UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Sistema de gestión para una asociación de la tercera edad

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Molero Castillo, Carmelo

Tutor/a: Insfrán Pelozo, César Emilio

CURSO ACADÉMICO: 2021/2022

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Sistema de gestión para una asociación de la tercera edad

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Carmelo Molero Castillo

Tutor: César Emilio Insfrán Pelozo

Curso académico: 2021/2022

Resumen

En la actualidad, la asociación para la tercera edad ‘El Jardín’ de Peñarroya Pueblonuevo cuenta con una aplicación desarrollada en VBA Excel para la gestión del club. En ella controlan altas y bajas de socios y pagos de cuotas. Debido a las características de la aplicación, esta depende tanto de Microsoft Windows como de Microsoft Excel para su ejecución. Además de tener una interfaz poco amigable y para nada adaptada el tipo de usuario que habitualmente la usa (personas de +65 años con poca experiencia en tecnologías).

En este documento se detalla la creación de una aplicación web para gestionar un club, adaptando la interfaz a las necesidades de los socios, responsive para el uso en cualquier tipo de dispositivo, multiplataforma y minimizando costes de implementación. Usando tecnologías web innovadoras como nextjs, nodejs con typescript, jest y cypress, entre otros.

Palabras clave: asociación, tercera edad; gestión integral, socios.

Abstract

Currently, the association for the elderly 'El Jardín' of Peñarroya Pueblonuevo has an application developed in VBA Excel for the management of the club. In it they will control registrations and cancellations of partners and payments of quotas. Due to the characteristics of the application, it depends on both Microsoft Windows and Microsoft Excel to run. In addition to having an unfriendly interface and not at all adapted to the type of user who usually uses it (people over 65 with little experience in technology).

This document details the creation of a web application to manage a club, adapting the interface to the needs of the members, responsive for use on any type of device, multiplatform and minimizing implementation costs. Using innovative web technologies like nextjs, nodejs with typescript, jest and cypress, among others.

Keywords: association, elderly, comprehensive management, partners.

Agradecimientos

Cada vez que recuerde estos cuatro magníficos años de mi vida, los primeros recuerdos que me vendrán a la cabeza serán los maratonianos fines de semana en las cabinas de la biblioteca central con el 'Teamblioteca'*, que después con el COVID-19, se volvieron en interminables sesiones de Discord, con apuntes, clases grabadas, series, películas, y alguna que otra captura de pantalla en épocas de exámenes.*

Desconocidos que pasaron a ser compañeros, y compañeros que se convirtieron en amigos.

A mi familia, que en un principio no tenían claro esto de dejar de trabajar para volver a las aulas como estudiante, otra vez. Pero que una vez tomada la decisión, nunca han dudado de mí, y jamás dejaron de apoyarme.

Y por supuesto, Álvaro Hernández Perales, este proyecto es casi tan mío como tuyo. Largas tardes configurando, programando, diseñando, que si no hubiera sido por tu paciencia, todavía no lo habría terminado (da igual cuando se lean estas palabras).

A todos, muchísimas gracias por ser y estar.

Índice de contenidos

Índice de imágenes

Índice de tablas

1 Introducción

La asociación para la tercera edad "*El Jardín*" de Peñarroya – Pueblonuevo necesita modernizar su sistema de gestión de socios ya que, hasta hace poco, todo el proceso se estaba haciendo mediante un sistema de fichas manuscritas, y donde todo el tratamiento de la información se hacía con sistemas propios de finales del siglo pasado.

Ante la imposibilidad de manejar la información de manera eficiente y en tiempo real (cobros y pagos pendientes, cantidad de socios activos y cantidad de socios con pagos de cuota pendientes), se creó un sistema informatizado bastante básico, mediante Excel, donde se pretendía simular una base de datos haciendo uso de *macros*. Este método, aunque cubre las necesidades básicas, no deja de ser un sistema poco adaptado a personas con pocos conocimientos informáticos, carece de una interfaz *responsive* y está muy lejos de ser amigable para el tipo de usuario al que va dirigido.

Por todo lo anterior, y habiendo interiorizado bastante rápido lo que la tecnología puede hacer por ellos, la asociación necesita una aplicación que, además de integrar lo poco que ya tienen desarrollado, también implemente nuevas funcionalidades. Algunos ejemplos de lo que les gustaría que la aplicación implementara son el uso de mensajería con sus socios (para informar de las actividades programadas), comunicación con el consistorio (del cual dependen), seguimiento de los socios con enfermedades por parte de los familiares autorizados, intolerancias alimenticias y alergias (para contratar los viajes y el régimen alimenticio apropiado) o la publicación de eventos o actividades.

En el presente documento se presenta una solución que consiste en la realización de una aplicación web adaptada a personas con bajos (o nulos) conocimientos informáticos, a fin de facilitar las labores típicas de dicha gestión (altas y bajas de socios, pagos de cuotas, balance de cuentas.). Y además implemente un cambio de paradigma, ya que actualmente solo tiene acceso a esta información los miembros de la directiva, pero se quiere permitir al resto de socios (o sus familiares) que puedan actualizar la información personal, apuntarse a las actividades o reservar plazas de viaje.

También se pretende aprovechar este desarrollo para la implementación de otros servicios más avanzados, como puede ser la notificación de eventos por mensajería instantánea (*WhatsApp*, *Telegram* o *SMS*), seguimiento por parte de los familiares en caso de imprevistos en las actividades donde sus parientes participen.

Contamos con un requisito bastante restrictivo, y es que, al tratarse de una organización sin ánimo de lucro, y debido a que cuentan con ingresos escasos, se debe minimizar al máximo los gastos en sistemas de gestión (es necesario llegar al coste 0, como requisito indispensable), por lo que se ha decidido hacer un uso de tecnologías con licencia libre y uso de servidores y servicios gratuitos.

1.1 Motivación

¿Por qué he elegido este proyecto? Debido a mi estrecha relación con los integrantes de la actual directiva de la asociación, soy plenamente consciente de las carencias derivadas de no tener un sistema informático adaptado a unas necesidades concretas.

Aunque en la actualidad hay sistemas informáticos que pueden cubrir con creces las labores de gestión informática requeridas para una asociación, nos encontramos con que, o están dirigidos a un tipo de usuario que nada tiene que ver con los integrantes de la asociación, con interfaces hombre-máquina complejas. O tienen unos costes que no pueden ser asumidos por la entidad, puesto que la principal fuente de ingresos depende de la recaudación de la cuota anual de sus socios. Por tanto, estos sistemas son incompatibles con el nivel de ingresos.

Se pretende crear una aplicación multiplataforma *Open Source*, totalmente gratuita, desplegada en un servidor web con alternativa de almacenamiento sin costo. Que no dependa de licencias corporativas como Windows o Microsoft Excel, capaz de ejecutarse desde dispositivos tan dispares como una *Raspberry pi 3b+*, una *tablet* o un *smartphone*.

Para mí, se trata de una gran oportunidad para desarrollar un software totalmente funcional, que forme parte de mi porfolio personal. Sin olvidar que ser trata de una gran oportunidad para mejorar mi conocimiento sobre tecnologías de desarrollo web, tanto de *frontend* como de *backend*, haciendo uso de *React*, *TailwindCSS* o *NodeJS* y *Typescript*.

Siendo mucho más ambicioso, me gustaría conseguir desarrollar un sistema simple, a la vez que robusto, que pueda ser usado por varias asociaciones de la localidad, facilitando la transición de los sistemas de gestión manuales a un sistema informatizado, con una curva de aprendizaje muy rápida, apto para el uso de cualquier tipo de usuario, homogeneizando la comunicación con el consistorio.

En resumen, me motiva construir una aplicación completa y totalmente funcional desde su inicio. Usando para ello tecnologías actuales y teniendo en mente que sea totalmente escalable, mantenible, adaptable y que cumpla con todos los requisitos requeridos por la parte interesada.

1.2 Objetivos

En la actualidad, el club, cuenta con una aplicación desarrollada en *VBA Excel*, totalmente dependiente de licencias de *Microsoft* ya que, para poder ejecutarla, es necesario tener habilitadas las macros en *Excel*, y esto, además de ser dependiente de *Microsoft Excel*, solo funciona bajo *Windows*.

Aunque dicha aplicación cumple con los requisitos que se especificaron en un principio, cuenta con varias limitaciones funcionales (intrínsecas a la naturaleza del lenguaje de programación usado). También hay que añadir que la ‘experiencia de usuario’ es bastante tosca, porque no se tuvo en cuenta las características del público para el que va dirigido (personas con más de 65 años, no acostumbradas a trabajar con entornos informáticos).

En la Figura 1 y Figura 2 se muestran las pantallas principales de la aplicación actual, donde se puede observar que se trata de un “simple” *Excel* que intenta aparentar ser una aplicación mucho más sofisticada y/o especializada.

Además de lo descrito anteriormente, el club necesita adaptar urgentemente el tratamiento de la información a la normativa de protección de datos, ya que se está cometiendo el error de no informar a los socios la finalidad con la que se recogen los datos personales necesarios, cuál es su tratamiento, las acciones de protección se están teniendo en cuenta para protegerlos frente a accesos no autorizados, etc.

La funcionalidad de la aplicación actual está centrada en la recolección de datos personales, (datos de contacto, dirección y estado del pago de cuotas de los socios). Pero el club está interesado en poder explotar al máximo las tecnologías de la información, a fin de poder ofrecer un buen servicio a todos sus socios (y a los familiares directos de estos). Por ejemplo, está interesado, entre otras funciones, en enviar información de los eventos que se celebran en el club por mensajería instantánea (*WhatsApp y/o Telegram*), tener una mejor organización de los eventos, poder controlar pagos y cobros (tanto de socios como de proveedores) para poder presentar cuentas a dichos socios y al ayuntamiento (del cual dependen). Dada las características de edad de los socios, también les gustaría contar con algún sistema de aviso en el que los familiares directos puedan saber la ubicación del socio cuando asisten a excursiones, o gestionar si tienen alergias e intolerancias alimenticias a la hora de contratar el catering de los hoteles.

Figura 1: Aplicación actual, pantalla de inicio.

Figura 2: Aplicación actual, panel principal.

En resumen, este proyecto tiene como propósito realizar un sistema para la gestión de la asociación para la tercera edad "El Jardín de Peñarroya - Pueblonuevo". En la actualidad, la asociación, cuenta con una aplicación muy rudimentaria desarrollada en *VBA Excel* con el que tienen el control de altas y bajas de los socios, así como los pagos de las cuotas. Sin embargo, esta aplicación tiene muchas limitaciones, por lo que se hace necesario desarrollar un sistema que satisfaga las necesidades y que sea más sencilla de utilizar.

También se desea que la aplicación sea lo suficientemente flexible para poder incorporar cambios en el futuro.

