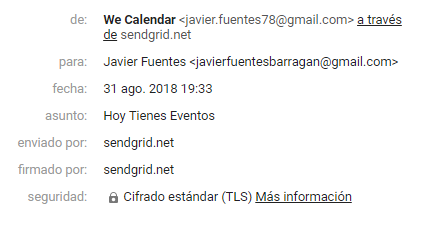


Ilustración 30. Correo Enviado desde Send Grid.

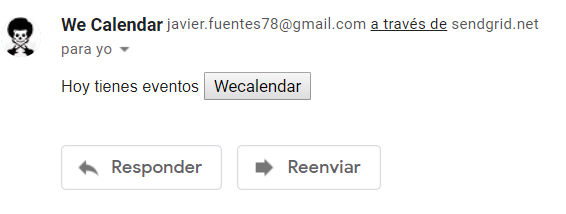


Ilustración 31. Contenido del correo.

Como se puede ver en el mensaje que se envíe con esta aplicación también podemos insertar código HTML para permitir hacer redirecciones a la página de una manera más limpia y sencilla para el usuario.

Gracias a esta aplicación se garantiza no solo la seguridad en el envío de mensajes, sino también una posible ampliación del flujo de mensajes semanales en el caso en que esta aplicación creciera en número de usuarios, únicamente mediante el cambio de suscripción en Send Grid.

## **7.4 SEGURIDAD**

Los factores que se han tenido en cuenta en este apartado solos factores relacionados con los datos de seguridad del usuario. A continuación, hablaremos del uso de las claves cifradas para guardar las contraseñas, las consultas mediante Token para garantizar anonimato y el hecho de que el código del Back-End ha sido compilado en el servidor, por lo que lo único que encontraremos al acceder a él, serán los archivos ejecutables del mismo.

CIFRADO DE CONTRASEÑAS

En este apartado hablaremos de la necesidad del cifrado de las contraseñas. Todas las contraseñas de los usuarios guardadas en la base de datos están cifradas de manera simétrica, estableciéndole a todas las mismas dimensiones de caracteres con un cifrado de 16bits, haciendo muy difícil las violaciones de seguridad mediante ataques de fuerza bruta.

TOKEN

Todos los usuarios, como se ha visto anteriormente, tienen almacenados una variable llamada Token, la cual nos servirá para realizar consultas. Esta variable es única y gracias a una función de C# (8) se puede generar de una forma sencilla.

La dinámica del Token consiste en que cada vez que el usuario inicie sesión, el sistema tras validar sus credenciales asignará un nuevo Token distinto al usuario, haciendo que cada vez que éste inicie sesión, el campo de búsqueda cambie evitando así posibles ataques de suplantación de identidad.

COMPILACIÓN DEL BACK-END

Aunque esta característica se ha planteado antes como una desventaja a la hora de elegir C# como lenguaje, hay que destacar que es una protección contra posibles hackeos al servidor, ya que, si una persona lograse entrar en nuestro servidor de Azure, lo único que encontraría sería un ejecutable, dificultando así la obtención de datos del código fuente.

## **7.5 USABILIDAD**

Al tratarse este proyecto como uno principalmente orientado al usuario final, es necesario pensar y orientar todas las acciones que se realizan en la página para proporcionarle al usuario una experiencia fácil y cómoda.

Para ello se ha tratado de reducir lo máximo posible el número de pulsaciones que un usuario debe realizar en la página para terminar la acción deseada. Se han implementado buscadores para filtrar y reducir lo máximo posible el desplazamiento por la pantalla, agilizando las búsquedas.

Otra ayuda que se le suministra al usuario para ayudar al uso es la disposición de la pestaña de los eventos, la cual le proporcionará una vista general de toda su organización mensual con un solo vistazo, centrando su vista en el eje principal de la aplicación, que es el calendario, y minimizando las búsquedas de los eventos.

