MÁSTER EN TECNOLOGÍAS Y APLICACIONES EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

UNIVERSIDAD DE ALMERÍA

ESCUELA SUPERIOR DE INGENIERÍA

"EV-Planner: Planificador de eventos"

Curso 2019/2020

Alumno/a:
Antonio Bordes Giménez

Director/es:
Antonio Becerra Terón

Agradecimientos

Quisiera dedicar unas líneas y agradecer a todas las personas que han contribuido de alguna manera a que este proyecto se haya realizado.

Primero, quiero agradecer a los profesores de la titulación de ingeniería informática y a los del Máster en Tecnologías y Aplicaciones en Ingeniería Informática todo lo que me han enseñado a lo largo de estos años, indudablemente gran parte de lo qué hay escrito en este documento es gracias a ellos y a todas sus lecciones.

También quiero agradecer a mi familia y amigos por su paciencia, porque entre el trabajo y la universidad no he tenido el suficiente tiempo libre que me hubiera gustado dedicarles.

Especialmente, quiero agradecer a mi tutor de este TFM, Antonio Becerra, por los ánimos y guía que me ha dado para la realización de este proyecto, es sin duda un gran docente y mejor persona.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. EL EFECTO DE LA TECNOLOGÍA EN LAS RELACIONES INTERPERSONALES

Con la llegada de internet y de los teléfonos inteligentes, las comunicaciones interpersonales han ido evolucionando, encaminándose a una sociedad intrínsecamente electrónica. Hace unos años las comunicaciones se realizaban principalmente en persona o por teléfono, ahora en cambio es cada vez más frecuente comunicarse mediante medios electrónicos como chats o redes sociales. Ello ha evidenciado que los medios por los cuales nos relacionamos socialmente están en continuo cambio y no podemos saber en un futuro hasta donde llegaremos para comunicarnos con otras personas.

No sería de extrañar que, dentro de unos años los teléfonos móviles y equipos fijos como tabletas u ordenadores, fueran capaces de grabar y reproducir hologramas, algo tan impensable hoy en día, como hace varias décadas lo era comunicarse a través de la voz con dispositivos portátiles *(Sierra, 2019).*

Lo que sí sabemos es que las personas queremos estar conectados con el resto de la sociedad en todo momento, sin importar el medio o la distancia que nos separe. Ello se hace patente en estudios que evidencian el incremento que se está produciendo en el uso de las redes sociales, como muestra los informes de 2019 realizados por *Hootsuite* y *We Are Social (Kemp, 2019).*

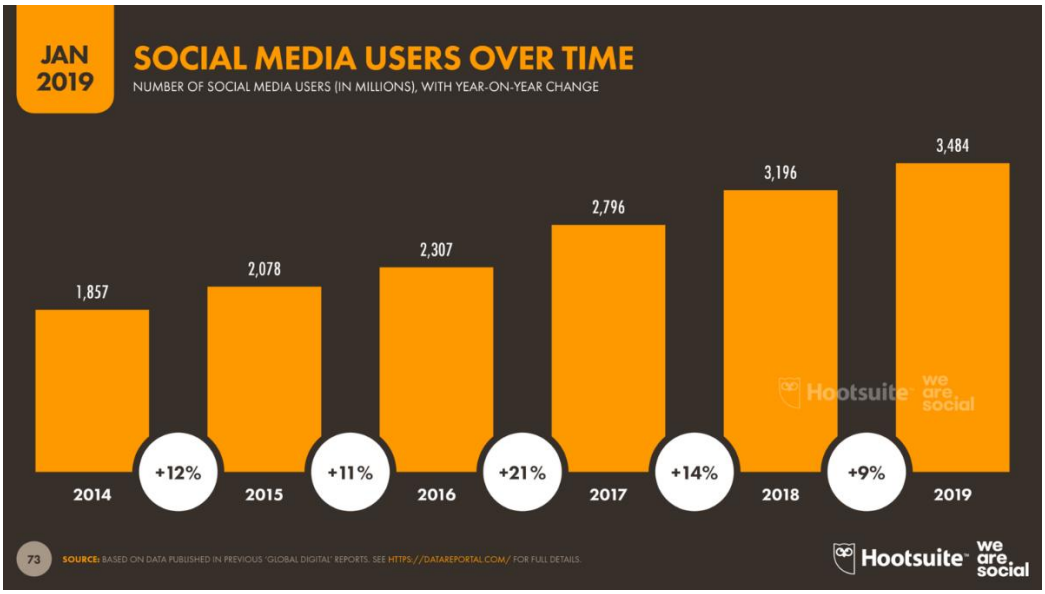


Ilustración 1: Incremento anual del uso de las redes sociales

La tecnología permite comunicarnos rompiendo la barrera del espacio e incluso la del tiempo, pero ¿la tecnología es realmente buena para las relaciones personales o es algo contraproducente?

La tecnología en sí no es buena o mala, depende del uso que se dé de ella. Cuando se usa demasiado puede producir una adicción, especialmente preocupante en niños y jóvenes. Según la psicóloga Francisca Rodríguez, "es muy positivo que los jóvenes sepan usar la tecnología, internet y las redes sociales, porque son canales que les sirven para comunicarse, pero los padres tienen la responsabilidad de enseñarle a sus hijos el buen uso de las herramientas tecnológicas, para que sepan utilizarlas a su favor y no en contra" *(Morales, 2012).*

1.2. EL MUNDO DE LA GESTIÓN DE EVENTOS

En la actualidad, afirmar que vivimos rodeados de eventos no es ninguna exageración. Desde cumpleaños hasta comidas de empresa o incluso quedar con unos amigos para ver una película, todos son eventos en los que participamos y al menos hay una persona encargada de organizarlos.

Las personas encargadas de organizar eventos de manera profesional son conocidos como *event planners*. Durante su trabajo se deben de tener en cuenta que buenas prácticas deben seguirse para lograr el éxito de un evento y que posibles errores deben evitarse. Es primordial trazar un mapa de rutas de todo el proceso del evento, promocionarlo por los medios adecuados, evitar en la medida de lo posibles la improvisación y finalmente evaluar los resultados del evento transcurrido y sí se han cumplido los objetivos propuestos (*"Todo lo que tienes que saber sobre la gestión de eventos", s.f.)*.

El auge de las aplicaciones móviles que se está produciendo en los últimos años también ha afectado al ámbito de la gestión de eventos, permitiendo facilitar y agilizar la comunicación con los participantes o consultar o interactuar con los eventos y sus detalles desde cualquier lugar *("Aplicaciones móviles para eventos ¿qué nos aportan?", 2019)*. Algunos ejemplos de estas aplicaciones son *Pro Party Planner*, *Doodle*, *Attendium* o *Eventbrite Organizer.*

Ilustración 2: Aplicaciones de eventos

Aunque existan herramientas profesionales para gestionar eventos, lo cierto es que gran parte de los eventos que se organizan, sobre todos los de ámbitos informales, se gestionan por medio de otros sistemas de comunicación, como pueden ser el correo electrónico o las aplicaciones de mensajería instantánea.

El gran referente mundial es WhatsApp que permite crear grupos de cualquier tema e invitar a los contactos de los administradores para participar en ellos, permitiendo acordar

mediante una conversación con las personas implicadas en un evento todos sus detalles. Este tipo de organización de eventos es muy flexible, pero puede llegar a ser muy caótico, sobre todo cuantas más personas participen en él.

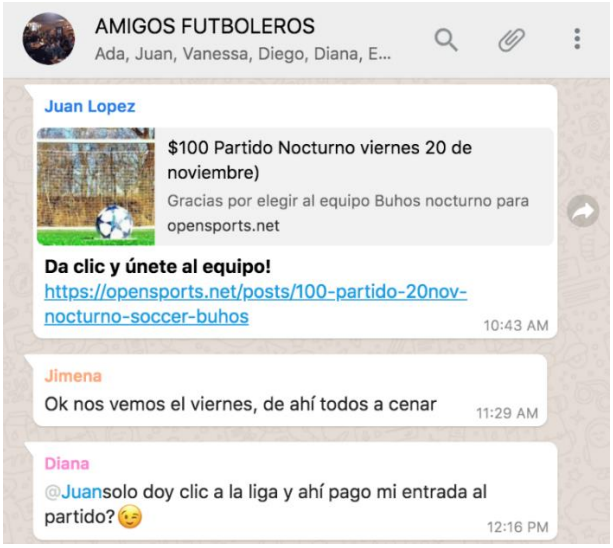


Ilustración 3: Grupos de WhatsApp

En conclusión, la tendencia en la organización de eventos se está dirigiendo a una vertiente virtual en la que los aspectos más relevantes se centran en la participación activa de los asistentes, la integración de correctos medios de comunicación y la estimulación de un componente lúdico durante todo el proceso de ejecución del evento *(Estanyol i Casals, 2012)*.

1.3. MARCOS DE DESARROLLO *FULL STACK*

El entorno de la computación y el de desarrollo está en auge, y debido a ello gran parte de las empresas se han trasladado al mundo digital, demandando profesionales con grandes conocimientos como los expertos *full stack*, uno de los perfiles de desarrollo más completos.

El concepto de *full stack* está relacionado con el cliente, el servidor y la base de datos, siendo necesario tener conocimientos en programación *back-end* y *front-end*, además de estar familiarizados con los sistemas operativos y sus componentes *("Full Stack Developer: cómo convertirse en el desarrollador más completo", 2018)*.

Uno de los enfoques clásicos de desarrollo *full stack* es el conocido como LAMP, un conjunto de software de código abierto que trabajan entre sí para poner en marcha y mantener servidores web. Su gran popularidad se debe principalmente a su bajo coste de implementación, ya que sus herramientas se encuentran en la mayoría de las distribuciones Linux *(García, 2016)*.

Su infraestructura está conformada por el sistema operativo Linux, el servidor web Apache, el gestor de base de datos MySQL y el lenguaje de programación PHP, Perl o python.

Ilustración 4: LAMP

En los últimos años, JavaScript se ha extendido con gran velocidad en los desarrollos web de todo el mundo debido a que es un lenguaje de programación fácil, versátil y adaptable a todas las plataformas. Esto se hace patente en MEAN, uno de los marcos de desarrollo *full stack* más populares *("Mean Stack: el pura sangre de los Stack", 2017)*.

Utilizando MEAN, acrónimo formado por las tecnologías que lo conforman: MongoDB, Express, Angular y Node.js, es posible implementar aplicaciones distribuidas en todas sus fases y capas con un mismo lenguaje, JavaScript *(Alarcón, 2015)*.

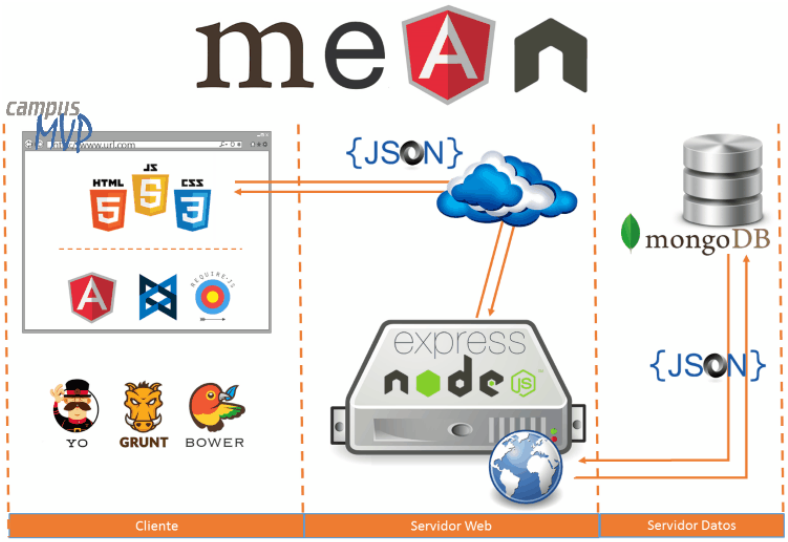


Ilustración 5: MEAN

Cada uno de los *stacks* explicados tienen sus defectos y sus virtudes, y a la hora de escoger uno para comenzar un nuevo proyecto se deben de tener todos en cuenta. Además hay otros factores de gran importancia que afectan a esta elección, como es el caso del entorno en el que se trabaja y la experiencia del desarrollador, no es lo mismo trabajar por cuenta propia que trabajar para una empresa, si una empresa suele trabajar con unos determinados lenguajes y tienen sus propios procedimientos, probablemente la balanza se decante por seguir utilizando las mismas tecnologías y herramientas que asegurarán que el equipo de desarrollo pueda seguir manteniendo el proyecto sin invertir en ello un exceso de recursos, pero cada proyecto es único y muchas veces para lograr asegurar su calidad se debe de salir de la zona de confort y apostar por las tecnologías que más se adecúen sus requisitos.

A continuación, voy a comparar las características de los dos *stacks* explicados en esta sección con el objetivo de servir de ayuda para escoger uno en la elaboración de un proyecto.

Por un lado, LAMP tiene mayor tiempo de recorrido y con ello una gran cantidad de librerías desarrolladas y una inmensa comunidad detrás, además es fácil de aprender y poner en práctica. Entre sus desventajas se encuentran que son dependientes de un sistema operativo y aunque es fácil de aprender es complejo de dominar, por ello muchas de las aplicaciones que se realizan no siguen buenas prácticas ni las medidas de seguridad necesarias, además suele estar integrado con MySQL, una base de datos relacional, que no es tan escalable como otras bases de datos NoSQL.

Por otro lado, se encuentra MEAN, su principal ventaja es el uso de JavaScript en todos sus componentes, eso lo hace muy versátil y permite que desarrolladores con un buen dominio del lenguaje realicen aplicaciones de gran calidad. Además, utiliza MongoDB, una base de datos no relacional que ofrece gran velocidad de lectura. También utiliza Angular y su filosofía de reutilización de componentes, permitiendo crear aplicaciones fáciles de mantener y de integrar con todo tipo de dispositivos como móviles o aplicaciones web de escritorio. Por el contrario, al ser relativamente nueva, muchas de las librerías que están disponibles no están del todo optimizadas y muchas empresas arraigadas en otras tecnologías todavía no han dado el paso para migrar a este nuevo modo de desarrollo.

En este proyecto se ha escogido MEAN para llevar a cabo la implementación de la aplicación. En el apartado 1.5. Motivación se argumenta porque he escogido este *stack* para realizar el proyecto y en el apartado 2.3.1. JavaScript se explica con más detalle cada componente que lo conforman.

1.4. OBJETIVOS

El objetivo principal de este proyecto es el desarrollo de una aplicación, basada en un marco de desarrollo *full stack*, que permita organizar eventos y compartirlos de manera pública o privada. Para llevarlo a cabo se han tenido que afrontar varios subobjetivos:

- Gestión de usuarios: El almacenamiento y consulta de la información de los usuarios de la aplicación.
- Gestión de eventos: El almacenamiento y consulta de los datos relacionados con los eventos, como es el caso de los detalles, asistentes o la localización vinculada.
- Gestión de notificaciones: Sistema de notificaciones relacionados con los eventos y métodos de compartición con otros usuarios.

Otro de los objetivos principales del proyecto es mostrar de manera detallada todo el proceso de ingeniería del software qué hay detrás para constituir una guía detallada de desarrollo y buenas prácticas.

1.5. MOTIVACIÓN

Basado en mi experiencia profesional en una gran empresa en la que empecé gracias a la Beca Talento-D-UAL de la universidad y en la que llevo trabajando más de un año, he podido comprobar de primera mano las necesidades que tienen las empresas en los tiempos actuales por digitalizar la mayoría de los servicios que ofrece, tanto externos como internos, y la gran demanda que hay en el desarrollo de aplicaciones nuevas y en el mantenimiento de las ya existentes.

Por fortuna, aunque en la empresa en la que trabajo predomina el desarrollo de aplicaciones LAMP, debido fundamentalmente a la facilidad de mantenimiento por los lenguajes utilizados por el departamento de software, siempre se apuesta por estudiar en cada nuevo proyecto que necesidades se deben cubrir y que tecnologías pueden dar resultados óptimos, permitiéndome analizar varias de las soluciones que se utilizan en los ambientes de desarrollo actual.

Gracias al máster, he podido introducirme en el desarrollo de aplicaciones utilizando la pila MEAN y ser consciente de todo lo que puede ofrecer:

- Node.js como entorno de ejecución permite crear aplicaciones escalables y es perfecta para manejar gran cantidad de tráfico.
- MongoDB como gestor de base de datos es a la hora de manejar gran cantidad de datos de las mejores opciones del mercado, permitiendo realizar gran cantidad de operaciones por segundo y distribuirse por diferentes máquinas para mejorar su escalabilidad.

- Express es un marco de desarrollo muy versátil que permite gestionar el *back-end* del servidor de una manera muy simple.

- Angular es uno de los marcos de desarrollo de *front-end* más completos, permitiendo crear un código basado en componentes, muy útil a la hora de mantener una aplicación.

Y lo mejor es que todos estos elementos hablan un mismo lenguaje, JavaScript (*Holmes & Harber, 2019*). Por ello una de las motivaciones principales de este trabajo es profundizar más en el desarrollo de aplicaciones basadas en el *stack* MEAN, mediante la creación una aplicación, que, acompañado con la documentación elaborada, en la que se definirán todos los requisitos que debe tener el sistema y como se ha realizado su correspondiente implementación, sirva de guía de desarrollo software y de buenas prácticas a seguir para asegurar la calidad y el mantenimiento de una aplicación. Todo esto, sin duda, me será muy útil en toda mi trayectoria profesional.

La motivación para escoger como tema principal de la aplicación a desarrollar la gestión de eventos es debido a lo explicado durante las secciones anteriores. Como se ha expuesto en el apartado 1.1. El efecto de la tecnología en las relaciones interpersonales, cada vez es más frecuente que las comunicaciones se realicen de manera electrónica, pero eso no quiere decir que se sustituya por completo el trato humano, es más, los medios electrónicos frecuentemente ayudan a fomentar este tipo de comunicaciones, la cantidad de eventos organizados de manera profesional o casual mediante plataformas o aplicaciones electrónicas son innumerables, sobre este punto se ha hablado de manera más detallada en el apartado 1.2. El mundo de la gestión de eventos, dónde se ha puesto como aplicación de referencia a la hora de organizar eventos a WhatsApp.

WhatsApp permite crear grupos de cualquier tema e invitar a los contactos de los administradores para participar en ellos, pero estos grupos están limitados, por una parte, no hay que olvidar que WhatsApp es una plataforma de mensajería electrónica lo que le otorga una gran flexibilidad para compartir opiniones y archivos multimedia, pero lo limita a la hora de gestionar los propios eventos, conocer quién va a asistir realmente o cuales son todos los detalles del evento se puede volver un completo caos sobre todo cuanta más gente participe. Citando a Pedro Luis Barcia, director de la Academia Argentina de Letras: “Antes del chateo, nunca nadie había escrito tanto -digitalmente- y tan mal.” *(Barcia, 2006).*

Además, actualmente estos grupos son privados y solo pueden participar los contactos de las personas encargadas de administrar los grupos. Por ello yo propongo una herramienta que permita organizar los eventos de una manera más clara y profesional, EV-Planner. Esta herramienta no pretende sustituir a otras plataformas que se utilizan actualmente, sino que su objetivo es complementarlas y potenciarlas.

Retomando el tema de los grupos de WhatsApp, se podrían gestionar los eventos de una manera más clara y concisa si por ejemplo en la descripción del grupo hubiera un enlace a una

página que permitiera a los participantes conocer la información del evento, sus últimas actualizaciones y confirmar su asistencia, y así permitir que el grupo de WhatsApp se centre en discutir los detalles del evento, que posteriormente, los administradores se encargaran de actualizar en la plataforma, o incluso enviar el enlace del evento por otros medios como el correo electrónico.

La aplicación no sería únicamente una herramienta para organizar eventos privados como reuniones o cumpleaños, sino que va más allá, ofreciendo la oportunidad de crear eventos públicos ligados a localizaciones, permitiendo a la gente consultar los eventos más próximos a ellos junto a sus condiciones y participar en ellos si así lo desean.

Ilustración 6: Ejemplo de ubicaciones de eventos

Las posibilidades son ilimitadas, desde organizar reuniones de personas afines a algún tema concreto, hasta organizar cursos de formación o incluso eventos deportivos que necesiten más participantes.

2. DESARROLLO DEL PROYECTO

2.1. ETAPAS

El desarrollo de la aplicación estará basado en una metodología ágil, dividiré las funcionalidades que debe tener en dos versiones, cada versión se desarrollará en un plazo de tiempo determinado y proporcionará software funcional al final de cada iteración. Las fases de desarrollo del proyecto se explican a continuación.

El primer paso del proyecto será elaborar la especificación del sistema a desarrollar y sus características.

Para el desarrollo de la aplicación me guiaré por las etapas fijadas en el SWEBOK *(Sánchez, S., Sicilia, M.A., & Rodríguez, D., 2011)*:

- Análisis de Requisitos y Planificación: Primero realizaré un análisis completo de los requisitos del sistema. Una vez definidos los requisitos, realizaré una planificación basada en una metodología ágil para distribuirlos en diferentes plazos de tiempo, permitiendo obtener diferentes versiones funcionales del producto en cada una de ellas. Al final de la realización de cada versión, realizaré una revisión del proyecto para poder tomar las medidas necesarias y así asegurarme de que el proyecto transcurra sin incidentes *(Kniberg, H., 2015)*.

- Diseño: En esta etapa, se realizará el diseño de las ventanas de la aplicación.

- Implementación: En el desarrollo de la aplicación se distinguen principalmente dos elementos *front-end* y *back-end*. Desarrollaré ambas partes por separado, implementando por un lado la interfaz y por el otro lado la API, para finalmente integrarlas. Al crear una API para manejar el acceso a los datos se facilitará el posterior mantenimiento de la aplicación y la reutilización de los datos en caso de que otros proyectos lo requieran.

- Pruebas: Probar la aplicación es indispensable antes de su despliegue, por ello elaboraré al final de la implementación de cada versión una batería de pruebas de aceptación que permita comprobar el correcto funcionamiento del sistema y la adecuación con los requisitos descritos en la especificación.

- Mantenimiento: El proceso de mantenimiento estará presente en cada etapa del desarrollo del software, me centraré en la aplicación de un mantenimiento preventivo, construyéndolo por si en el futuro debe sufrir posibles modificaciones. Además, documentaré detalladamente el código y las características de cada versión para permitir que se pueda modificar la aplicación de una forma más sencilla, aunque pasase por manos de otros desarrolladores.

2.2. TECNOLOGÍAS Y HERRAMIENTAS

En cada una de las etapas explicadas en la sección anterior, utilizaré diversas herramientas que me permitan desarrollar un proyecto de calidad en un plazo de tiempo aceptable.

2.2.1. VISUAL PARADIGM

Ilustración 7: Logo de Visual Paradigm

Visual Paradigm es una herramienta CASE que facilita la representación de los distintos componentes de un sistema software mediante diagramas UML. En este proyecto lo utilizaré para desarrollar los diagramas de casos de uso que me permitirán identificar los requisitos funcionales del sistema.

2.2.2. REDMINE

Ilustración 8: Logo de Redmine

Redmine es una aplicación web de gestión de proyectos de código abierto. Integrado junto al plugin *Agile*, lo utilizaré para realizar la planificación de las versiones y el seguimiento de las tareas del proyecto *(Lesyuk, A., (2013)*.

2.2.3. MICROSOFT WORD

Ilustración 9: Logo de Microsoft Word

Microsoft Word es uno de los programas de procesador de texto más extendidos por el mundo. En mi proyecto lo utilizaré para definir las plantillas de los distintos requisitos del sistema y las baterías de pruebas de aceptación de cada versión.

2.2.4. GIT

Ilustración 10: Logo de git

Git es un software de control de versiones que permite distribuir el código en distintas ramas que se combinan y separan según el estado de implementación del proyecto, facilitando la implementación y mantenimiento del código. Lo utilizaré para gestionar el control de versiones de la aplicación.

La estrategia de ramificación que seguiré en este proyecto se centrará en las versiones planificadas: cada vez que se inicie una nueva versión de la aplicación se creará una rama, cuando esa versión esté terminada y probada se fusionará con la rama principal del proyecto, teniendo en todo momento en esa rama software funcional sin errores de la última versión implementada. Esta estrategia permite tener un software estable en una rama mientras que en otra rama se está realizando la implementación de nuevas características. Además, gracias a la flexibilidad de este control de versiones se puede volver a un estado anterior del código, pudiendo así corregir errores de una mala actualización del código.

2.2.5. GITHUB

Ilustración 11: Logo de GitHub

GitHub es una aplicación web que permite visualizar y gestionar el control de versiones del código desde una interfaz web. Lo utilizaré para gestionar las ramas del repositorio y los registros de los *commits* realizados.

2.2.6. VISUAL STUDIO CODE

Ilustración 12: Logo de Visual Studio Code

Visual Studio Code es un editor de código optimizado y gratuito que permite crear y depurar modernas aplicaciones web. Lo utilizaré como entorno de desarrollo del código de la aplicación.

2.2.7. MONGODB COMPASS

Ilustración 13: Logo de MongoDB Compass

MongoDB Compass es una interfaz gráfica de usuario que permite gestionar visualmente las bases de datos de MongoDB.

2.3. LENGUAJES

En lo referido a los lenguajes que utilizaré en el desarrollo de la aplicación se explican en las siguientes subsecciones.

2.3.1. JAVASCRIPT

Ilustración 14: Logo de JavaScript

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web *("JavaScript", 2019).* Lo utilizaré para implementar tanto la parte del *front-end* como la del *back-end*, permitiendo el desarrollo de gran parte del código con un único lenguaje. Esto será realizado siguiendo el marco de desarrollo MEAN.

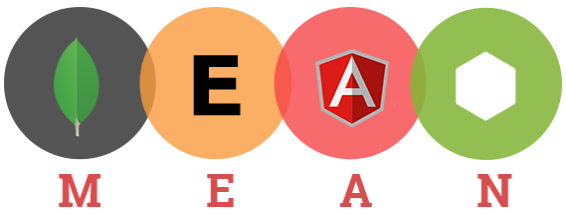


Ilustración 15: Logo de MEAN

Como se ha comentado anteriormente, MEAN está compuesto por varios componentes que tienen como nexo de unión el lenguaje, JavaScript en este caso. Sus elementos se explican a continuación:

- MongoDB: Es una base de datos distribuida, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube *("La base de datos líder del mercado para aplicaciones modernas", s.f.).*
- Express.js: Es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web, móviles y APIs *("Express - Infraestructura de aplicaciones web Node.js", s. f.).*

- Angular: Es un marco de desarrollo para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página, permitiendo crear de manera flexible el *front-end* de las aplicaciones *("Angular (framework)", s. f.)*.

- Node.js: Es un entorno en tiempo de ejecución de código abierto JavaScript, utilizado para la creación de aplicaciones de red a tiempo real, principalmente para la capa del servidor. En la actualidad su uso es ampliamente extendido por todo el mundo debido entre otras características a su ligereza y escalabilidad *("¿Qué es y para que sirve NodeJS?", 2015)*.

2.3.2. HTML

Ilustración 16: Logo de HTML

HTML (HyperText Markup Language) es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad *("HTML", 2019).* Utilizaré en el proyecto HTML5, la última versión de HTML, para definir la estructura y elementos de la aplicación.

2.3.3. CSS

Ilustración 17: Logo de CSS

CSS (Cascading Style Sheets) es el lenguaje utilizado para describir la presentación de documentos HTML o XML *("CSS", 2019)*. En el proyecto utilizaré CSS3, la última versión de CSS, para definir el estilo de las ventanas de la aplicación.

2.4. PLANIFICACIÓN

Una aproximación de la planificación del proyecto realizada antes de comenzar la implementación de la aplicación se muestra en la siguiente imagen.

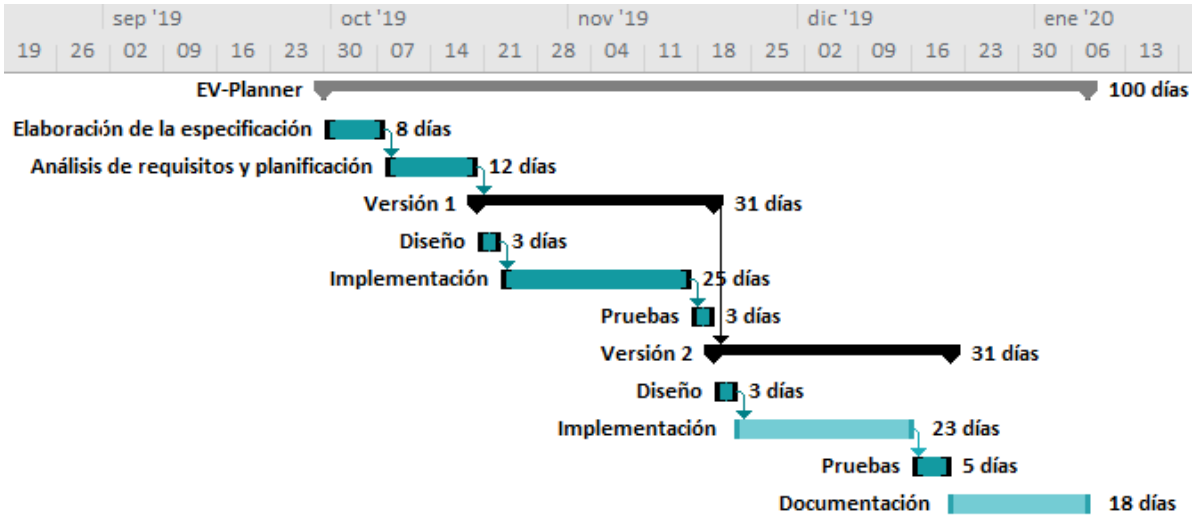


Ilustración 18: Planificación previa del proyecto

Como se puede observar en la imagen, la planificación previa del proyecto abarcaba una duración estimada de 100 días, incluyendo días laborables, festivos y fines de semana. Comenzando el día 30 de septiembre, el proyecto debería de estar terminado el 8 de enero. En lo referido al cómputo de horas he estimado, en base a un trabajo constante, un total de 308 horas, en el que de lunes a viernes se trabajaría 2 horas y sábados y domingos 6 horas.

Durante el desarrollo del proyecto, la planificación mostrada anteriormente ha sufrido varias modificaciones debido a elementos no contemplados como periodos vacacionales o imprevistos que han surgido durante su elaboración. Además, la documentación en lugar de realizarla al final del proyecto se ha realizado de manera paralela al desarrollo de la aplicación. En la siguiente imagen se muestra las etapas y tiempos reales del desarrollo del proyecto.

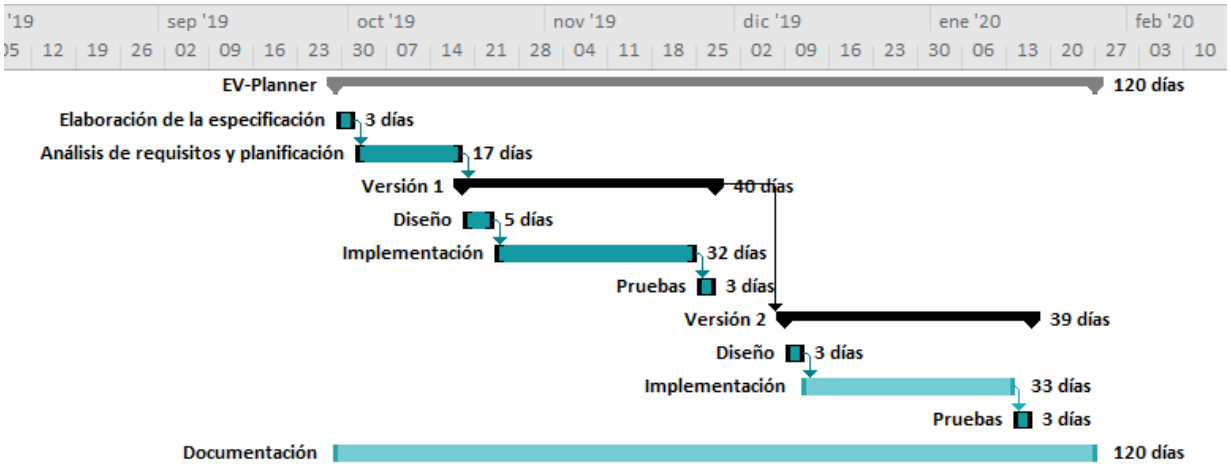


Ilustración 19: Planificación posterior del proyecto

Al final el proyecto ha durado 120 días, 20 días más de lo planificado, esto es debido a que en muchos de los días planificados originalmente no se han podido trabajar y algunos de ellos se ha trabajado menos tiempo del propuesto, extendiéndose con ello más en el tiempo. En realidad, se ha realizado menos horas por día de promedio, pero se han trabajado más días, por ello las horas empleadas efectivas han sido similares o superiores a las planificadas.

3. ESPECIFICACIÓN DE REQUISITOS

3.1. INFORMACIÓN DEL DOMINIO DEL PROBLEMA

EV-Planner es una aplicación destinada a la planificación de eventos públicos y privados, permitiendo a los organizadores gestionar sus detalles y al resto de usuarios unirse a ellos ya sea por invitación o buscándolos por su cuenta en caso de ser públicos.

Los eventos pueden estar ligados a ubicaciones, permitiendo a los usuarios consultar los eventos públicos más próximos a ellos.

Los tipos de eventos podrán ser variados como deportivos, sociales o de otra índole, permitiendo filtrar las búsquedas de eventos según los intereses del usuario.

Las cuentas de usuario estarán ligados a un correo electrónico donde recibirán las notificaciones de los eventos a los que este asociado en caso de que así se deseé.

3.2. OBJETIVOS DEL NEGOCIO

Los principales objetivos de la aplicación se pueden visualizar en las tablas mostradas en esta sección.

Tabla 1: OBJ 001 - Gestionar eventos

OBJ 001	Gestionar eventos
Versión	1.0
Descripción	El sistema deberá permitir gestionar todos los eventos de la aplicación.