Además de las funcionalidades básicas indicadas anteriormente, se llevará un control de las actividades organizadas (cursos, excursiones, etc.) así también se explorará la posibilidad de incorporar módulos para actividades online, grupos de amigos, integración con redes sociales, etc.

1.2.1 Objetivos generales

- Entender y documentar los requisitos de la asociación y de las personas afectadas.
- Diseñar e implementar un sistema Web para la gestión de la asociación.
- Realizar las pruebas y la migración de los datos del sistema actual.

1.2.2 Objetivos específicos

- Desarrollar una interfaz adaptada a personas con alto desconocimiento en las nuevas tecnologías.
- Desarrollar integración de la aplicación con *socialmedia* (*WhatsApp*, *Telegram*, *Facebook*, *Instagram*, …).
- Desarrollar una aplicación escalable.
- Implementar un sistema fácilmente mantenible.
- Desarrollar una web *responsive* que se adapte a la mayoría de los dispositivos móviles.

1.3 Estado del arte / Estudio estratégico

En el mercado hay una gran variedad de programas de escritorio y aplicaciones web dedicadas a la gestión de asociaciones y *ONG's* de distinta índole. Todas estas tienen en común que se tratan de herramientas muy extensas y bastante complejas para el tipo de usuario al que estamos dar una solución.

De entre todas las soluciones actuales que hay en el mercado, hemos recogido y resumido las más conocidas de este sector en los siguientes apartados.

1.3.1 Berrly [1]

Este sistema se encarga de la gestión de todo tipo de organizaciones, clubs, federaciones, *ONG's*, etc., y se ejecuta como una aplicación web, ya que está situado en la nube.

No cuenta con ningún plan gratuito, siendo la tarifa de 35€ / mes la más económica, a los que hay que añadir otros 40€ / mes si queremos que los socios puedan tener acceso desde una aplicación móvil (precios sin IVA).

Sistema con múltiples funcionalidades para mejorar la gestión de cualquier organización no empresarial de cualquier tipo, desde asociaciones culturales hasta clubs deportivos.

A grandes rasgos, *Berrly* cuenta con:

- Correos personalizados como método de comunicación.

- Un gestor documental, con almacenamiento en la nube. Para poner a salvo toda la documentación importante del socio.
- Zona privada para los socios.
- Control de eventos con entradas y control de aforo.
- Digitalización de la tesorería y cobro de cuotas.
- Encuestas y votaciones para conocer la opinión de sus socios.

1.3.2 Playoff [2]

En este caso, la aplicación es muy parecida a la anterior, ya que ofrece lo mismo. Se trata de un sistema de gestión para organizaciones, asociaciones culturales, centros cívicos, fallas, hermandades, etc, y también está situado en la nube.

Esta aplicación no cuenta con ningún plan gratuito, siendo la tarifa de 25€ / mes la más económica, a los que hay que añadir otros 35€ / mes si queremos que los socios puedan tener acceso desde una aplicación móvil (precios sin IVA).

Esta aplicación no cuenta con ningún plan gratuito, siendo la tarifa de 25€ / mes la más económica, a los que hay que añadir otros 35€ / mes si queremos que los socios puedan tener acceso desde una aplicación móvil (precios sin IVA).

Las ventajas de *Playoff* son, entre otras:

- Alertas por impagos.
- Preinscripción anticipada de forma online.
- Comunicaciones segmentadas, dependiendo del tipo de socio al que va dirigido.
- Control de gastos y remesas de recibos.
- Gestión de grupos familiares.

1.3.3 Cucunver [3]

En este caso estamos en una aplicación específica para asociaciones sin ánimo de lucro. Pero que al igual que las anteriores, también se encuentra alojada en la nube.

Esta aplicación tampoco cuenta con plan gratuito, siendo la tarifa de 25€ / mes la más económica. La gran diferencia es que este sistema no cuenta con ninguna aplicación para dispositivos móviles (precios sin IVA).

Los puntos fuertes de *Cucunver* son:

- Cobro de cuotas a través de tarjeta bancaria
- Base de datos centralizada.
- Añadir tutores legales y supervisores a los usuarios que necesiten una persona a su cargo.
- Importación de datos.
- Soporte a través de la aplicación.
- Logo/escudo de la entidad personalizado.
- Añadir etiquetas a los usuarios para organizarlos y filtrarlos de forma sencilla (alumnos, voluntarios, profesores, colaboradores, etc.).
- Inscripción online.

- Exportar informes de ingresos y gastos en *pdf, MS Excel.*
- Aceptar o denegar las inscripciones online de los usuarios a las actividades.
- Lista de asistencia a actividades.
- Exportar el libro de asociados.
- Anotaciones ilimitadas en todas las secciones.
- Crear carpetas en las distintas secciones para organizar la documentación de la asociación (estatutos, actas, etc.).
- Comunicación interna entre la entidad y los usuarios.
- Permisos de privacidad para los editores de la entidad.
- Control de los bienes de la asociación mediante inventario.

1.3.4 Taclia [4]

Esta aplicación es totalmente gratuita, y está orientada a facilitar la digitalización e informatización de los negocios que aun queden rezagados en plena época digital.

Entre sus características más relevantes se encuentran:

- Gestión de clientes (CRM).
- Seguimiento de tareas y citas.
- Presupuestos y facturas.
- Compras y tickets de gasto.
- Control horario y fichaje.
- Formularios y plantillas.
- Localización de visitas y servicios.
- Inventario y activos.
- Informes automáticos.
- Toma decisiones para tu negocio con datos centralizados.

1.4 Impacto esperado

1.4.1 Impacto en la directiva de la asociación

- Eficiencia en el tratamiento de los datos. No estarán sujetos a fichas manuscritas ni a programas obsoletos para poder consultar y/o modificar la información relativa al estado de cuentas, estado de socios, eventos, etc.
- Información actualizada en tiempo real, desde cualquier dispositivo con conexión a internet.
- No estarán sujetos al uso a ningún programa o sistema operativo propietario, por tanto, no necesitarán pagar por software.

1.4.2 Impacto en los socios

- Pasarán a ser una parte más activa en la asociación.
- Podrán hacer el seguimiento del estado de eventos en tiempo real.
- Modificación de la información asociada a su perfil.
- Indicarán sus alergias e intolerancias en su perfil, y podrá ser tenido en cuenta en todo momento.

1.4.3 Impacto en los familiares de los socios

- Notificación en tiempo real de los contratiempos que hayan sucedido en los eventos.
- Localización de sus seres queridos en viajes.

1.4.4 Impacto en el ayuntamiento

- Auditorías en tiempo real, sin desplazamientos del personal responsable al local de la asociación. Se evitarán los intermediarios en el tratamiento de la información, ya que estará siempre disponible.

1.5 Estructura del documento

El presente documento cuenta con la siguiente estructura:

- **Especificación de requisitos:** En este apartado se partirá de la descripción del problema y a partir de técnicas estándar o adaptadas (IEEE 830, Casos de Uso, Historias de Usuario, Pruebas de Aceptación), se desarrollará el modelado conceptual y se elegirán las tecnologías a usar.
- **Proceso software:** Documenta el proceso software (clásico o ágil) elegido en el desarrollo del proyecto.
- **Desarrollo e implementación:** Documentar tecnologías, herramientas y lenguajes utilizados. También muestra las implementaciones y/o configuraciones más relevantes.
- **Pruebas y calidad del software:** Diseño, aplicación y resultados de las pruebas que se van a realizar como verificación tanto de que la aplicación funciona de acuerdo con los requisitos como que la aplicación es mantenible.
- **Mantenimiento y gestión de versiones:** Documentación sobre las herramientas de gestión de versiones y mantenimiento usadas.
- **Conclusiones:** Análisis de lo desarrollado en el TFG, tareas pendientes a desarrollar en un futuro, posibles mejoras a partir de lo ya implementado.
- ANEXO I: ODS.
- **ANEXO II: Especificación software:** Contiene todo el desarrollo de la especificación software, ya que en la memoria solo incluiremos los apartados más relevantes.

2 Especificación de Requisitos

2.1 Descripción del problema

En primera instancia, el presidente del club nos informó sobre las características que necesitan para la nueva versión del programa, y que se muestra en la Figura 3.

El presidente

Club Municipal "El Jardín" Peñarroya-Pueblonuevo

Peñarroya-Pueblonuevo 26 febrero 2022

Como ya hemos comentado con anterioridad te especifico a mi poco entender las funciones que estamos interesados que recoja la nueva versión del sistema informático que nos gustaría que elaborases:

- Base de datos para la gestión de socios (nombre, apellidos, DNI, domicilio y número, teléfonos fijo y móvil, numero de socio, estado civil, población y código postal) siguiendo el criterio de la versión actual.
- Poder entrar al programa fácil y a datos de socios (ficha) con el número de socio, DNI o 1º apellido.
- Que se pueda ejecutar desde cualquier ordenador propiedad del club. Y que la ventana se adapte a la pantalla independientemente de la pulgadas del ordenador.
- Facilitar el alta, modificación y baja de los socios, sin muchas complicaciones.
- Que el estado del socio se actualice momentáneamente, en todos los ordenadores donde esté ejecutándose la aplicación
- Los socios activos solo deben aparecer los que hayan abonado la cuota del año actual, teniendo en cuenta que la cuota se puede pagar entre el 1 de enero hasta el 30 de septiembre el año en curso.
- Se debe tener en cuenta que hay algunos socios con categoría especial (honoríficos) y que se encuentran exentos de pago anual de por vida (con carácter retroactivo, es decir, se le perdona las deudas que tengan).
- Poder sacar listados con la información totalmente actualizada, y que de la opción de configurar la página mediante vista previa (vertical u horizontal, A3 o A4). Que se pueda seleccionar los campos que se muestren en dicho listado.
- En caso de ser necesario, poder imprimir la ficha del club con el formato que ya establecido (en A5).
- Poder imprimir carné con tamaño DNI y con formato igual al actual carné, en caso de que tengamos instalada la impresora adecuada.
- Poder enviar vía WhatsApp información de interés a socios y controlar quien contesta a la información.
- Indicador para saber cuáles socios quedan pendientes de firmar la autorización de protección de datos. Dar opción de imprimir un formulario estándar con los datos que se tratarán y para qué son necesarios.
- Ya que se trata de una asociación anexa al Ayuntamiento, debemos contar con la opción de transferir información al ayuntamiento, situado en otro emplazamiento de la localidad.
- Detectar posibles errores en datos personales del socio (teléfono erróneo, código postal que no exista, etc.).
- Sacar un estado de cuentas a través de la aplicación, para poder imprimirlo mensualmente.
- Poder sustituir las fotos existentes de los socios usando la webcam del pc o móviles personales, y que automáticamente se redimensione.
- Incluir un apartado con los estatutos del club.
- Sincronizar la aplicación con todas las televisiones Smart tv del club para poder emitir por ellas videos de YouTube, Bingo o cualquier otra.