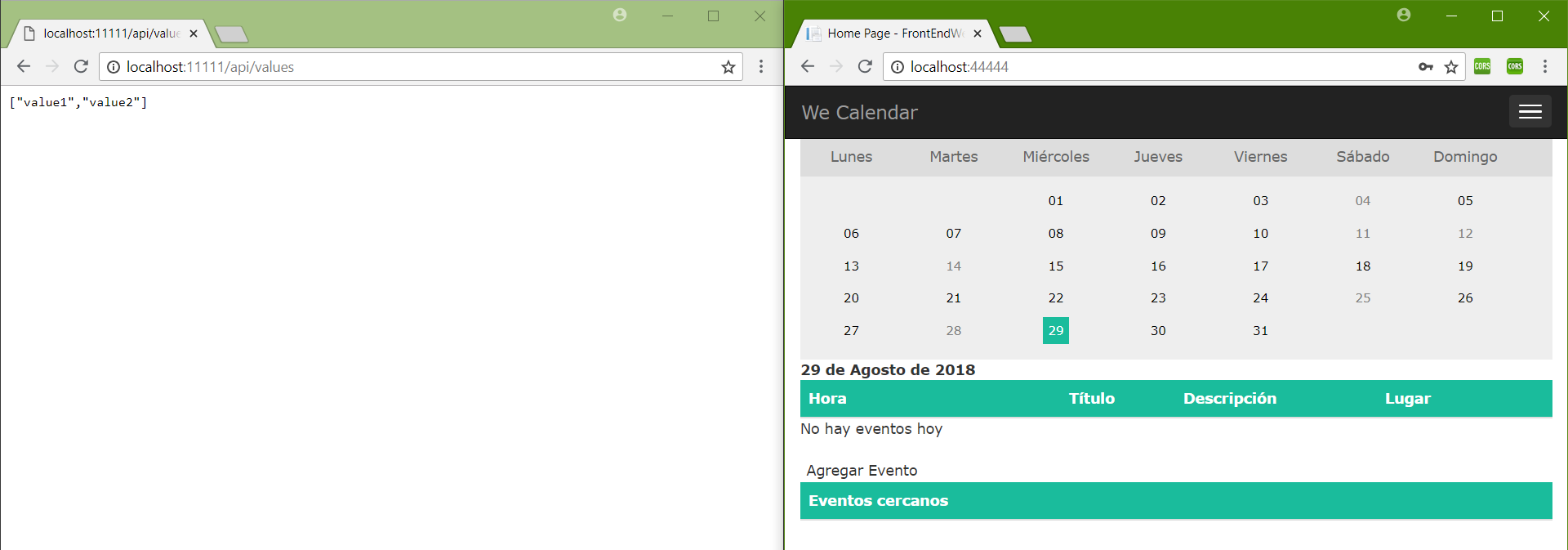


Ilustración 32. Ejecución del Back-End y el Front-End en local.

Se puede apreciar que en la página predominan los colores verde y azul esto es debido a lo que dice (4) *“el verde es el color que los ojos procesan mejor. Úsalo para crear un efecto relajante o de calma”* y también *“el azul suele asociarse con empresas grandes y bancos porque no es invasivo y se asocia con la seriedad”*. Con estos dos factores a tener en cuanta se ha decidido emplear una combinación de esos dos colores para transmitirle al usuario calma y organización ya que son dos atributos necesarios a la hora de realizar una correcta organización de los eventos.

## **7.6 PERSISTENCIA**

En este caso el servidor de Azure que estamos utilizando para almacenar el Back-End y el Front-End se trata de una arquitectura *Serverless* (9) , gracias a este modelo podemos abstraernos de los servidores, sin preocuparnos del sistema operativo. Este modelo se basa en la reacción a los eventos en tiempo real en la nube. Las facturas derivadas de este servicio están relacionadas al tiempo que el código se encuentra en ejecución.

Gracias a este modelo se crea una estructura mucho más robusta, resistente a las amenazas y con una disponibilidad muy alta. A parte también tiene muchísima flexibilidad a la hora de realizar escalados. *“Un servicio Serverless puede empezar de nada hasta abarcar decenas de miles de funciones concurrentes casi al instante”*.