Tabla 2: OBJ 002 - Gestionar usuarios

OBJ 002	Gestionar usuarios
Versión	1.0
Descripción	El sistema deberá permitir gestionar todos los usuarios registrados en la aplicación.

Tabla 3: OBJ 003 - Gestionar notificaciones

OBJ 003	Gestionar notificaciones
Versión	1.0
Descripción	El sistema deberá permitir gestionar el servicio de notificaciones de la aplicación.

Tabla 4: OBJ 004 - Gestionar información de la aplicación

OBJ 004	Gestionar información de la aplicación
Versión	1.0
Descripción	El sistema deberá permitir gestionar los métodos de contacto y la información que se muestra de la aplicación a los usuarios.

3.3. CASOS DE USO DEL SISTEMA

En esta sección se expone la especificación de los actores del sistema y las funcionalidades que pueden realizar en la aplicación, mediante el uso de diagramas de casos de uso y plantillas de casos de uso y actores.

3.3.1. DIAGRAMAS DE CASOS DE USO DEL SISTEMA

Los diagramas de casos de uso creados representan las funcionalidades que pueden realizar cada tipo de usuario en el sistema. Para facilitar su visualización se muestran en las siguientes imágenes el diagrama completo del sistema y los diagramas individuales de cada tipo de usuario.

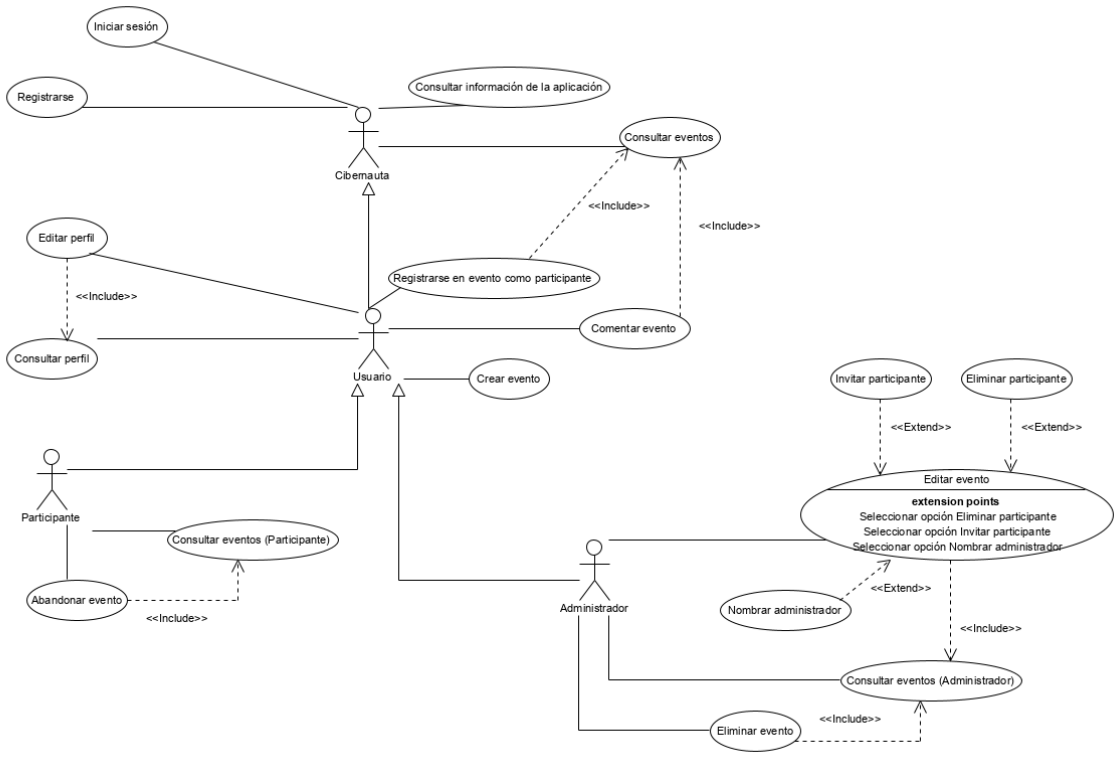


Ilustración 20: Diagrama de casos de uso del sistema

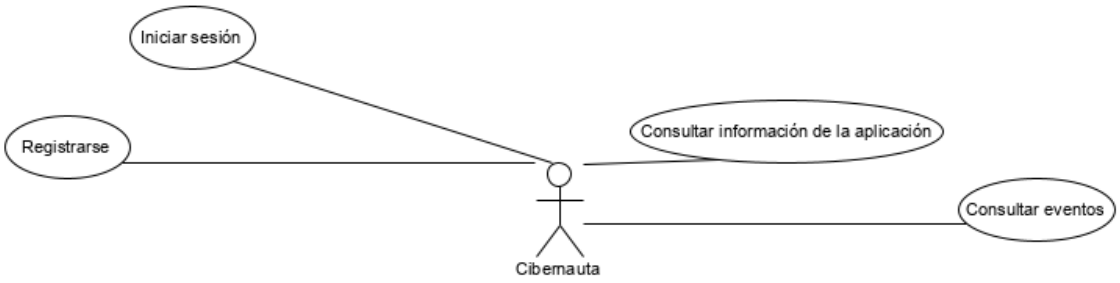


Ilustración 21: Diagrama de casos de uso del cibernauta

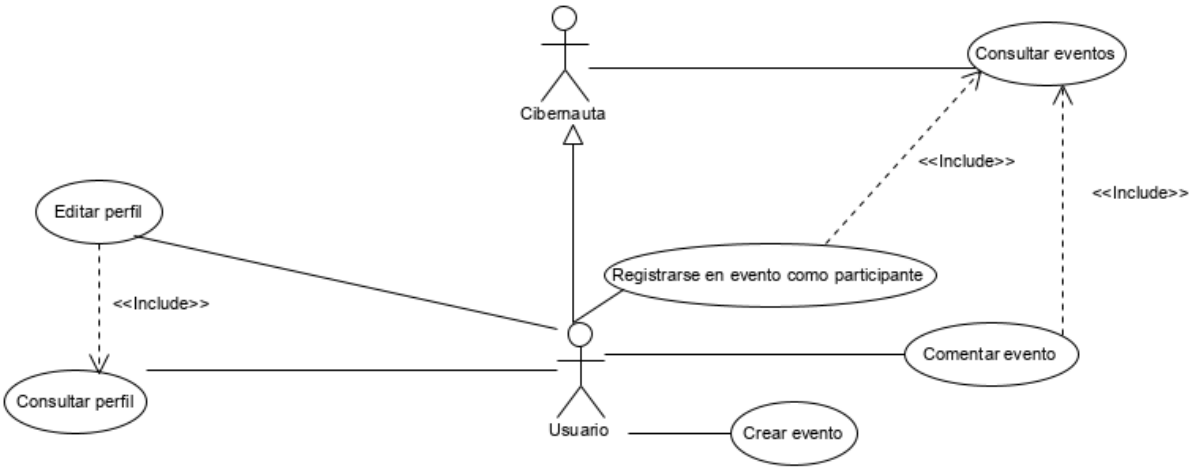


Ilustración 22: Diagrama de casos de uso del usuario

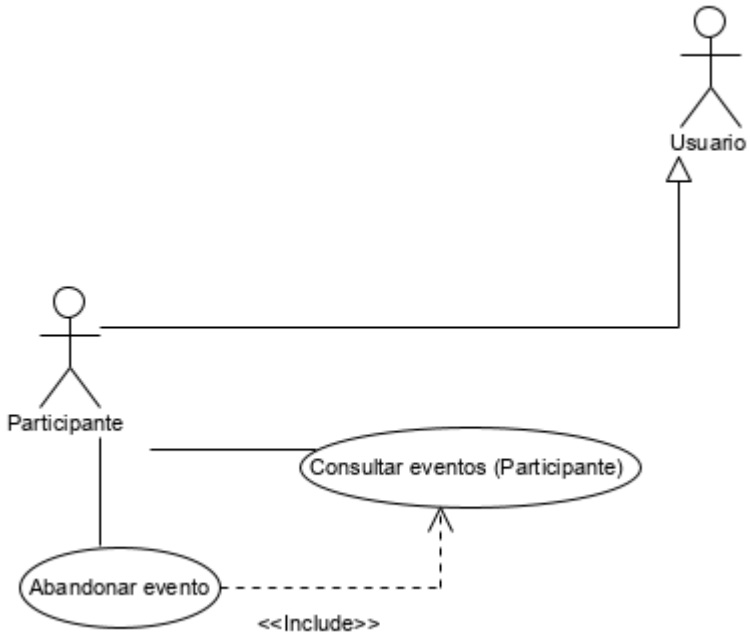


Ilustración 23: Diagrama de casos de uso del participante

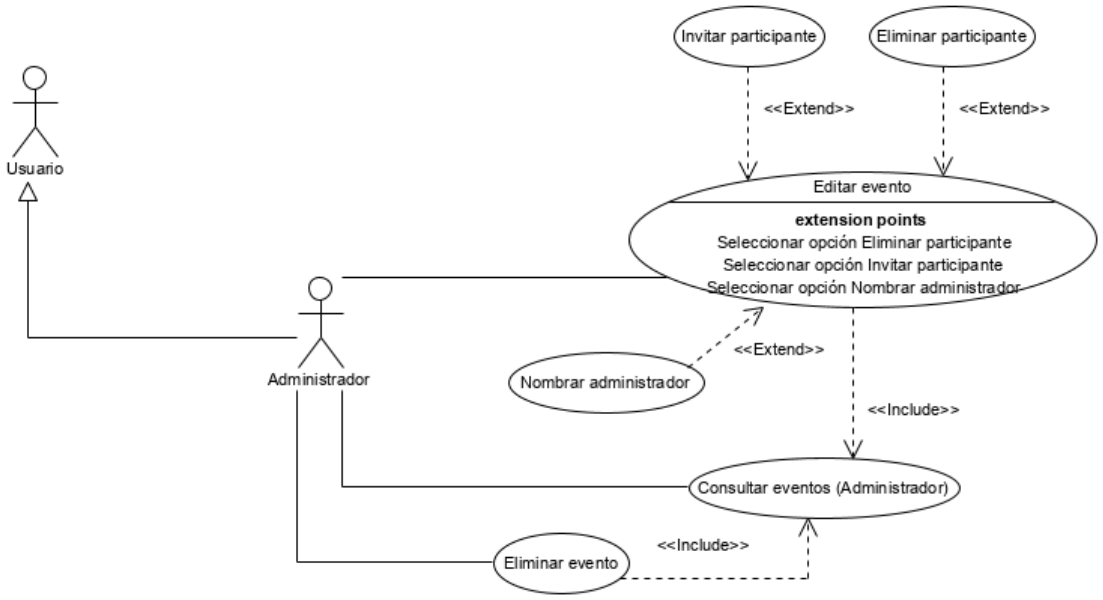


Ilustración 24: Diagrama de casos de uso del administrador

3.3.2. ESPECIFICACIÓN DE ACTORES DEL SISTEMA

Tabla 5: ACT 001 - Cibernauta

ACT 001	Cibernauta
Versión	1.0
Descripción	Este actor representa a todas las personas que navegan en la aplicación sin haber iniciado sesión.

Tabla 6: ACT 002 - Usuario

ACT 002	Usuario
Versión	1.0
Descripción	Este actor representa a todas las personas con cuentas en la aplicación que han iniciado sesión.

Tabla 7: ACT 003 - Administrador de evento

ACT 003	Administrador de evento
Versión	1.0
Descripción	Este actor representa a los usuarios que han organizado eventos en la aplicación.

Tabla 8: ACT 004 - Participante de evento

ACT 004	Participante de evento
Versión	1.0
Descripción	Este actor representa a los usuarios que se han inscrito como participantes de eventos en la aplicación.

3.3.3. ESPECIFICACIÓN DE CASOS DE USO DEL SISTEMA

Tabla 9: UC 001 - Registrarse

UC 001	Registrarse
Versión	1.0
Dependencias	OBJ 002 – Gestionar usuarios IRQ 001 - Información de los usuarios
Descripción	El sistema deberá permitir a los cibernautas registrarse en el sistema.
Precondición	Utilizar un correo y cuenta de usuario no existente en la aplicación.
Secuencia normal	1. El cibernauta presiona el botón de registrarse desde la pantalla de inicio de sesión de la aplicación. 2. El sistema muestra el formulario de registro. 3. El usuario rellena el formulario de registro con sus datos de acuerdo al requisito de información *IRQ 001 - Información de los usuarios* y presiona el botón de crear cuenta. 4. El sistema envía una notificación por correo al usuario con la confirmación de la creación de la nueva cuenta. 5. El sistema redirige al usuario a la ventana de inicio de sesión.
Postcondición	El nuevo usuario es creado en la aplicación.
Excepciones	-
Prioridad	Alta
Comentarios	Ninguno.

Tabla 10: UC 002 - Iniciar sesión

UC 002	Iniciar sesión
Versión	1.0
Dependencias	OBJ 002 – Gestionar usuarios IRQ 001 - Información de los usuarios
Descripción	El sistema deberá permitir a los usuarios registrados iniciar sesión en la aplicación.
Precondición	El usuario debe haberse registrado previamente en la aplicación.
Secuencia normal	1. El usuario rellena los campos de cuenta y contraseña desde la pantalla de inicio de la aplicación. 2. El usuario presiona el botón de iniciar sesión. 3. El sistema redirige al usuario a la ventana principal de la aplicación.
Postcondición	El usuario ha iniciado sesión en la aplicación.
Excepciones	3. Si el usuario y/o la contraseña son incorrectas aparece un mensaje en la ventana explicando que las credenciales introducidas no son válidas y no se inicia sesión en la aplicación.
Prioridad	Alta
Comentarios	Ninguno.

Tabla 11: UC 003 - Consultar perfil

UC 003	Consultar perfil
Versión	1.0
Dependencias	OBJ 002 – Gestionar usuarios IRQ 001 - Información de los usuarios
Descripción	El sistema deberá permitir a los usuarios del sistema consultar su perfil en la aplicación.
Precondición	El usuario debe haber iniciado sesión en la aplicación.
Secuencia normal	1. El usuario navega a la sección "Mi perfil". 2. La aplicación muestra los datos del perfil del usuario de acuerdo al requisito de información *IRQ 001 - Información de los usuarios*.
Postcondición	-
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 12: UC 004 - Editar perfil

UC 004	Editar perfil
Versión	1.0
Dependencias	OBJ 002 – Gestionar usuarios IRQ 001 - Información de los usuarios
Descripción	El sistema deberá permitir a los usuarios editar algunos de los datos de su perfil.
Precondición	El usuario debe haber iniciado sesión en la aplicación.
Secuencia normal	1. Se realiza el caso de uso *UC 003 – Consultar perfil.* 2. El usuario presiona el botón de editar perfil. 3. El usuario edita sus datos de perfil de acuerdo al requisito de información *IRQ 001 - Información de los usuarios*. 4. El usuario presiona el botón "Actualizar mis datos de perfil". 5. El sistema muestra la ventana del perfil de usuario con los nuevos datos actualizados.
Postcondición	Los datos del usuario se han actualizado con los datos introducidos por el usuario.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 13: UC 005 - Consultar eventos

UC 005	Consultar eventos
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir consultar a los usuarios todos los eventos públicos del sistema.
Precondición	El usuario debe haber iniciado sesión en la aplicación.
Secuencia normal	1. El usuario navega a la sección de eventos. 2. El sistema muestra varios eventos registrados en la aplicación. 3. Si el usuario cambia uno de los filtros de búsqueda, el sistema muestra los correspondientes eventos en la aplicación. 4. El usuario selecciona un evento. 5. El sistema muestra los datos del evento.
Postcondición	-
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 14: UC 006 - Registrar en evento como participante

UC 006	Registrarse en evento como participante
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios registrar como participante en los eventos públicos o aquellos a los que ha sido invitado que dispongan de plazas libres de inscripción.
Precondición	El usuario debe haber iniciado sesión en la aplicación y el evento consultado debe tener plazas libres.
Secuencia normal	1. Se realiza el caso de uso *UC 005 - Consultar eventos.* 2. El usuario selecciona la opción inscribirse como participante. 3. El sistema muestra los detalles del evento con los datos del nuevo participante actualizado.
Postcondición	El participante es registrado en el evento.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 15: UC 007 - Comentar evento

UC 007	Comentar evento
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios de la aplicación realizar comentarios en los eventos.
Precondición	El usuario debe haber iniciado sesión en la aplicación.
Secuencia normal	1. Se realiza el caso de uso *UC 005 - Consultar eventos.* 2. El usuario accede a la sección de comentarios. 3. El usuario escribe un comentario y presiona el botón de enviar. 4. El sistema muestra la sección de comentario con el nuevo comentario añadido.
Postcondición	El comentario es registrado en el evento.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 16: UC 008 - Crear evento

UC 008	Crear evento
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios crear eventos.
Precondición	El usuario debe haber iniciado sesión en la aplicación.
Secuencia normal	1. El usuario accede a la sección de eventos. 2. El sistema muestra varios eventos registrados en la aplicación. 3. El usuario presiona el botón de crear evento. 4. El sistema muestra el formulario de creación de evento de acuerdo al requisito de información *IRQ 002 - Información de los eventos.* 5. El usuario rellena el formulario y presiona el botón de creación de evento. 6. El sistema redirige al usuario a una ventana que muestra los datos del nuevo evento creado.
Postcondición	El evento es registrado en el sistema y el usuario creador es registrado como administrador del evento.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 17: UC 009 - Consultar eventos (Participante)

UC 009	Consultar eventos (Participante)
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios consultar los eventos en los que está inscrito como participante.
Precondición	El usuario debe haber iniciado sesión en la aplicación y estar inscrito como participante en al menos un evento.
Secuencia normal	1. El usuario navega a la sección “Mis eventos”. 2. El sistema muestra los eventos en los que el usuario esta registrado como participante o administrador. 3. Si el usuario cambia uno de los filtros de búsqueda, el sistema muestra los correspondientes eventos en la aplicación. 4. El usuario selecciona un evento. 5. El sistema muestra los datos del evento.
Postcondición	-
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 18: UC 010 - Abandonar evento

UC 010	Abandonar evento
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios abandonar los eventos en los que participa.
Precondición	El usuario debe haber iniciado sesión en la aplicación y estar inscrito como participante en al menos un evento.
Secuencia normal	1. Se lleva a cabo el caso de uso *UC 009 – Consultar eventos (Participante).* 2. El usuario presiona el botón de abandonar evento. 3. El sistema redirige al usuario a la sección “Mis eventos”.
Postcondición	El usuario ya no está registrado en el evento.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 19: UC 011 - Consultar eventos (Administrador)

UC 011	Consultar eventos (Administrador)
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios consultar los eventos en los que está inscrito como administrador.
Precondición	El usuario debe haber iniciado sesión en la aplicación y estar inscrito como administrador en al menos un evento.
Secuencia normal	1. El usuario navega a la sección "Mis eventos". 2. El sistema muestra los eventos en los que el usuario esta registrado como participante o administrador. 3. Si el usuario cambia uno de los filtros de búsqueda, el sistema muestra los correspondientes eventos en la aplicación. 4. El usuario selecciona un evento. 5. El sistema muestra los datos del evento.
Postcondición	-
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 20: UC 012 - Nombrar administrador

UC 012	Nombrar administrador
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los administradores de un evento nombrar a otros administradores durante su edición (*UC 013 – Editar evento)*.
Precondición	El usuario debe haber iniciado sesión en la aplicación, estar inscrito como administrador en al menos un evento y haber seleccionado la opción de nombrar administrador durante el caso de uso *UC 013 – Editar evento*.
Secuencia normal	1. El sistema muestra una lista con los participantes del evento. 2. El usuario selecciona a un participante y confirma que desea hacerlo administrador.
Postcondición	Cuando se termine la edición del evento el participante es registrado como administrador.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 21: UC 013 - Editar evento

UC 013	Editar evento
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios editar los eventos en los que está registrado como administrador
Precondición	El usuario debe haber iniciado sesión en la aplicación y estar inscrito como administrador en al menos un evento.
Secuencia normal	1. Se realiza el caso de uso *UC 011 – Consultar eventos (Administrador).* 2. El usuario presiona el botón de editar evento. 3. El sistema muestra un formulario con los datos del evento de acuerdo con el requisito de información *IRQ 002 - Información de los eventos.* 4. El usuario edita los campos del formulario. 5. El usuario presiona el botón de guardar los datos del evento se actualizan.
Postcondición	Los datos del evento se han actualizado.
Excepciones	4. a. Si el usuario presiona el botón de invitar participante se realiza el caso de uso *UC 014 – Invitar participante.* b. Si el usuario presiona el botón de eliminar participante se realiza el caso de uso *UC 015 – Eliminar participante.* c. Si el usuario presiona el botón de nombrar administrador se realiza el caso de uso *UC 013 – Nombrar administrador.*
Prioridad	Media
Comentarios	Ninguno.

Tabla 22: UC 014 - Invitar participante

UC 014	Invitar participante
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los administradores de un evento invitar a participantes durante su edición (*UC 013 – Editar evento)*.
Precondición	El usuario debe haber iniciado sesión en la aplicación, estar inscrito como administrador en al menos un evento y haber seleccionado la opción de invitar participante durante el caso de uso *UC 013 – Editar evento*.
Secuencia normal	1. El sistema muestra un cuadro de búsqueda de usuarios. 2. El usuario busca al usuario que desea invitar y lo selecciona. 3. El sistema muestra en el listado de participantes el nuevo usuario agregado.
Postcondición	Cuando se termine la edición del evento, el usuario seleccionado es registrado como participante.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 23: UC 015 - Eliminar participante

UC 015	Eliminar participante
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los administradores de un evento eliminar a los participantes que no sean administradores durante su edición (*UC 013 – Editar evento)*.
Precondición	El usuario debe haber iniciado sesión en la aplicación, estar inscrito como administrador en al menos un evento y haber seleccionado la opción de eliminar participante durante el caso de uso *UC 013 – Editar evento*.
Secuencia normal	1. El sistema muestra un listado con los participantes. 2. El usuario confirma el participante (no administrador) que desea eliminar.
Postcondición	Cuando se termine la edición del evento el usuario seleccionado es eliminado del evento.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 24: UC 016 - Eliminar evento

UC 016	Eliminar evento
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos IRQ 002 - Información de los eventos
Descripción	El sistema deberá permitir a los usuarios eliminar los eventos en los que está registrado como único administrador.
Precondición	El usuario debe haber iniciado sesión en la aplicación y estar inscrito como único administrador en al menos un evento.
Secuencia normal	1. Se realiza el caso de uso *UC 011 – Consultar eventos (Administrador).* 2. El usuario presiona el botón de editar evento. 3. El sistema muestra un formulario con los datos del evento de acuerdo al requisito de información *IRQ 002 - Información de los eventos.* 4. El usuario presiona el botón de eliminar evento.
Postcondición	El evento es eliminado del sistema.
Excepciones	-
Prioridad	Media
Comentarios	Ninguno.

Tabla 25: UC 017 - Consultar información de la aplicación

UC 017	Consultar información de la aplicación
Versión	1.0
Dependencias	OBJ 004 – Gestionar información de la aplicación IRQ 003 - Información de la aplicación
Descripción	El sistema deberá permitir a los usuarios consultar la información de la aplicación.
Precondición	-
Secuencia normal	1. El usuario navega hasta la sección “Contacto”. 2. El sistema muestra la información de la aplicación de acuerdo al requisito de información *IRQ 003 - Información de la aplicación*.
Postcondición	-
Excepciones	-
Prioridad	Media
Comentarios	Información adicional de la aplicación también estará disponible en el *footer* y en la sección “Sobre nosotros”.

3.4. REQUISITOS DE INFORMACIÓN DEL SISTEMA

En las plantillas de esta sección se detallan que tipo de datos se deben gestionar en la aplicación y que campos lo conforman.

Tabla 26: IRQ 001 - Información de los usuarios

IRQ 001	Información de los usuarios
Versión	1.0
Dependencias	OBJ 001 – Gestionar usuarios
Descripción	El sistema deberá almacenar información correspondiente a los usuarios del sistema. En concreto:
Datos específicos	▪ Nombre ▪ Apellidos ▪ Correo electrónico ▪ Nombre de cuenta ▪ Localización
Comentarios	Ninguno.

Tabla 27: IRQ 002 - Información de los eventos

IRQ 002	Información de los eventos
Versión	1.0
Dependencias	OBJ 001 – Gestionar eventos
Descripción	El sistema deberá almacenar información correspondiente a los eventos del sistema. En concreto:
Datos específicos	▪ Nombre ▪ Descripción ▪ Localización ▪ Participantes ▪ Categoría ▪ Administradores ▪ Imágenes ▪ Fecha y hora de comienzo ▪ Fecha y hora de fin ▪ Aforo máximo ▪ Tipo (público o privado) ▪ Requisitos/Materiales necesarios
Comentarios	Ninguno.

Tabla 28: IRQ 003 - Información de la aplicación

IRQ 003	Información de la aplicación
Versión	1.0
Dependencias	OBJ 004 – Gestionar información de la aplicación
Descripción	El sistema deberá gestionar la información relativa a la aplicación. En concreto:
Datos específicos	▪ Nombre ▪ Descripción ▪ Correo de contacto ▪ Número de teléfono ▪ Redes sociales asociadas ▪ Logo ▪ Términos y políticas de uso
Comentarios	Ninguno.

3.5. REQUISITOS NO FUNCIONALES DEL SISTEMA

En esta sección se detallan que requisitos no funcionales se deben tener en cuenta durante el desarrollo de la aplicación.

Tabla 29: RNF 001 - Usabilidad - Manual de usuario

RNF 001	Usabilidad - Manual de usuario
Versión	1.0
Descripción	El sistema debe contar con un manual de usuario que contenga detalladamente las funciones del sistema.
Prioridad	Media
Comentarios	Ninguno.

Tabla 30: RNF 002 - Usabilidad - Mensajes de usuario

RNF 002	Usabilidad - Mensajes de usuario
Versión	1.0
Descripción	El sistema deberá mostrar mensajes de confirmación o de advertencia en determinadas operaciones en las que interactúe con el sistema.
Prioridad	Media
Comentarios	Ninguno.

Tabla 31: RNF 003 - Usabilidad – Diseño adaptativo

RNF 003	Usabilidad – Diseño adaptativo
Versión	1.0
Descripción	El sistema deberá disponer de un diseño adaptativo que permita su correcta visualización en ordenadores, tabletas y móviles.
Prioridad	Media
Comentarios	Ninguno.

Tabla 32: RNF 004 - Internalización – Idiomas

RNF 004	Internalización – Idiomas
Versión	1.0
Descripción	La interfaz de la aplicación deberá estar disponible en inglés y español.
Prioridad	Media
Comentarios	No se incluirán traducciones de elementos que cree el usuario como el nombre de los eventos o sus descripciones.

4. VERSIONES DE LA APLICACIÓN

4.1. GESTIÓN DE REQUISITOS

Como se ha comentado en la sección 2.1. Etapas, se ha pensado dividir el desarrollo de la aplicación en dos versiones funcionales, dejándola preparada para continuar su desarrollo como posible trabajo futuro.

Toda la gestión de los requisitos se ha llevado a cabo en la aplicación Redmine, como se explica en la sección 2.2.2. Redmine. El tiempo de desarrollo de cada una de las versiones es aproximadamente de un mes, intentando agrupar en cada una de ellas características que estén relacionadas, priorizando las que tienen mayor urgencia o son necesarias para realizar otras.

Antes de comenzar la implementación de las primeras versiones de la aplicación he planificado una versión preliminar que contiene tareas relacionadas con la preparación del entorno de trabajo. En las siguientes imágenes se muestran las tareas y los tiempos planificados.

#	Tipo	Estado	Prioridad	Asunto	Categoría	Fecha de inicio	Fecha fin
138	Tareas	Resuelta	Normal	Elaboración de especificación del proyecto	Análisis de requisitos	2019-09-29	2019-10-18
137	Tareas	Resuelta	Normal	Configuración de entorno de trabajo	Herramientas	2019-09-29	2019-09-30

Ilustración 25: Tareas de la versión de preparación

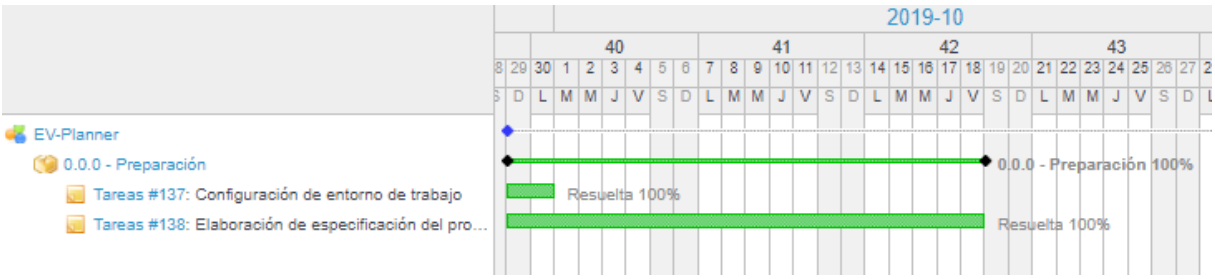


Ilustración 26: Diagrama de Gantt de la versión de preparación

La versión de preparación incluye las tareas de configuración del entorno de trabajo y la elaboración inicial de la especificación del proyecto. La versión comenzó el 29 de septiembre y termino el 18 de octubre.

En el apartado de planificación de las secciones 4.2. Versión 1.0. y 4.3. Versión 2.0. se expone con más detalle que requisitos están vinculados a cada una de las versiones.

4.2. VERSIÓN 1.0.

4.2.1. PLANIFICACIÓN

Las tareas escogidas para formar parte de la versión 1.0 han sido seleccionadas en base a la prioridad que se ha establecido en cada una de ellas y a las dependencias que se deben cumplir durante la implementación de dichas características. Por ello, se han escogido tareas que están relacionadas entre sí, como es el caso de la gestión inicial de usuarios y eventos.

El orden de realización de tareas se ha realizado en base al motivo comentado anteriormente, debido a ello las primeras tareas en realizarse han sido las relacionadas con el registro de usuario e inicio de sesión, posteriormente la consulta de datos del usuario y edición de su perfil y finalmente la consulta, creación y edición inicial de eventos. También se ha añadido una tarea relacionada con la información de la aplicación, en la que se ha implementado la estructura inicial de las ventanas que tendrá las primeras versiones de la aplicación. Esta tarea se ha realizado de manera paralela al resto de tareas.

En la siguiente imagen se muestran los requisitos dados de alta en la herramienta de gestión de requisitos para esta versión.

#	Tipo	Estado	Prioridad	Asunto	Categoría	Fecha de inicio	Fecha fin
155	Tareas	Resuelta	Normal	Consultar información de la aplicación	Global	2019-11-03	2019-11-24
140	Tareas	Resuelta	Alta	Inicio de sesión de usuarios	Gestión de usuarios	2019-10-24	2019-11-03
139	Tareas	Resuelta	Alta	Registro de usuarios	Gestión de usuarios	2019-10-24	2019-11-03
141	Tareas	Resuelta	Normal	Consulta de perfil de usuario	Gestión de usuarios	2019-11-03	2019-11-09
142	Tareas	Resuelta	Normal	Edición de perfil de usuario	Gestión de usuarios	2019-11-04	2019-11-12
146	Tareas	Resuelta	Normal	Creación de evento	Gestión de eventos	2019-11-13	2019-11-18
143	Tareas	Resuelta	Normal	Consulta de eventos	Gestión de eventos	2019-11-13	2019-11-18
150	Tareas	Resuelta	Normal	Edición de evento	Gestión de eventos	2019-11-19	2019-11-23

Ilustración 27: Tareas de la versión 1.0.

4.2.2. DISEÑO

Antes de comenzar la fase de implementación, se ha realizado un diseño previo de las ventanas que tendrá esta versión de la aplicación. El diseño de las ventanas de registro e inicio de sesión se ha realizado mediante la herramienta de diseño online *creately*, el resto de las ventanas se ha realizado en HTML, permitiendo en algunos casos aprovechar componentes del diseño en la fase de implementación. A continuación, se muestran las imágenes de las ventanas diseñadas.

La ventana de inicio en un principio se diseñó para que se mostraran imágenes de eventos que fueran cambiando con el tiempo. En el caso de que el usuario no hubiera iniciado sesión se mostrarían dos enlaces para que iniciara sesión o se registrara en la aplicación.

Ilustración 28: Diseño - Ventana de inicio

Al registrarse el usuario, además de introducir sus datos, tendría la opción de acceder a los términos y políticas de la aplicación.

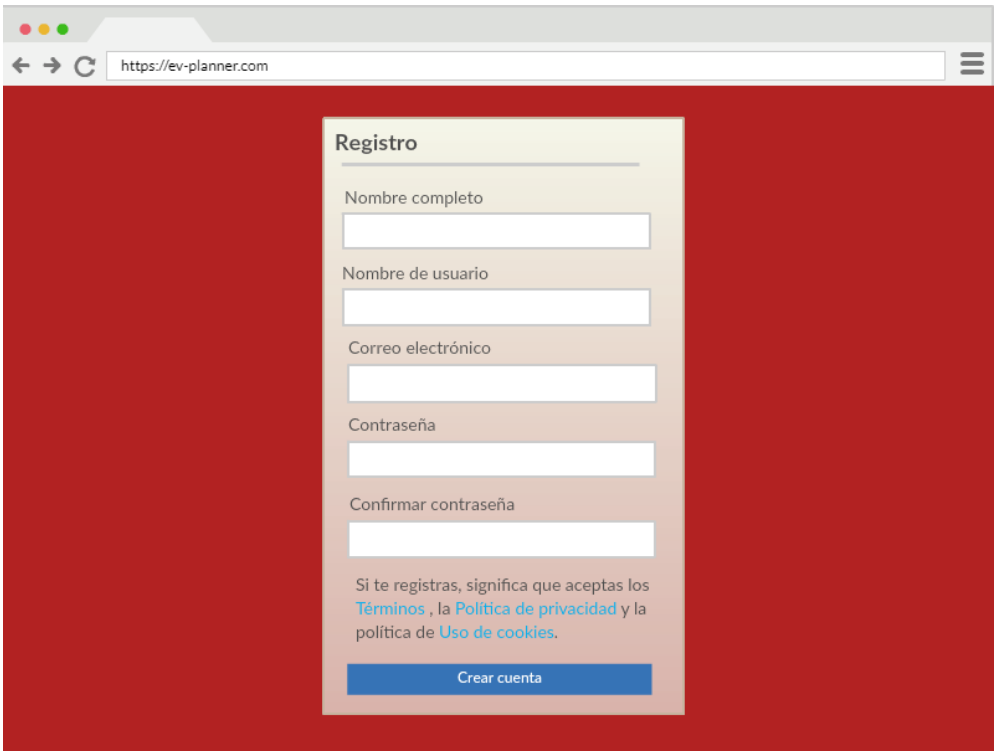


Ilustración 29: Diseño - Ventana de registro

En lo referido al inicio de sesión las credenciales utilizadas serían nombre de usuario y contraseña.