Figura 3: Descripción del problema, carta de la directiva del club.

A partir de la documentación anterior, nos pusimos en contacto con ellos varias veces y recopilamos la siguiente información:

El club (que aunque tiene identidad propia, forma parte del ayuntamiento de la localidad), está constituida como organización sin ánimo de lucro, y por tanto, tiene un CIF, una razón social, unos estatutos, un domicilio fiscal, una fecha de constitución, una junta directiva y socios. En ocasiones, y para el envío de documentación, se suele usar un logotipo de la asociación y un sello con el que se firman la correspondencia postal con la que actualmente se ponen en contacto con los socios.

La visualización de la nueva versión de la aplicación debe poder adaptarse a la ventana del dispositivo, independientemente al tamaño o tipo de dispositivo donde se muestre. También debe perder redimensionarse el tamaño de letra a voluntad del usuario.

El club debe cumplir lo descrito en la Ley Orgánica 1/2002, de 22 de marzo, reguladora del Derecho de Asociación.

Para ser socio del club es necesario estar jubilado de acuerdo a la edad estipulada por el gobierno (según la fecha de nacimiento), aunque en ocasiones especiales se puede pertenecer estando en la condición de prejubilado (debido a las condiciones socio culturales de la comarca), los datos necesarios que todo socio debe proporcionar son el nombre y los apellidos completos, el DNI (NIE o cualquier documento nacional de identificación español), la fecha de nacimiento, un domicilio donde recibir la correspondencia, datos de contacto (teléfono fijo y/o móvil), información sobre intolerancias y alergias alimenticias (para la organización de eventos culinarios y/o viajes) y al menos una persona de contacto. También tiene asociado una foto tamaño carné para su identificación, esta será opcional.

Todos los socios están identificados con el número de socio, que es personal y que no debe reutilizarse (incremental y con longitud igual a 5). Un socio solo pierde su número de socio cuando, por razones de impago, se le da de baja, y por ello pierde la antigüedad en el club. En caso de que con el tiempo quieran volver a ser socio, se le asignará un nuevo id y perderá la antigüedad y todos los derechos asociados a ella. Solo en casos puntuales se debe permitir la reutilización del id (y la NO perdida de antigüedad asociada).

Las personas de contacto pueden ser o no socios, aunque se recomienda que, además de un socio (como puede ser la pareja), también faciliten los datos de algún otro familiar directo, para poder contactar con ellos en caso de incidencias durante algún viaje o evento. Las personas de contacto que no sean socios deberán facilitar el nombre y apellidos completos, y algún teléfono (fijo y/o móvil) de contacto, así como el parentesco que tienen con el socio. En caso de que efectivamente, si pertenezcan a la asociación, deberán facilitar el número de socio de dicha persona de contacto, para poder enlazar esta información, y el parentesco.

También se debe asignar una cuota a cada socio, así como poder visualizar si están corrientes del pago o la fecha de cuál fue la última cuota que se pagó.

Hay en algunos casos en los que los socios tendrán un tratamiento especial y serán catalogados como “honoríficos”. Esta designación lleva asociada una series de ventajas, como puede ser la exención de cuotas (o cuotas bonificadas reducidas) con carácter retroactivo.

También hay que tener un control de cuales socios están al corriente en la aceptación de la ley de protección de datos.

Para formar parte de la directiva de la asociación, es indispensable ser socio del club y estar al corriente del pago de las cuotas. La directiva está formada por un mínimo de 3 cargos (presidente, tesorero y secretario, que pueden recaer sobre la misma persona) y un máximo de 6 cargos (a los anteriores hay que añadir 3 vocales). También hay que tener en cuenta que se pueden tener hasta un máximo de 3 cargos más como reservas, que pueden formar parte de cualquier cargo anterior, en caso de dimisión de cualquiera de los integrantes.

Para la gestión económica, la asociación quiere poder registrar tanto los pagos que se le hace a los proveedores/acreedores como los ingresos provenientes de cuotas, eventos, rifas y demás actividades.

Los gastos deben registrar la fecha, el número de factura, el CIF y razón social, la cantidad y el precio total.

Se considerará como ingreso, tanto las cuotas de los socios (con la fecha y el número de orden asociado) como las eventos que se hacen (rifas, subvenciones, lotería, etc.). Deberán contar con campos como la fecha, el concepto, la cantidad y el montante total recaudado.

A la aplicación deberá tener acceso distintos tipos de usuario:

- Los socios de la directiva activa, que tendrán acceso a toda la funcionalidad de la aplicación, mientras que su cargo esté activo.
- El administrador del ayuntamiento, que tendrá acceso a las características que nos reclame, a priori, estas serán el acceso a la dirección de los socios activos, y el estado de cuentas.
- Socios, con acceso restringido para que puedan modificar solo sus datos personales, puedan visualizar el estado de los eventos a los que están apuntados (viajes, pago que tiene hecho, cuanto le queda, fecha de la realización, etc.) y los eventos futuros que se están preparando.
- Familiares de socios, restringido a poder modificar los datos de dicho familiar, y si les interesa recibir información del estado de su pariente durante determinados eventos.

También nos gustaría poder visualizar (y configurar para la impresión) distintos tipos de listados (por direcciones, por socios en activo, que se pueda filtrar por eventos comunes, etc.).

Además de imprimir listados, también nos gustaría poder visualizar (e imprimir) la ficha estándar de socios (tal y como se hace ahora) y tener una vista del carné de socio, con la opción de poder imprimirlo cuando se compre una impresora compatible para esa función.

Para la correspondencia por carta ordinaria, nos gustaría poder imprimir las etiquetas de estas, sin direcciones repetidas, por ejemplo, para socios, que comparten vivienda aparecen dos veces la misma dirección y, por tanto, enviaríamos la misma carta, siendo un desperdicio económico y ecológico (imprimir en papel).

A la hora de dar de alta un socio, nos gustaría que se pudiese rellenar automáticamente los campos más comunes, por ejemplo, los de población y provincia cuando se selecciona el código postal. La calle si ya ha sido insertada con anterioridad, etc.

De cara a las elecciones de cambio de directiva nos gustaría hacer un formulario donde aparezcan los grupos de personas que se presentan, junto a los nombres, fotos (si existen en la base de datos), si están al corriente del pago de la cuota y el cargo que ocupará en la directiva.

Otra funcionalidad que echamos muy en falta es la de poder hacer modificaciones masivas de los socios, por ejemplo, la de poder dar de baja a todos los socios que no han pagado en un periodo de tiempo.

O buscar a los socios haciendo uso de varios campos, como, por ejemplo, el número de socio, el apellido, el nombre o el DNI. Actualmente solo se puede hacer por N.º de socio.

También sería interesante tener integrar en el programa la cuenta de correo electrónico, a fin de facilitar el acceso a él. Bastará con un visor de correos, tanto leídos como no leídos, y que se pueda responder desde el mismo programa. Actualmente la cuenta de correo es de Gmail, pero nos gustaría que estuviese abierta a otros servidores como Outlook.

Para los socios que así lo deseen (y así esté configurado en su perfil de acuerdo con el documento de protección de datos), nos gustaría poder enviarles noticias por WhatsApp o Telegram sobre los eventos que se van a realizar próximamente en el club. Estos eventos van desde actividades en el centro (como los días de bingo, concursos esporádicos, etc.) hasta los próximos viajes que la asociación tiene planificados.

Como última instancia, y dada la edad de los socios, queremos implementar un servicio para los familiares que así lo deseen, para poder consultar los datos de su familiar durante los eventos que así lo selecciones, haciendo uso del GPS de su dispositivo móvil, o mediante mensajería de la directiva en el evento en concreto. Para ello, ambas partes deberán tener firmado un consentimiento de rastreo para evitar problemas legales.

2.2 Análisis del problema

En el mayor nivel de abstracción, y teniendo en cuenta las necesidades descritas en el punto anterior, es primordial actualizar la aplicación actual (creada para ejecutarse mediante *VBA* macros), por una aplicación distribuida, con acceso desde varios equipos a la vez y de distinta índole (portátiles, móviles o Tablet). También debe ser accesible desde distintas redes (la local de la asociación, desde el ayuntamiento o desde cualquier dispositivo móvil con conexión a internet)

Debe tener capacidad *CRUD* para modificar la información en tiempo real. Y deberá soportar el uso con distintos tipos de usuarios y permisos (roles).

El diseño debe ser *responsive*, con iconografía adaptada a personas mayores con poco o nulo conocimiento informático.

2.2.1 Comparativa entre aplicaciones

En primera instancia, vamos a comparar las funcionalidades más relevantes y los precios de las aplicaciones mostradas en el apartado 1.3 Estado del arte / Estudio estratégico, junto a la aplicación actual desarrollada en VBA y las funcionalidades que pretendemos implementar en nuestro proyecto.

Funcionalidad	Berrly	Playoff	Cucunver	Taclia	Actual VBA	TFG
Coste versión básica	35€/mes	45€/mes	20€/mes	Gratis	Gratis	Gratis
Cantidad máxima de usuarios administradores	5	4	3	∞	∞	∞
Cantidad máxima de socios registrados	∞	150	200	∞	∞	∞
Envío de comunicados	✓	✓	✓	✓	✗	✓
Informes y análisis de datos	✓	✓	✓	✓	✓	✓
Seguridad y cifrado	✓	✓	✓	✓	✗	✓
Carné para socio con código QR	✓	✓	✗	✗	✗	✓
Confirmación de asistencia a eventos	✓	✓	✓	✗	✗	✓
Sistema de cobro de cuotas integrado	✓	✓	✓	✗	No necesario	No necesario
Espacio de almacenamiento en la nube	1GB 500 archivos	✗	✗	✗	No necesario	No necesario
App móvil	iOS / Android 40€/mes	iOS / Android 35€/mes	✗	Android Gratis	✗	Web Responsive
Interfaz adaptada tercera edad	✗	✗	✗	✗	✗	✓
Seguimiento especializado de socios por familiares	✗	✗	✗	✗	✗	✓
Acceso a socios para actualizar su información	✓	✓	✓	✗	✗	✓
Integración con cualquier gestor de correo electrónico	✗	✗	✗	✗	✗	✓

Tabla 1: Comparativa de aplicaciones actuales en el mercado

2.2.2 Priorización de funcionalidades. Método MoSCoW

Para poder priorizar la realización de todas las funcionalidades que debe recoger nuestro aplicativo, haremos uso del método MoSCoW [5], ya que es una técnica que sirve para priorizar requerimientos y tomar decisiones en la gestión de proyectos, el desarrollo de software y aplicaciones, asignando una prioridad a cada uno de los requerimientos.