## **7.7 MANUAL DE USUARIO**

A continuación, se va a disponer un pequeño manual de usuario para poder entender mejor como se establecen las transiciones dentro de la página y como se realizarán correctamente todas las funciones.

**Inicio y registro**

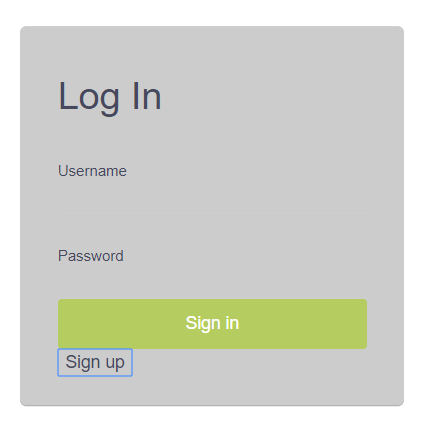
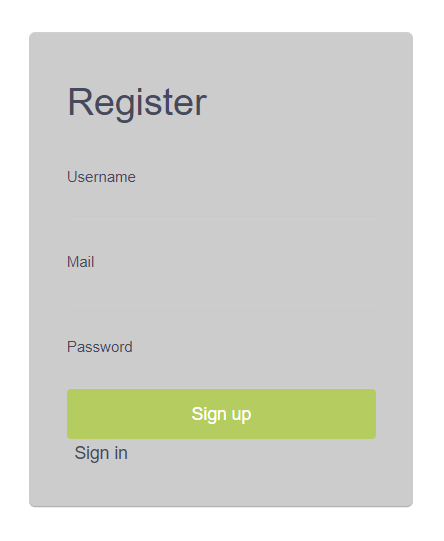


Ilustración 33. Pantalla de registro de usuario.

Ilustración 34. Pantalla de inicio de sesión.

Estas dos pantallas será las primeras con las que los usuarios tendrán contacto, tanto si el usuario es nuevo en el sistema y desea registrarse, como si el usuario ya se encuentra registrado en el sistema y lo que desea es acceder a él. El usuario una vez que se encuentre en cualquiera de estas dos plantillas podrá alternar entre estas mediante los botones *sign up* y *sign in*.

**Home**

Ilustración 35. Pantalla de Home.

Nada más iniciar sesión en la aplicación vamos a poder ver una pestaña con un menú lateral con todas las demás opciones de la aplicación.

Lo principal de esta pestaña es el calendario que aparece en el centro de la pestaña. Justo debajo del calendario se despliega una tabla con los eventos que tendrán lugar en el día seleccionado, que por defecto será el día actual. Se ha optado por este diseño del calendario para que el usuario tenga de un solo vistazo una visión global de la planificación del mes y a la vez pueda ver de una forma más específica cada uno de los eventos en el día.

En la tabla de los eventos podemos ver la información del evento correspondiente a la hora de inicio y fin del evento, su nombre y descripción y el lugar. Justo después tenemos dos botones para borrar o ver con más detalla el evento.

Justo debajo podemos ver también una pequeña tabla donde nos aparecerán los eventos más cercanos al día actual para tener presenta no solo lo que acontecerá ese día si no también, lo que pasará próximamente.

**Notas**

****

Ilustración 36. Pantalla de Notas.

Aquí podemos ver la pestaña de notas, se ha optado por un modelo de notas adhesivas para que se asemeje más a un tablero de notas casero. Dentro de esta pestaña se dispondrán todas las notas del usuario. Al pasar el ratón por encima de la nota esta aumentará ligeramente de tamaño y rotará ligeramente hasta enderezarse para que sea mucho más fácil de leer. La última nota que aparecerá en esta pestaña será el formulario para generar nuevas notas.

Se puede apreciar que hay notas con un color distinto de las demás, eso significa que esa nota ha sido compartida en algún grupo como podrá verse más adelante.

**Usuario**

****

Ilustración 37. Pantalla de Usuario.