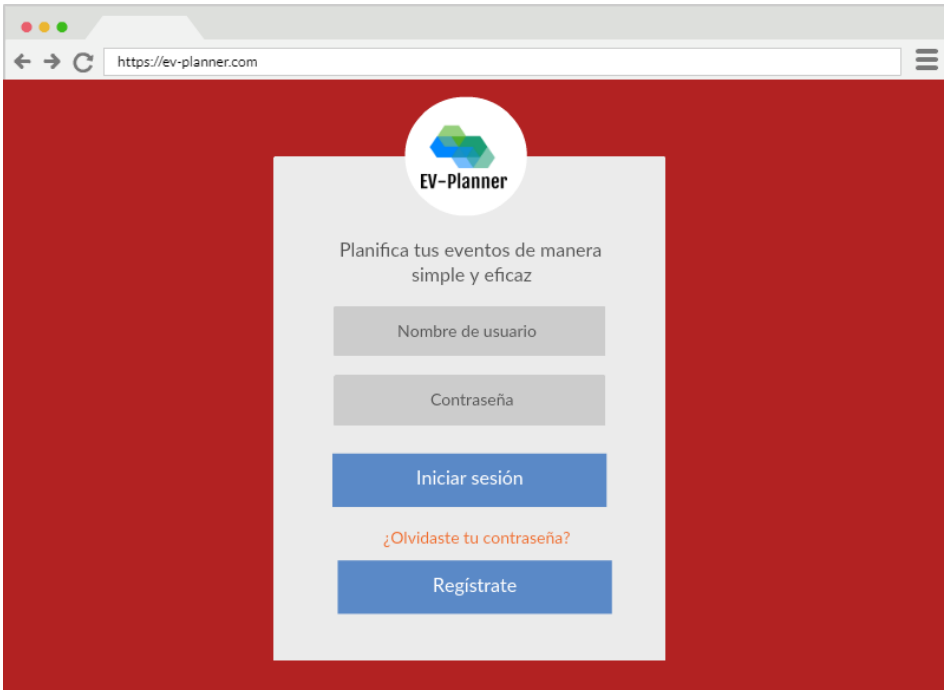


Ilustración 30: Diseño - Ventana de inicio de sesión

La ventana de perfil de usuario mostraría los datos que introdujo el usuario al registrarse y algunos datos adicionales que permitan completar su perfil.

Ilustración 31: Diseño - Ventana de perfil de usuario

Al seleccionar la opción de editar perfil, se mostrarían los mismos datos del perfil para que se puedan editar y además se permitiría el cambio de su contraseña.

Ilustración 32: Diseño - Ventana de edición de perfil

Desde la sección de eventos se podría consultar todos los eventos que se han registrado en el sistema.

Ilustración 33: Diseño - Ventana de listado de eventos

Al crear un nuevo evento, se dispondría de un formulario que se debería rellenar con los detalles de dicho evento.

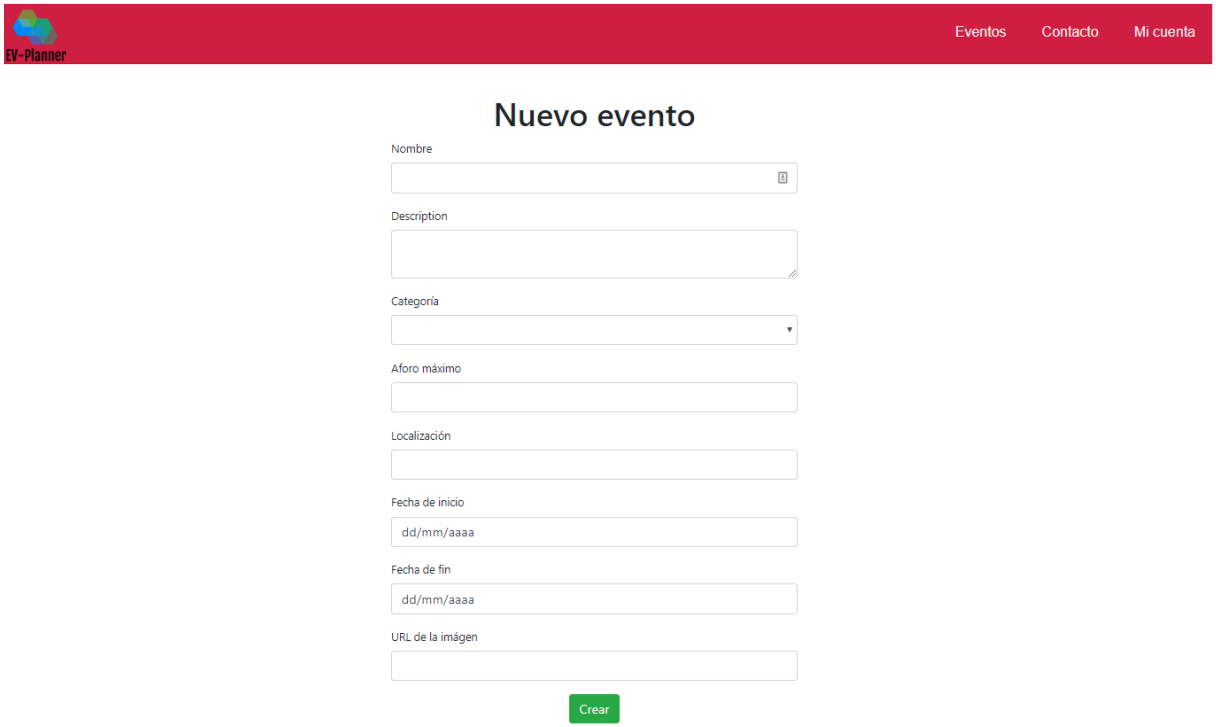


Ilustración 34: Diseño - Ventana de creación de evento

Desde la ventana del evento se podrían consultar los detalles del evento introducidos durante su creación.

Ilustración 35: Diseño - Ventana de detalles de evento

4.2.3. IMPLEMENTACIÓN

El proyecto está formado principalmente por tres elementos, que se relacionan y comunican entre sí:

- APP: Es el programa final con el que el usuario interactúa, en este caso a través de un navegador web. La implementación de la APP incluye el desarrollo de toda la parte de *front-end* de la aplicación, la parte de *back-end* de despliegue del servicio y la comunicación con la API para obtener los datos.

- API: Es la interfaz que comunica los datos registrados en la base de datos y las aplicaciones. El acceso a parte de las consultas a la API estará restringido para que solo puedan operar sobre ellas las aplicaciones autorizadas.

- Base de datos: Es un conjunto de datos agrupados y almacenados, en este caso en forma de documentos en una base de datos no relacional (MongoDB).

En la siguiente imagen se puede observar una simplificación de la relación entre los elementos mencionados anteriormente.

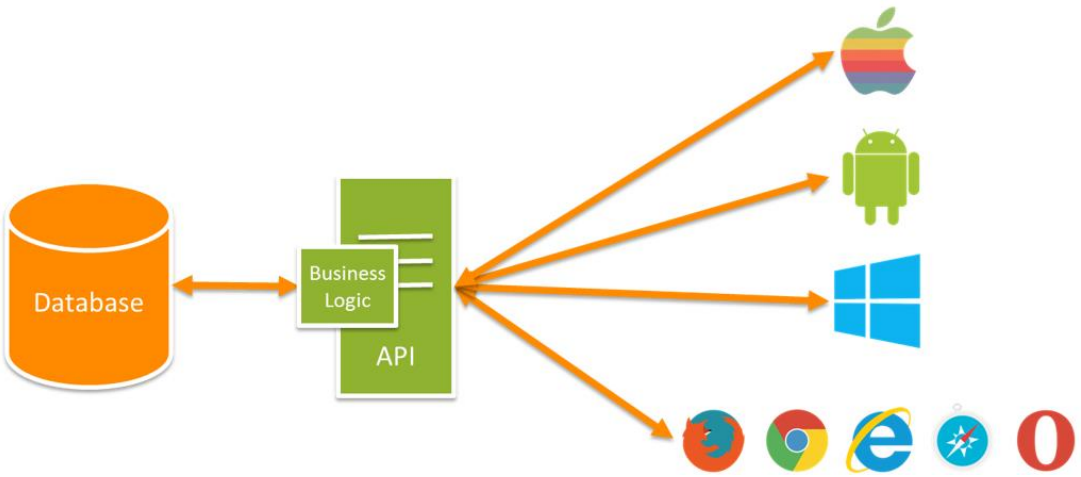


Ilustración 36: Relación APP, API y BD

En lo referido a la estructura del código, he procurado dividirlo en distintos módulos para favorecer su mantenimiento.

Por un lado, la API, tienes tres módulos principales:

- Archivos de configuración: En ellos se registran variables del entorno, como puede ser el puerto en el que se despliega el servicio o la clave secreta y tiempo de expiración de los tokens de acceso (JWT).

- Rutas: En estos archivos se establecen las rutas de las consultas que se pueden realizar a la API y que función de ejecución tienen asociadas.

- Modelos de datos: Estos archivos contienen las estructuras de los datos que gestiona la API. En la siguiente imagen se muestra un parte del modelo de datos de los usuarios.

```
var userSchema = new mongoose.Schema({
    fullName: {
        type: String,
        required: 'Nombre completo no puede estar vacío'
    },
    userName: {
        type: String,
        required: 'Nombre de usuario no puede estar vacío'
    },
    publicInfo: {
        type: String
    },
    email: {
        type: String,
        required: 'Dirección de correo electrónico no puede estar vacío',
        unique: true
    },
    password: {
        type: String,
        required: 'La contraseña no puede estar vacía',
        minlength: [4, 'La contraseña debe tener al menos 4 carácteres de longitud']
    },
    saltSecret: String
});
```

Ilustración 37: Ejemplo de modelo de datos

- Controladores: Contienen las funciones encargadas de operar con la base de datos para realizar consultas, registros, eliminaciones o actualizaciones de datos.

En la siguiente imagen puede verse la estructura del código de la primera versión de la API.

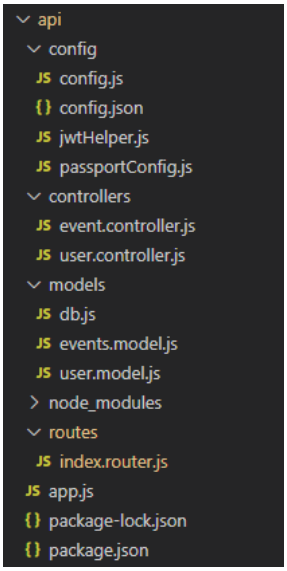


Ilustración 38: Estructura del código de la API (Versión 1)

Por otro lado, el código de la aplicación tiene cinco elementos que me gustaría destacar:

- Módulos de configuración: Contienen las declaraciones e importaciones de los módulos y componentes de la aplicación.
- Modelos de datos: Contienen los campos de los datos que gestiona la aplicación.
- Servicios: Contienen las funciones que definen las peticiones que se realizan a la API.
- Rutas: Contienen las direcciones de las páginas de la aplicación y los componentes que tienen asociados.
- Componentes: La parte del *front-end* de la aplicación se ha desarrollado con Angular y su filosofía de creación y reutilización de componentes, respetando el patrón MVC (Modelo, Vista, Controlador). Cada componente se divide esencialmente en tres partes: diseño (css), estructura (html) y controlador (ts). El controlador se comunica con el servicio para obtener los datos que después se muestran en la vista. En la siguiente imagen se muestra un ejemplo de la estructura de los componentes en la aplicación.

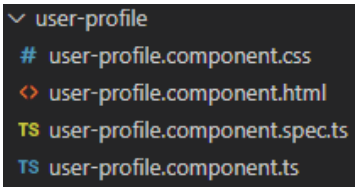


Ilustración 39: Estructura de componente

En la siguiente imagen puede verse parte de la estructura del código de la primera versión de la aplicación.

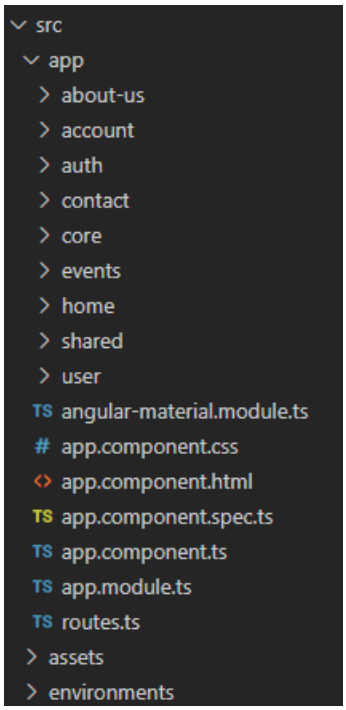


Ilustración 40: Estructura del código de la APP (Versión 1)

A continuación, se explican cómo se han implementado varias de las características de esta versión de la aplicación.

4.2.3.1. SEGURIZACIÓN DE LA API MEDIANTE JWT

Una de las partes más importantes de prácticamente cualquier aplicación que se publica en el mercado es la seguridad, especialmente la referida a los datos que se gestionan.

En la aplicación que he desarrollado, el acceso a los datos se gestiona mediante una API. Para asegurar que el acceso a los datos lo realice únicamente el personal autorizado, he implementado un sistema de seguridad basado en JWT (JSON Web Token).

Un JWT está representado por tres cadenas: encabezado, contenido y firma, como se muestra en la siguiente imagen.

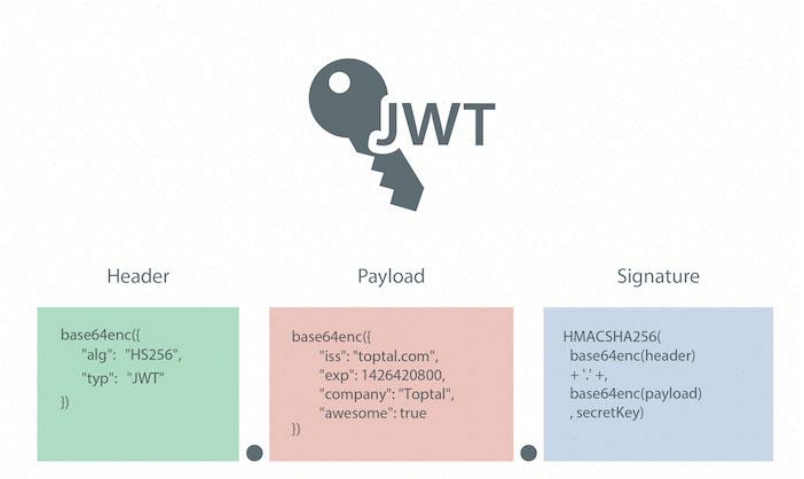


Ilustración 41: Cadenas que forman JWT

Cuando un usuario inicia sesión en la aplicación, la API comprueba que sus credenciales son correctas y devuelve un token con fecha de expiración. Mientras ese token este vigente el usuario podrá realizar peticiones a la API adjuntando dicho token *(Vaati, 2018)*. En el siguiente diagrama se muestra el funcionamiento explicado.

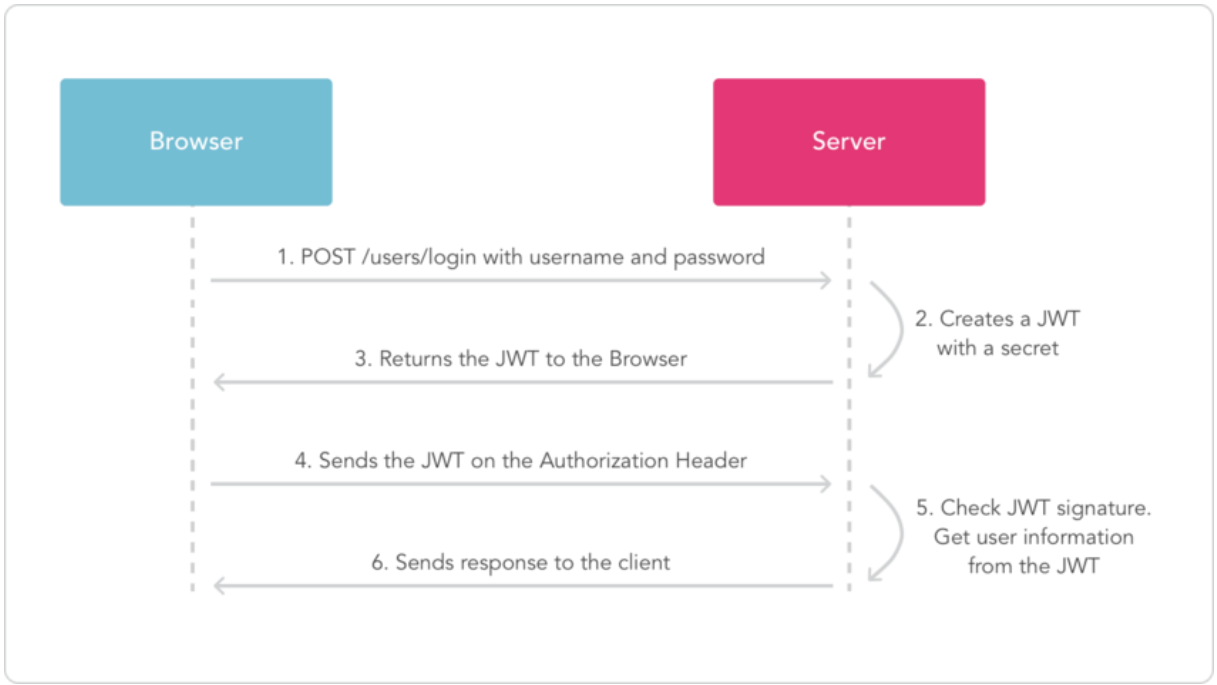


Ilustración 42: Diagrama de funcionamiento JWT

A continuación, se explica cómo se ha implementado JWT en el sistema y su funcionamiento

Cuando el usuario intenta iniciar sesión en la aplicación se llama a la ruta */authenticate* de la API, que tiene una función asociada encargada de verificar si el usuario está registrado en el sistema.

```
router.post('/authenticate', ctrlUser.authenticate);
```

Ilustración 43: Código - Ruta de autentificación en la API (JWT)

```
module.exports.authenticate = (req, res, next) => {
    // call for passport authentication
    passport.authenticate('local', (err, user, info) => {
        // error from passport middleware
        if (err) return res.status(400).json(err);
        // registered user
        else if (user) return res.status(200).json({ "token": user.generateJwt() });
        // unknown user or wrong password
        else return res.status(404).json(info);
    })(req, res);
}
```

Ilustración 44: Código - Función de autentificación de usuario

Si el usuario está registrado en el sistema se genera un token, indicando una clave secreta y un tiempo de expiración, y se envía como respuesta de la petición de autentificación para que se utilice en futuras consultas a la API.

```
userSchema.methods.generateJwt = function () {
    return jwt.sign({ _id: this._id},
        process.env.JWT_SECRET,
    {
        expiresIn: process.env.JWT_EXP
    });
}
```

Ilustración 45: Código - Generación de JWT

En el archivo que gestiona las rutas de la API se pueden proteger las rutas que se deseen mediante una función de verificación que compruebe que en la cabecera de la petición realizada a la API contenga el token JWT.

```
router.get('/userProfile',jwtHelper.verifyJwtToken, ctrlUser.userProfile);
```

Ilustración 46: Código - Ruta protegida por JWT

```
module.exports.verifyJwtToken = (req, res, next) => {
    var token;
    if ('authorization' in req.headers)
        token = req.headers['authorization'].split(' ')[1];

    if (!token)
        return res.status(403).send({ auth: false, message: 'No token provided.' });
    else {
        jwt.verify(token, process.env.JWT_SECRET,
            (err, decoded) => {
                if (err)
                    return res.status(500).send({ auth: false, message: 'Token authentication failed.' });
                else {
                    req._id = decoded._id;
                    next();
                }
            }
        )
    }
}
```

Ilustración 47: Código - Función de verificación de JWT

En la aplicación, cuando un usuario inicia sesión y se recibe el token, se almacena dicho token el almacenamiento local. Cuando se realizar una petición a la API, si se requiere utilizar el token, se adjunta a su cabecera. En el caso de que el token haya expirado o no se haya generado se redirige al usuario a la página de inicio de sesión.

```
intercept(req: HttpRequest<any>, next: HttpHandler) {

    if (req.headers.get('noauth'))
        return next.handle(req.clone());
    else {
        const clonedreq = req.clone({
            headers: req.headers.set("Authorization", "Bearer " + this.userService.getToken())
        });
        return next.handle(clonedreq).pipe(
            tap(
                event => { },
                err => {
                    if (err.error.auth == false) {
                        this.router.navigateByUrl('/login');
                    }
                })
        );
    }
}
```

Ilustración 48: Código - Interceptor de peticiones a la API

4.2.3.2. REGISTRO DE USUARIOS

Primero, en la API se ha creado un modelo con los campos necesarios para que un usuario se registre y sus restricciones asociadas.

```
var userSchema = new mongoose.Schema({
    fullName: {
        type: String,
        required: 'Nombre completo no puede estar vacío'
    },
    userName: {
        type: String,
        required: 'Nombre de usuario no puede estar vacío'
    },
    publicInfo: {
        type: String
    },
    email: {
        type: String,
        required: 'Dirección de correo electrónico no puede estar vacío',
        unique: true
    },
    password: {
        type: String,
        required: 'La contraseña no puede estar vacía',
        minlength: [4, 'La contraseña debe tener al menos 4 carácteres de longitud']
    },
    saltSecret: String
});
```

Ilustración 49: Modelo de datos de usuario

Cuando se llama a la ruta */registerUser* de la API se ejecuta una función encargada de registrar al usuario según el modelo mostrado en la imagen anterior.

```
router.post('/registerUser', ctrlUser.register);
```

Ilustración 50: Ruta - Registro de usuario

```
module.exports.register = (req, res, next) => {

    var user = new User();
    user.fullName = req.body.fullName;
    user.userName = req.body.userName;
    user.email = req.body.email;
    user.password = req.body.password;

    user.save((err, doc) => {
        if (!err)
            res.send(doc);
        else {
            if (err.code == 11000)
                res.status(422).send(['Dirección de correo electrónico ya existente.']);
            else
                return next(err);
        }
    });
}
```

Ilustración 51: Función de registro de usuario

Para aumentar la seguridad de los datos, cuando se va a registrar a un usuario en la base de datos se encripta la contraseña antes de que eso suceda.

```
userSchema.pre('save', function (next) {
    bcrypt.genSalt(10, (err, salt) => {
        bcrypt.hash(this.password, salt, (err, hash) => {
            this.password = hash;
            this.saltSecret = salt;
            next();
        });
    });
});
```

Ilustración 52: Encriptación de contraseña

Para realizar la encriptación de la contraseña he usado bcrypt, una función que se encarga de transformar la contraseña en un código hash. Este hash es generado incluyendo una cadena aleatoria denominada salt, ello asegura que dos contraseñas iguales no estén registradas en la base de datos con el mismo valor y con ello poder aumentar la seguridad frente a ataques por fuerza bruta *(Vicente, 2017)*.

Finalmente, se ha creado un formulario en la aplicación con los campos necesarios para que un usuario se registre, en este caso nombre completo, nombre de usuario, correo electrónico y contraseña.

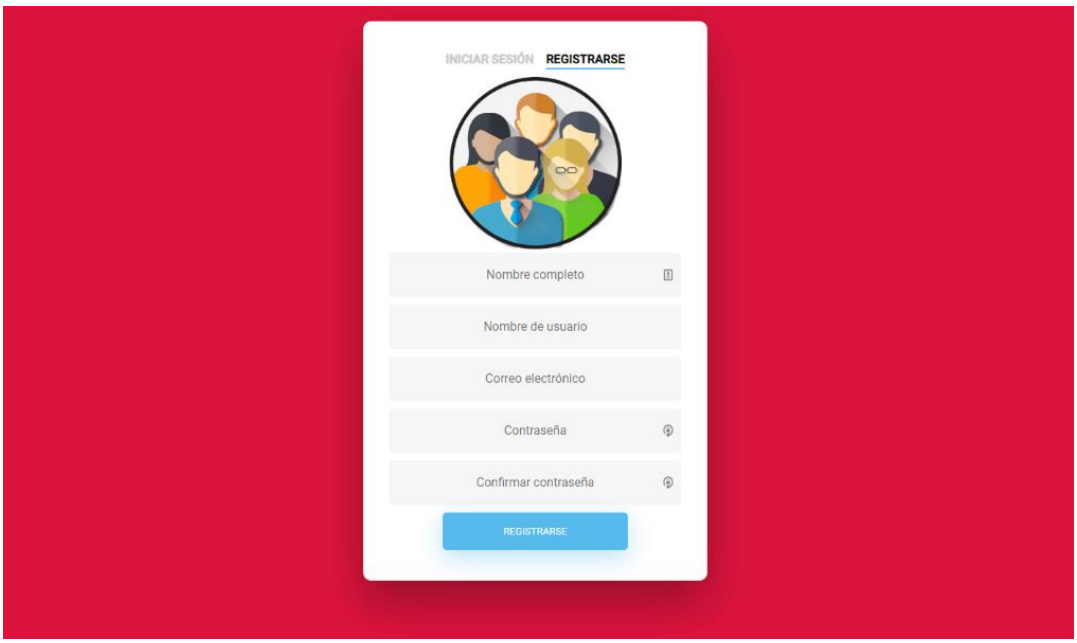


Ilustración 53: Formulario de registro de usuarios (Versión 1)

Cuando el usuario pulsa el botón de registrarse los datos del formulario, se envían a la API y el usuario se guarda en la base de datos. En el caso de que los datos introducidos no cumplan con las restricciones definidas en el modelo de datos, se mostrarían los detalles de los errores en el formulario.

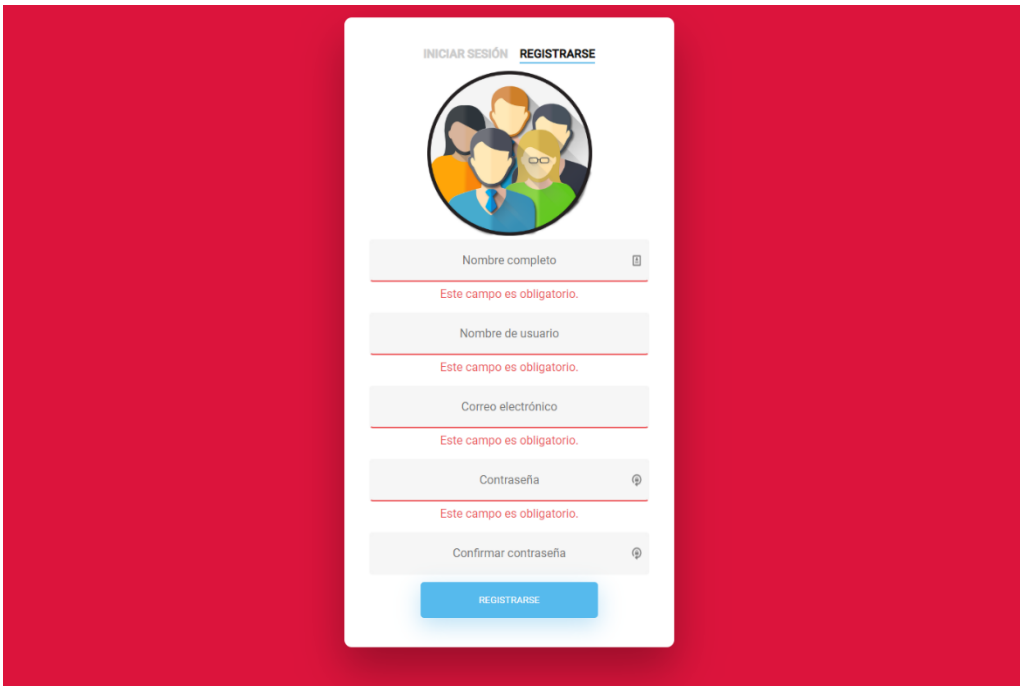


Ilustración 54: Formulario de registro con errores

4.2.3.3. INICIO DE SESIÓN

Para permitir iniciar sesión a los usuarios en la aplicación, he implementado en la API un sistema de autentificación mediante el middleware *Passport*.

El sistema de autentificación que he decidido implementar está basado en una estrategia local en la que se valida que las credenciales de los usuarios corresponden a los registrados en el sistema, en este caso correo electrónico y contraseña. Cuando la aplicación llama a la ruta */authenticate* de la API se ejecuta su función asociada que realiza la autentificación del usuario basándose en la estrategia definida.

```
router.post('/authenticate', ctrlUser.authenticate);
```

Ilustración 55: Ruta - Inicio de sesión

```
module.exports.authenticate = (req, res, next) => {
    // call for passport authentication
    passport.authenticate('local', (err, user, info) => {
        // error from passport middleware
        if (err) return res.status(400).json(err);
        // registered user
        else if (user) return res.status(200).json({ "token": user.generateJwt() });
        // unknown user or wrong password
        else return res.status(404).json(info);
    })(req, res);
}
```

Ilustración 56: Función de inicio de sesión

```
passport.use(
    new localStrategy({ usernameField: 'email' },
        (username, password, done) => {
            User.findOne({ email: username },
                (err, user) => {
                    if (err)
                        return done(err);
                    // unknown user
                    else if (!user)
                        return done(null, false, { message: 'Correo electrónico no registrado' });
                    // wrong password
                    else if (!user.verifyPassword(password))
                        return done(null, false, { message: 'Contraseña incorrecta.' });
                    // authentication succeeded
                    else
                        return done(null, user);
                });
        })
);
```

Ilustración 57: Estrategia de autentificación

Como se puede observar en la imagen que muestra la estrategia de autentificación, la verificación de la contraseña se realiza con una función diferente, esto es debido a que esta encriptada con la función *bcrypt*, como se explicó en la sección anterior, 4.2.3.2. Registro de usuarios, y solo se puede desencriptar con el mismo tipo de algoritmo, como se muestra en la siguiente imagen.

```
userSchema.methods.verifyPassword = function (password) {
    return bcrypt.compareSync(password, this.password);
};
```

Ilustración 58: Función de verificación de contraseña

Si la autentificación del usuario es exitosa, la API genera un token que permite a los usuarios realizar operaciones que necesitan autentificación, tal como se explica en la sección 4.2.3.1. Segurización de la API mediante JWT.

En la aplicación se ha creado un formulario que permite a los usuarios introducir sus credenciales para iniciar sesión en la aplicación, en este caso correo electrónico y contraseña.

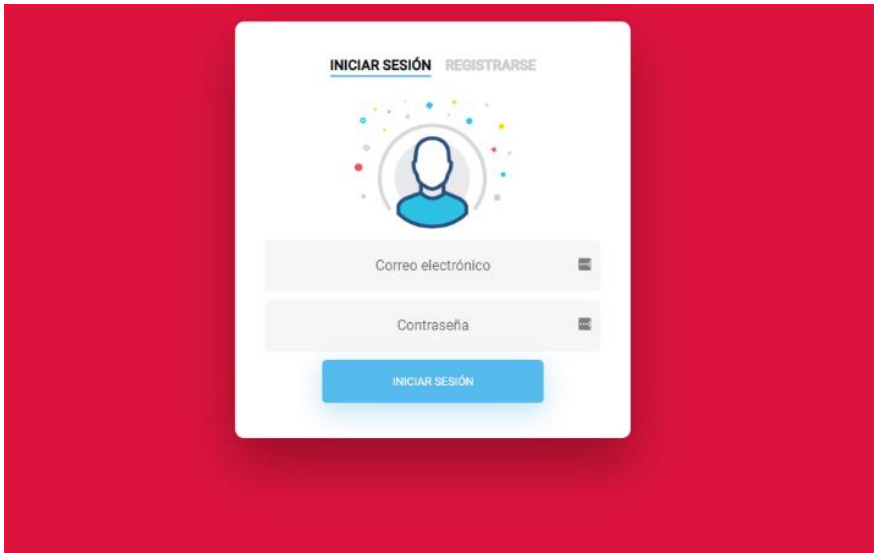


Ilustración 59: Formulario de inicio de sesión

Si el usuario introduce credenciales inválidas, se muestran los errores que se han producido en el mismo formulario. En cambio, si las credenciales son correctas se redirige al usuario a la ventana que muestra los datos de su perfil.