MUST (Debe tener)	✓ Iniciar / Cerrar sesión. ✓ *CRUD* socio. ✓ *CRUD* evento. ✓ *CRUD* gastos. ✓ *CRUD* ingresos. ✓ *CRUD* asociación. ✓ Gestor de integrantes de la directiva. ✓ Gestor de permisos a los integrantes y socios. ✓ Diseño *responsive*.
SHOULD (Debería tener)	✓ *CRUD* personal externo autorizado a consultas (para personal del consistorio). ✓ *CRUD* proveedores/acreedores. ✓ Seguimiento de los contratiempos de los eventos desde el perfil de la organización. ✓ Configuración e impresión de listados personalizados, ajustados a distintos criterios. ✓ Impresión de ficha de socio. ✓ Vista previa del carnet de socio en la ficha de dicho socio. ✓ Gestor de estatutos del club. ✓ Mensajería por *WhatsApp*, *Telegram*, *email*. ✓ Gestor de socios que están al corriente de haber firmado la autorización de tratamiento de sus datos según la ley de protección de datos.
COULD (Podría tener)	?? Modificación masiva de socios según criterio preestablecido. ?? Integración con el perfil de los socios, para usar el *GPS* en tiempo real (y no solo con los miembros de la directiva) para hacer el seguimiento de los eventos directamente a los familiares. ?? Distintos modos de mostrar la información en el programa. ?? Integración con el mail de la asociación, para centralizarlo todo en la misma aplicación. ?? Poder deshacer/rehacer mientras que se actualizan los campos. ?? Crear hojas de candidaturas con los datos de los socios que se presentan a la directiva.
WON'T (No tendrá)	× Alojamiento de documentación por parte de los socios (tipo nube). × Impresión del carnet de socio mediante impresora (es necesario tener en cuenta los drivers de dicha impresora). × Sincronización de la aplicación con las *SmartTV* del local, para visionar videos de *YouTube.*

Tabla 2: Prioridades MoSCoW

2.3 Modelado Conceptual

2.3.1 Fuente de requisitos

2.3.1.1 Stakeholders relevantes

- Socios en activo.
- Familiares de los socios.
- Miembros de la directiva.
- Ayuntamiento de la localidad.

2.3.1.2 Documentos propios de la asociación

- Estatutos de consolidación de la Asociación: en ellos están contenidas todas las características de dicha sociedad, y nos servirá de base para establecer la estructura básica del aplicativo.

2.3.1.3 Reglamentos y legislación

- Ley Reguladora del Derecho de Asociación [6]: Al igual que los estatutos, nos servirá para establecer la estructura básica del aplicativo.
- Ley de Protección de Datos [7]: La aplicación consiste en el tratamiento automatizado de datos personales de los socios, y por tanto está sometida a esta normativa.

2.3.2 Perfil de *stakeholders*

Nombre / Rol	Usuario Directo	Interés
Socio	Si	Mejora la calidad de la información y de los servicios que la asociación ofrece.
Familiares	Si	Implementa información en tiempo real de los familiares que autoricen la monitorización, así pueden saber si ha ocurrido algo durante un evento.
Directiva	Si	Mayor control sobre la información de la que se dispone. Mejora la organización de los eventos, mejora la transparencia del estado de cuentas ante socios y otros organismos públicos de los cuales depende.
Ayuntamiento	Si	Reduce los desplazamientos hacia la sede de la asociación, mejora la consulta del estado de cuentas del estado de los socios.

Tabla 3: Perfil de stakeholders

2.3.3 Diagrama de contexto

Con el diagrama de contexto podemos identificar los límites de nuestro sistema software pretendemos resaltar los factores externos para tener en cuenta en la aplicación.

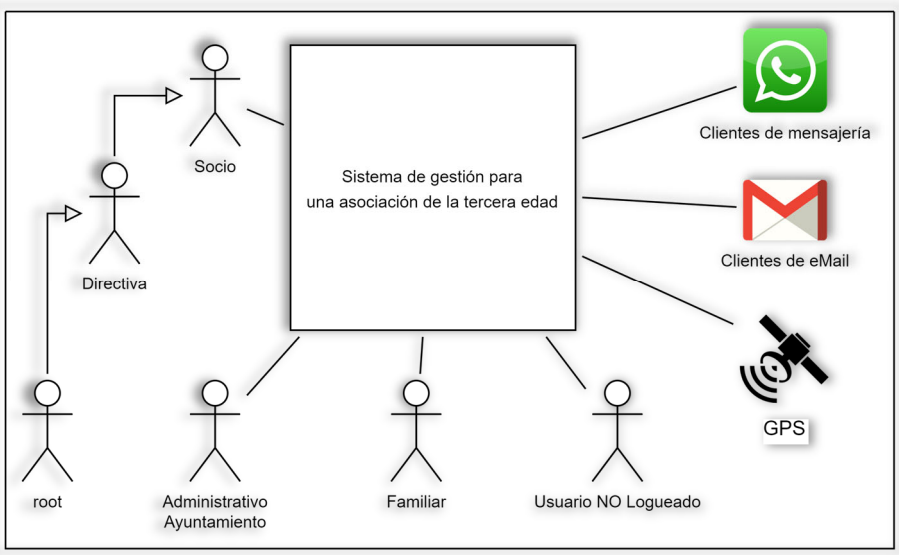


Figura 4: Diagrama de contexto.

Revisar el documento *ANEXO II: Especificación software* apartado *1 Glosario* donde se encuentran todos los términos técnicos que aparecen en la Figura 4.

2.3.4 Modelo de dominio

En este apartado representaremos las clases conceptuales del mundo real, así podremos explicar los conceptos clave y el vocabulario del dominio del problema.

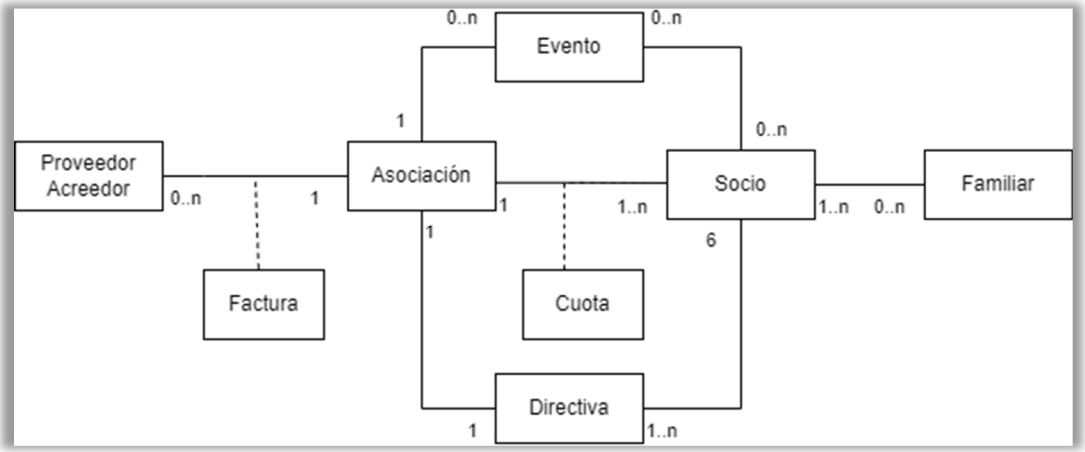


Figura 5: Modelo de dominio.

En el *ANEXO II: Especificación software* apartado *1 Glosario* se encuentran definidos todos los términos indicados en la Figura 5.

2.3.5 Requisitos funcionales

Con los requisitos funcionales [8] queremos describir explícitamente el comportamiento que debe tener la solución de software que vamos a desarrollar y la información que debe manejar.

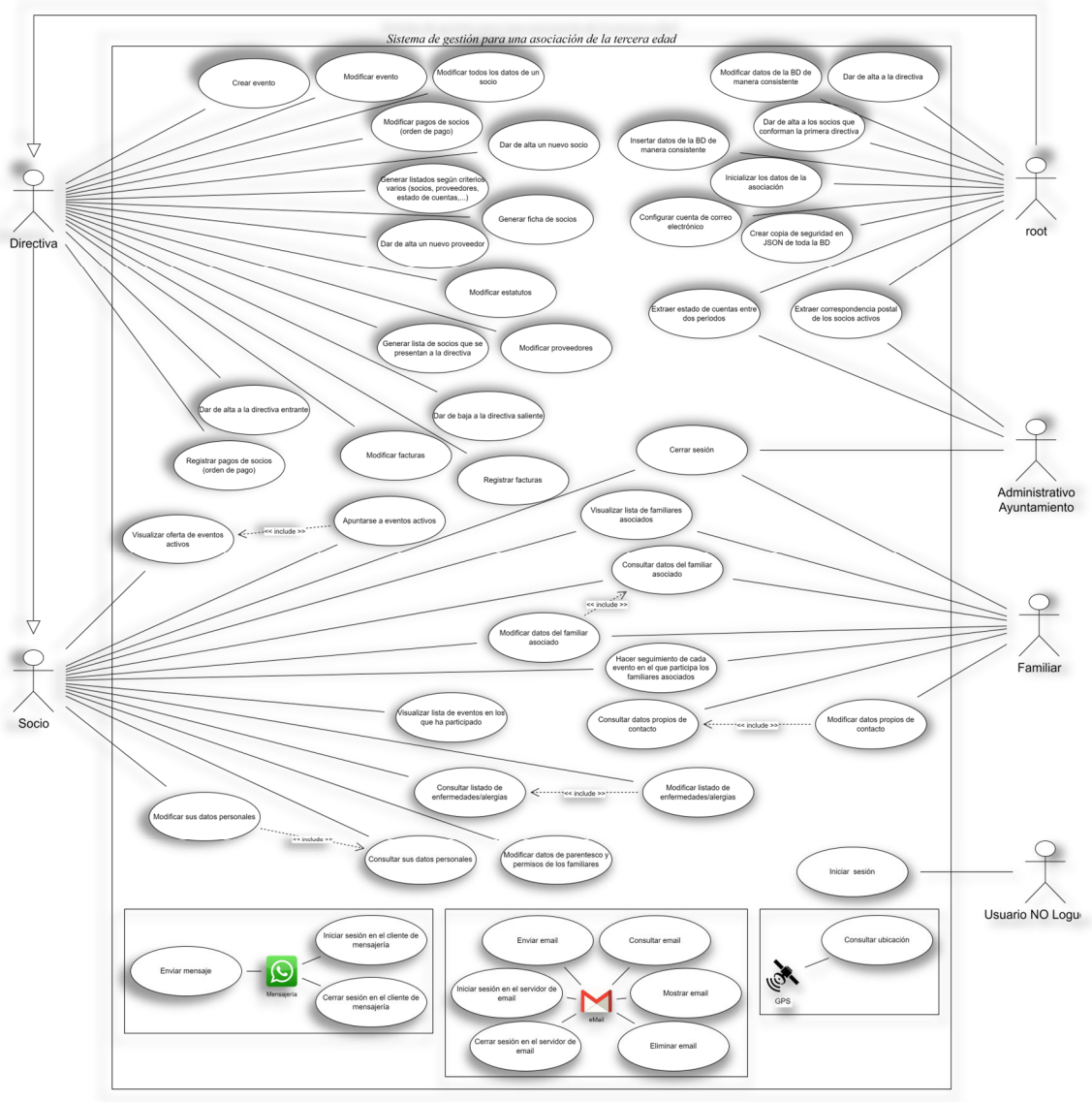


Figura 6: Diagrama de casos de uso.