En esta pantalla se puede ver la información referente al usuario que se encuentra registrado en el sistema en este momento. A parte de este perfil en el menú aparecerá el nombre del usuario. En esta pestaña se ve una imagen de perfil de usuario, su nombre y correo y la posibilidad de recibir las notificaciones, en esta última existe la opción de silenciarla si queremos para que el usuario ya no reciba más notificaciones al correo.

**Chat**



Ilustración 38. Pantalla de Chat.

En esta pestaña podemos ver la sala de chat que nos permitirá comunicarnos con el resto de los usuarios del sistema. Lo primero que nos encontramos son los mensajes de un grupo seleccionado. Estos mensajes disponen de una barra de scroll para poder ver todos los mensajes que se encuentren. Debajo de los mensajes tenemos la barra para escribir el mensaje el cual una vez enviado aparecerá con el nombre y la fecha en la que se envió.

A la derecha se puede ver la lista de los grupos, aquí se mostrarán todos los grupos en los que el usuario participa. Cuando el usuario pincha en alguno de los grupos se cambia el contenido del chat para así poder ver los mensajes correspondientes a dicho grupo.

Debajo de la barra del chat de puede apreciar otro tablero similar al tablero que había en la sección de notas. En este tablero se mostrarán las notas compartidas en el grupo seleccionado, las cuales hacen referencia a las notas azules de la pestaña de notas. También encontraremos un formulario para añadir una nota directamente al grupo seleccionado.

**Eventos**



Ilustración 39. Pantalla de Eventos.

Como se ha comentado anteriormente en el caso de uso **CU-20** los usuarios registrados en el sistema tendrán la posibilidad de ver los eventos públicos, los cuales aún no hayan terminado, y tras visualizarlos el usuario podrá marcar que desea asistir a ellos y este evento será añadido a su calendario. En este caso nos encontramos con una pestaña en la cual se dispondrán los eventos por orden mostrando los datos tales como el nombre, la descripción y el lugar y finalmente un botón para asistir. Se puede apreciar también, que hay una barra de búsqueda, la cual está justo encima de los eventos, esto hace referencia al **CU-21**.



Ilustración 40. Ejemplo de buscador.

Aquí les proporcionamos a los usuarios una barra de búsqueda para que puedan filtrar los eventos cuando la lista de los eventos públicos crezca mucho. Este filtro se activa al introducir cualquier parámetro de búsqueda en la barra del buscador inmediatamente filtrando las coincidencias como se ve en la *ilustración 32.*

**Gestión de eventos pendientes**

Todo lo mostrado anteriormente corresponde con el sistema en su estado normal en un momento en el que el usuario no haya recibido ninguna notificación de evento pendiente. Ahora vamos a ver las transiciones que un usuario debe hacer cuando ha sido invitado a un evento.

Primero, el usuario apreciará que en su barra de navegación aparece un elemento distinto. En este caso una alerta de que tiene un número de eventos pendientes, este número irá variando en función de la cantidad de eventos.

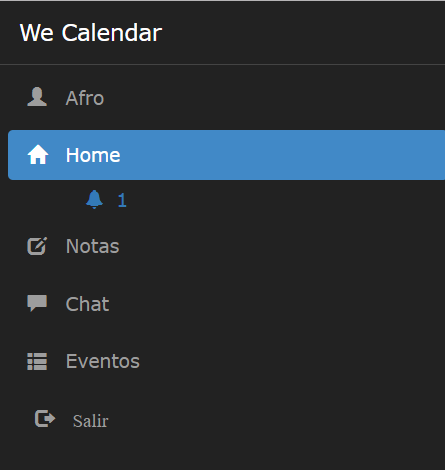


Ilustración 41. Notificación de evento pendiente.

Una vez el usuario se percate de esta notificación podrá pulsar sobre esta para ser redirigido a la pestaña de los eventos pendientes. En esta pestaña se le dispondrá un calendario en el que estarán marcadas las fechas de los eventos que están pendientes de validación.



Ilustración 42. Pantalla de eventos pendientes.