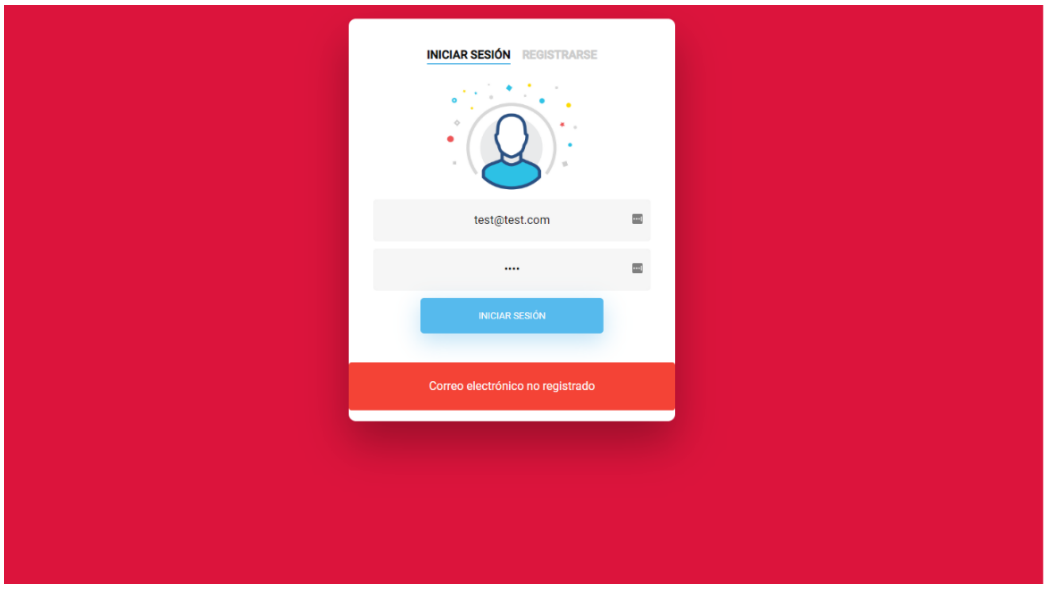


Ilustración 60: Formulario de inicio de sesión con errores

4.2.3.4. CONSULTA Y EDICIÓN DE PERFIL

Las funciones de consulta y edición de perfil están protegidas en la API y solo están disponibles cuando el usuario inicia sesión en la aplicación.

Para el caso de la consulta de los datos del perfil del usuario, se ha establecido en la API la ruta */userProfile*. Su función asociada se encarga de buscar al usuario pasado como parámetro con la petición y recuperar los datos de su perfil, en este caso nombre completo, nombre de usuario, información pública y correo electrónico.

```
router.get('/userProfile',jwtHelper.verifyJwtToken, ctrlUser.userProfile);
```

Ilustración 61: Ruta - Perfil de usuario

```
module.exports.userProfile = (req, res, next) => {
    User.findOne({ _id: req._id },
        (err, user) => {
            if (!user)
                return res.status(404).json({ status: false, message: 'Usuario no encontrado.' });
            else
                return res.status(200).json({ status: true,
                    user: _.pick(user, ['_id', 'fullName', 'userName', 'publicInfo', 'email']) });
        }
    );
}
```

Ilustración 62: Función de consulta de datos de usuario

En lo referido a la actualización de los datos de los usuarios, tras comprobar que el usuario existe, se cambian los valores de sus datos por los pasados como parámetro, y al igual que ocurría al registrar un usuario en la base de datos, los campos se tienen que validar para que cumplan las restricciones del modelo.

```
router.put('/updateUser',jwtHelper.verifyJwtToken,  ctrlUser.updateUser);
```

Ilustración 63: Ruta - Actualización de usuario

```
module.exports.updateUser = (req, res, next) => {
    User.updateOne({ _id: req._id },
        {
            $set: {
                fullName: req.body.fullName,
                userName: req.body.userName,
                publicInfo: req.body.publicInfo,
                email: req.body.email
            }
        },
        (err, user) => {
            if (!user) {
                return res.status(404).json({ status: false, message: 'Usuario no encontrado.' });
            } else if (!err) {
                res.send(user);
            }
            else {
                if (err.code == 11000)
                    res.status(422).send(['Dirección de correo electrónico ya existente.']);
                else
                    return next(err);
            }

        });
}
```

Ilustración 64: Función de actualización de usuario

Desde la aplicación, tras navegar a la ventana que muestra los datos del perfil de usuario, el controlador de la vista llama a la función encargada de comunicarse con la API y obtener los datos del usuario que ha iniciado sesión en el sistema.

```
ngOnInit() {
  this.userService.getUserProfile().subscribe(
    res => {
      this.userDetails = res['user'];
    },
    err => {
      console.log(err);
    }
  );
}
```

Ilustración 65: Carga de datos de perfil de usuario

Esos datos se vinculan con los campos del formulario del perfil de usuario con una técnica denominada *Data Binding*, que consiste en sincronizar los datos entre el controlador y la vista, de tal manera que, si en uno de ellos cambia el valor de los datos, en la otra parte se actualiza con el nuevo valor *(Díaz, 2016)*.

```
<input type="text" #fullName="ngModel" [(ngModel)]="userDetails.fullName" name="fullName"
    placeholder="Nombre completo" required class="form-control here"
    [ngClass]="{'invalid-textbox' :updateProfileForm.submitted && !fullName.valid }">
```

Ilustración 66: Ejemplo de campo sincronizado mediante *Data Binding*

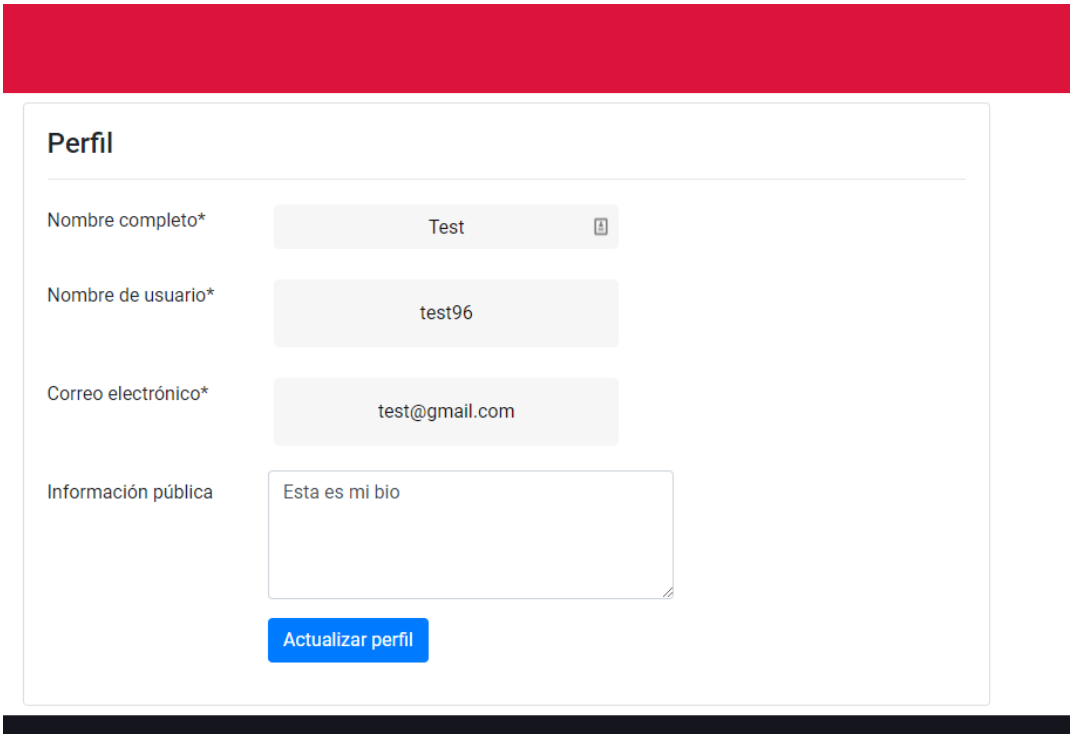


Ilustración 67: Formulario de perfil de usuario

Cuando se presiona el botón de actualizar perfil, el controlador envía una petición a la API con los nuevos valores del usuario para que se actualicen.

```
onSubmit(form: NgForm) {
  this.userService.putUser(form.value).subscribe(
    res => {
      this.showSucessMessage = true;
      setTimeout(() => this.showSucessMessage = false, 4000);
    },
    err => {
      if (err.status === 422) {
        this.serverErrorMessages = err.error.join('<br/>');
      }
      else
        this.serverErrorMessages = 'Algo fue mal.Por favor, contacte con el administrador.';
    }
  );
}
```

Ilustración 68: Función de envío de datos de usuario a la API

4.2.3.5. GESTIÓN DE EVENTOS

La gestión de eventos implementada en la versión 1.0 comprende las funciones de consulta, creación y actualización de eventos.

El modelo definido para los datos de los eventos se muestra en la siguiente imagen.

```
var eventSchema = new mongoose.Schema({
    name: {
        type: String,
        required: 'Nombre no puede estar vacío'
    },
    description: {
        type: String,
        required: 'Descripción no puede estar vacía'
    },
    start_date: {
        type: String,
        required: 'Fecha de inicio no puede estar vacía'
    },
    end_date: {
        type: String,
        required: 'Fecha de fin no puede estar vacía'
    },
    photo:{
        type: String
    },
    location:{
        type: String
    },
    type: {
        type: String,
        required: 'Tipo no puede estar vacío'
    }
});
```

Ilustración 69: Modelo de datos de evento

Las operaciones de gestión de eventos funcionan de manera similar a las de gestión de usuarios explicadas anteriormente. La principal diferencia es el modelo de datos que tienen vinculados dichas operaciones.

Desde la sección de eventos, se pueden visualizar todos los eventos que se han registrado en el sistema.

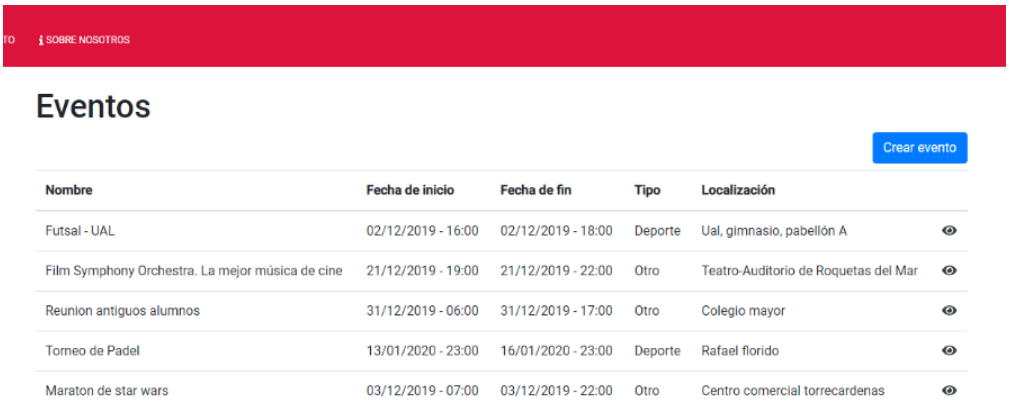


Ilustración 70: Lista de eventos

Si se selecciona el icono de visualización de uno de los eventos de la lista, se muestran los detalles del evento, si en cambio, se pulsa el botón de crear evento se redirige al usuario al formulario de creación de evento.

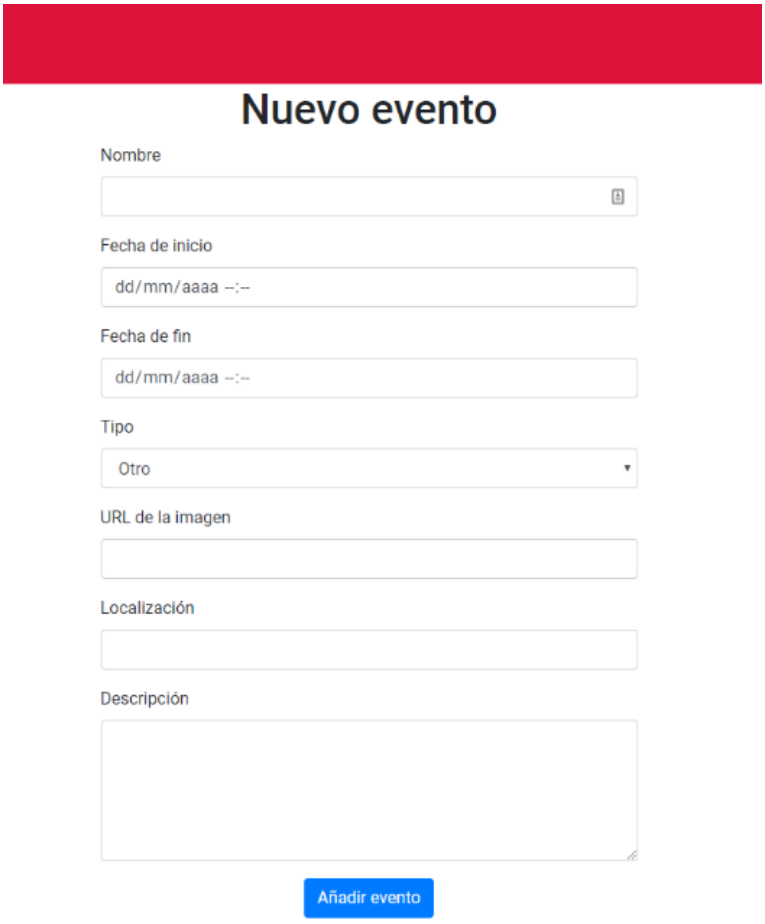


Ilustración 71: Formulario de creación de evento

Para la primera versión, los campos que componen el registro de un evento son: nombre, fecha de inicio, fecha de fin, tipo de evento, URL de una imagen, localización del evento (en formato texto) y descripción.

La ventana que muestra los detalles del evento en esta primera versión se puede observar en la siguiente imagen.

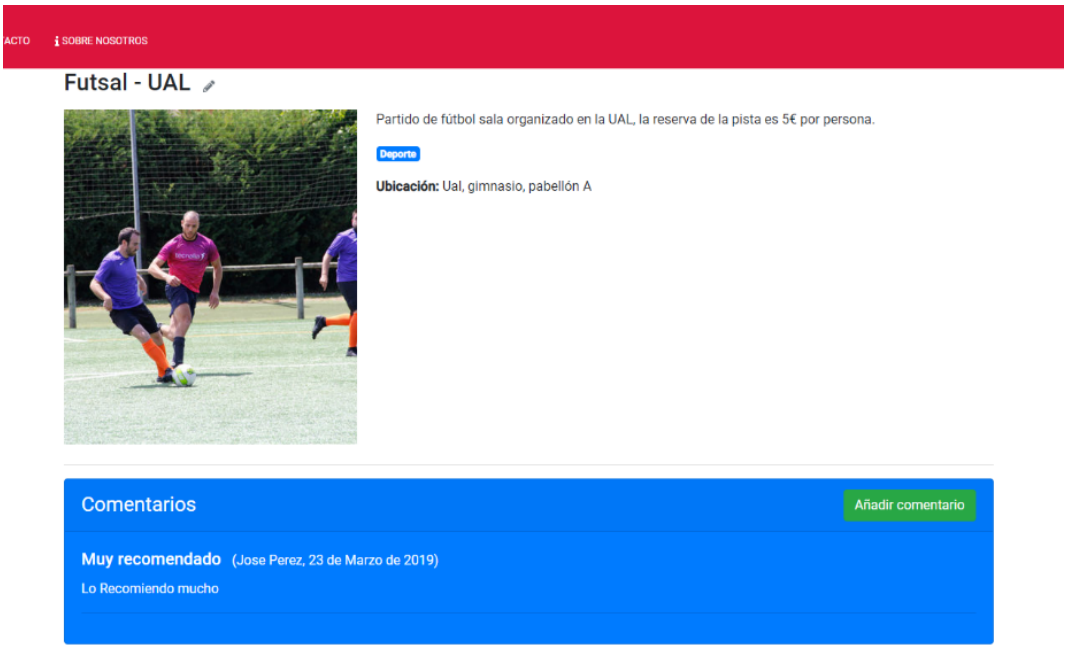


Ilustración 72: Detalles de un evento

Como se puede observar, aunque la gestión de comentarios en esta versión no está implementada, si se ha creado su diseño inicial.

Las consultas de la información de los eventos no están protegidas por un token, de esta forma las aplicaciones que lo deseen pueden consultar estos datos.

```
router.get('/events', ctrlEvent.events);
router.get('/events/id/:id', ctrlEvent.eventInfo);
```

Ilustración 73: Rutas - Consultas de eventos

Si se presiona el icono de edición desde la ventana de detalles del evento, se muestra un formulario con los datos del evento precargados. Si se modifican sus campos y se presiona el botón con el texto “Actualizar evento”, la información del evento es actualizada.

Ilustración 74: Formulario de edición de evento

Los formularios de creación y edición de eventos son en realidad un mismo componente que se ha reutilizado. Dependiendo de la ruta que se escoja en la aplicación se cargara un formulario vacío o un formulario con los datos de un evento precargado.

```
{
    path: 'events/id/:id/update', component: EventFormComponent
},
{
    path: 'events/new', component: EventFormComponent
},
```

Ilustración 75: Rutas que cargan el formulario de eventos

Si la ruta contiene el id de un evento, se consulta a la API para obtener sus detalles y poder mostrarlos en el formulario.

```
if (this.rutaActiva.snapshot.params.id != null) {
  this.edition = true;
  this.id_event = this.rutaActiva.snapshot.params.id;
  this.eventService.getEventInfo(this.id_event).subscribe(
    res => {
      this.eventService.selectedEvent = res['event'];
      this.start_date = this.eventService.selectedEvent.start_date;
      this.end_date = this.eventService.selectedEvent.end_date;
    },
```

Ilustración 76: Consulta de los datos de un evento

Cuando se termina de rellenar el formulario, si se había navegado desde la ruta de creación, se realiza una operación *post*, si en cambio se había navegado desde una ruta de actualización, se realiza una operación *put.*

```
if (this.creation) {
  this.eventService.postEvent(form.value).subscribe(
    res => {
      this.showSucessMessage = true;
      setTimeout(() => this.showSucessMessage = false, 2000);
      this.resetForm(form);
      setTimeout(() => this.router.navigateByUrl('/events/id/' + res['_id']),2000);
    },
```

Ilustración 77: Registro de un evento

```
else if (this.edition) {
form.value["_id"] = this.id_event;
this.eventService.putEvent(form.value).subscribe(
  res => {
    this.showSucessMessage = true;
    setTimeout(() => this.showSucessMessage = false, 4000);
    this.resetForm(form);
    setTimeout(() => this.router.navigateByUrl('/events/id/' + res['_id']),4000);
  },
```

Ilustración 78: Actualización de un evento

4.2.4. PRUEBAS

En esta sección se muestran las plantillas de pruebas de aceptación utilizadas en esta versión.

Tabla 33: Prueba - Registro de usuario (Correcto)

Nombre	Registro de usuario (Correcto)	**Versión**	1.0
Dependencias	UC 001 - Registrarse		
Descripción	El usuario debe poder registrarse en la aplicación, utilizando un correo electrónico no existente en el sistema y una contraseña de más de 4 caracteres.		
Pasos	1. El usuario navega a la sección de registro. 2. El usuario rellena el formulario de registro. 3. El usuario pulsa el botón “Registrarse”.		
Datos introducidos	• Nombre completo: “Francisco Fernández Sánchez” • Nombre de usuario: “fran96” • Correo electrónico: “fran96@gmail.com” • Contraseña: “Test_96” • Confirmar contraseña: “Test_96”		
Resultado esperado	El usuario es registrado en el sistema. Aparece un aviso confirmando el registro del usuario en la aplicación.		
Resultado obtenido	El usuario se almacena en la base de datos del sistema y en la aplicación aparece un aviso con el mensaje “Registro realizado correctamente”.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 34: Prueba - Registro de usuario (Incorrecto – Campos en blanco)

Nombre	Registro de usuario (Incorrecto – Campos en blanco)	**Versión**	1.0
Dependencias	UC 001 - Registrarse		
Descripción	El sistema debe impedir el registro de un usuario, si los datos introducidos no cumplen las restricciones del sistema, en este caso no se admiten campos en blanco.		
Pasos	1. El usuario navega a la sección de registro. 2. El usuario pulsa el botón “Registrarse”.		
Datos introducidos	Ninguno.		
Resultado esperado	El sistema no registra al usuario en la base de datos. Aparece un aviso en la aplicación indicando que no se admiten campos en blancos.		
Resultado obtenido	El sistema no registra al usuario en la base de datos. Debajo de cada campo del formulario aparece el mensaje “Este campo es obligatorio”.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 35: Prueba - Registro de usuario (Incorrecto – Correo electrónico existente)

Nombre	Registro de usuario (Incorrecto – Correo electrónico existente)	**Versión**	1.0
Dependencias	UC 001 - Registrarse		
Descripción	El sistema debe impedir el registro de un usuario, si los datos introducidos no cumplen las restricciones del sistema, en este caso el correo electrónico no debe de estar registrado en el sistema.		
Pasos	1. El usuario navega a la sección de registro. 2. El usuario rellena el formulario de registro. 3. El usuario pulsa el botón “Registrarse”.		
Datos introducidos	• Nombre completo: “Fran Fernández Soria” • Nombre de usuario: “fran96” • Correo electrónico: “fran96@gmail.com” • Contraseña: “Test_96” • Confirmar contraseña: “Test_96”		
Resultado esperado	El usuario no es registrado en el sistema. Aparece un aviso informando de que el correo electrónico ya esta registrado en el sistema.		
Resultado obtenido	El usuario no se almacena en la base de datos del sistema y en la aplicación aparece un aviso con el mensaje “Dirección de correo electrónico ya existente.”.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 36: Prueba - Registro de usuario (Incorrecto – Contraseña demasiado corta)

Nombre	Registro de usuario (Incorrecto – Contraseña demasiado corta)	**Versión**	1.0
Dependencias	UC 001 - Registrarse		
Descripción	El sistema debe impedir el registro de un usuario, si los datos introducidos no cumplen las restricciones del sistema, en este caso la contraseña debe tener más de 3 caracteres.		
Pasos	1. El usuario navega a la sección de registro. 2. El usuario rellena el formulario de registro. 3. El usuario pulsa el botón “Registrarse”.		
Datos introducidos	• Nombre completo: “Fran Fernández Soria” • Nombre de usuario: “fran96” • Correo electrónico: “fran96@gmail.com” • Contraseña: “Tes” • Confirmar contraseña: “Tes”		
Resultado esperado	El sistema no registra al usuario en la base de datos. Aparece un aviso en la aplicación indicando que la contraseña debe tener más de 3 caracteres.		
Resultado obtenido	El sistema no registra al usuario en la base de datos. Debajo del campo de contraseña aparece el mensaje “Introduzca al menos 4 caracteres.”.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 37: Prueba - Inicio de sesión (Correcto)

Nombre	Inicio de sesión (Correcto)	**Versión**	1.0
Dependencias	UC 002 – Iniciar sesión		
Descripción	El sistema debe permitir a un usuario registrado en la aplicación iniciar sesión con sus credenciales.		
Pasos	1. El usuario navega a la sección de inicio de sesión. 2. El usuario rellena el formulario de inicio de sesión. 3. El usuario pulsa el botón “Iniciar Sesión”.		
Datos introducidos	• Correo electrónico: “fran96@gmail.com” • Contraseña: “Test_96”		
Resultado esperado	El usuario inicia sesión con éxito en la aplicación y se muestra su perfil del usuario.		
Resultado obtenido	La aplicación redirige al usuario a la sección de su perfil, donde se muestran sus datos.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 38: Prueba - Inicio de sesión (Incorrecto – Credenciales no válidas)

Nombre	Inicio de sesión (Incorrecto)	**Versión**	1.0
Dependencias	UC 002 – Iniciar sesión		
Descripción	El sistema no debe permitir a un usuario iniciar sesión con credenciales incorrectas.		
Pasos	1. El usuario navega a la sección de inicio de sesión. 2. El usuario rellena el formulario de inicio de sesión. 3. El usuario pulsa el botón “Iniciar Sesión”.		
Datos introducidos	• Correo electrónico: “fran96@gmail.com” • Contraseña: “prueba_96”		
Resultado esperado	El usuario no inicia sesión y se muestra en la aplicación un aviso explicando que las credenciales no están registradas en el sistema.		
Resultado obtenido	El usuario no inicia sesión en la aplicación. Se muestra un mensaje con el texto “Contraseña incorrecta.”.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 39: Prueba - Consultar perfil

Nombre	Consultar perfil	**Versión**	1.0
Dependencias	UC 003 – Consultar perfil		
Descripción	Un usuario que ha iniciado sesión en el sistema debe poder consultar los datos de su perfil.		
Pasos	1. El usuario navega a la sección de perfil de usuario.		
Datos introducidos	Ninguno.		
Resultado esperado	El sistema muestra un formulario con los datos de perfil del usuario.		
Resultado obtenido	El sistema muestra un formulario con los datos de perfil del usuario.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 40: Prueba - Editar perfil (Correcto)

Nombre	Editar perfil (Correcto)	**Versión**	1.0
Dependencias	UC 004 – Editar perfil		
Descripción	Un usuario que ha iniciado sesión en el sistema debe poder editar los datos de su perfil de usuario.		
Pasos	1. El usuario navega a la sección de perfil de usuario. 2. El usuario rellena el formulario. 3. El usuario presiona el botón “Actualizar perfil”.		
Datos introducidos	• Nombre completo: “Antonio Soria” • Nombre de usuario: “asor96” • Correo electrónico: “asor@gmail.com” • Información pública: “Esta es mi bio”		
Resultado esperado	Los datos del usuario se actualizan en el sistema. Se muestra un aviso que informa al usuario que sus datos se han actualizado.		
Resultado obtenido	Los datos del usuario se actualizan y se muestra en pantalla el mensaje “Perfil actualizado correctamente”.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 41: Prueba - Editar perfil (Incorrecto – Correo electrónico existente)

Nombre	Editar perfil (Incorrecto – Correo electrónico existente)	**Versión**	1.0
Dependencias	UC 004 – Editar perfil		
Descripción	Los datos del perfil de un usuario no se deben de actualizar si el correo electrónico introducido ya existe en el sistema.		
Pasos	1. El usuario navega a la sección de perfil de usuario. 2. El usuario rellena el formulario. 3. El usuario presiona el botón “Actualizar perfil”.		
Datos introducidos	• Nombre completo: “Antonio Soria” • Nombre de usuario: “asor96” • Correo electrónico: “test@example.com” • Información pública: “Esta es mi bio”		
Resultado esperado	Los datos del usuario no se actualizan en el sistema. Se muestra un aviso que informa al usuario que el correo electrónico introducido ya existe en el sistema.		
Resultado obtenido	Los datos del usuario no se actualizan y se muestra en pantalla el mensaje “Dirección de correo electrónico ya existente.”.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 42: Prueba - Consultar lista de eventos

Nombre	Consultar lista de eventos	**Versión**	1.0
Dependencias	UC 005 – Consultar eventos		
Descripción	Los usuarios deben poder consultar la lista de todos los eventos registrados en el sistema.		
Pasos	1. El usuario navega a la sección de eventos.		
Datos introducidos	Ninguno.		
Resultado esperado	En la aplicación se muestra una lista de los eventos registrados en el sistema.		
Resultado obtenido	En la aplicación se muestra una lista de los eventos registrados en el sistema.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 433: Prueba - Crear evento

Nombre	Crear evento	**Versión**	1.0
Dependencias	UC 008 – Crear evento		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema crear eventos.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario presiona el botón “Crear evento” 3. El usuario rellena el formulario. 4. El usuario presiona el botón “Añadir evento”.		
Datos introducidos	• Nombre: “Concierto – Club 8” • Fecha de inicio: “12/01/2020 22:00” • Fecha de fin: “13/01/2020 01:00” • Tipo: “Otro” • URL de la imagen: “https://thumbs.dreamstime.com/z/logotipo-del-club-de-la-m%C3%BAsica-67814384.jpg” • Localización: “Calle cebada, nº 23” • Descripción: “Concierto de artista invitado sorpresa el 12 de enero a las 22. Entrada 5€. ¡Te esperamos!”		
Resultado esperado	Se registra en el sistema el evento creado y se muestran los detalles del evento.		
Resultado obtenido	Se registra el evento en el sistema, se muestra el mensaje “Registro realizado correctamente” y la aplicación muestra la ventana con los detalles del evento creado.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 444: Prueba - Consultar detalles de evento

Nombre	Consultar detalles de evento	**Versión**	1.0
Dependencias	UC 005 – Consultar eventos		
Descripción	Los usuarios deben poder consultar los detalles de un evento registrado en el sistema.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos.		
Datos introducidos	Ninguno.		
Resultado esperado	En la aplicación se muestran los detalles del evento seleccionado.		
Resultado obtenido	En la aplicación se muestran los detalles del evento seleccionado.		
Fecha	12/01/2020	**Estado**	Realizada con éxito

Tabla 45: Prueba - Editar evento

Nombre	Editar evento	**Versión**	1.0
Dependencias	UC 013 – Editar evento		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema editar la información de un evento.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario presiona el icono de edición. 4. El usuario rellena el formulario. 5. El usuario presiona el botón “Actualizar evento”.		
Datos introducidos	• Nombre: “Partido de baloncesto” • Fecha de inicio: “12/01/2020 17:00” • Fecha de fin: “12/01/2020 20:00” • Tipo: “Deporte” • URL de la imagen: “https://yt3.ggpht.com/a/AGF-l7-lOEF-W1hj338rG3v7O0lRToLQXgYLc71SQQ=s900-c-k-c0xffffffff-no-rj-mo” • Localización: “Avenida del mediterráneo, Rafael Florido” • Descripción: “Partido de baloncesto. Nos faltan 5 personas. 2€ por persona la reserva de la cancha.”		
Resultado esperado	Se actualizan en el sistema los datos del evento y se muestran los detalles del evento.		
Resultado obtenido	Se actualizan los datos el evento en el sistema, se muestra el mensaje “Registro realizado correctamente” y la aplicación muestra la ventana con los detalles del evento actualizado.		
Fecha	11/01/2020	**Estado**	Realizada con éxito

4.2.5. REVISIÓN DE LA VERSIÓN

En esta versión se han implementado todas las características que se habían propuesto en su planificación en un plazo de tiempo aceptable. El inicio de la implementación se produjo el 24 de Octubre y finalizó el 24 de Noviembre, como se muestra en el diagrama de Gantt de la siguiente imagen.

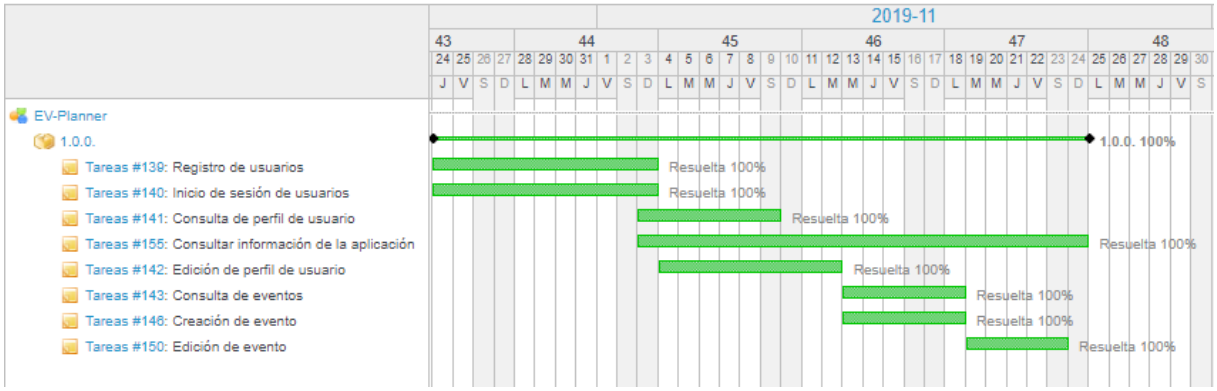


Ilustración 79: Diagrama de Gantt de la versión 1.0.

A continuación, se muestran los componentes y ventanas finales de esta versión.