En el caso que nos concierne, se contemplan los siguientes:

- Solo se puede usar la aplicación si se está registrado.
- Solo te puede registrar un usuario *root* o un miembro de una directiva activa.
- Para los usuarios que no están registrados solo está disponible la página principal de *logueo*, que tendrá tres opciones, *logueo* como socio y/o familiar, como directiva o como tercero (este último incluye el usuario *root* y el administrativo del ayuntamiento).
    - *Logueo como socio, número de socio y pin de 4 dígitos.*
    - *Logueo como directiva, número de DNI y pin de 6 dígitos.*
    - *Logueo como usuario con privilegios, número de DNI y contraseña de 8 dígitos, una letra minúscula, una letra mayúscula y un carácter especial.*

- Los socios solo pueden modificar su información personal.
- Los socios se pueden apuntar a cualquier cantidad de eventos, siempre que cumplan con el requisito de estar al corriente del pago de la cuota.
- Los socios pueden ver cualquier evento que esté publicado.
- La directiva puede insertar/modificar eventos.
- La directiva puede insertar/modificar socios.
- La directiva puede insertar/modificar nuevas directivas.
- Solo puede haber como máximo dos directivas activas, en caso de elecciones, una saliente y una entrante.
- La directiva puede insertar/modificar acreedores/proveedores.
- Al dar de alta un nuevo evento o modificarlo, se enviará un mensaje (por el canal de mensajería elegido por el socio y si la directiva así lo decide) con la información relativa a dicho evento.

Se amplía información sobre los requisitos funcionales, en el apartado *"ANEXO II: Especificación software"* en este mismo documento.

2.3.6 Requisitos no funcionales

Los requisitos no funcionales [9] nos aportan las características generales y restricciones del sistema (o aplicación aplicación) que nos disponemos a desarrollar.

Para esta aplicación tendremos en cuenta las siguientes características:

- Seguir las directrices de los estatutos de consolidación de la Asociación.
- Seguir las directrices de la Ley Orgánica 1/2002, de 22 de marzo, reguladora del Derecho de Asociación [6].
- Seguir las directrices de la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos.
- El usuario no puede experimentar más de dos fallas por mes en la aplicación.
- La aplicación web debe estar disponible 24/7.
- La aplicación de recuperar la información del usuario y mostrarla en menos de 3 segundos.
- La aplicación web debe funcionar al menos en *Firefox*, *Chrome*, *Safari* y *IE*.
- La aplicación web debe funcionar en PC, tabletas y dispositivos móviles (*Android* e *iOS*).
- El formato de intercambio de datos con la aplicación del cliente usará el estándar *JSON*.

2.4 Tecnologías por utilizar

2.4.1 Sitios para el alojamiento

Con las tecnologías que hemos elegido, se puede apreciar el modelo en tres capas que hemos elegido para el desarrollo de nuestra aplicación, ya que desde un principio se están separando *frontend*, *backend* y persistencia de datos.

2.4.1.1 Frontend

Vamos a usar los servicios de *Vercel* [10], ya que permite enlazar los repositorios alojados en *github*, y por tanto, al hacer un *commit* desde nuestro entorno de trabajo, automáticamente se actualizará en sus servidores, agilizando la resolución de bugs.

Otra ventaja es que cuenta con una alternativa gratuita con los servicios necesarios para que nuestra aplicación funcione.

Como desventaja contamos con que no cuenta con integración de bases de datos relacionales, por lo que la persistencia de nuestra aplicación deberemos hacerla con otros servicios

2.4.1.2 Backend

Para montar el *backend* vamos a contar con los servicios de *railway* [11], que entre sus ventajas se encuentra sus 1024mb de alojamiento gratuito para nuestro proyecto, a compartir con la base de datos y cuenta con tecnologías como *MySQL* o *PostgreSQL*, integradas.

Railway también permite enlazar nuestro repositorio de *github*, por tanto se consigue una estandarización en el proceso de programación. Ya que tanto *frontend* como *backend* se gestiona de la misma manera, con un repositorio en *github* y con los servidores enlazados a la rama *main* del proyecto.

2.4.1.3 Persistencia de datos

Para la persistencia de los datos (a partir de ahora BD) también usaremos servicios de *railway* [11], ya que como hemos comentado en el punto anterior, permite la integración dentro del propio proyecto, sin *addons* de terceros.

Aunque para optimizar el uso de la BD, hemos considerado las siguientes consideraciones:

1. No almacenar ningún archivo (imágenes o *pdf's*) en formato *BLOB*, como alternativa, estos se alojarán en la cuenta de *Dropbox* de la asociación, y en BD solo se almacenará la *url*.
2. Los datos con información estática, que no varían a lo largo del tiempo (códigos postales de poblaciones, códigos de provincia, etc.) se guardarán como archivos *json* en el propio proyecto *backend*.

2.4.2 Elección de la tecnología de desarrollo

Nos hemos decidido en desarrollar una aplicación web, ya que permite escalabilidad y disponibilidad. De esta manera se permite la consulta, inserción y modificación de datos de forma concurrente, desde cualquier dispositivo, adaptando la interfaz con diseño *responsive* al navegador de cualquier tipo de dispositivo.

Debido a que la aplicación se trata de una arquitectura cliente-servidor. Hemos tomado la decisión de diseñar la aplicación en tres capas:

- ***Capa de datos:*** donde usaremos un base de datos *MySQL*.
- **Capa de negocio:** para backend usaremos NodeJS con Typescript.
- **Capa de presentación:** para frontend usaremos nextJS, React y *TailwindCss*.

Con esta metodología se consigue un mayor desacoplamiento informático, permitiendo una mejor escalabilidad.

2.4.3 Herramientas

2.4.3.1 Visual Studio Code (VS Code)

Visual Studio Code [12] es un editor de texto desarrollado por *Microsoft*. Es software libre y está disponible para *Windows*, *GNU/Linux* y *macOS*.

VS Code integra a *Git*, dispone de un sinfín de extensiones para ampliar su funcionalidad y cuenta con soporte para depuración de código. Siendo imprescindible para editar cualquier lenguaje de programación.

2.4.3.2 GitHub

Github [13] es una herramienta que permite tener nuestros repositorios de Git en la nube. Permitiendo poder colaborar con otros compañeros o socios de otras organizaciones.

2.4.3.3 GitHub Desktop

GitHub Desktop [14] consiste en un cliente de escritorio para facilitar el uso de *Github* en nuestra máquina local.

2.4.3.4 Microsoft Word

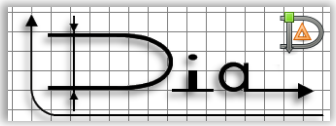
Word [15] es un programa de procesamiento de textos, diseñado para a crear documentos de calidad profesional. Ayuda a organizar y escribir documentos de forma más eficaz.

2.4.3.5 Microsoft Power-Point

Power Point [16] es un programa para hacer presentaciones de diapositivas, animaciones de texto e imágenes prediseñadas.

2.4.3.6 diaw.exe

Diaw.exe [17] permite diagramas estructurados. Alternativa con Licencia Pública General de GNU a aplicaciones como *LucidChart* y *Draw.io*.



Nos ha permitido realizar todos los diagramas relacionados al modelaje *UML*.

2.4.3.7 Draw.io

Draw.io [18] al igual que la anterior, es una herramienta gratuita que nos ayuda en el desarrollo de distintos tipos de diagrama.

2.4.3.8 Trello

Trello [19] es una herramienta que nos ayuda en la gestión del trabajo. Esta ideada para trabajar en equipo, organizar flujos de trabajo, colaborar en proyectos y hacer un seguimiento del progreso de una manera visual.

2.4.3.9 MySQL Workbench

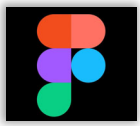
MySQL Workbench [20] consiste en un entorno gráfico de diseño de bases de datos, servidores, administración y mantenimiento para el sistema *MySQL*. Permite modelar, gestionar y generar bases de datos de manera visual o gráfica.

2.4.3.10 PostMan

Postman [21] agiliza la colaboración para que pueda crear mejores *API's*, más rápido.

2.4.3.11 Figma

Figma [22] consiste en un editor de gráficos vectorial y herramienta de prototipado multiplataforma



3 Proceso Software

Las metodologías ágiles son aquellas que permiten dividir el trabajo a realizar en partes que se adaptan sobre la evolución del trabajo y admiten la resolución de necesidades en poco tiempo. Por esta razón nos hemos decantado por este tipo de metodología.

Dentro de este tipo de técnicas, nos hemos decantado por el método *Kanban* [23], aunque no es una técnica de planificación sino de organización del trabajo. Aunque, debido a la envergadura del proyecto, también le hemos añadido matices *Scrum* para así poder despiezar el proyecto en varios sprint para que la asociación pueda empezar a usar la aplicación lo antes posible y poco a poco añadirle el resto de las funcionalidades, priorizando según las necesidades que nos indiquen.

La herramienta que se ha usado para organizar el trabajo ha sido Trello, ya que facilita mucho esta labor y además es gratuita.