Como se puede ver en la *ilustración 19* solamente nos aparece un evento en la pantalla marcado en naranja. En este caso se han elegido dos tipos de colores para los eventos, el naranja para el evento pendiente que, si puede agregarse sin problema al calendario, y el rojo para el evento que no puede agregarse o es necesario realizar algún cambio en la planificación para que el evento pueda agregarse a nuestro calendario.

Una vez un evento pasa a estado pendiente pueden suceder varias cosas. Primero, podemos no querer asistir al evento, en ese caso podemos simplemente rechazarlo y este quedaría fuera de nuestra planificación y borrado de la lista de pendientes. Si deseamos añadir el evento entonces tenemos dos posibilidades, la primera es que esté disponible el margen de horas que precisa el evento, en ese caso bastará con aceptarlo y el evento quedará añadido. En el caso de que deseemos añadir el evento, pero no podamos el sistema nos proporcionará las herramientas para editar los eventos creados por nosotros o cancelar la asistencia a los elementos que hemos sido invitados. Una vez hecho esto, podremos aceptar el evento tranquilamente.

**Administración**

En la pestaña de administración vamos a tener un menú con dos opciones, usuarios y eventos. Como se ha mencionado antes la función del administrador será la de ver y eliminar los eventos de los usuarios y ver y banear a los usuarios del sistema. A continuación, se mostrará la pantalla correspondiente a esas funciones.