Ilustración 80: Header - Cibernauta (Versión 1)

Ilustración 81: Header - Usuario (Versión 1)

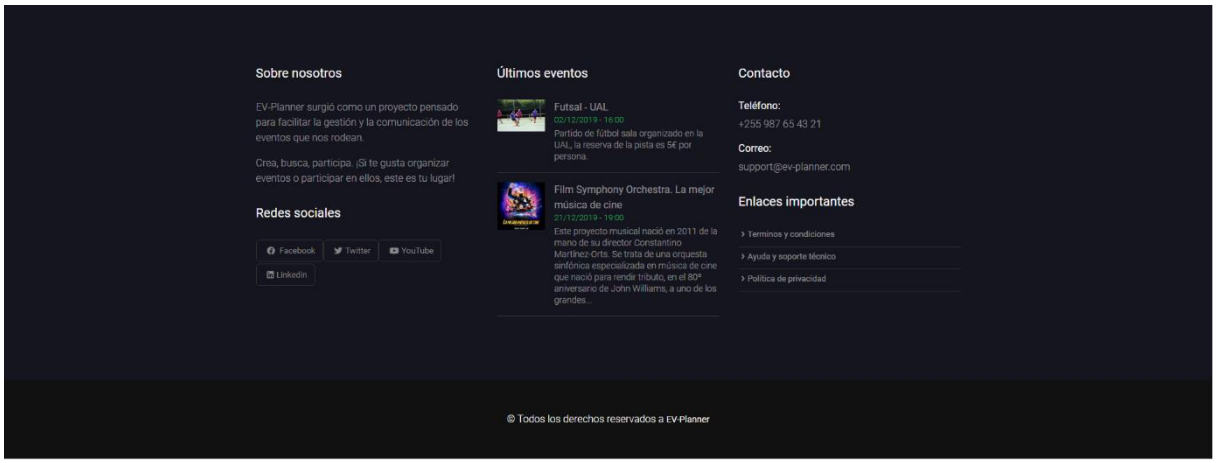


Ilustración 82: Footer (Versión 1)

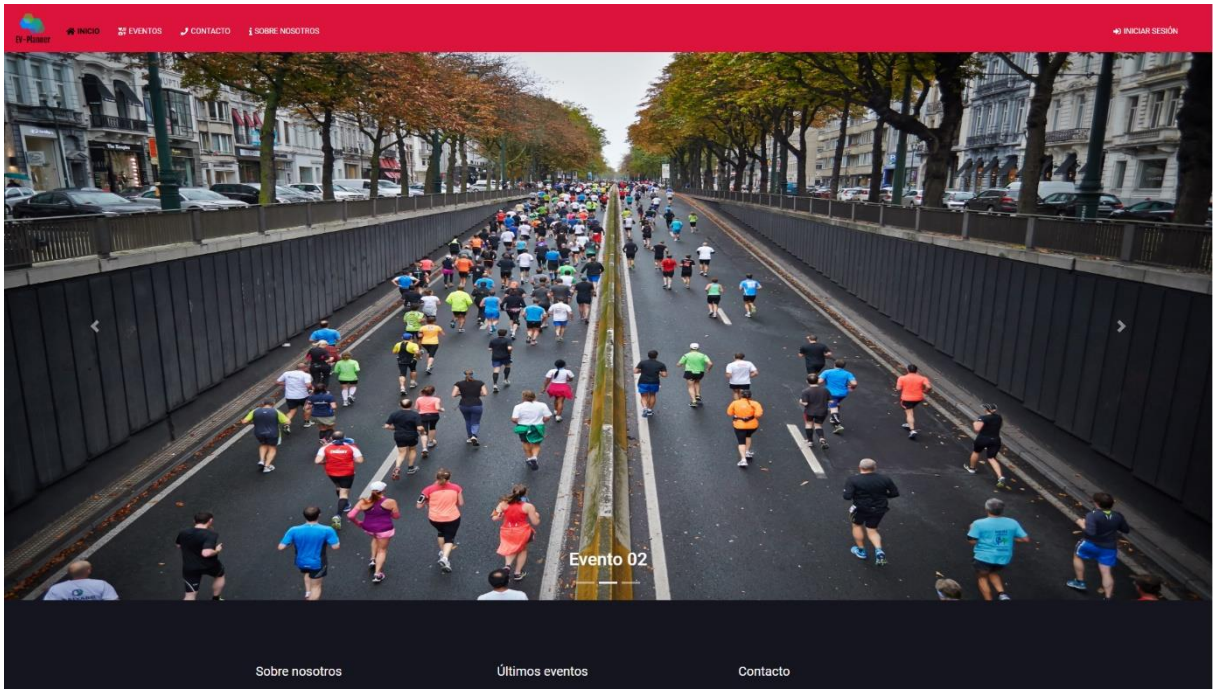


Ilustración 83: Ventana - Inicio (Versión 1)

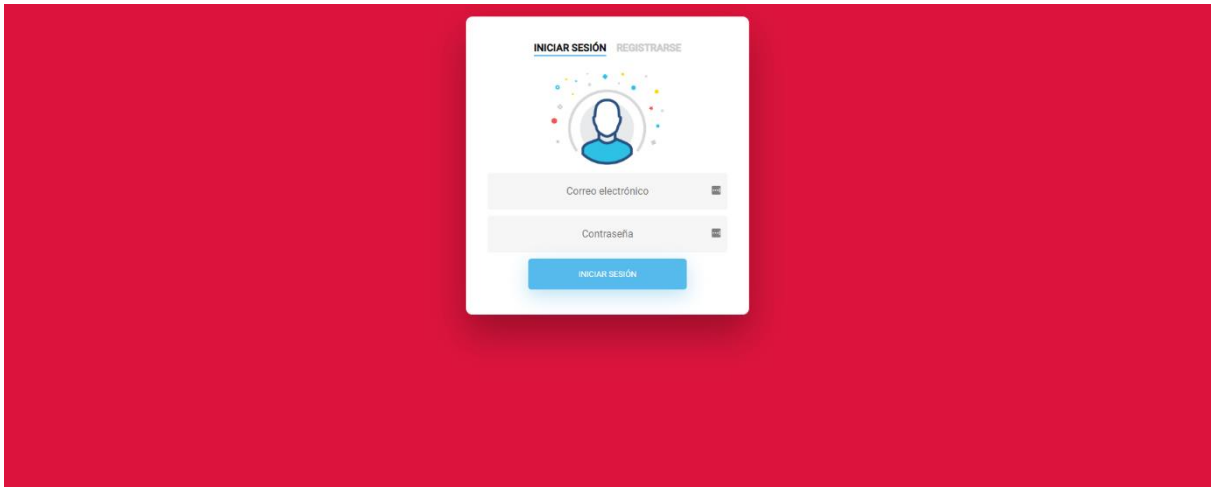


Ilustración 84: Ventana - Inicio de sesión (Versión 1)

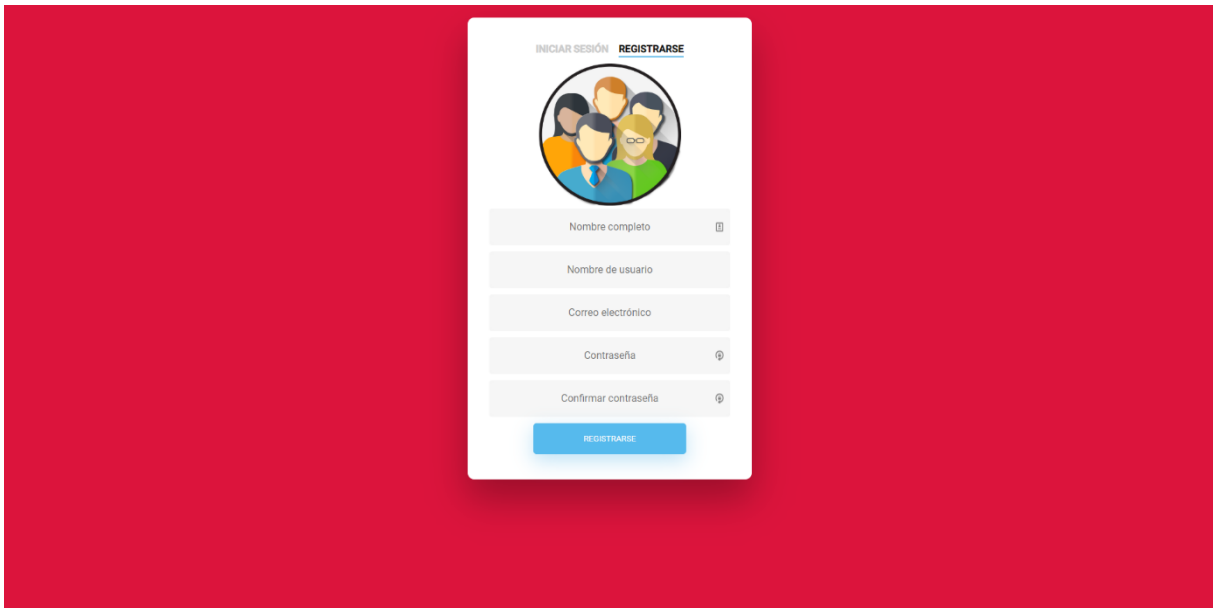


Ilustración 85: Ventana - Registro (Versión 1)

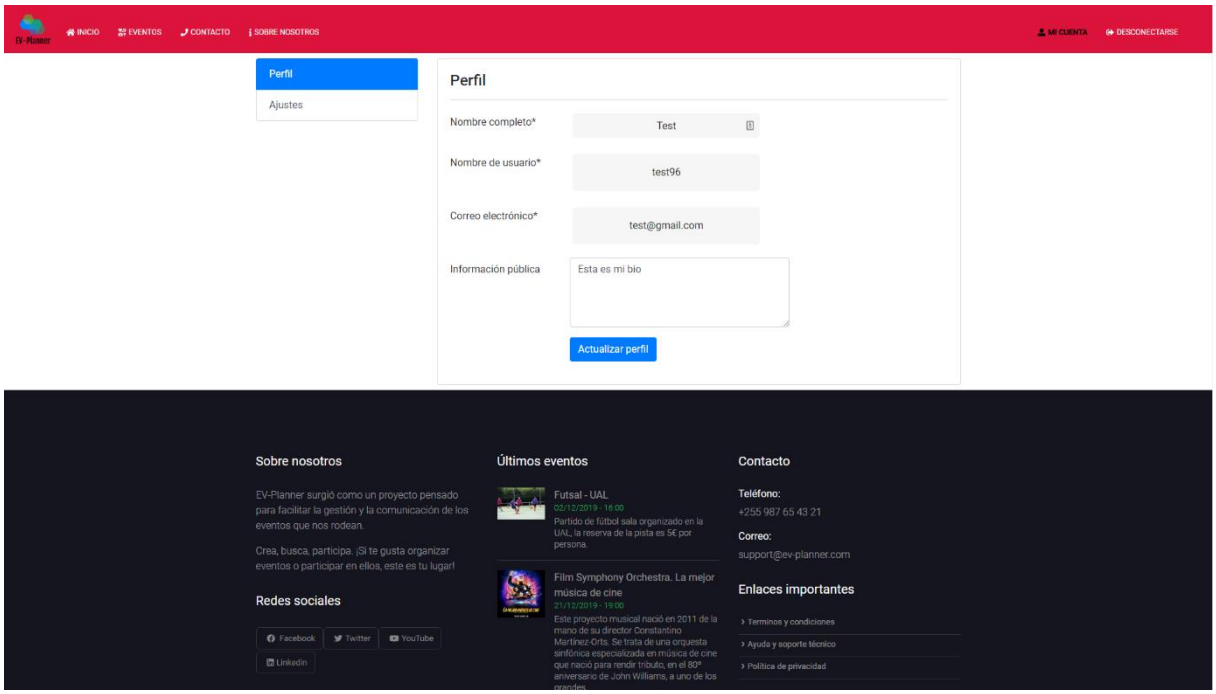


Ilustración 86: Ventana - Perfil (Versión 1)

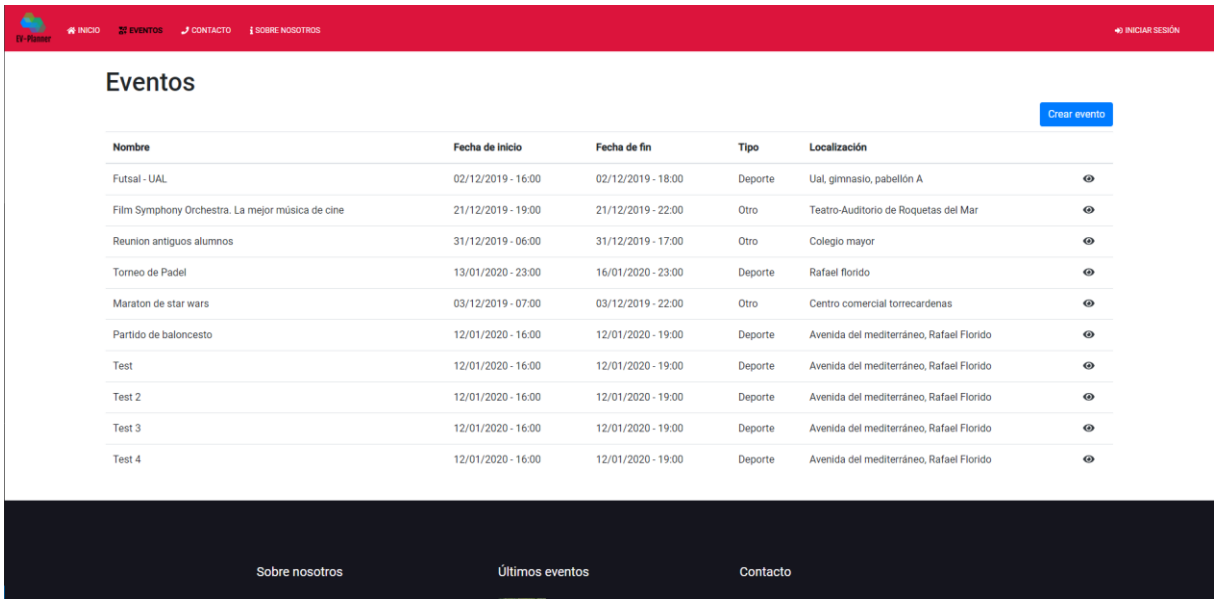


Ilustración 87: Ventana - Lista de eventos (Versión 1)

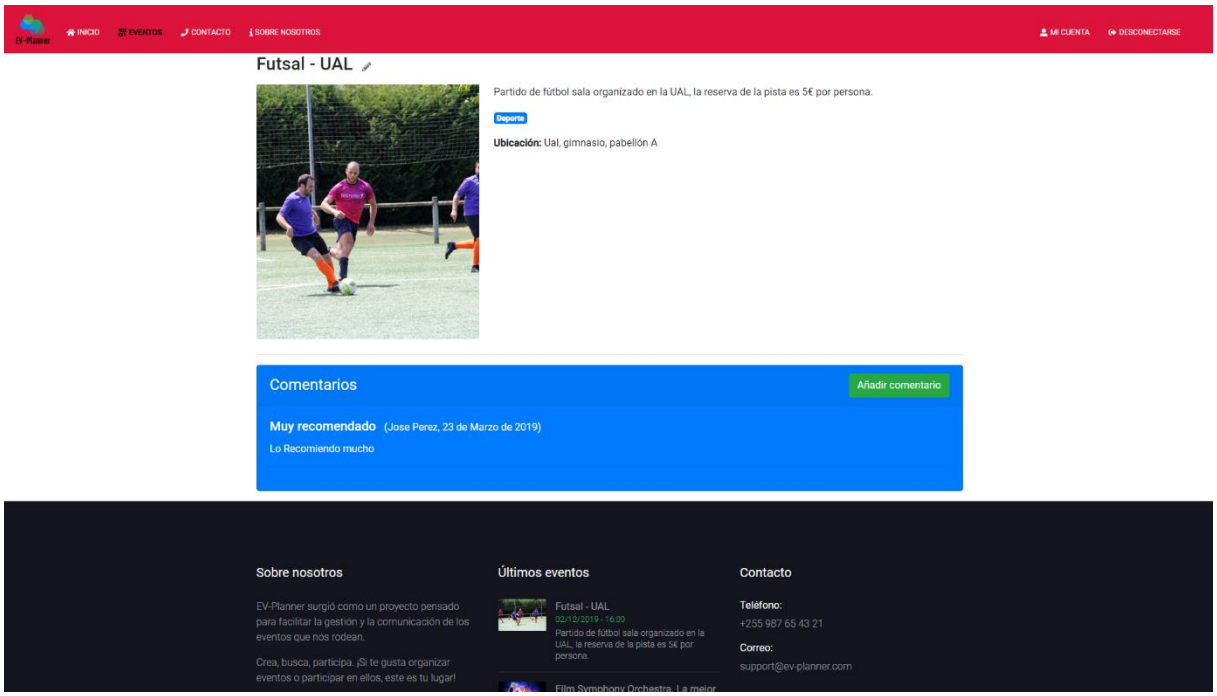


Ilustración 88: Ventana - Detalles de evento (Versión 1)

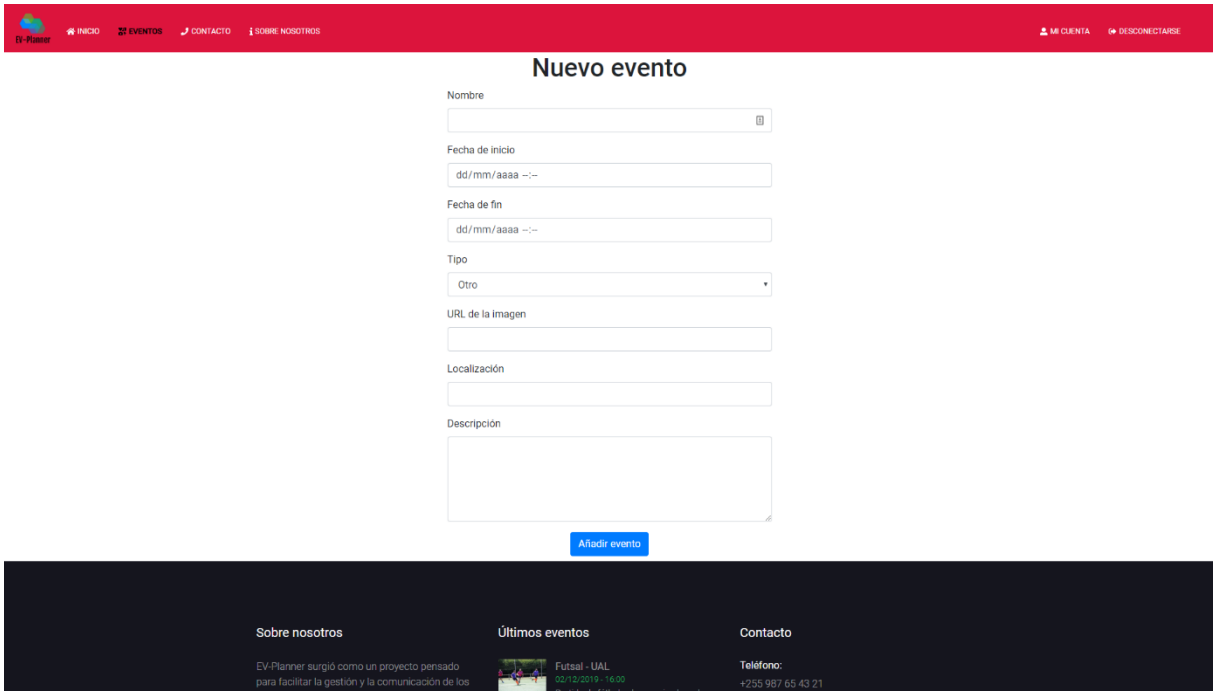


Ilustración 89: Ventana - Creación de evento (Versión 1)

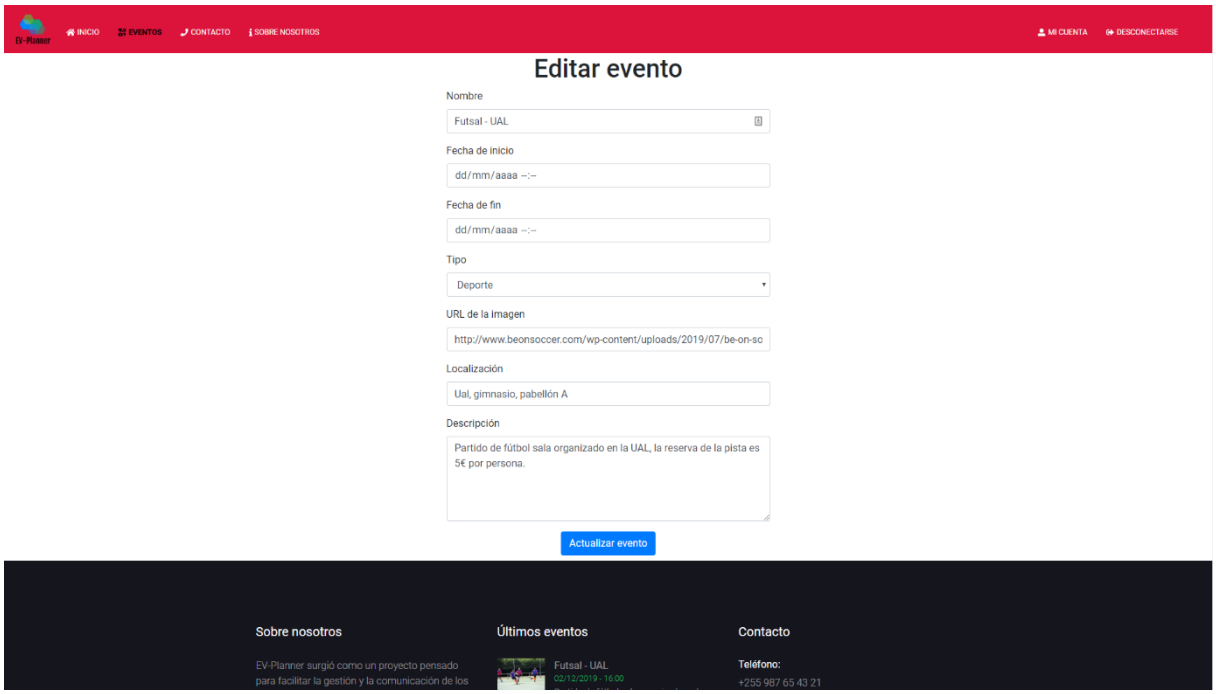


Ilustración 90: Ventana - Edición de evento (Versión 1)

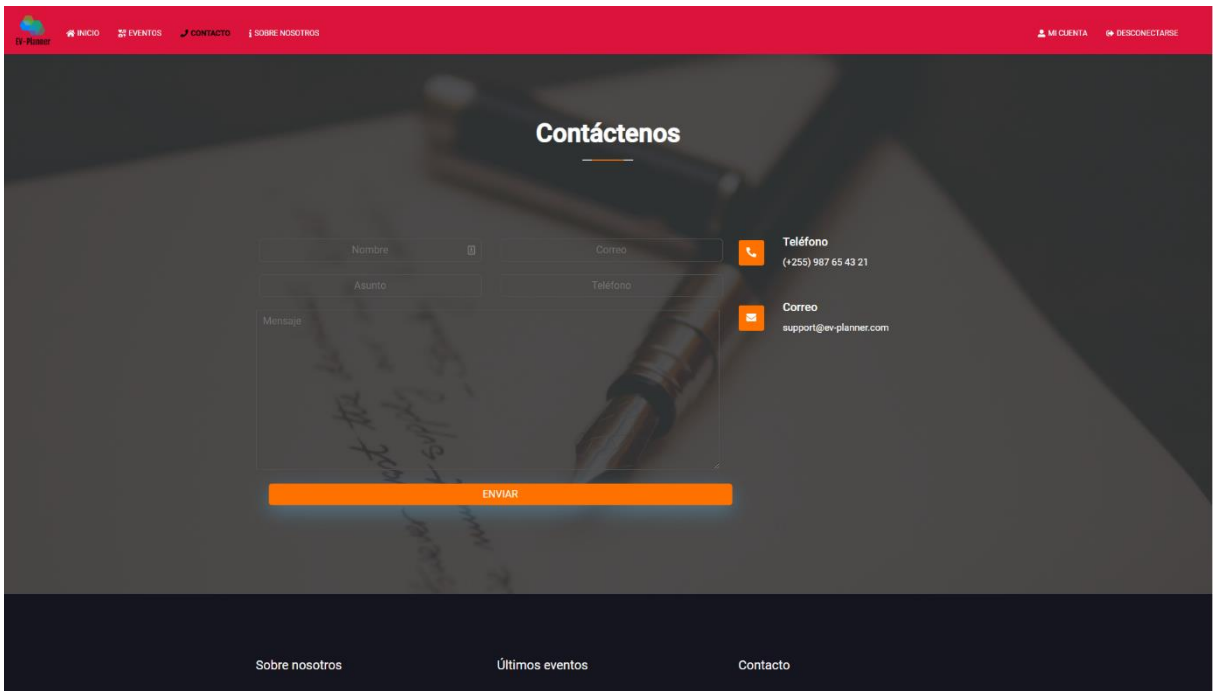


Ilustración 91: Ventana - Contacto (Versión 1)

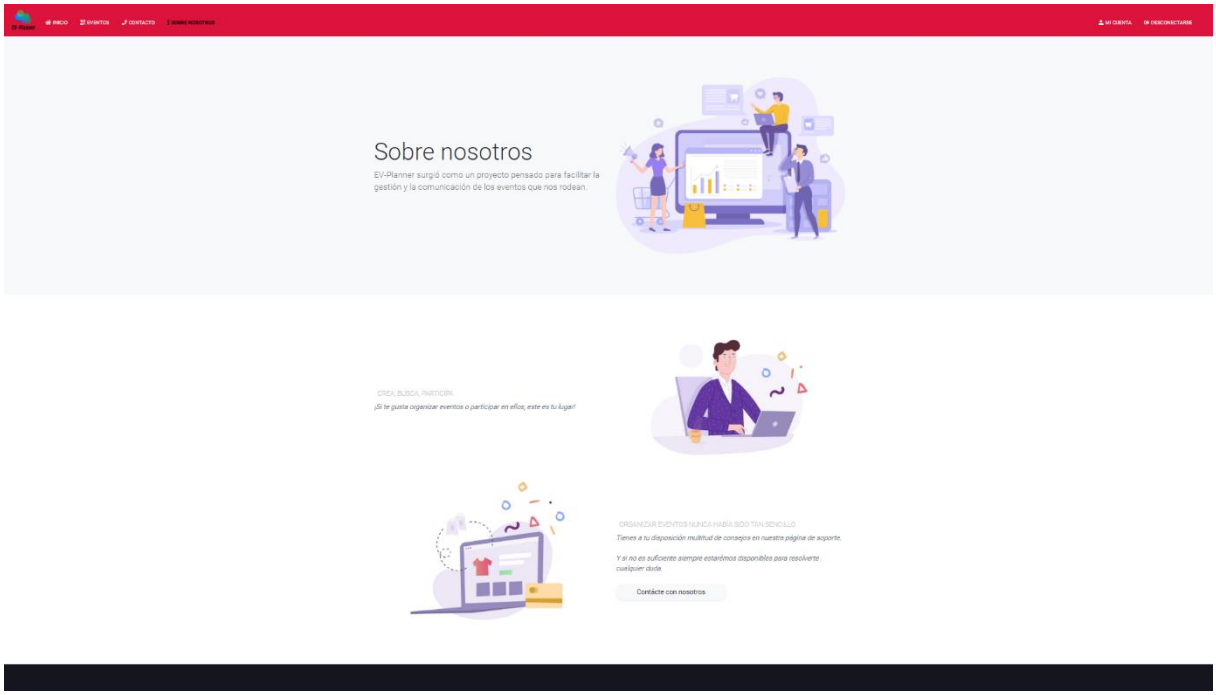


Ilustración 92: Ventana - Sobre nosotros (Versión 1)

4.3. VERSIÓN 2.0.

4.3.1. PLANIFICACIÓN

Para esta versión se han escogido todos los requisitos relacionados con la gestión de eventos que faltaban por implementar, particularmente los vinculados a la gestión de participantes y comentarios. También se ha añadido uno nuevo relacionado con el cambio de la propiedad de ubicación del evento para que permita ser manipulado en un mapa. En la siguiente imagen se muestran los requisitos escogidos.

#	Tipo	Estado	Prioridad	Asunto	Categoría	Fecha de inicio	Fecha fin
154	Tareas	Resuelta	Normal	Eliminación de evento	Gestión de eventos	2019-12-12	2019-12-14
151	Tareas	Resuelta	Normal	Nombrar administrador	Gestión de eventos	2019-12-14	2019-12-15
152	Tareas	Resuelta	Normal	Invitar a participante	Gestión de eventos	2019-12-15	2019-12-22
153	Tareas	Resuelta	Normal	Eliminación de participante	Gestión de eventos	2019-12-17	2019-12-22
144	Tareas	Resuelta	Normal	Registrar participante en evento	Gestión de eventos	2019-12-21	2019-12-26
148	Tareas	Resuelta	Normal	Abandono de evento	Gestión de eventos	2019-12-26	2019-12-28
147	Tareas	Resuelta	Normal	Consulta de eventos de los participantes	Gestión de eventos	2020-01-02	2020-01-04
149	Tareas	Resuelta	Normal	Consulta de eventos de administradores	Gestión de eventos	2020-01-02	2020-01-04
145	Tareas	Resuelta	Normal	Añadir comentarios a eventos	Gestión de eventos	2020-01-04	2020-01-07
156	Tareas	Resuelta	Normal	Integración con google maps	Gestión de eventos	2020-01-08	2020-01-13

Ilustración 93: Tareas de la versión 2.0.

4.3.2. DISEÑO

Para agilizar la implementación de esta versión, se han realizado varias anotaciones de diseño en las ventanas de las versiones anteriores, proponiendo dónde situar los elementos relacionados con las nuevas características a implementar.

Para gestionar los eventos que puede ver cada usuario, se ha pensado en añadir a la sección que muestra la lista de eventos unas pestañas que permitan mostrar los eventos en los que participa el usuario, los que administra o todos los públicos del sistema.

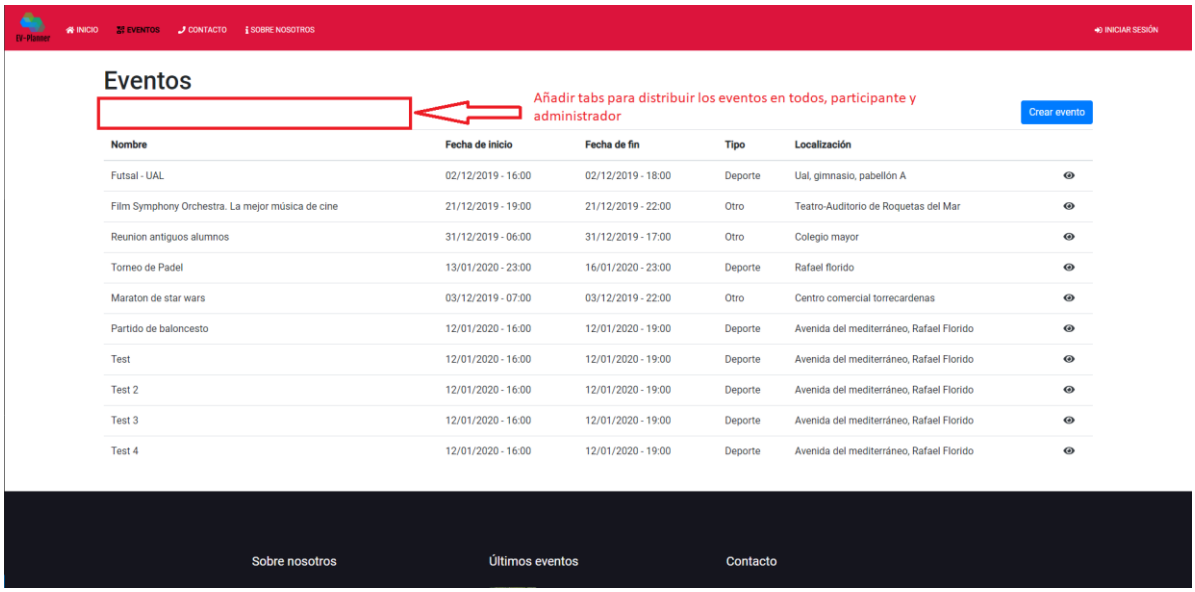


Ilustración 94: Diseño - Pestañas de eventos (Versión 2)

Para la función de eliminación de eventos, se podría añadir un botón en el formulario de edición de eventos que permita eliminarlo tras la correspondiente confirmación del usuario.

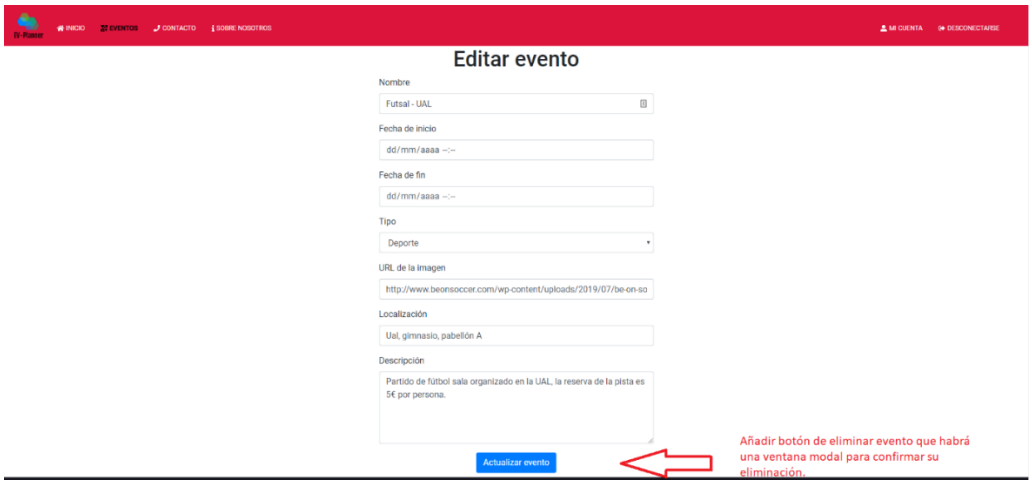


Ilustración 95: Diseño - Eliminación de evento (Versión 2)

La parte de más peso de esta versión es la relacionada con la gestión de participantes de eventos. Se ha pensado en añadir a la ventana de detalles del evento, una sección para gestionar los participantes de tal modo que los participantes podrán abandonar o inscribirse en un evento y los administradores podrán echar o añadir participantes a un evento que administren. También se ha decidido distribuir los datos que se muestran en esta vista de otra manera, incluir nuevos campos y cambiar el campo de localización de formato texto a coordenadas para mapearlo con un mapa de *Google maps*.

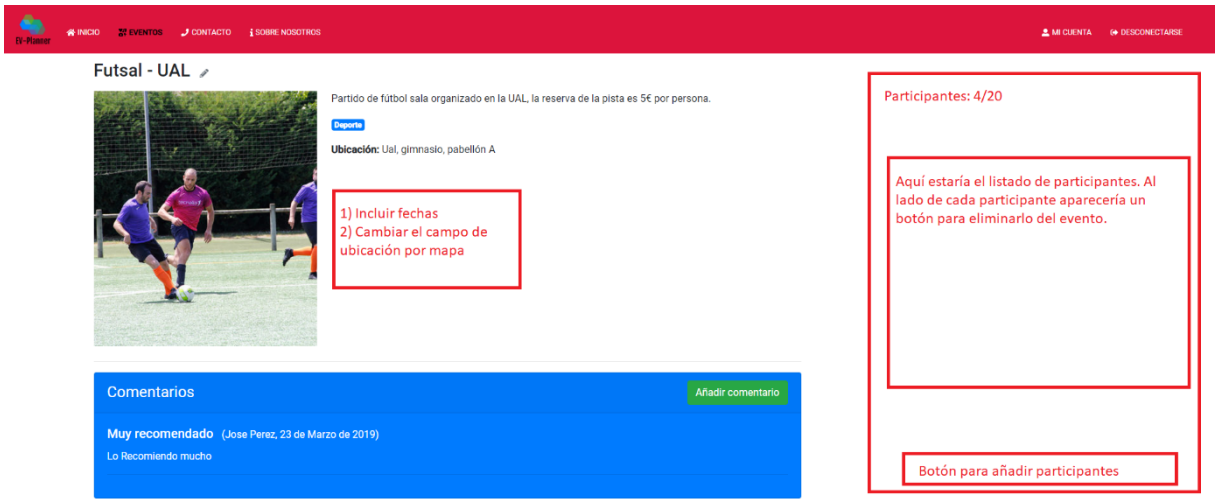


Ilustración 96: Diseño - Detalles de evento y gestión de participantes (Versión 2)

4.3.3. IMPLEMENTACIÓN

4.3.3.1. GESTIÓN DE COMENTARIOS

Se ha implementado en la vista de información de un evento un apartado para añadir comentarios entre usuarios registrados en la aplicación.