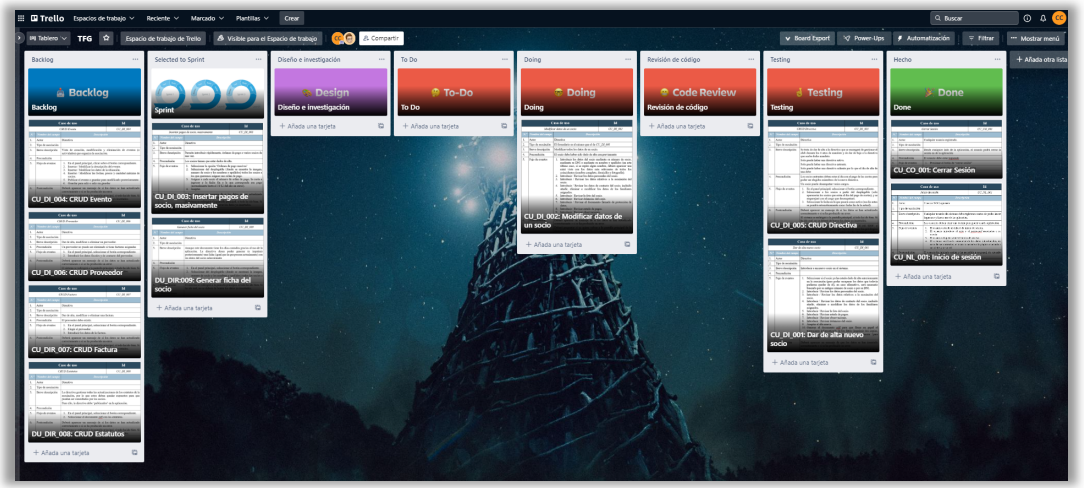


Figura 7: Tablero Kanban, en Trello

En nuestro caso en particular, hemos dividido el tablero *Kanban* en varias columnas que representan el *workflow* por el que irá transitando cada caso de uso (de aquí en adelante CU) que hemos diseñado como si de unidades de trabajo (a partir de ahora UT) se tratara.

Cada columna recoge los siguientes CU:

- ***Backlog:*** Recoge todos los CU que se van diseñando y especificando.
- ***Sprint:*** Aquí promocionan los CU que han sido seleccionados para ser desarrollados en el sprint.
- ***Design:*** CU que están en proceso de diseño o creación de mockup's.
- ***To Do:*** CU que estén pendientes de empezar a programar.
- ***Doing:*** CU que ya están programándose.
- ***Code Review:*** CU que necesitan una revisión de código. Pueden venir de las columnas '*Doing*' o '*Testing*'. Estos CU tienen prioridad frente a otros para evitar cuellos de botella.
- ***Testing:*** Aquí se colocan los CU después de la revisión de código. Se tiene en cuenta tanto los test diseñados como el diseño inicial.
- ***Done:*** Aquí se encuentran todos los CU que ya se han terminado completamente.

4 Desarrollo/Implementación

4.1 Arquitectura Software y Diseño de la solución

Como bien nos resumen en *DotNet2021* [24], hay cuatro tipos de arquitectura software más comúnmente usadas, y que se muestran en la Figura 8.

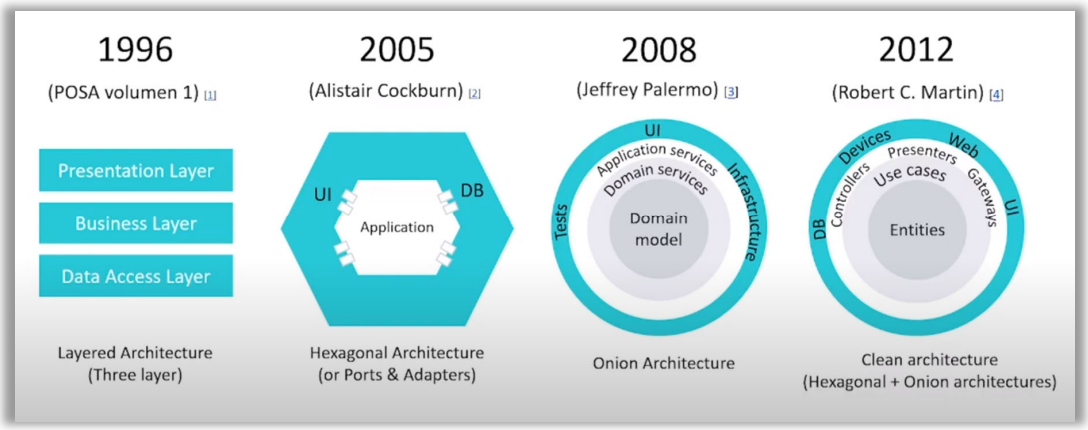


Figura 8: Tipos de arquitectura software

- 3-Capas: Divide la aplicación en tres partes, capa de presentación (lo que el usuario ve), capa de negocio (tratamiento de los datos) y capa de persistencia (almacenamiento de los datos).
- Hexagonal: Divide la aplicación en un núcleo central que se conecta con la persistencia y la presentación a través de puertos definidos en la propia aplicación. Hay una inversión de prioridad que se resuelve mediante el uso de interfaces
- *Onion*: Arquitectura que evoluciona de la hexagonal, y al igual que en ella la UI y la infraestructura se debe adaptar a la aplicación, y no al revés.
- *Clean*: Revisión que recopila todas las características de sus antecesoras y ajusta el núcleo de la arquitectura.

4.1.1 Arquitectura

Para las aplicaciones web de pequeño tamaño suelen implementarse una arquitectura cliente/servidor, pero este sistema, con el pasar del tiempo, ha demostrado que tiene una evolución poco escalable, ya que la lógica de la aplicación suele acabar mezclada con los detalles de la interfaz de usuario.

Por tanto hemos decidido crear la aplicación en tres niveles, ya que de esta manera evitaremos el problema descrito, además de mejorar la escalabilidad notablemente. Por el contrario, hay más trabajo que hacer al tener que desarrollar una capa independiente más [25].

4.1.1.1 Frontend o capa de presentación

Se trata de la parte visible de la aplicación, ya que es la interacciona con el usuario. Como ya hemos dicho en ocasiones anteriores, tratará de una interfaz web, por lo que hará falta un navegador para poder acceder a la aplicación.

En este apartado, hay que tener en cuenta que se trata de una web *responsive*, agradable y usable.

Se desarrollará usando *nextJS*, *React* y *TailwindCss* y estará alojada en *Vercel*.

4.1.1.2 Backend o capa de lógica de aplicación

Es el corazón del sistema, ya que es la responsable de realizar las tareas para las cuales se diseña el sistema.

Usaremos *NodeJS* con *Typescript*, como lenguaje orientado a objetos y estará alojado en *Railway*.

4.1.1.3 Persistencia de datos o capa de acceso a datos

Se encarga del almacenamiento de los datos, donde usaremos un gestor de bases de datos relacionales basado en *MySQL*. En el caso que nos compete, usaremos la integración que nos ofrece *Railway* y *MySQL*.

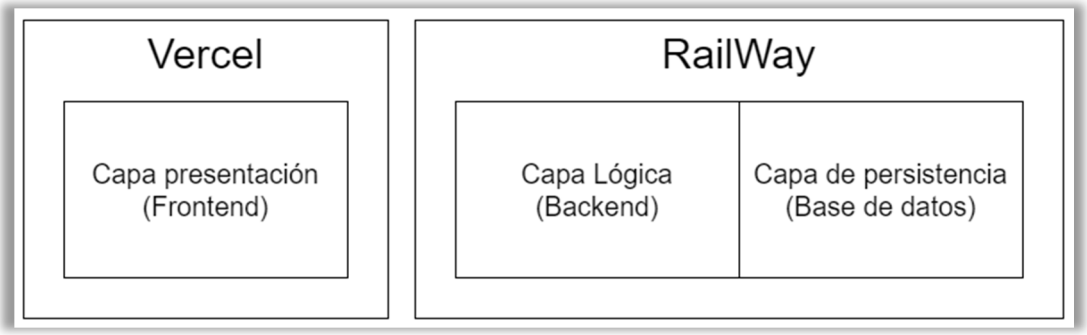


Figura 9: Arquitectura de la aplicación

4.1.2 Diagrama de clases

Un diagrama de clases [26] en Lenguaje Unificado de Modelado (UML) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

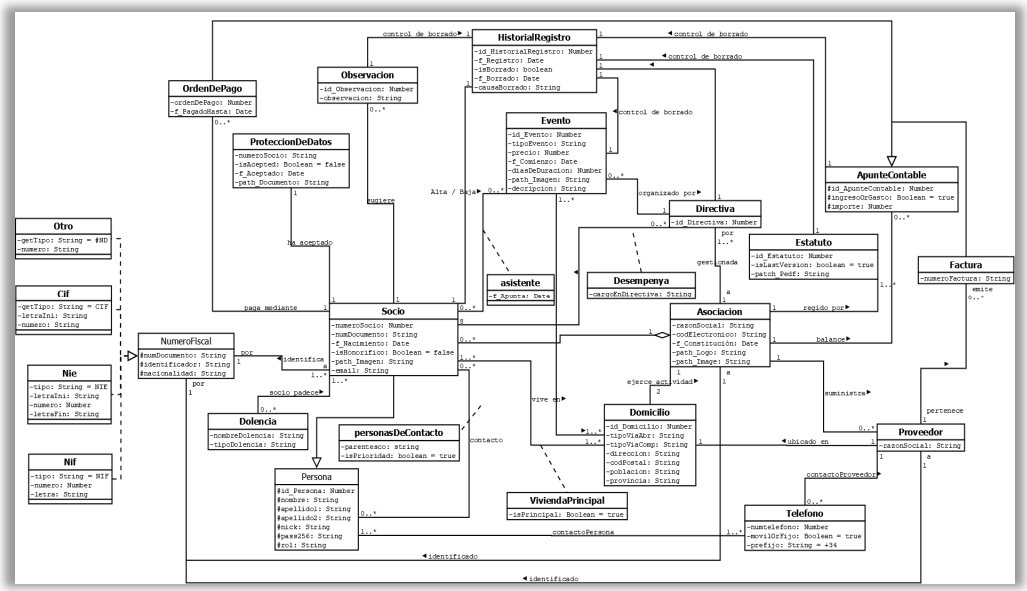


Figura 10: Diagrama de clases.

4.1.3 Diagrama Entidad-Relación

Un modelo entidad-relación [27] es una herramienta para el modelo de datos, la cual facilita la representación de entidades de una base de datos.