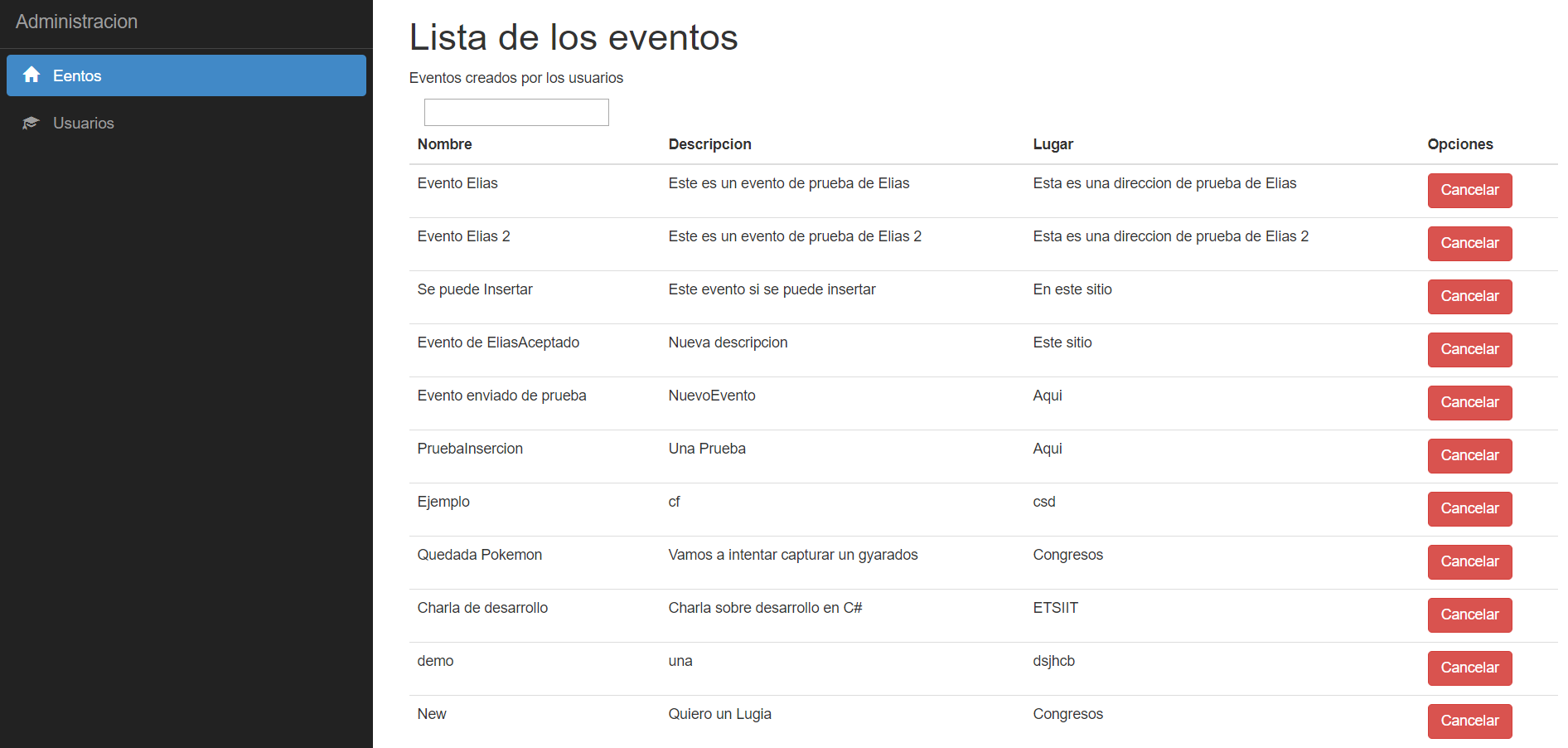


Ilustración 43. Pantalla de Administración de Eventos.

Aquí podemos ver la pantalla en la que los administradores podrán ver la lista de eventos del sistema. En la pantalla se observa una barra de búsqueda para facilitar la tarea del administrador y en cada uno de los eventos una opción de cancelar con la cual el evento quedará suspendido.