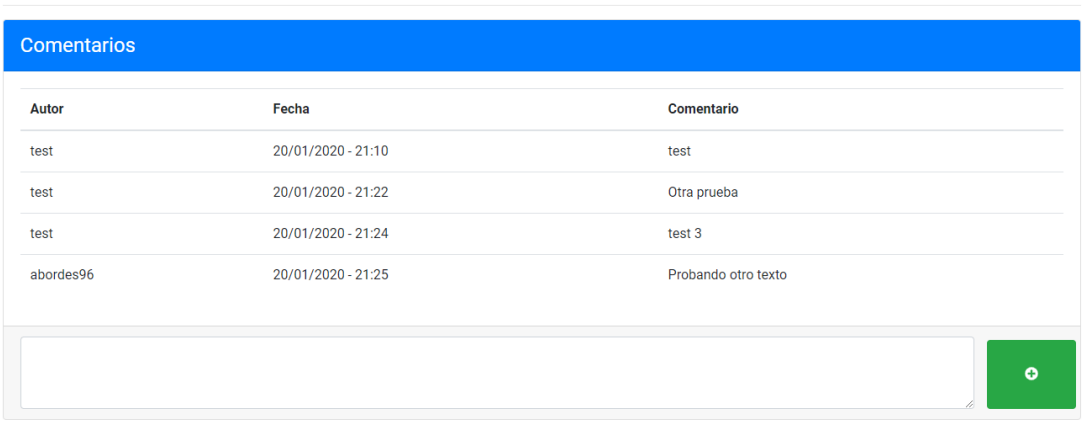


Ilustración 97: Tabla de comentarios

Al modelo de eventos se ha añadido como propiedad un array de comentarios. La estructura de datos de los comentarios es definida en su propio esquema.

```
var commentSchema = new mongoose.Schema({
    author: String,
    comment: String,
    created_date: {
        type: Date,
        "default": Date.now
    }
});
```

Ilustración 98: Esquema de datos de comentario

Cuando se manda la petición a la API de creación de un comentario, el controlador se encarga de buscar el evento correspondiente y agregar a la propiedad de comentarios el nuevo registro.

```
module.exports.addComment = (req, res, next) => {
    let event = Event.findOne({ _id: req.params.id },
        (err, event) => {
            if (!event) {
                return res.status(404).json({ status: false, message: 'Evento no encontrado.' });
            } else {
                event.comments.push(req.body);
                event.save((err, doc) => {
                    if (!err)
                        res.send(doc);
                    else {
                        return next(err);
                    }

                });
            }
        }
    );
}
```

Ilustración 99: Creación de comentario en la API

En la APP, en la vista que muestra los detalles de un evento se ha agregado el componente de comentarios, pasando como parámetros la información del evento y la del usuario actual.

```
<!-- --------------------Comments------------------------------ -->
<app-comments [userLogin]="userLogin" [event]="event" [user]="user"></app-comments>
```

Ilustración 100: Inclusión del componente de gestión de comentarios

El componente de comentarios mapea las variables pasadas de la vista padre y las utiliza para cargar los comentarios en la vista con el correspondiente formato.

```
@Input() event: Event;
@Input() user: User;
@Input() userLogin: boolean;
```

Ilustración 101: Variables mapeadas en el componente de comentarios

```
loadComments() {
  this.event.comments.forEach(comment => {
    this.userService.getUserInfo(comment.author).subscribe(
      res => {
        this.comments.push({
          'author' : res['user'].userName,
          'comment': comment.comment,
          'date': comment.created_date
        });
      },
      err => {
        console.log(err);
      }
    );
  });
}
```

Ilustración 102: Carga de los comentarios en su vista

El apartado de comentarios tiene un cuadro de texto vinculado con una variable de su controlador, cuando se pulsa el botón de añadir comentario el contenido del cuadro de texto se envía a la API para que se añada a la base de datos.

```
<div class="card-footer">
    <div class="row">
        <div class="col-sm-12 col-md-11">
            <textarea name="comment" [(ngModel)]='comment' rows="3" class="form-control"></textarea>
        </div>
        <button (click)="addComment()" class="col-md-1 col-sm-12 btn btn-success mt-1">
            <fa-icon class="mx-2 text-white" [icon]="faPlusCircle"></fa-icon>
        </button>
    </div>
```

Ilustración 103: Sincronización de la variable comentarios con su controlador

4.3.3.2. GESTIÓN DE PARTICIPANTES

Al igual que en la gestión de comentarios, la gestión de participantes se ha integrado en la vista que muestra los detalles de un evento.

```
<div id="right-side" class="col-lg-3 col-m-12 col-sm-12 mb-2">
    <!-- ------------------------Participants------------------------- -->
    <app-participants  [userLogin]="userLogin" [event]="event" [user]="user"></app-participants>
</div>
```

Ilustración 104: Inclusión del componente de gestión de participantes

Dependiendo de si el usuario es administrador del evento, un participante o un cibernauta las operaciones que pueden realizar son diferentes.

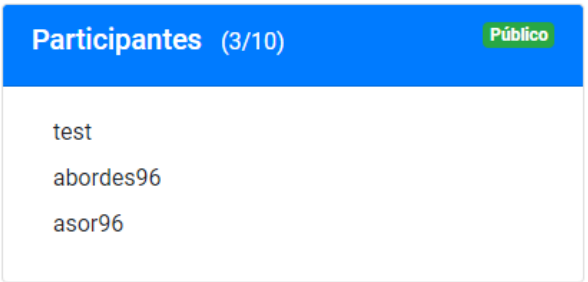


Ilustración 105: Gestión de participantes (Cibernauta)

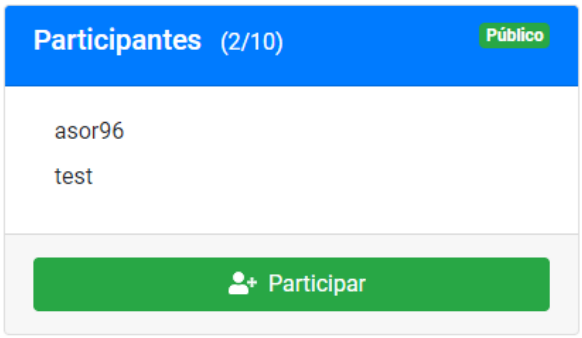


Ilustración 106: Gestión de participantes (Usuario)

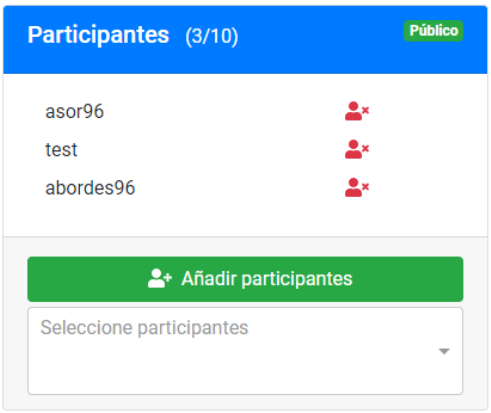


Ilustración 107: Gestión de participantes (Administrador)

Los participantes se han añadido como una propiedad del modelo de datos de los eventos y consiste en un array que contiene los id de los usuarios que participan en el evento.

Cuando se añade un participante, el controlador de eventos de la API busca el evento correspondiente y añade a su array de participantes el id del nuevo usuario. Si al añadir un nuevo participante se supera el aforo máximo del evento, en su lugar la API manda un código de error a la aplicación con el mensaje correspondiente.

```
event.participants.push(req.body._id);
if (event.participants.length > event.capacity) {
    res.status(422).send(['No se puede superar el aforo máximo del evento.']);
} else {
    event.save((err, doc) => {
```

Ilustración 108: Inserción de participante

Para eliminar un participante se recorre la lista de participantes del evento y cuando se localiza el id del usuario se elimina del array.

```
for (var i = 0; i < event.participants.length; i++) {
    if (event.participants[i] === req.body._id) {
        event.participants.splice(i, 1);
    }
}
event.save((err, doc) => {
```

Ilustración 109: Eliminación de participante

En el caso de los usuarios que no administran el evento, al pulsar el botón “Participar” o “Abandonar” se ejecuta su función asociada que se encarga de comunicarse con la API y llevar a cabo la función correspondiente contra la base de datos, tal y como se explicó al comienzo de esta sección.

En el caso de los administradores, cuando presionan el icono de eliminar participante, al lado del nombre de usuario de cada participante, se ejecuta la misma función que cuando un usuario abandona un evento, solo que en este caso el id del usuario pasado como parámetro es el del usuario seleccionado y no el del usuario que ha iniciado sesión en el sistema.

La función de añadir participantes, en cambio, está vinculado con un desplegable de selección múltiple. Tras pulsar el botón de añadir participantes para cada uno de los usuarios seleccionados se llama a la función encargada de comunicarse con la API y añadir a los participantes.

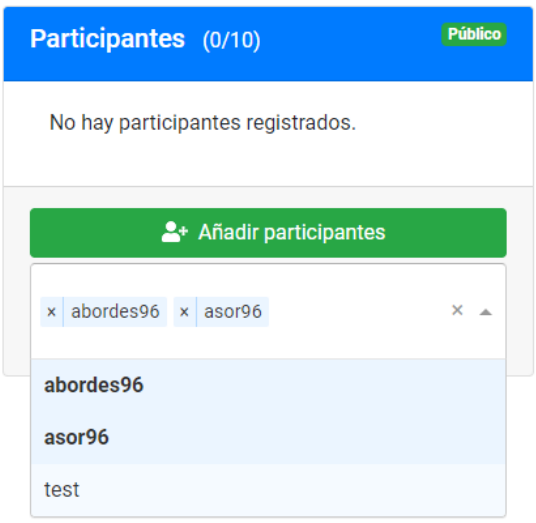


Ilustración 110: Selección múltiple de participantes

4.3.3.3. INTEGRACIÓN CON GOOGLE MAPS

La propiedad de ubicación de evento del modelo de datos ha cambiado de ser en formato texto a un objeto que contiene el nombre de la ubicación y sus coordenadas.

```
location:{
    address: String,
    longitude: Number,
    latitude: Number
},
```

Ilustración 111: Propiedad localización (Versión 2)

En la aplicación se ha definido un componente encargado de gestionar el mapa desde el que se seleccionan las coordenadas de la ubicación.

En el formulario utilizado para la creación y edición de eventos se ha sustituido el campo de texto de la ubicación por un mapa con un cuadro de búsqueda. El usuario puede buscar la ubicación por el cuadro de texto o directamente con el marcador del mapa, cuando uno de esos elementos cambia, el otro elemento se actualiza con su valor.

Ilustración 112: Mapa del formulario de creación / edición de evento

El mapa y el marcador tienen asociada su posición a los valores de latitud y longitud, cuando se cambia de posición el marcador, esos valores se actualizan y con ello la posición del mapa.

```
<agm-map [latitude]="latitude" [longitude]="longitude" [zoom]="zoom">

  <agm-marker *ngIf="section == 'creation' || section == 'edition'"
   [latitude]="latitude" [longitude]="longitude"
   [markerDraggable]="true" (dragEnd)="markerDragEnd($event)"></agm-marker>
```

Ilustración 113: Mapa y marcador (HTML)

```
markerDragEnd($event: MouseEvent) {
  //console.log($event);
  this.latitude = $event.coords.lat;
  this.longitude = $event.coords.lng;
  this.updateCoordinates();

  this.getAddress(this.latitude, this.longitude);
}
```

Ilustración 114: Evento de cambio de posición del marcador

En el evento que se activa al cambiar la posición del marcador, se llama a la función encargada de obtener la dirección vinculada a esas coordenadas y actualizarla, como el campo de dirección está vinculado al cuadro de texto de búsqueda este también se actualiza.

```
getAddress(latitude, longitude) {
  this.geoCoder.geocode({ 'location': { lat: latitude, lng: longitude } }, (results, status) => {
    if (status === 'OK') {
      if (results[0]) {
        this.zoom = 13;
        this.address = results[0].formatted_address;
        this.updateCoordinates();
      } else {
        window.alert('No results found');
      }
    } else {
      window.alert('Geocoder failed due to: ' + status);
    }
  });
}
```

Ilustración 115: Función de obtención de dirección a partir de coordenadas

```
<input (change)="updateCoordinates()" type="text" class="form-control"
(keydown.enter)="$event.preventDefault()" placeholder="Introduzca una ubicación" autocorrect="off"
 autocapitalize="off" spellcheck="off" type="text" [(ngModel)]="address" #search>
```

Ilustración 116: Cuadro de búsqueda de ubicación (HTML)

Este componente también se ha reutilizado en la vista que muestra los detalles de un evento, pero limitando sus funcionalidades para que no se pueda cambiar la ubicación seleccionada por el administrador.

Ilustración 117: Mapa de la vista de detalles de un evento

4.3.3.4. ADMINISTRACIÓN Y CLASIFICACIÓN DE EVENTOS

En esta versión además de implementar la gestión de participantes se ha añadido a los eventos un campo que controla si un evento es público o privado, de tal manera que los eventos privados solo se muestren a los administradores y a sus participantes.

La sección de eventos ha sido dividida por pestañas para permitir clasificar los distintos eventos de la aplicación.

Nombre	Fecha de inicio	Fecha de fin	Tipo	Ubicación	Aforo
Futsal - UAL	02/12/2019 - 09:00	03/12/2019 - 15:00	Deporte	Calle Maestro Lecuona, 8, 29006 Málaga, España	3/10
Film Symphony Orchestra. La mejor música de cine	21/12/2019 - 19:00	21/12/2019 - 22:00	Otro	Calle Jesús de Perceval, 75, 04003 Almería, España	0/15
Torneo de Padel	13/01/2020 - 23:00	16/01/2020 - 23:00	Deporte	Ctra. de Ronda, 74, 04006 Almería, España	0/5
Maraton de star wars	03/12/2019 - 07:00	03/12/2019 - 22:00	Otro	Calle Ulloa, 4, 04002 Almería, España	0/5
Partido de baloncesto	12/01/2020 - 16:00	12/01/2020 - 19:00	Deporte	AL-3202, 04120 Almería, España	1/10
tttt	18/01/2020 - 00:00	01/01/2020 - 00:01	Otro	Calle Circun Pz Toros, 5, 04008 Almería, España	1/1
TTT	24/01/2020 - 00:00	23/01/2020 - 00:00	Otro	Calle Santiago, 44, 04006 Almería, España	0/10

Ilustración 118: Lista de eventos dividida por pestañas

Al iniciar la vista dependiendo del usuario que interactúa con la aplicación se cargan los datos correspondientes las pestañas y eventos que puede visualizar.

```
loadAllEventsPublic() {
  this.eventService.getEventsPublic().subscribe(
    res => {
      this.tabs.push({
        name: 'Públicos',
        events: res['events']
      });
    },
    err => {
      console.log(err);
    }
  );
}
```

Ilustración 119: Carga de datos de la pestaña de eventos públicos

4.3.4. PRUEBAS

En esta sección se muestran las plantillas de pruebas de aceptación utilizadas en esta versión.

Tabla 46: Prueba - Consultar lista de eventos (Versión 2)

Nombre	Consultar lista de eventos	**Versión**	2.0
Dependencias	UC 005 – Consultar eventos UC 009 – Consultar eventos (Participante) UC 011 – Consultar eventos (Administrador)		
Descripción	Los usuarios deben poder consultar la lista de todos los eventos registrados en el sistema. Estos eventos deberán estar agrupados para mostrar los públicos, los que el usuario administra y aquellos en los que participa.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario navega por las pestañas “Públicos”, “Administro” y “Participo”.		
Datos introducidos	Ninguno.		
Resultado esperado	En la aplicación se muestra una lista de los eventos registrados en el sistema, el usuario al navegar por las distintas pestañas puede observar los eventos correspondientes al grupo consultado.		
Resultado obtenido	Se muestran los eventos correspondientes a cada pestaña consultada.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 47: Prueba - Crear evento (Versión 2)

Nombre	Crear evento	**Versión**	2.0
Dependencias	UC 008 – Crear evento UC 012 – Nombrar administrador		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema crear eventos.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario presiona el botón “Crear evento” 3. El usuario rellena el formulario. 4. El usuario presiona el botón “Añadir evento”.		
Datos introducidos	• Nombre: “Concierto – Club 8” • Privacidad: “Público” • Fecha de inicio: “22/01/2020 22:00” • Fecha de fin: “23/01/2020 01:00” • Tipo: “Otro” • Aforo: “50” • URL de la imagen: “https://thumbs.dreamstime.com/z/logotipo-del-club-de-la-m%C3%BAsica-67814384.jpg” • Localización: “Calle Calz. de Castro, 83, 04006 Almería, España” • Descripción: “Concierto de artista invitado sorpresa el 22 de enero a las 22. Entrada 5€. ¡Te esperamos!”		
Resultado esperado	Se registra en el sistema el evento creado y se muestran los detalles del evento.		
Resultado obtenido	Se registra el evento en el sistema, se muestra el mensaje “Registro realizado correctamente” y la aplicación muestra la ventana con los detalles del evento creado.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 48: Prueba - Editar evento (Versión 2)

Nombre	Editar evento	**Versión**	2.0
Dependencias	UC 013 – Editar evento		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema editar la información de un evento.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario presiona el icono de edición. 4. El usuario rellena el formulario. 5. El usuario presiona el botón “Actualizar evento”.		
Datos introducidos	• Nombre: “Concierto – Paseo marítimo” • Privacidad: “Privado” • Fecha de inicio: “30/01/2020 20:00” • Fecha de fin: “30/01/2020 22:00” • Tipo: “Otro” • Aforo: “15” • URL de la imagen: “https://yt3.ggpht.com/a/AGF-l7-lOEF-W1hj338rG3v7O0lRToLQXgYLc71SQQ=s900-c-k-c0xffffffff-no-rj-mo” • Localización: “Paseo Marítimo Carmen de Burgos, Almería, España” • Descripción: “Concierto privado en la playa del paseo marítimo, enfrente del café París. Si no podéis asistir añadir un comentario.””		
Resultado esperado	Se actualizan en el sistema los datos del evento y se muestran los detalles del evento.		
Resultado obtenido	Se actualizan los datos el evento en el sistema, se muestra el mensaje “Registro realizado correctamente” y la aplicación muestra la ventana con los detalles del evento actualizado.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 49: Prueba - Comentar evento

Nombre	Comentar evento	**Versión**	2.0
Dependencias	UC 007 – Comentar evento		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema añadir comentarios a los eventos.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario rellena el área de texto del apartado de comentarios. 4. El usuario presiona el botón de añadir comentario.		
Datos introducidos	• Comentario: “Prueba de comentario”		
Resultado esperado	Se agrega el comentario al evento y se muestra en la tabla de comentarios.		
Resultado obtenido	Se actualizan los datos del evento en el sistema con el nuevo comentario añadido. En la tabla de comentarios también se muestra el nuevo comentario.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 50: Prueba - Registrarse en evento como participante

Nombre	Registrarse en evento como participante	**Versión**	2.0
Dependencias	UC 006 – Registrarse en evento como participante		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema inscribirse como participante en un evento.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario presiona el botón “Participar”.		
Datos introducidos	Ninguno.		
Resultado esperado	Se agrega el participante al evento y se muestra en la tabla de participantes.		
Resultado obtenido	Se actualizan los datos el evento en el sistema con el nuevo participante añadido. En la tabla de participantes también se muestra el nuevo participante.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 51: Prueba - Abandonar evento

Nombre	Abandonar evento	**Versión**	2.0
Dependencias	UC 010 – Abandonar evento		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema y participan en un evento, abandonar ese evento.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario presiona el botón “Abandonar”.		
Datos introducidos	Ninguno.		
Resultado esperado	Se elimina el participante del evento y de la tabla de participantes.		
Resultado obtenido	Se actualizan los datos el evento en el sistema, eliminando al participante. En la tabla de participantes también se ha eliminado al participante.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 52: Prueba - Registrar a participante en evento

Nombre	Registrar a participante en evento	**Versión**	2.0
Dependencias	UC 014 – Invitar participante		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema y administran un evento, añadir participantes a ese evento.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario selecciona varios usuarios al desplegable del apartado de participantes. 4. El usuario presiona el botón “Añadir participantes”.		
Datos introducidos	Ninguno.		
Resultado esperado	Se actualizan los datos el evento en el sistema con los nuevos participantes añadidos. En la tabla de participantes también se muestran los nuevos participantes. En el desplegable de usuarios no aparece ninguno de los participantes.		
Resultado obtenido	Se actualizan los datos el evento en el sistema con los nuevos participantes añadidos. En la tabla de participantes también se muestran los nuevos participantes. En el desplegable de usuarios no aparece ninguno de los participantes.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 53: Prueba - Registrar a participante en evento (Aforo máximo)

Nombre	Registrar a participante en evento (Aforo máximo)	**Versión**	2.0
Dependencias	UC 014 – Invitar participante		
Descripción	La aplicación no debe permitir a los usuarios que han iniciado sesión en el sistema y administran un evento, añadir participantes a ese evento si se supera el aforo máximo del evento.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario selecciona varios usuarios al desplegable del apartado de participantes. 4. El usuario presiona el botón “Añadir participantes”.		
Datos introducidos	Ninguno.		
Resultado esperado	Se actualizan los datos el evento en el sistema con los nuevos participantes añadidos hasta que se alcance el aforo máximo. En la tabla de participantes también se muestran los nuevos participantes. En el desplegable de usuarios no aparece ninguno de los participantes que se han añadido. Se muestra en la aplicación un mensaje indicando que el aforo máximo se ha alcanzado.		
Resultado obtenido	Se actualizan los datos el evento en el sistema con los nuevos participantes añadidos hasta que se alcance el aforo máximo. En la tabla de participantes también se muestran los nuevos participantes. En el desplegable de usuarios no aparece ninguno de los participantes que se han añadido. Se muestra en la aplicación el mensaje “Aforo máximo alcanzado”.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 54: Prueba - Eliminar participante de evento

Nombre	Eliminar participante de evento	**Versión**	2.0
Dependencias	UC 015 – Eliminar participante		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema y administran un evento, eliminar participantes del evento.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario presiona el icono de eliminar junto al nombre de uno de los participantes del evento.		
Datos introducidos	Ninguno.		
Resultado esperado	Se actualizan los datos el evento en el sistema eliminando el participante seleccionado. En la tabla de participantes no se debe mostrar el participante eliminado. En el desplegable de usuarios debe aparecer el participante eliminado.		
Resultado obtenido	Se actualizan los datos el evento en el sistema eliminando el participante seleccionado. En la tabla de participantes no se muestra al participante eliminado. En el desplegable de usuarios aparece el participante eliminado.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

Tabla 55: Prueba - Eliminar evento

Nombre	Eliminar evento	**Versión**	2.0
Dependencias	UC 016 – Eliminar evento		
Descripción	La aplicación debe permitir a los usuarios que han iniciado sesión en el sistema y administran un evento eliminarlo.		
Pasos	1. El usuario navega a la sección de eventos. 2. El usuario selecciona el icono de visualizar de uno de los eventos. 3. El usuario presiona el icono de edición. 4. El usuario presiona el botón “Eliminar”.		
Datos introducidos	Ninguno.		
Resultado esperado	Se elimina el evento del sistema.		
Resultado obtenido	Se elimina el evento del sistema.		
Fecha	22/01/2020	**Estado**	Realizada con éxito

4.3.5. REVISIÓN DE LA VERSIÓN

Las características planificadas para esta versión se han implementado en un marco de tiempo razonable, comenzando la implementación el 12 de Diciembre, acabo el 13 de Enero. En la siguiente imagen se muestra el diagrama de Gantt de la versión.

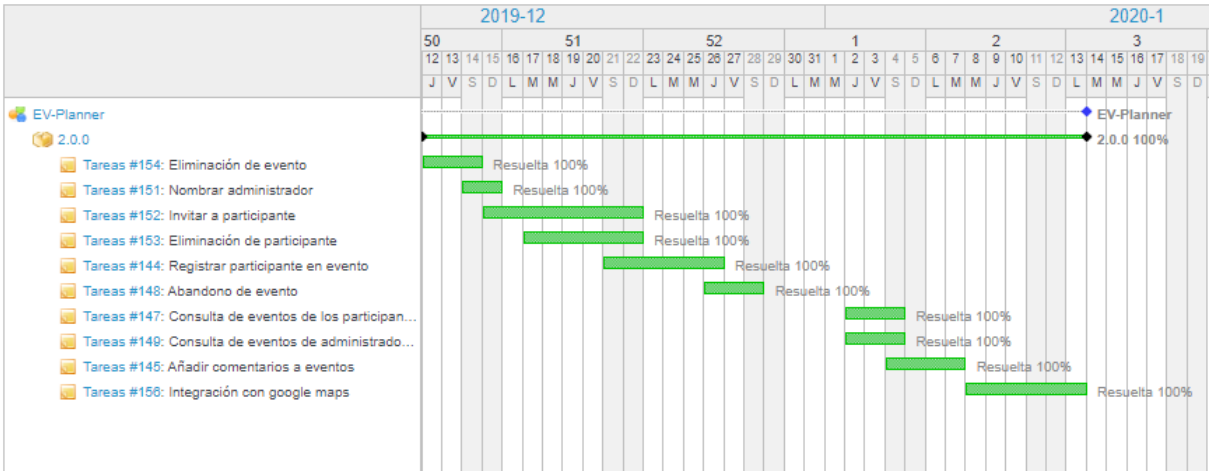


Ilustración 120: Diagrama de Gantt de la versión 2.0.

Las ventanas que se han modificado en esta versión se muestran a continuación.

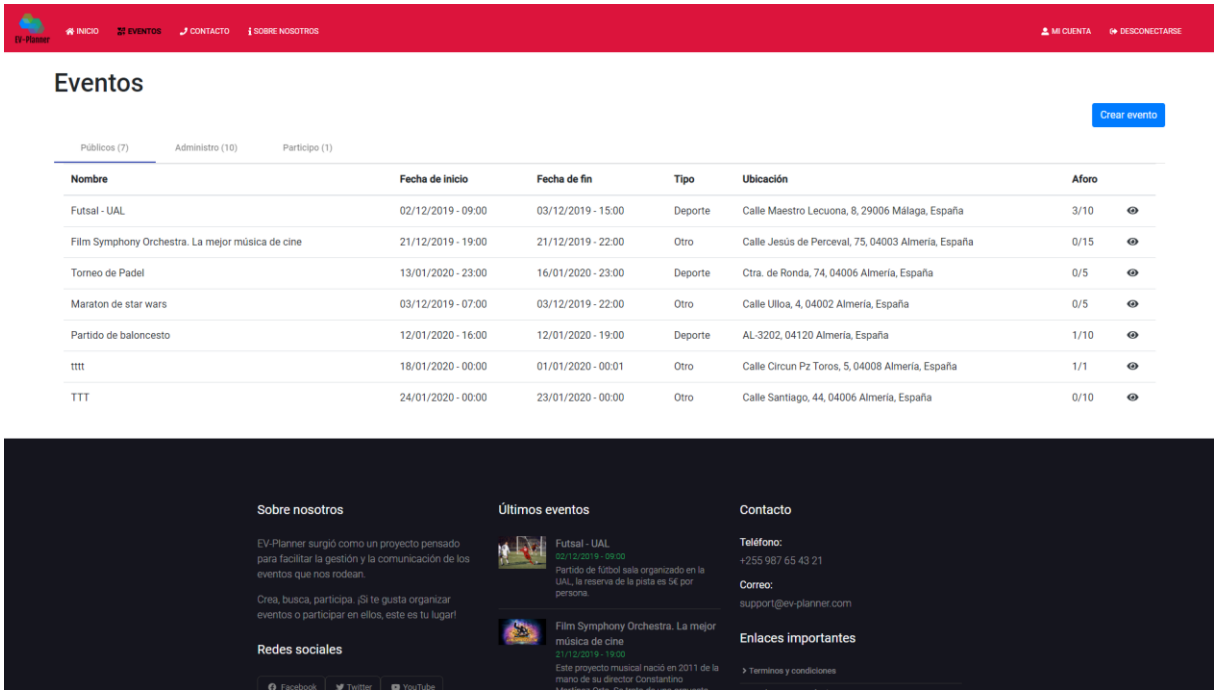


Ilustración 121: Ventana - Lista de eventos (Versión 2)

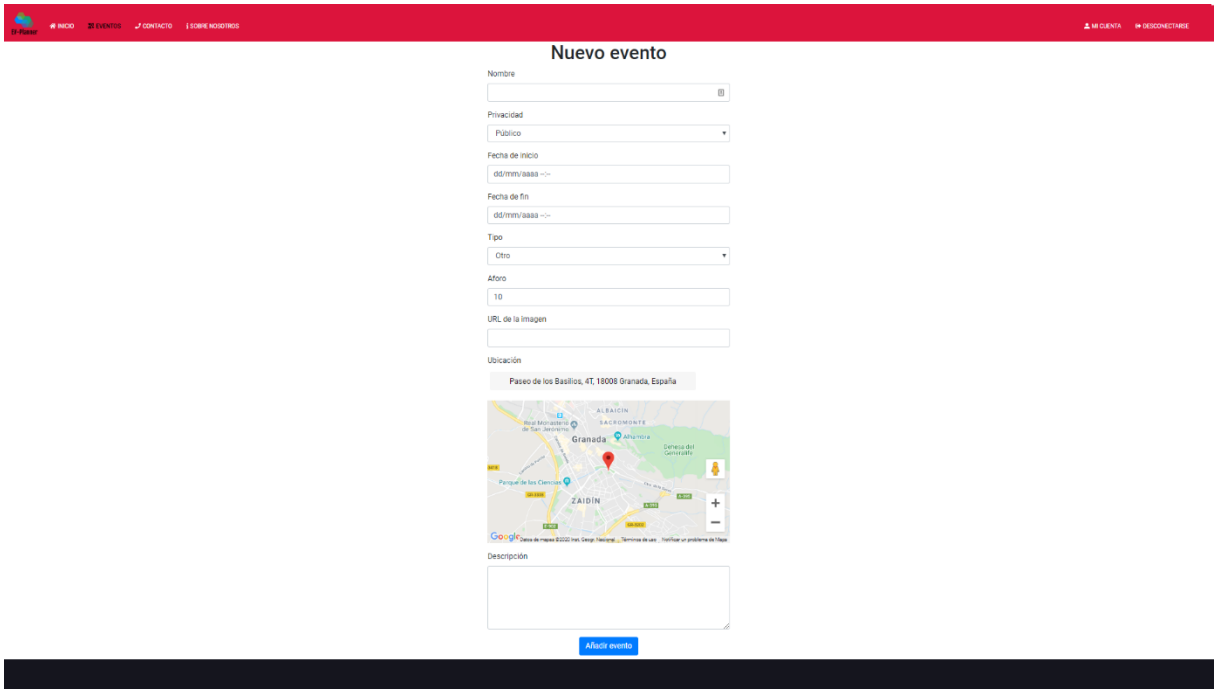


Ilustración 122: Ventana - Formulario de eventos (Versión 2)

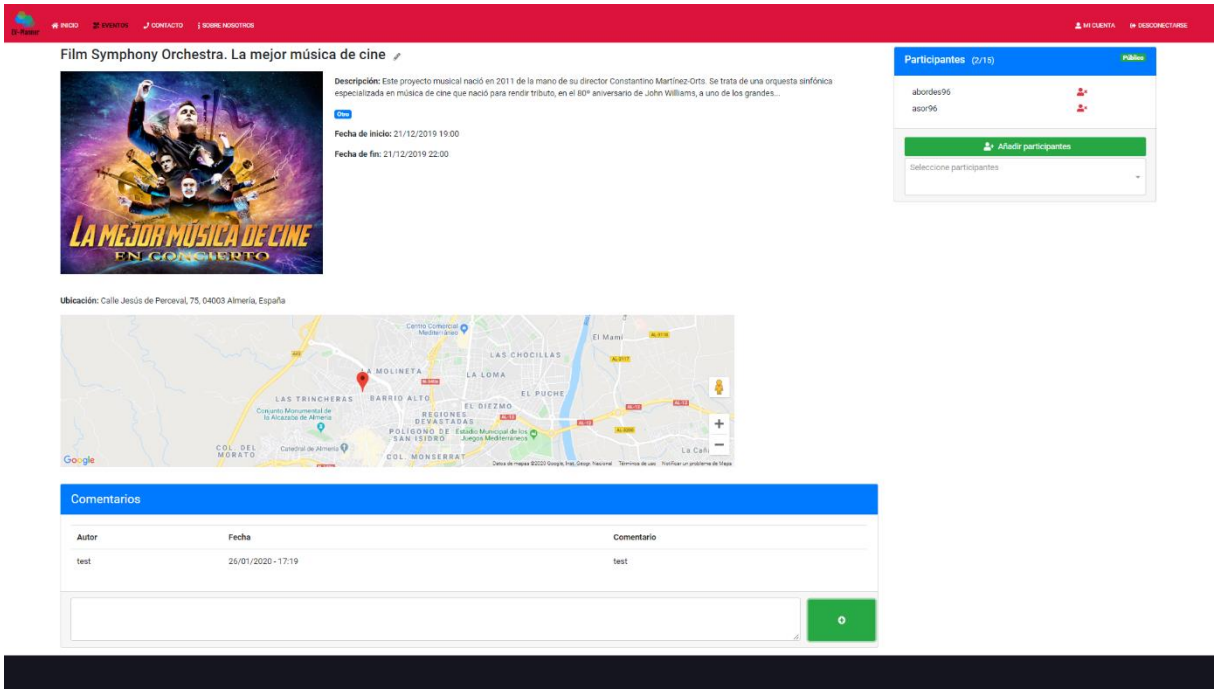


Ilustración 123: Ventana - Detalles de evento de administrador (Versión 2)