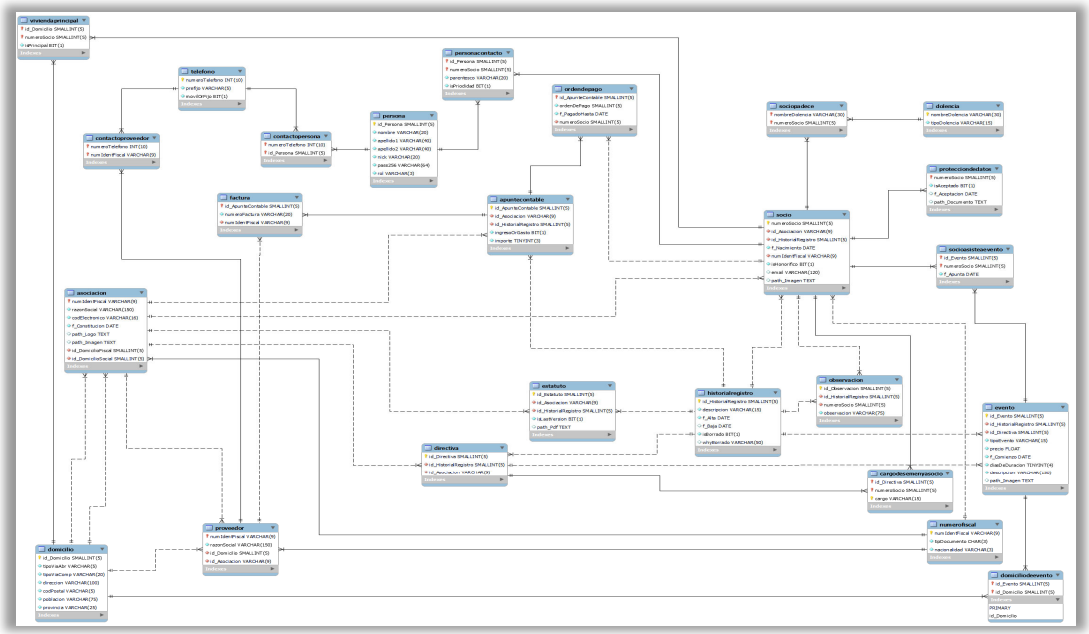


Figura 11: Diagrama Entidad-Relación

4.1.3.1 Descripción de cada tabla

- **NumeroFiscal:** Contiene la información de los números fiscales (NIF, NIE, CIF u OTRO), es decir, el número, así como cuál es el tipo al que corresponde, y la nacionalidad de la persona (aunque en contadas ocasiones, hay algunos socios que son franceses.
  Persona: Contiene los datos de toda persona registrada en la aplicación. Puede corresponder tanto a socios (que si pertenecen a la asociación) como a no socios (que no pertenecen a la asociación). Estos últimos pueden ser los administradores de la aplicación. Aquí se guardan las credenciales de acceso a la aplicación y el rol que desempeña.
- **Socio:** Extiende a la tabla Persona y contiene los datos de los socios. El campo 'numIdentFiscal' permite los duplicados ya que un socio se puede dar de baja y al tiempo se puede volver a dar de alta, pero perdiendo toda la antigüedad.
- **Domicilio:** Contiene los datos relacionados con los domicilios.
- **ViviendaPrincipal:** Relación binaria [Muchos-Muchos] entre [Socio-Domicilio], con atributo de unión [isPrincipal]. Debido a que un Socio puede tener varias residencias, puede indicarlas y decidir en cuál de ellas prefiere recibir la correspondencia.
- **Dolencia:** Regula la cantidad de dolencias. Sirve para tener en cuentas intolerancias alimenticias y alergias.
- **SocioPadece:** Relación binaria [Muchos-Muchos] entre [Socio-Dolencia].
- **ProteccionDeDatos:** Recoge si el socio ha firmado el documento de aceptación de tratamiento de datos.
- **Asociación:** Contiene los datos fiscales de la asociación.
  PersonaContacto: Relación binaria [Muchos-Muchos] entre [Socio-Persona], con atributo de unión [isPriodidad]. Sirve para saber quién es el familiar de quien y si tiene prioridad a la hora de contactarlo. Queda pendiente restringir por software que una persona no su propia persona de contacto.
- **Telefono:** Almacena todos los números de teléfono, indicando si es móvil o fijo (hace esa distinción para poder usar el teléfono con *WhatApp* o *Telegram*). También guarda el prefijo del país.
- **ContactoPersona:** Relación binaria [Muchos-Muchos] entre [Teléfono-Persona].
- **Proveedor:** Tabla que recoge los datos fiscales de los proveedores que tiene la asociación.
- **ContactoProveedor:** Relación binaria [Uno-Muchos] entre [Teléfono-Proveedor].
- **ApunteContable:** Guarda los ingresos o gastos que tiene la asociación. Se desglosan en Factura (los pagos) o OrdenDePago (ingreso).
- **Factura:** Pagos a acreedores/proveedores.
- **OrdenDePago:** Ingresos de cuotas de socios.
- **HistorialRegistro:** Datos que marcan la trazabilidad de cuando se registra algo y cuando marca como borrado. Por ejemplo, para un Socio indica cuando se da de alta y/o baja del sistema; para una Observación indica cuando se hizo, o si se borró porque se trataba de un error…
- **Observación:** Registra las observaciones que ha indicado el socio.
- **Estatuto:** Guarda en *pdf* el histórico de los estatutos de la asociación, es decir, guarda cada actualización.
- **Evento:** Guarda los eventos organizados. La duración se refiere en días, siendo 1 la duración mínima correspondiente a 1 día.

- **SocioAsisteAEvento:** Relación binaria [Muchos-Muchos] entre [Evento-Socio], con atributo de unión [f_Apunta].
- **DomicilioDeEvento:** Relación binaria [Muchos-Muchos] entre [Evento-Domicilio].
- **Directiva:** Datos de la directiva.
- **CargoDesemenyaSocio:** Relación binaria [Muchos-Muchos] entre [Evento-Domicilio], con atributo de unión.

4.2 Desarrollo de la solución

Con el proyecto ya implementado y empezando a desarrollar la aplicación, nos encontramos con que los términos de uso de *Heroku*, nuestro principal candidato para implementar el *backend*, han cambiado, dejando de ofrecer el servicio gratuito que hizo que nos decantáramos por él en primera instancia. Así que tuvimos que buscar una alternativa rápidamente, para poder implementar cambios mayores, si fuese necesario. Finalmente, a nivel de código solo cambió la configuración de las variables de entorno.

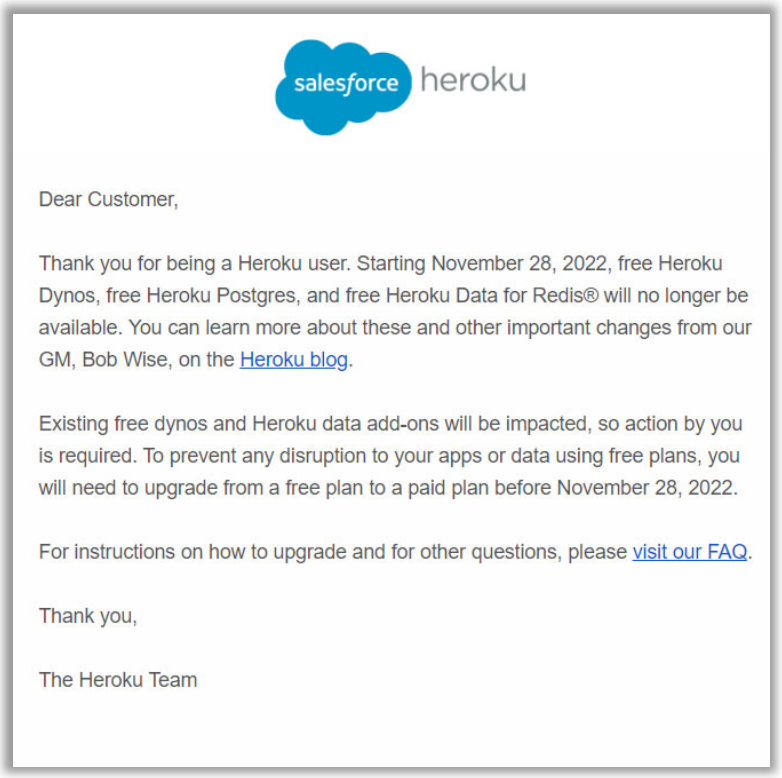


Figura 12: Notificación cambio en los términos de Heroku

En la Figura 13 se muestra la Api de conexión con el *backend*. Basadas en peticiones web mediante métodos GET, POST, PUT y DELETE.

index.js

```
264  //  //  //  //  //
265  //
266  //  API ADDRESS
267  //
268  //  //  //  //  //
269
270  app.get('/api/getAddressByID', (req, res) => {
271      if (utilities.getNumber(req.query.id_address) == -1) {
272          return res.send('El id no tiene un formato correcto')
273      }
274      address.getAddressByID(db,req.query.id_address).then(response => {
275          res.send(response)
276      })
277  })
278
279  app.get('/api/getAllAddressOfActivities', (req, res) => {
280      address.getAllAddressOfActivities(db,).then(response => {
281          res.send(response)
282      })
283  })
284
285  app.post('/api/createNewAddress', (req, res) => {
286      address.createNewAddress(db, req).then(response => {
287          res.send(response)
288      })
289  })
290
291  app.put('/api/updateAddress', (req, res) => {
292      address.updateAddress(db, req).then(response => {
293          res.send(response)
294      })
295  })
```

Figura 13: API de conexión con el Backend

El *backend* se ha organizado en varias carpetas, donde se puede distinguir las consultas *sql* contiene el código para crear la estructura de la base de datos, así como los datos, valga la redundancia, para poder ejecutar los test.

En la carpeta *src* se encuentran todos los módulos de cada uno de los artefactos que conforman la estructura de la aplicación.