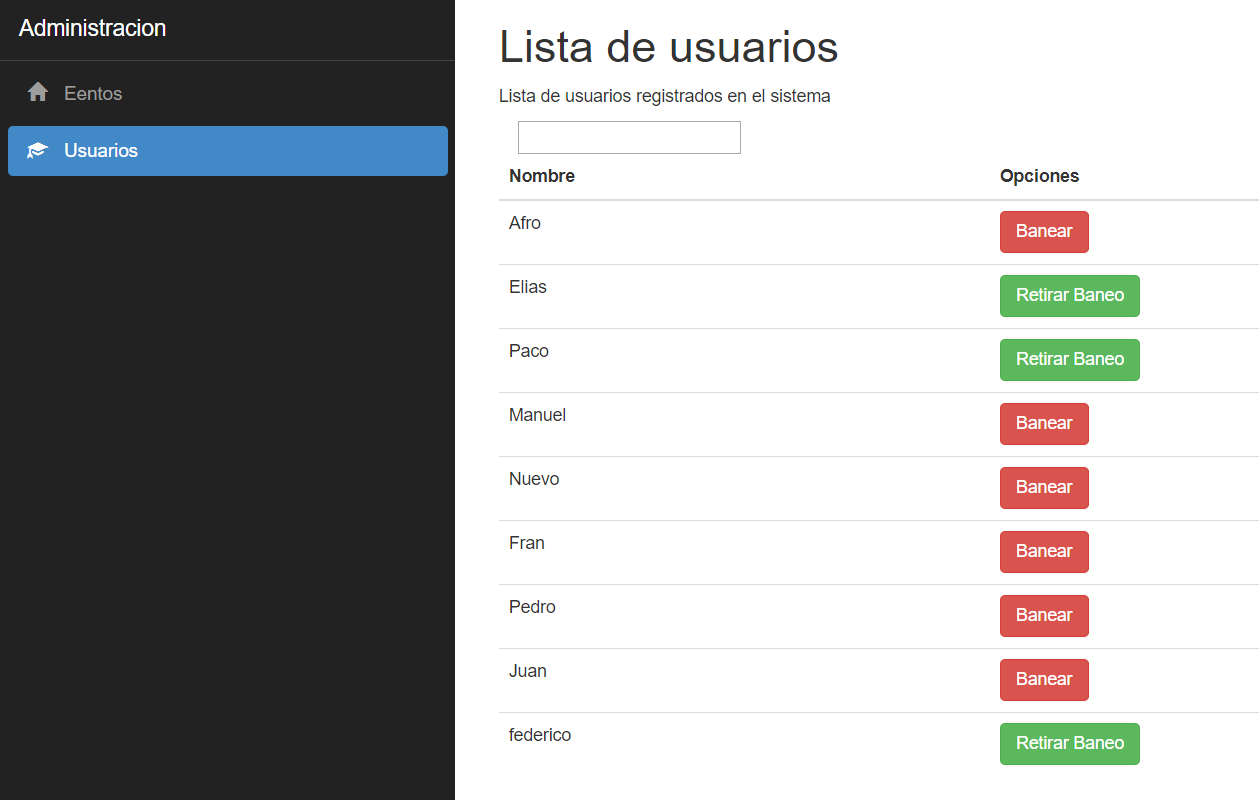


Ilustración 44. Pantalla de Administración de Usuarios.

En esta pantalla los administradores encontrarán una lista de los usuarios que se encuentren registrados en el sistema. Igual que en la pantalla anterior podemos ver una barra de búsqueda para filtrar los usuarios. Cada usuario tendrá la opción de baneo al lado de su nombre, una vez el usuario quede baneado se dispondrá de una opción de retirar baneo para devolverle el acceso al sistema.

**CAPÍTULO 8**

# **CONCLUSIONES**

Se ha conseguido desarrollar desde cero un novedoso sistema de organización personal que facilita la gestión y administración de eventos en el que hay que sincronizar la participación de varios asistentes mediante una plataforma web multiplataforma responsiva que puede ser manejada desde ordenadores pc a dispositivos móviles y tablets. Los usuarios pueden llevar a cabo una correcta organización, tanto de eventos personales, como organizar eventos públicos y administrar correctamente sus tareas diarias, solos o en grupo.

Gracias al calendario, a las notas y a las salas de chat hacemos más fácil la coordinación entre usuarios, gracias a las herramientas de búsqueda agilizamos el proceso de asignación de eventos y gracias a los formularios y a la información que manejamos en ellos evitamos el solapamiento de los mismos, que es uno de nuestros objetivos.

El sistema además ha podido ser desplegado sobre servicios Azure, lo que proporciona una forma rápida de llevar a producción un sistema sin necesidad de tener que instalar, configurar o mantener un servidor. Por tanto, se puede reducir el tiempo de entrega del producto e incrementar de forma dinámica las necesidades del servidor según sea mayor o menor el número de accesos concurrente.

No sólo se implementado una nueva plataforma. El proyecto demuestra cómo se pueden abordar esta clase de proyectos con las herramientas de Microsoft, aprovechando las facilidades que ofrecen para el desarrollo rápido de plataformas web.

A continuación, voy a analizar uno por uno los objetivos que había propuesto, identificando en cada uno de ellos mi nivel de satisfacción personal con el resultado y la razón de ese nivel.