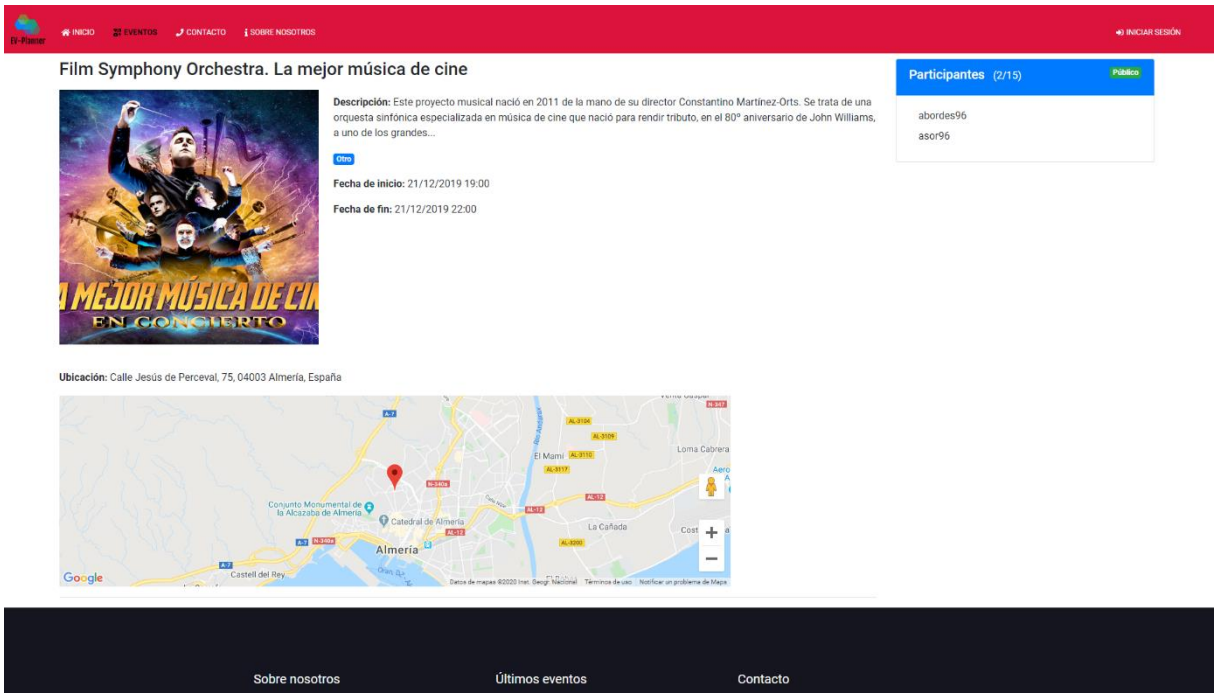


Ilustración 124: Ventana - Detalles de evento de participante (Versión 2)

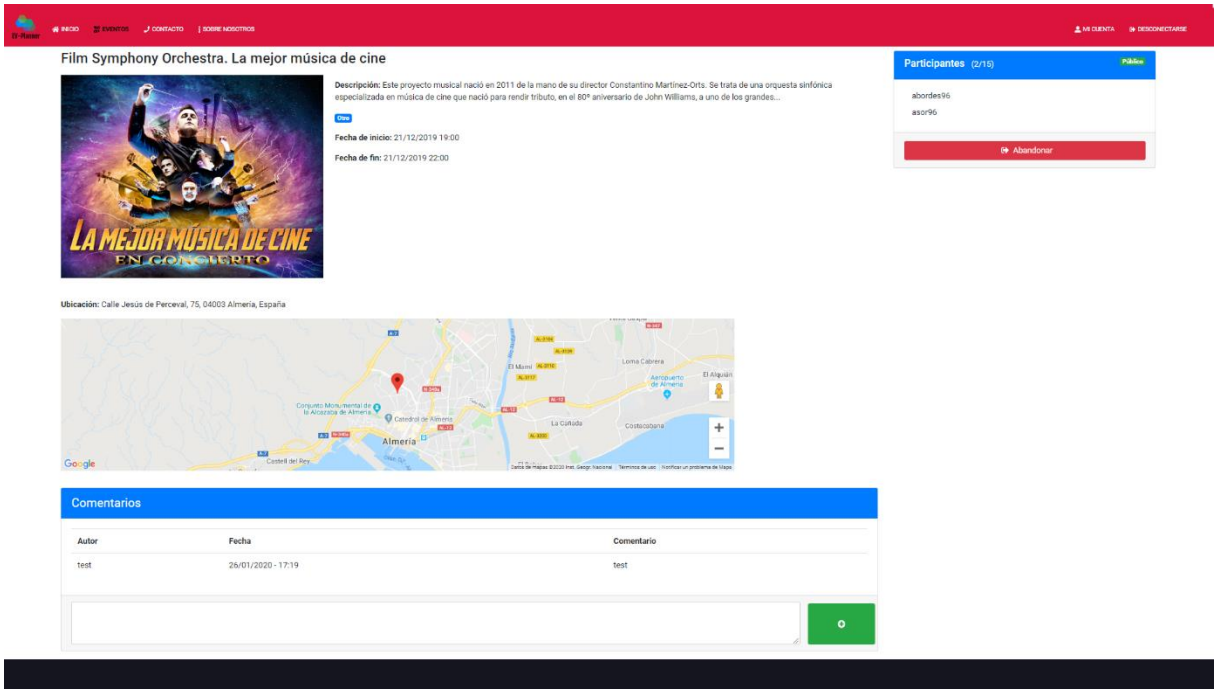


Ilustración 125: Detalles de evento de cibernauta (Versión 2)

5. RESULTADOS

5.1. FUNCIONAMIENTO DE LA APLICACIÓN

Después de desarrollar la versión 2.0, se han realizado algunos ajustes en el diseño de las ventanas. En esta sección se mostrarán todas las ventanas de la aplicación divididas por la funcionalidad que puede realizar cada tipo de usuario y que han sido definidas en la especificación de requisitos (3.3. Casos de uso del sistema).

5.1.1. CIBERNAUTA

- Registrarse

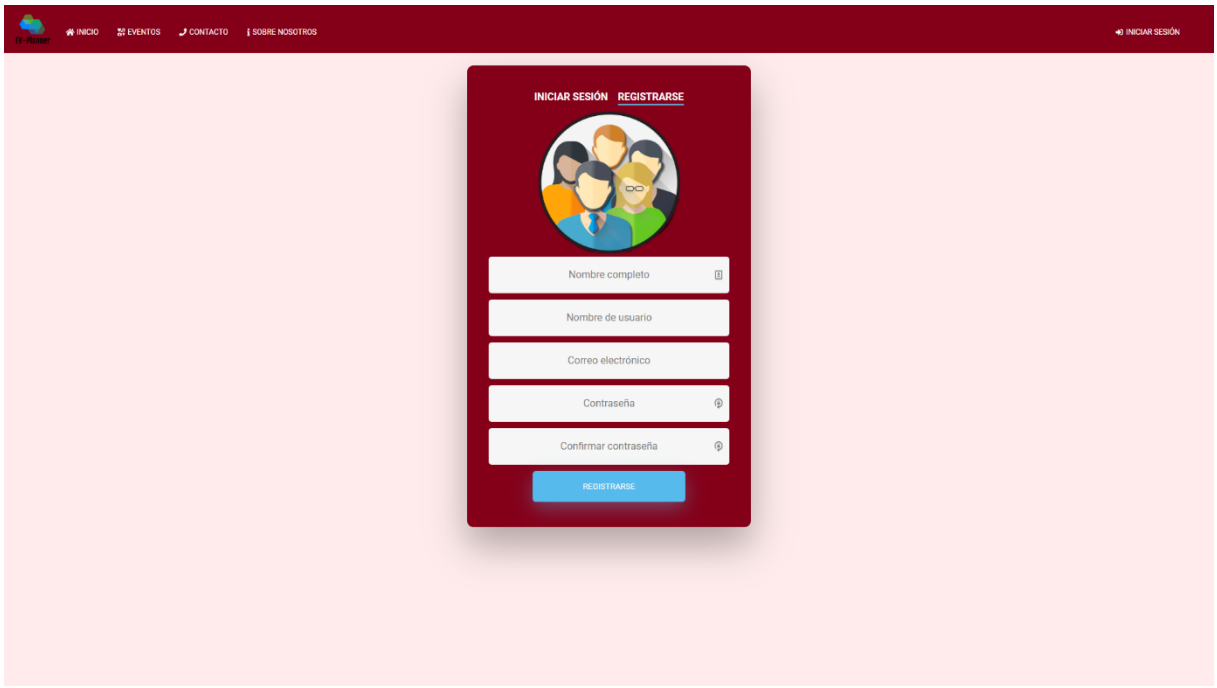


Ilustración 126: Ventana - Registro

- Iniciar sesión

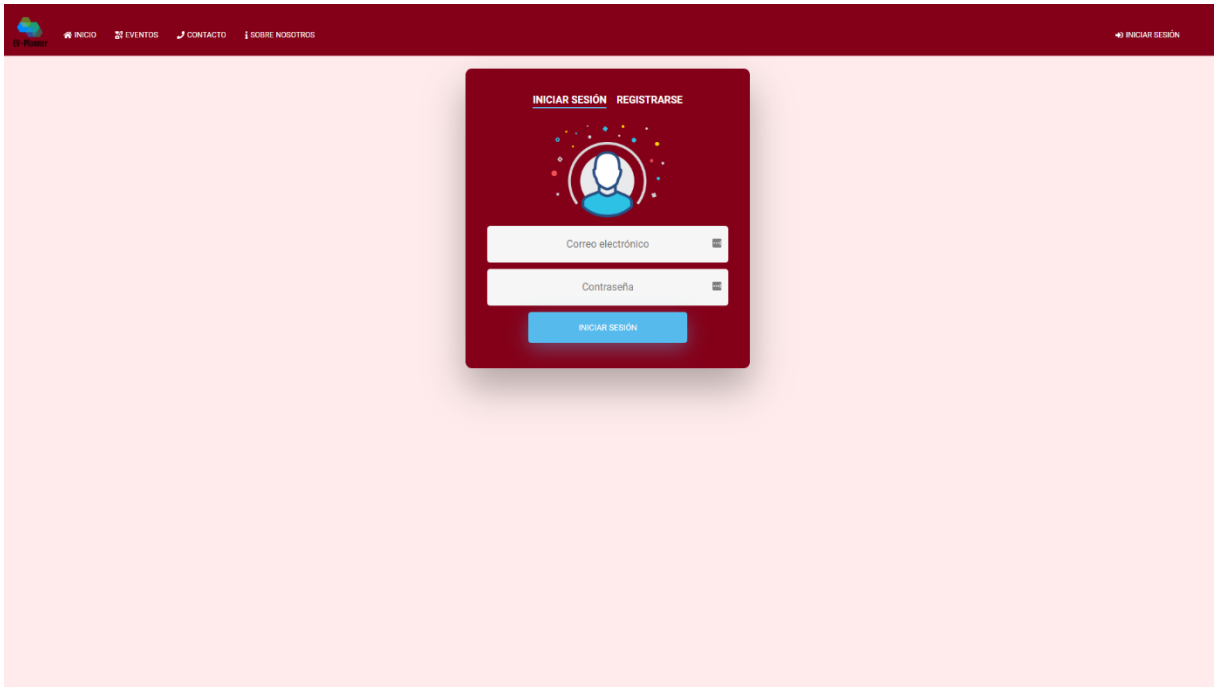


Ilustración 127: Ventana - Inicio de sesión

- Consultar información de la aplicación (Ilustraciones 128 – 131)

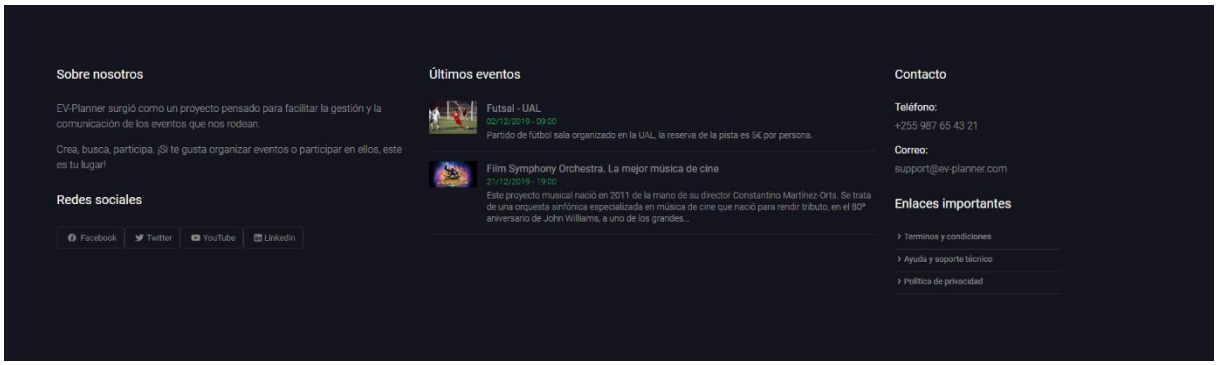


Ilustración 128: Ventana - Footer

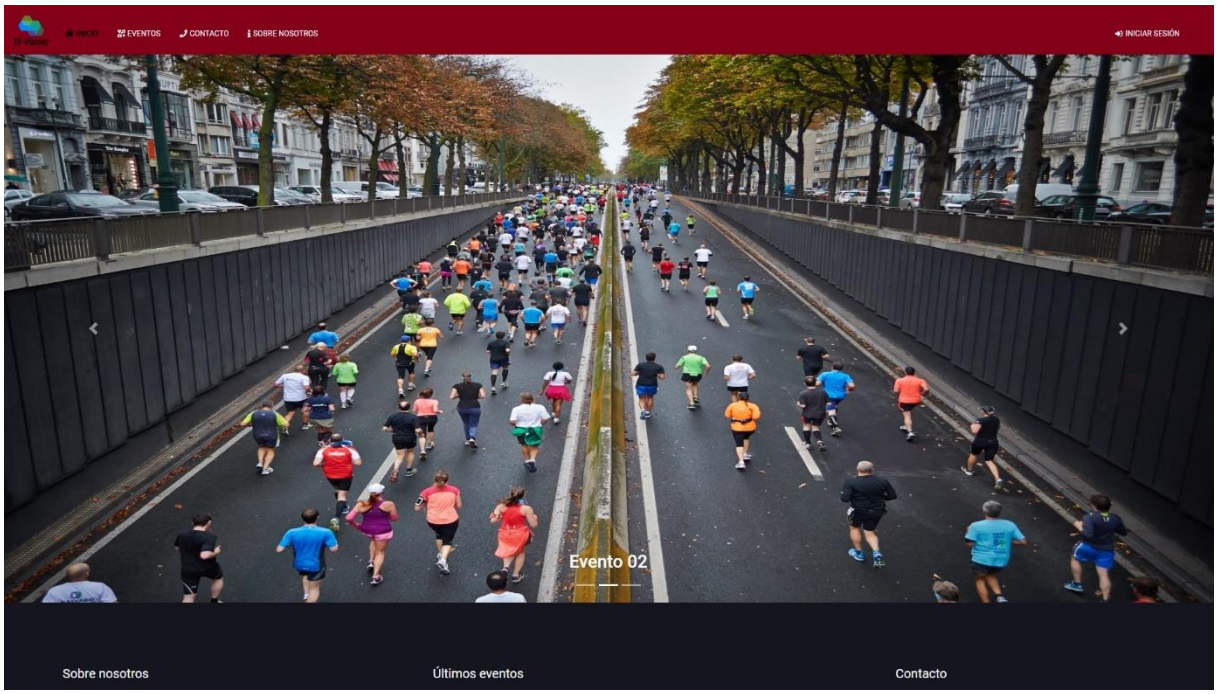


Ilustración 129: Ventana - Inicio

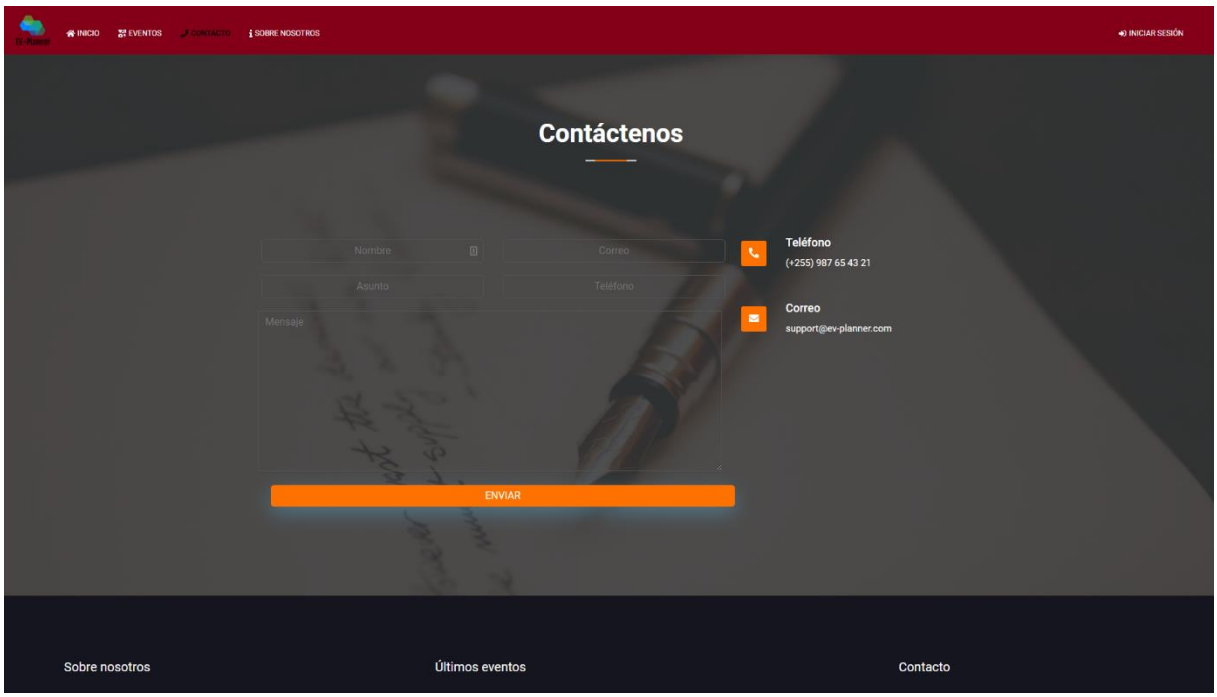


Ilustración 130: Ventana - Formulario de contacto

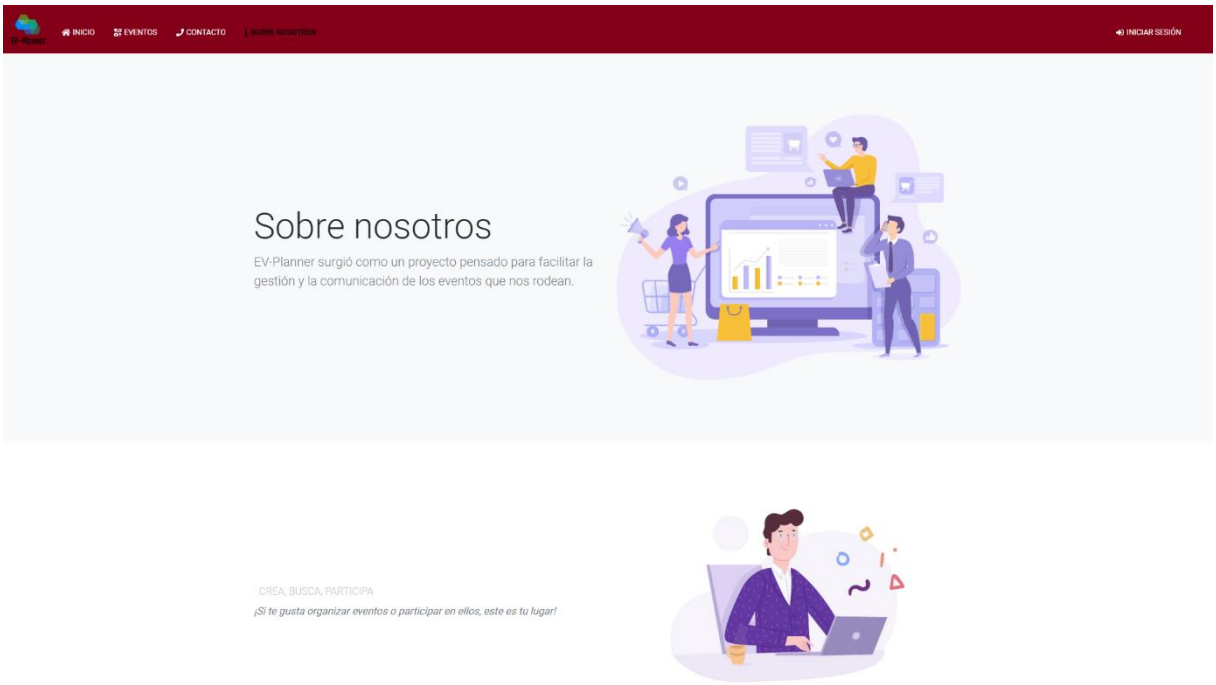


Ilustración 131: Ventana - Sobre nosotros

- Consultar eventos (Ilustraciones 132 y 133)

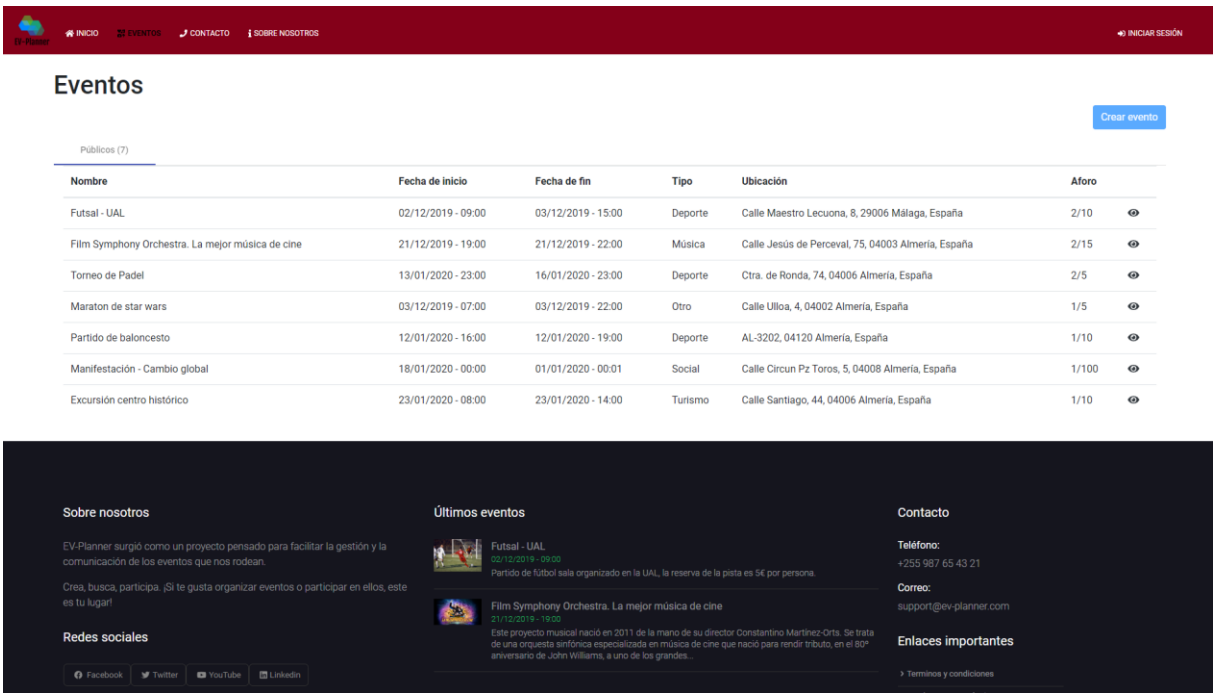


Ilustración 132: Ventana - Lista de eventos públicos

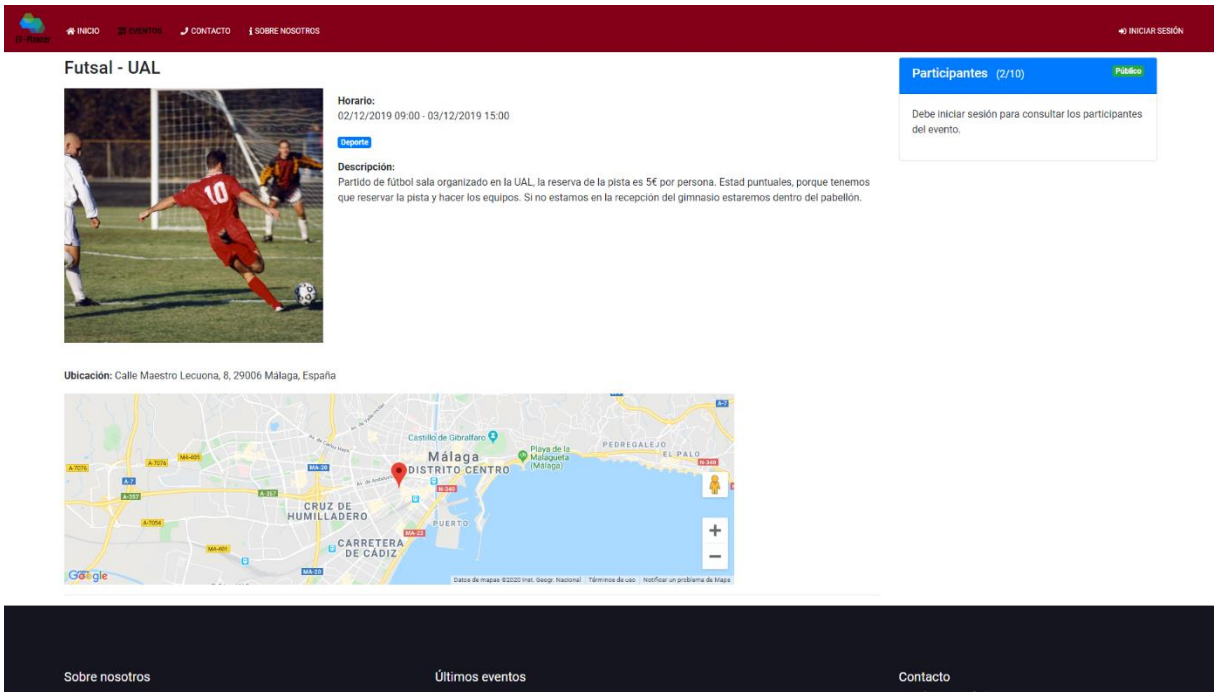


Ilustración 133: Ventana - Detalles de evento (Cibernauta)

5.1.2. USUARIO

- Consultar / Editar perfil

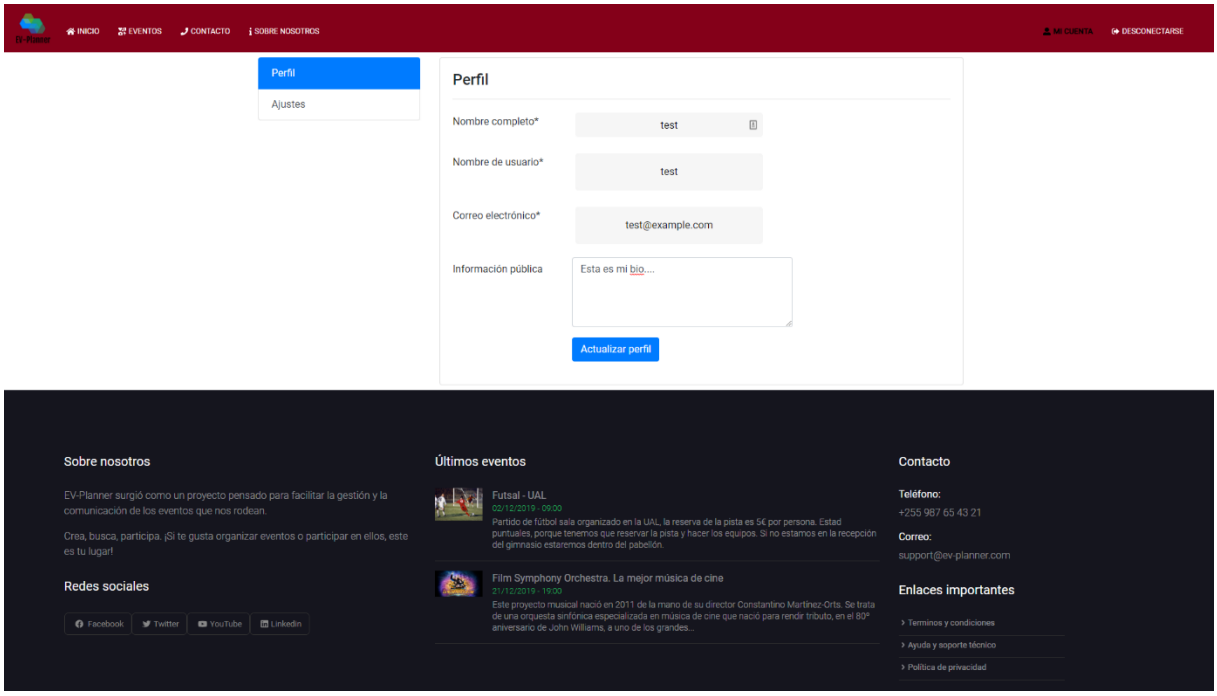


Ilustración 134: Ventana - Perfil del usuario

- Crear evento

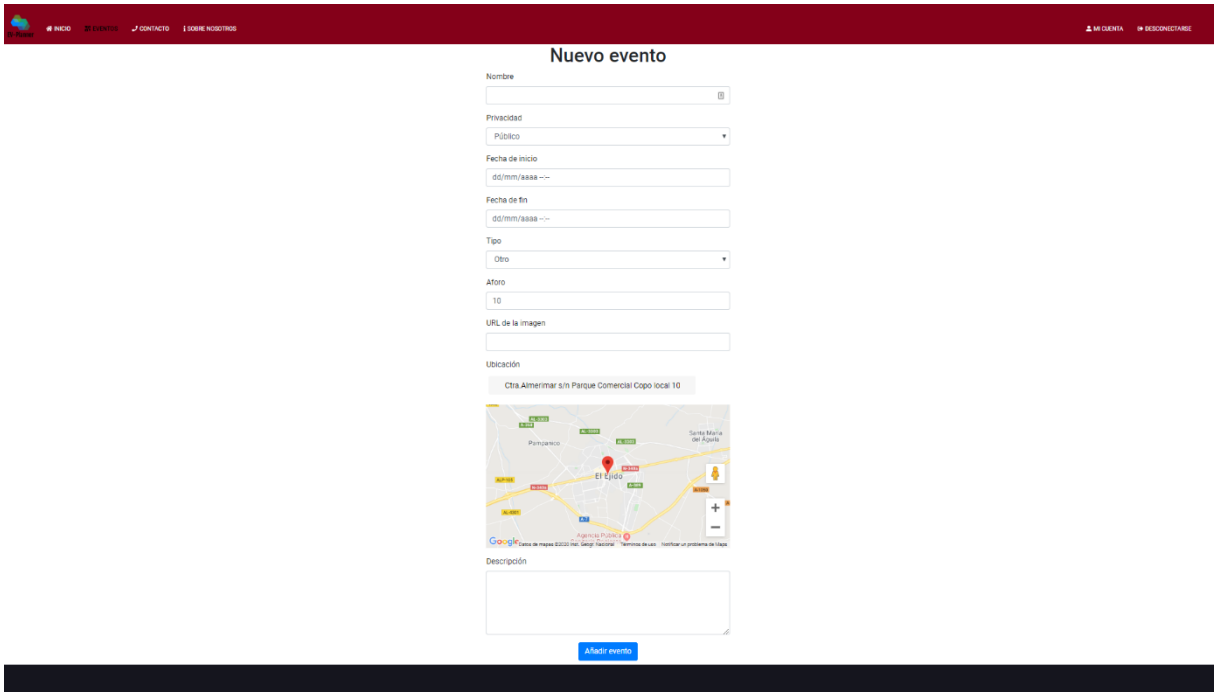


Ilustración 135: Ventana - Creación de evento

- Registrar en evento como participante / Comentar evento

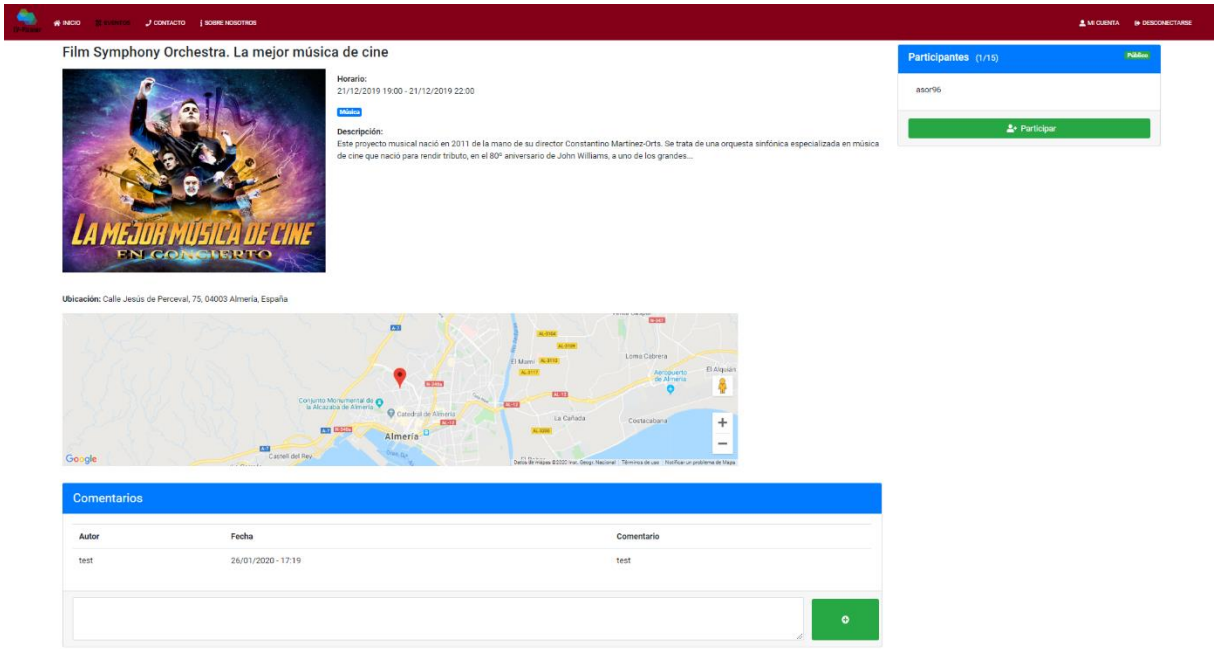


Ilustración 136: Ventana - Detalles de evento (Usuario)