|  |  |  |
| --- | --- | --- |
| OBJETIVO | NIVEL DE LOGRO | OBSERVACIONES |
| Sistema de inicio de sesión y registro | 100% | Como era necesario, para que los usuarios inicien sesión en el sistema y a su vez puedan registrarse en la plataforma en el caso de tratarse de nuevos usuarios, se ha dispuesto de un sistema de inicio de sesión y de registro de nuevos usuarios. |
| Establecer un calendario personal | 100% | Se ha conseguido establecer para el usuario final un calendario en el cual podrá agregar los eventos, visualizarlos y editarlos. |
| Control de eventos invitados | 100% | Se ha añadido un mecanismo para que el usuario de forma visual pueda ver los eventos invitados dándole a la opción de asistir o no a dicho evento. |
| Control del perfil y amigos | 100% | Se les ha proporcionado a los usuarios un sistema que les permita ver la información básica de su perfil y editar aquella que sea relevante para el funcionamiento de la aplicación. También en la misma pestaña se podrá llevar un control de los amigos que tiene el usuario. |
| Sistema de notas | 100% | El usuario dispondrá de un sistema de notas adhesivas que le permitirá añadir recordatorios de una manera sencilla. |
| Sala de chat | 80% | Aunque se ha suministrado una sala de chat completamente funcional para el usuario, lo más probable es que el mismo acabe recurriendo a otras aplicaciones de mensajería más estandarizadas para la comunicación en la mayoría de los casos. |
| Compartir notas | 100% | Este puede que sea el mayor atractivo para que un usuario use el sistema de mensajería interna del sistema, cada sala de chat dispondrá de un tablero para poder publicar notas, las cuales serán visibles por el resto de los usuarios que se encuentren en el grupo. |
| Buscador de eventos | 100% | Se ha conseguido implementar una barra de búsqueda que será capaz de filtrar los eventos en tiempo real permitiendo al usuario agilizar el proceso de búsqueda de un evento concreto. |
| Administración | 70% | Aunque se ha proporcionado un sistema de administración para poder controlar las acciones de los usuarios dentro del sistema cumpliendo con las especificaciones previas, este sistema puede crecer en futuras actualizaciones en la página volviéndose más complejo y requiriendo más funciones, pero de momento cumple con las necesidades que se le exigen a un administrador. |
| Sistema de notificaciones | 100% | Este sistema cuenta con una herramienta de mensajería a disposición de los usuarios registrados que, diariamente, se encargará de recordar los eventos que el usuario desee que sean recordados, siempre y cuando el propio usuario desee recibir notificaciones. |
| Multiplataforma | 100% | En este caso se ha conseguido que toda la estética de la web sea totalmente responsiva para facilitar su uso desde cualquier lugar |
| Uso de las tecnologías de Microsoft | 100% | El desarrollo íntegro de la plataforma se ha seguido empleando únicamente tecnologías de Microsoft específicas para esta clase de desarrollo. |

Tabla 3. Tabla de objetivos.

# **Bibliografía**

1. **Google.** Google Calendar. [En línea] [Citado el: 28 de Agosto de 2018.] https://calendar.google.com/calendar/r.

2. **Doodle.** Doodle. [En línea] [Citado el: 28 de Agosto de 2018.] https://doodle.com/es/.

3. **César de la Torre Llorente, Unai Zorrilla Castro, Javier Calvarro Nelson, Miguel Ángel Ramos Barroso.** Guía Arquitectura N-Capas Orientada al Dominio. *Guía Arquitectura N-Capas Orientada al Dominio - Microsoft Architecture .* s.l. : Krasis press, 2010, pág. 534.

4. **Bowman, Dawn.** Psicología del color en el diseño web. [En línea] [Citado el: 10 de Febrero de 2015.] https://es.jimdo.com/2015/02/10/psicología-del-color-en-el-diseño-web-parte-1/.

5. **MDN Web Docs.** CORS. *MDN Web Docs.* [En línea] [Citado el: 8 de Agosto de 2018.] https://developer.mozilla.org/es/docs/Web/HTTP/Access\_control\_CORS.

6. **Axios.** Github. *Github.* [En línea] [Citado el: 15 de Julio de 2018.] https://github.com/axios/axios.

7. **SendGrid.** API C# Code Example. *API C# Code Example.* [En línea] [Citado el: 29 de Agosto de 2018.] https://sendgrid.com/docs/for-developers/sending-email/v2-csharp-code-example/.

8. **Microsoft.** Guid. *Developer Network.* [En línea] [Citado el: 16 de Agosto de 2018.] https://msdn.microsoft.com/es-es/library/system.guid.newguid(v=vs.110).aspx.

9. **Microsoft Azure**. *Microsoft Azure.* [En línea] [Citado el: 1 de Septiembre de 2018.] https://azure.microsoft.com/en-us/overview/serverless-computing/.

10. **FluentSchedule.** Github. *Github.* [En línea] [Citado el: 15 de Julio de 2018.] https://github.com/fluentscheduler/FluentScheduler.

11. **StackShare.** StackShare. *PHP vs. Python vs. C#.* [En línea] [Citado el: 20 de Febrero de 2018.] https://stackshare.io/stackups/c-sharp-vs-php-vs-python.