5.1.3. PARTICIPANTE DE EVENTO

- Consultar eventos (Participante)

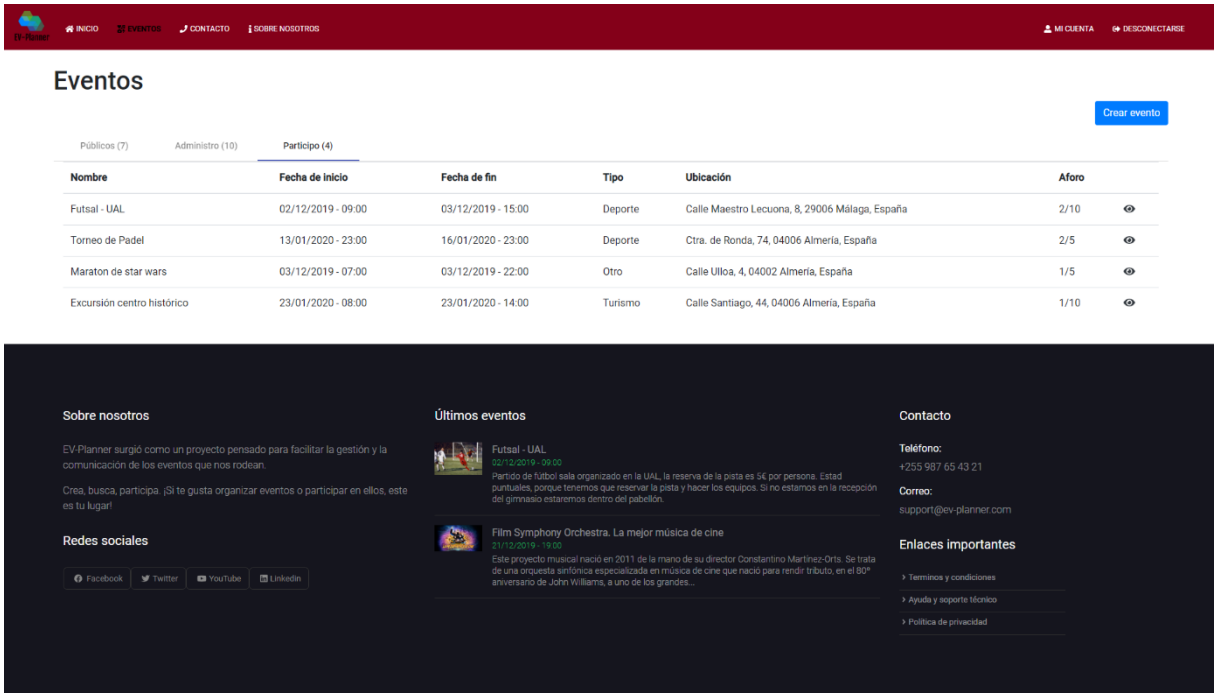
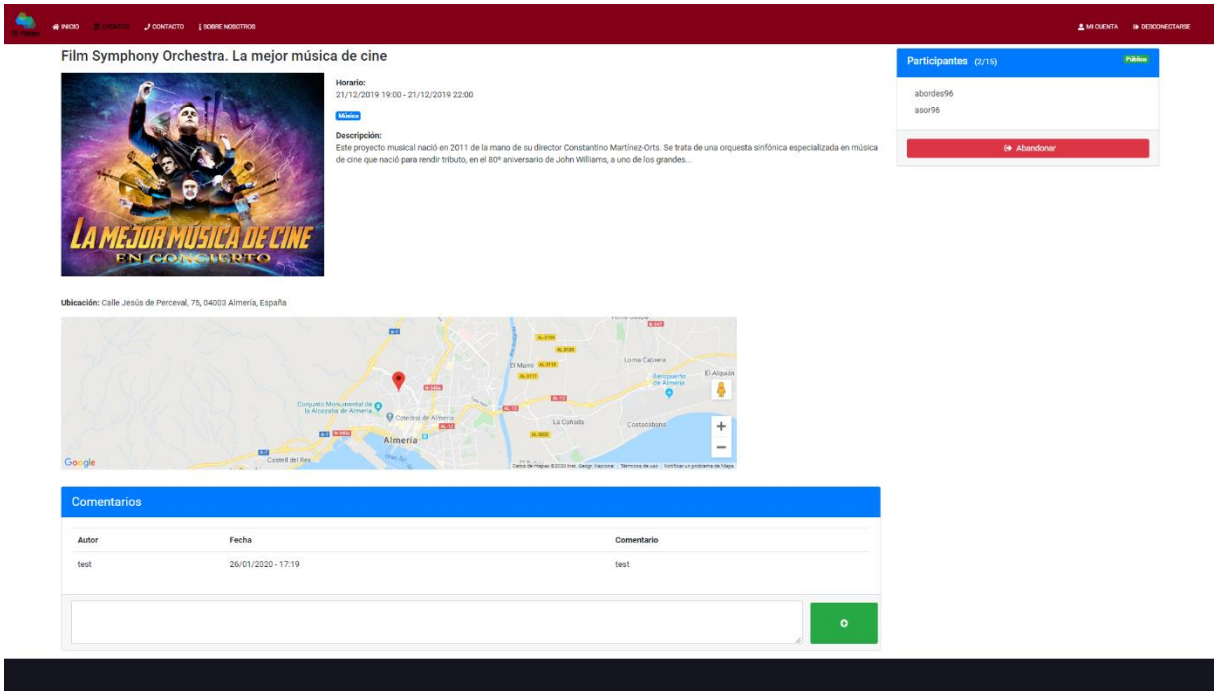


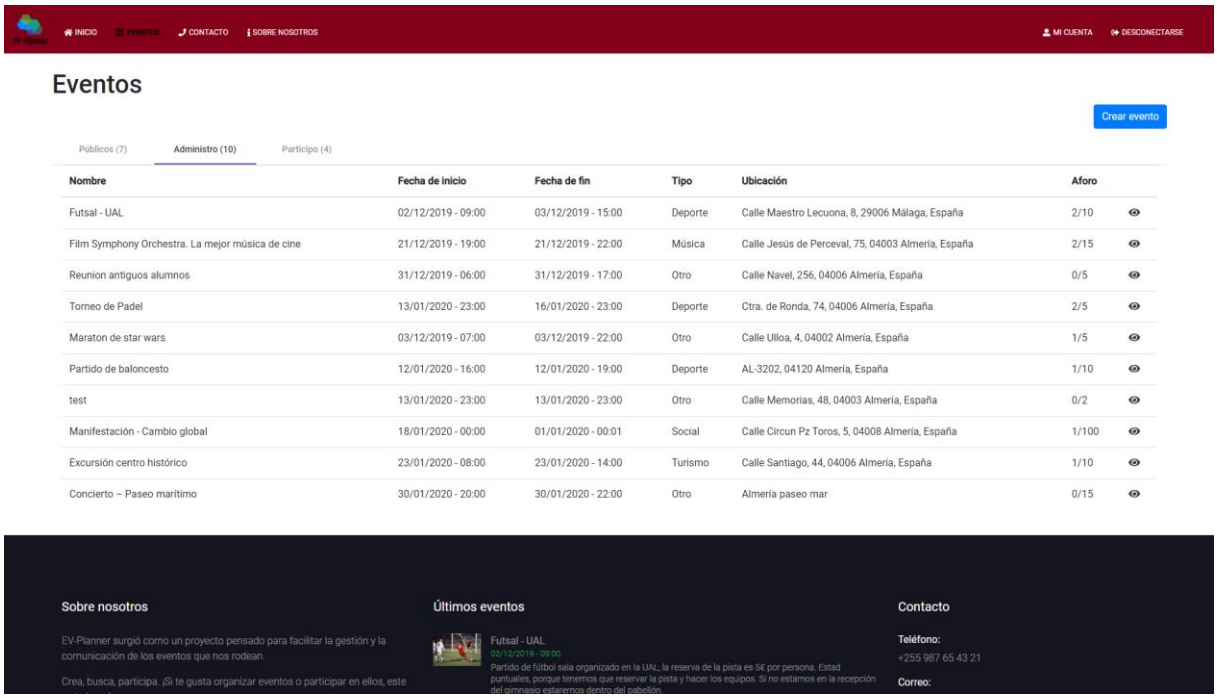
Ilustración 137: Ventana - Lista de eventos (Participante)

- Consultar eventos (Participante) / Abandonar evento

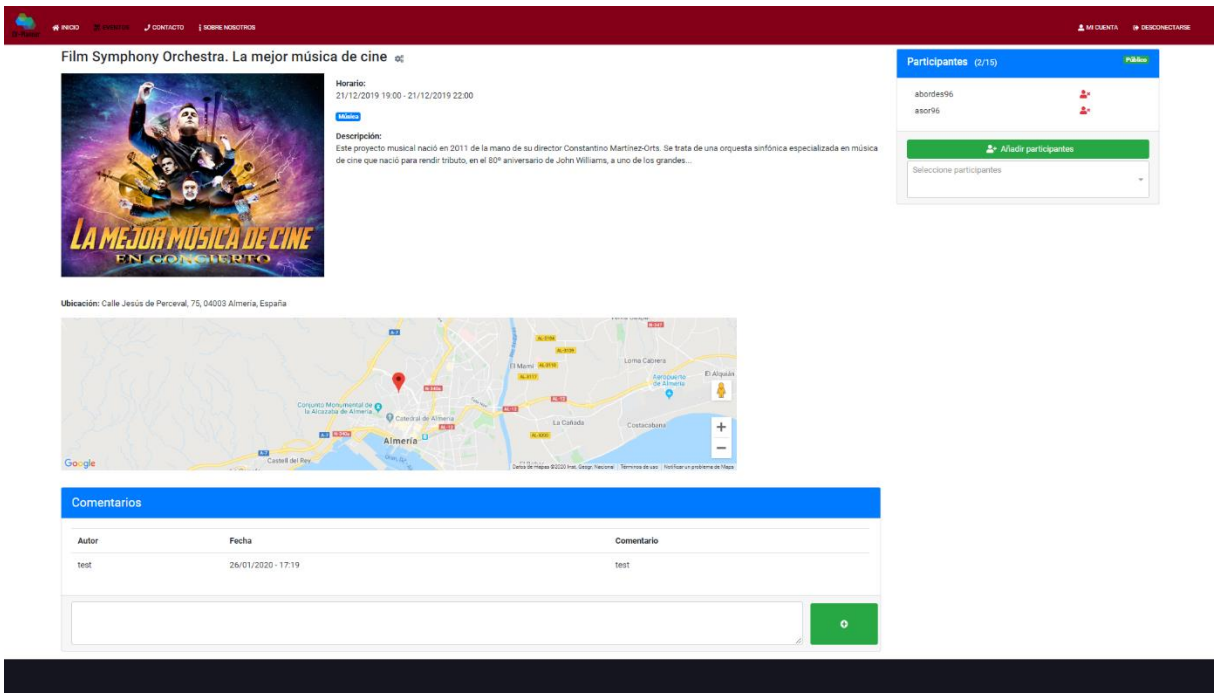


5.1.4. ADMINISTRADOR DE EVENTO

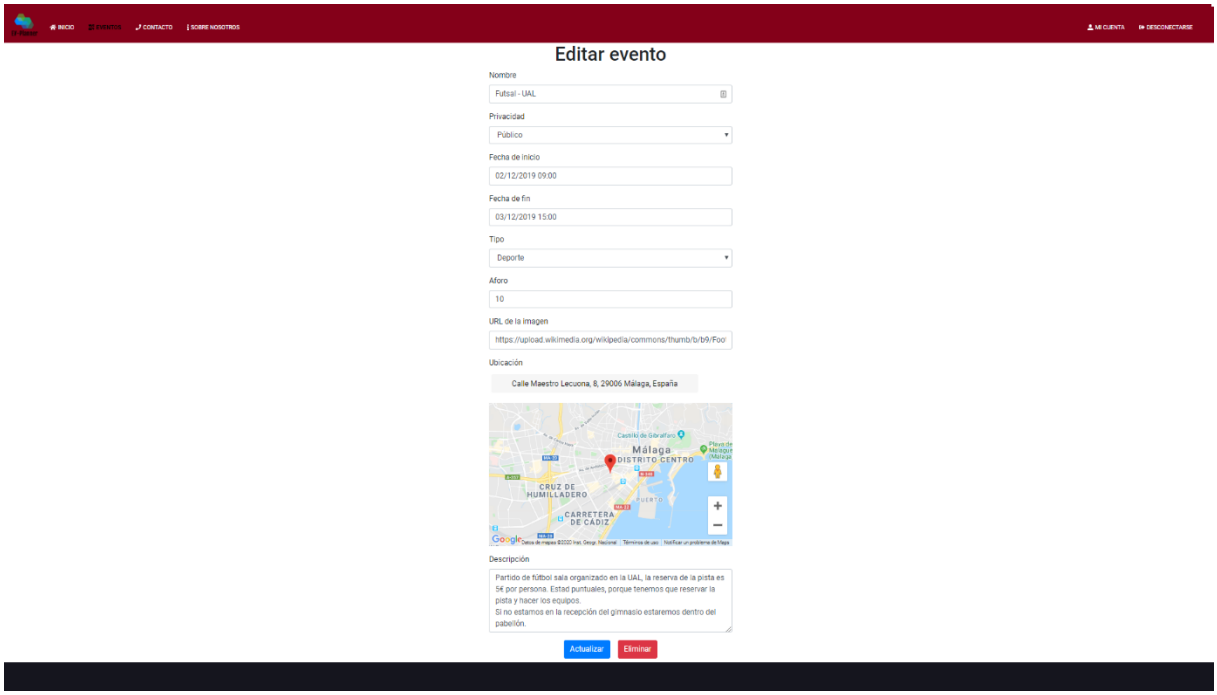
- Consultar eventos (Administrador)



- Consultar eventos (Administrador) / Invitar participante / Eliminar participante



- Editar evento / Eliminar evento



5.2. DOCUMENTACIÓN

Una parte fundamental del desarrollo software es la documentación, especialmente útil en la fase de mantenimiento, donde para implementar una nueva funcionalidad se debe conocer cómo afecta a los componentes de nuestro sistema.

La principal documentación de este proyecto se encuentra en la especificación de requisito realizada, en ella se definen quien puede interactuar con el sistema y que funciones puede realizar (3. Especificación de requisitos).

Otra fuente de documentación es la relacionada con la definición de las consultas que se pueden realizar a la API y que modelos de datos se encarga de gestionar. Esto es imprescindible si otras aplicaciones quieren interactuar con los datos de la API desarrollada. En este proyecto he utilizado Swagger para realizar todo el proceso de documentación de la API.

Los modelos definidos se pueden observar en las siguientes ilustraciones.

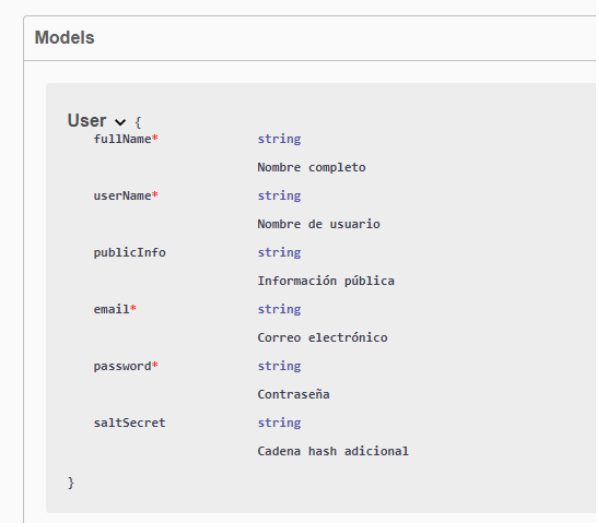


Ilustración 138: Swagger - Modelo de Usuario

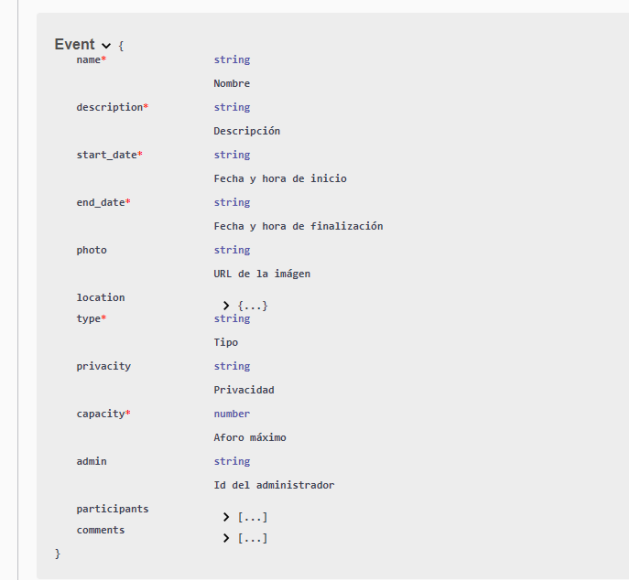


Ilustración 139: Swagger - Modelo de Evento

Ilustración 140: Swagger - Modelo de Comentario

Se han definido todas las consultas que se han implementado en API relacionadas con los usuarios y eventos.

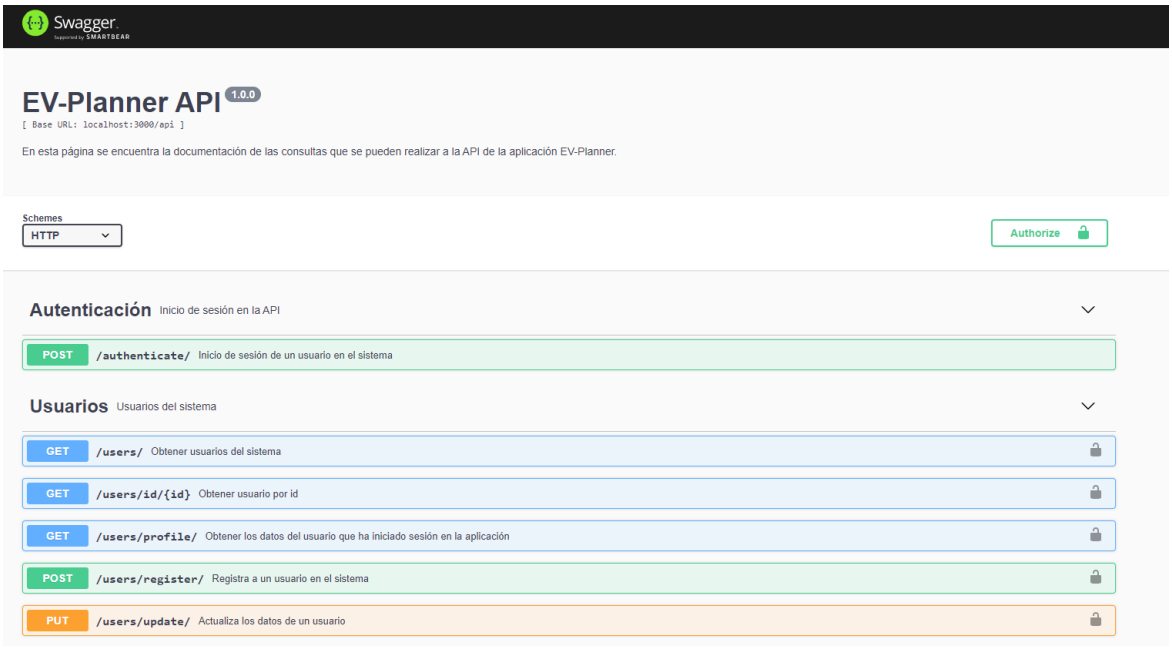


Ilustración 141: Swagger - Consultas de Usuarios

Ilustración 142: Swagger - Consultas de Eventos

Una de las partes más interesantes de Swagger es que se pueden probar las consultas de la API en línea. En el caso de que las consultas estén protegidas, como es el caso de muchas de las implementadas en el proyecto, se pueden autorizar desde la misma página introduciendo el token de autentificación.

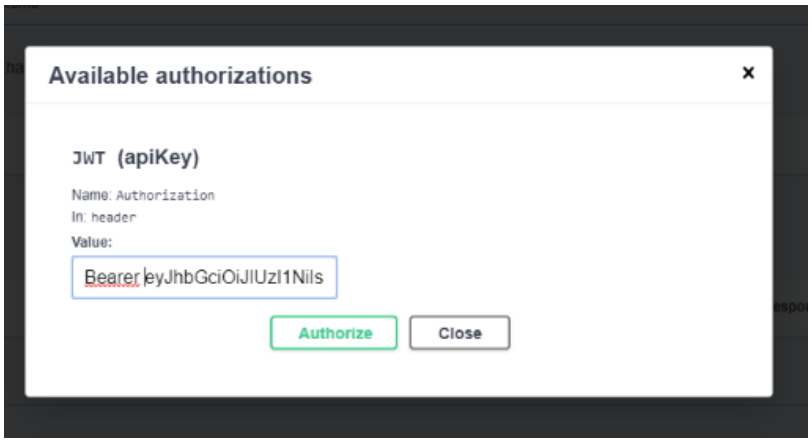


Ilustración 143: Swagger – Autententificación

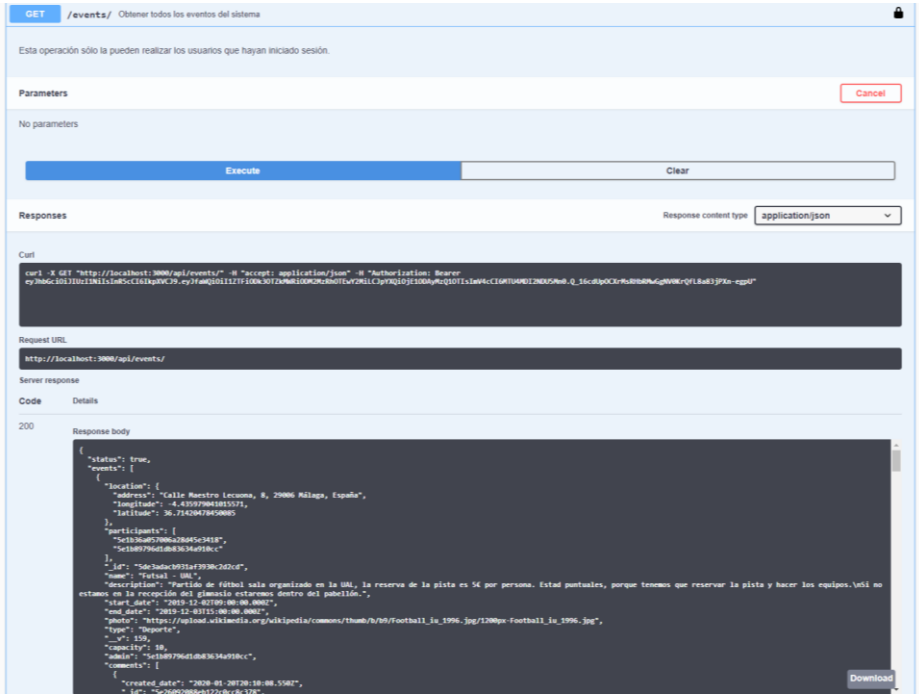


Ilustración 144: Swagger - Consulta realizada

5.3. DISEÑO ADAPTATIVO

El diseño es lo primero que se ve de una aplicación y es lo que marca las primeras impresiones del usuario, que colores se han utilizado, que fuente de letra o como de sobrecargada están las ventanas son ejemplos de elementos a tener en cuenta, pero el diseño va más allá, una de los principales factores a tener en cuenta es el modelo mental que desarrolla el usuario al interactuar con la aplicación y esto está relacionado con la usabilidad y posición de los elementos en la interfaz, es aquí donde un diseño adaptativo tiene un papel crucial.

Hoy en día, interactuamos con aplicaciones en cualquier lugar y con cualquier dispositivo, desde móviles hasta ordenadores o tabletas. Cada dispositivo tiene sus características como es el tamaño de la pantalla, por norma general en un móvil caben menos elementos al mismo tiempo en pantalla que en una tableta y esta que en un ordenador, por eso hay que procurar que las aplicaciones se adapten al dispositivo que se ejecuten para facilitar su usabilidad.

La aplicación desarrollada es un planificador de eventos y es deseable que se puedan consultar las ubicaciones de los eventos en los que los usuarios participan o administrar los eventos que se crean desde cualquier lugar, por ello he utilizado Bootstrap para realizar un diseño adaptativo, en las siguientes imágenes se ven ejemplos de la ventana de detalles de un evento ejecutándose con un tamaño de pantalla de una tableta y de un móvil.

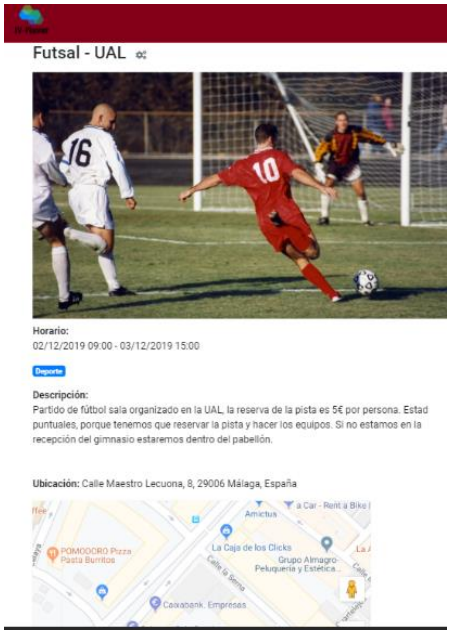


Ilustración 145: Aplicación en tableta

Ilustración 146: Aplicación en móvil (1)

Ilustración 147: Aplicación en móvil (2)

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. CONCLUSIONES

Con este proyecto se han completado todos los objetivos que se han planteado en la sección 1.4. Objetivos de este documento: Por un lado se ha desarrollado una aplicación de gestión de eventos a lo largo de dos versiones de desarrollo, incluyendo una API, utilizando una metodología *full-stack*, y por otra parte, se ha documentado de manera detallada todo el proceso de ingeniería software que hay detrás y un conjunto de buenas prácticas a seguir , como es el caso de la elaboración de la especificación de requisitos, la extracción y reutilización de componentes, seguir un diseño adaptativo, asegurar el acceso a los datos, documentar adecuadamente la API desarrollada o la realización de pruebas.

6.2. TRABAJO FUTURO

La aplicación, aunque es funcional y ya se han implementado las principales características que se han propuesto en la especificación de requisitos descrita en este documento, todavía pueden realizarse muchas mejoras que aseguren una mejora en su calidad, usabilidad y atractivo como las que se citan a continuación.

- Gestión de notificaciones: La validación del registro de usuarios por correo electrónico y la creación de enlaces de invitación a eventos.

- Traducción: Traducir la aplicación a varios idiomas como el inglés o el francés.

- Diseño: Hacerla más atractiva y fácil de usar para los usuarios.

- Despliegue: Analizar el tráfico de datos en la aplicación y decidir qué hardware utilizar para alojar los datos y permitir su escalabilidad.

- Nuevas funcionalidades: Sistema de amigos, eventos próximos en ubicación, subscripción a canales de eventos e incluso desde una perspectiva comercial incluir eventos promocionados, son algunas de las innumerables posibilidades que se pueden implementar.

- Pruebas automatizadas: Aunque se han realizado pruebas de aceptación, realizar pruebas automatizadas pueden servir de mucha ayuda cuando se implementan nuevas características para comprobar que todo sigue funcionando correctamente.

7. BIBLIOGRAFÍA

7.1. LIBROS

- Holmes, S., & Harber, c. (2019). Getting MEAN with Mongo, Express, Angular, and Node, Second Edition.
- Kniberg, H., (2015). Scrum and XP from the Trenches - 2nd Edition: Editorial InfoQ.
- Lesyuk, A., (2013). Mastering Redmine: Editorial PACKT Publishing.
- Sánchez, S., Sicilia, M.A., & Rodríguez, D., (2011). Ingeniería del Software. Un enfoque desde la guía SWEBOK: Editorial Garceta.

7.2. MISCELÁNEA WEB

- ¿Qué es y para que sirve NodeJS?. (2015). Apasionados: Marketing en Internet Valencia y Desarrollo Web Valencia. Recuperado el 24 de Diciembre de 2019, desde https://apasionados.es/blog/nodejs-4430/
- Alarcón, J. (2015). Qué es el stack MEAN y cómo escoger el mejor para ti - campusMVP.es. Recuperado el 17 de Diciembre de 2019, desde https://www.campusmvp.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx
- Angular (framework). Es.wikipedia.org. Recuperado el 24 de Diciembre de 2019, desde https://es.wikipedia.org/wiki/Angular_(framework)
- Aplicaciones móviles para eventos ¿qué nos aportan?. (2019). Recuperado el 15 de Diciembre de 2019, desde https://bevents-spain.com/app-eventos/
- Barcia, P. (2006). 'No existe un lenguaje del chat'. Recuperado el 1 de Septiembre de 2019, desde https://www.lagaceta.com.ar/nota/162606/informacion-general/no-existe-lenguaje-chat.html
- CSS. (2019). Documentación web de MDN. Recuperado el 24 de Diciembre de 2019, desde https://developer.mozilla.org/es/docs/Web/CSS
- Díaz, A. (2016). Angular 2, Conceptos Core: Data Bindings - Desarrollando sobre SharePoint. Blogs.encamina.com. Recuperado el 6 de Enero de 2020, desde https://blogs.encamina.com/desarrollandosobresharepoint/angular-2-bindings/

- Estanyol i Casals, E. (2012). Nuevas tendencias en organización de eventos. Recuperado el 15 de Diciembre de 2019, desde https://comein.uoc.edu/divulgacio/comein/es/numero08/articles/Article-Elisenda-Estanyol.html
- Express - Infraestructura de aplicaciones web Node.js. Expressjs.com. Recuperado el 24 de Diciembre de 2019, desde https://expressjs.com/es/
- Full Stack Developer: cómo convertirse en el desarrollador más completo. (2018). [Blog]. Recuperado el 15 de Diciembre de 2019, desde https://www.digitaltechinstitute.com/expertos-full-stack-developer/
- García, J. (2016). ¿Qué es LAMP Stack? aprende a instalar los componentes para crear servidores web en Ubuntu. Recuperado el 15 de Diciembre de 2019, desde https://rootear.com/ubuntu-linux/que-es-lamp-stack-instala-componentes-servidores
- HTML. (2019). Documentación web de MDN. Recuperado el 24 de Diciembre de 2019, desde https://developer.mozilla.org/es/docs/Web/HTML
- JavaScript. (2019). Documentación web de MDN. Recuperado el 24 de Diciembre de 2019, desde https://developer.mozilla.org/es/docs/Web/JavaScript
- Kemp, S. (2019). DIGITAL 2019: GLOBAL DIGITAL OVERVIEW. Recuperado el 14 de Diciembre de 2019, desde https://datareportal.com/reports/digital-2019-global-digital-overview
- La base de datos líder del mercado para aplicaciones modernas. Recuperado el 24 de Diciembre de 2019, desde https://www.mongodb.com/es
- Mean Stack: el pura sangre de los Stack. (2017). Recuperado el 16 de Diciembre de 2019, desde https://www.digitaltechinstitute.com/mean-stack/
- Morales, C. (2012). La tecnología y las relaciones interpersonales: ¿Cómo nos afecta? Recuperado el 14 de Diciembre de 2019, desde https://www.fayerwayer.com/2012/07/la-tecnologia-y-las-relaciones-interpersonales-como-nos-afecta/
- Sierra, M. (2019). Comunicación con hologramas, el siguiente paso tras el 5G en el que trabaja la industria. Recuperado el 1 de Septiembre de 2019, desde https://www.vozpopuli.com/economia-y-finanzas/Comunicacion-hologramas-5G-trabaja-industria_0_1179782335.html
- Todo lo que tienes que saber sobre la gestión de eventos. Recuperado el 14 de Diciembre de 2019, desde https://www.ar-hotels.com/blog/todo-lo-que-tienes-que-saber-sobre-la-gestion-de-eventos

- Vaati, E. (2018). Autenticación Angular Con JWT. Code Envato Tuts+. Recuperado el 2 de Enero de 2020, desde https://code.tutsplus.com/es/tutorials/jwt-authentication-in-angular--cms-32006
- Vicente, D. (2017). Encriptación de password en NodeJS y MongoDB: bcrypt. Solid GEAR. Recuperado el 5 de Enero de 2020, desde https://solidgeargroup.com/password-nodejs-mongodb-bcrypt/?lang=es

Resumen/Abstract

Nuestra sociedad y la forma en la que nos comunicamos están fuertemente vinculados al mundo electrónico en el que vivimos, pero eso no quiere decir que la tecnología nos restrinja en nuestras relaciones interpersonales. EV-Planner, la aplicación qué he desarrollado, es una herramienta que permite gestionar los eventos de nuestro entorno, permitiendo desde crear eventos e invitar a participantes hasta buscar los eventos más próximos a nuestra ubicación. Otro de los objetivos principales del proyecto es mostrar de manera detallada el proceso de desarrollo de una aplicación utilizando una metodología ágil y un marco de desarrollo *full stack* basado en JavaScript, como es el caso de MEAN, utilizado en este proyecto.

Our society and the way in which we communicate are strongly linked to the electronic world in which we live, but that does not mean that technology restricts us in our interpersonal relationships. EV-Planner, the application that I have developed, is a tool that allows us to manage the events of our environment, allowing from creating events and inviting participants to find the events closest to our location. Another of the main purposes of the project is to show in detail the process of developing a project using an agile methodology and a *full stack* development framework based on JavaScript, as is the case of MEAN, used in this project.