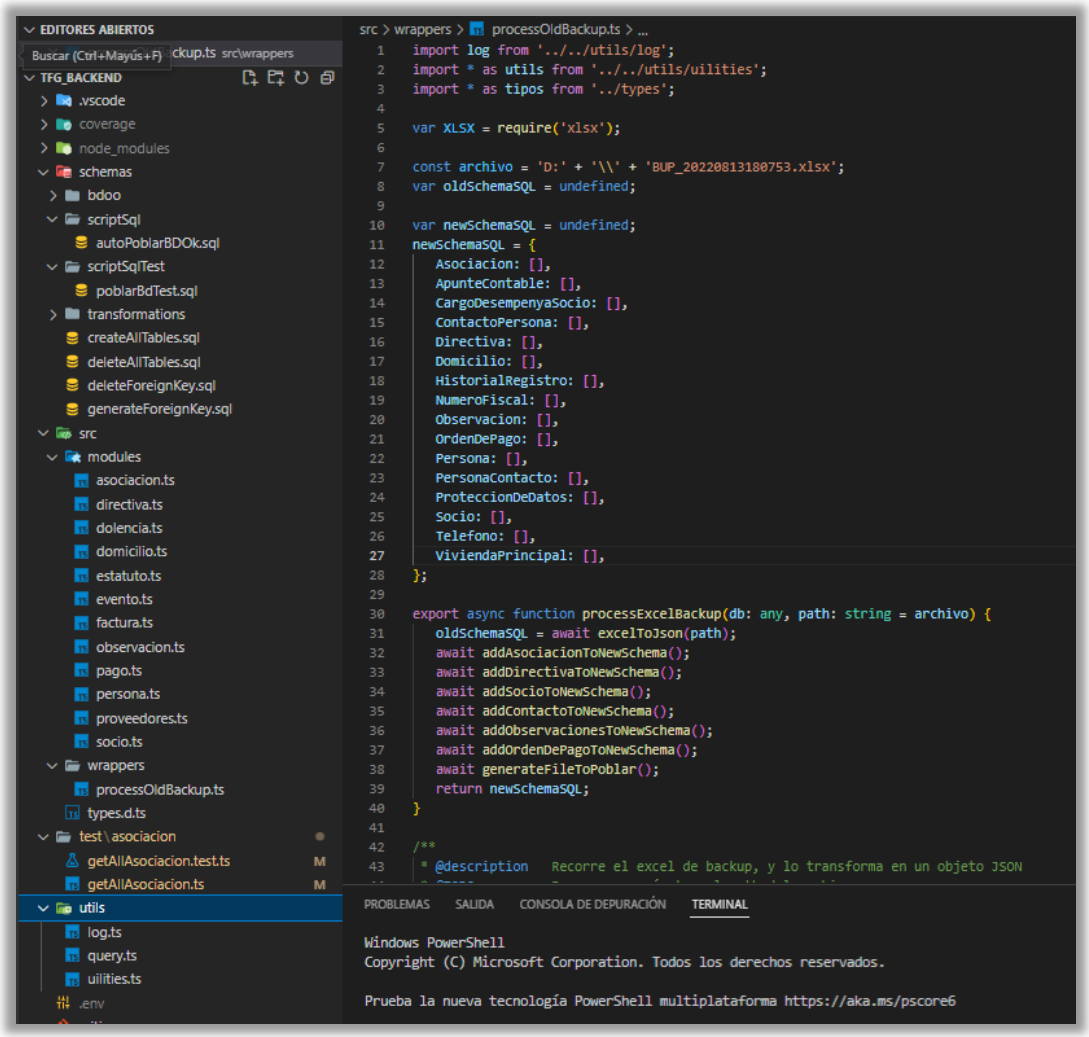


Figura 14: Estructura del backend

En el *frontend*, es donde se configura los estilos CSS, nos serviremos de *tailwingCSS* ya que con él la programación embebida en *html* y los diseño de *figma*, nos permite tener un mayor control en el diseño estético de nuestra web.

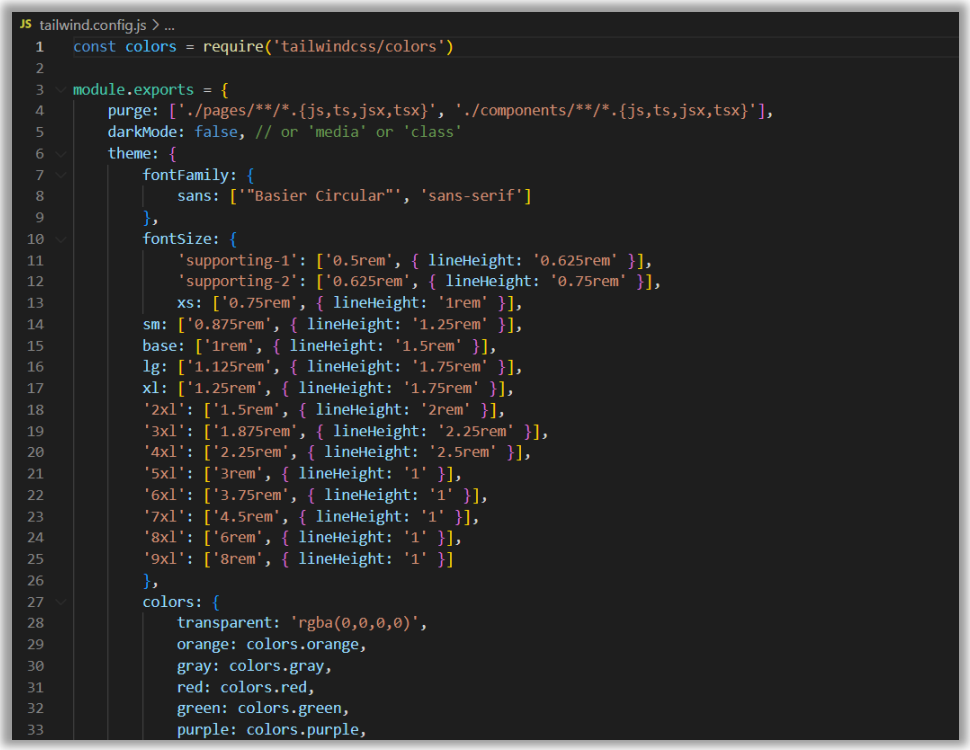


Figura 15: Frontend, archivo de configuración de tailwingCSS

En el *frontend*, hemos organizado todas las vistas en la carpeta *pages*, que contendrá todas las páginas que se vayan desarrollando, así como las funcionalidades de estas.

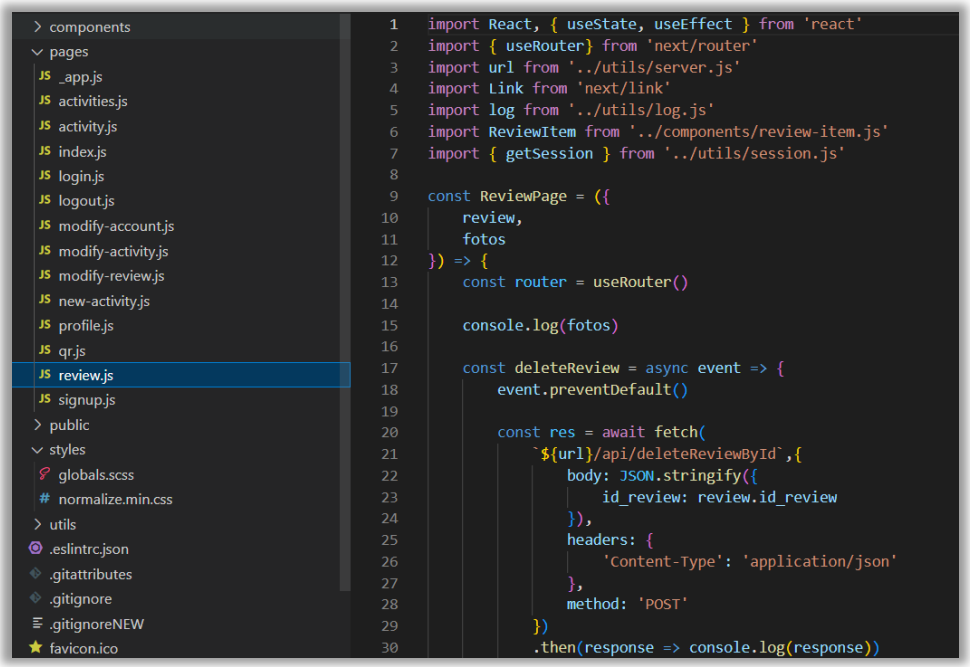


Figura 16: Estructura frontend

4.3 Implementación

4.3.1 Pasos iniciales

Lo primero que hemos tenido que hacer es configurar el entorno de desarrollo con las dependencias iniciales (dependencias que se han ido ampliando para amoldarse a las necesidades y evolución del proyecto). Todo esto, *nodeJS* lo registra en el archivo *package.json*.

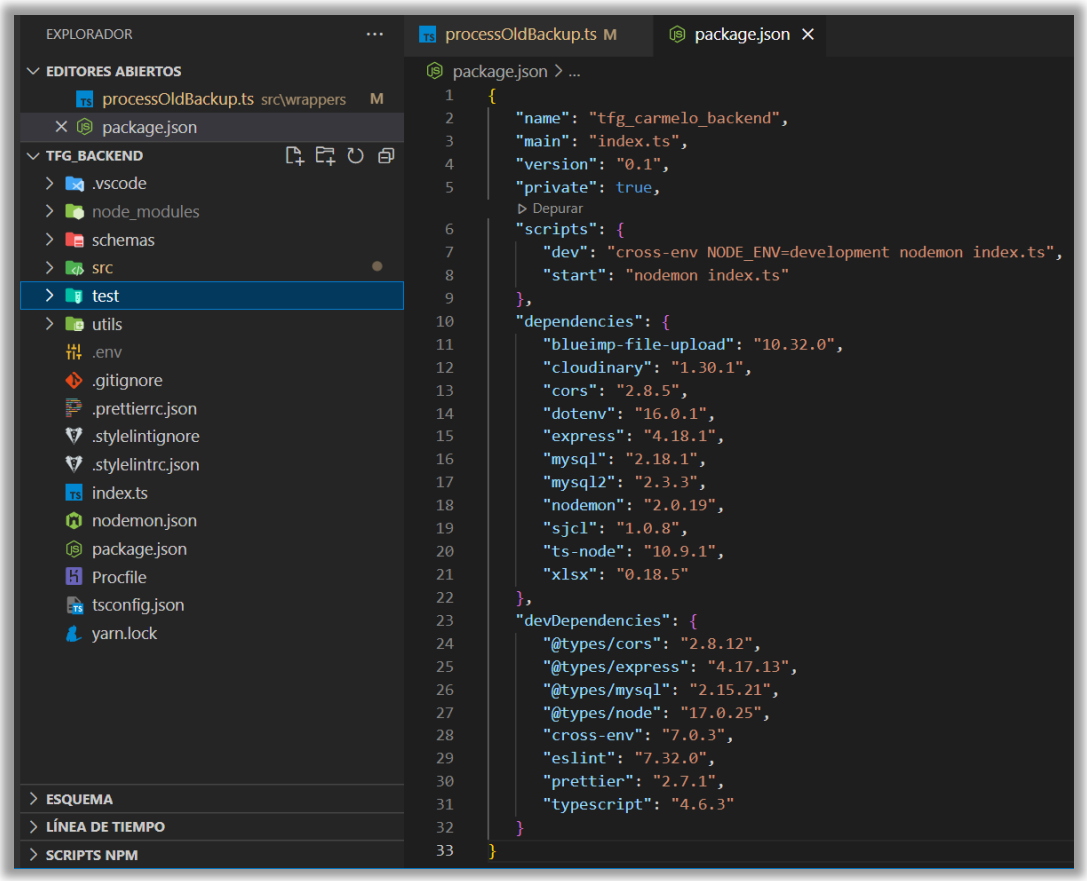


Figura 17: Configuración nodeJS, package.json

En estos momento, también hemos tenido que configurar una cuenta en *vercel* y otra en *railway* para poder poner en funcionamiento tanto el *frontend* como el *backend*, respectivamente.

Estos generan unas variables de entorno necesarias para poder establecer una comunicación segura entre ellos. Estas variables van en la carpeta *'.env'* para no compartirlas en *git* (ya que son privadas, y crearían un agujero de seguridad).

El dominio registrado para el uso de la aplicación es: https://eljardinpyapvo.vercel.app/

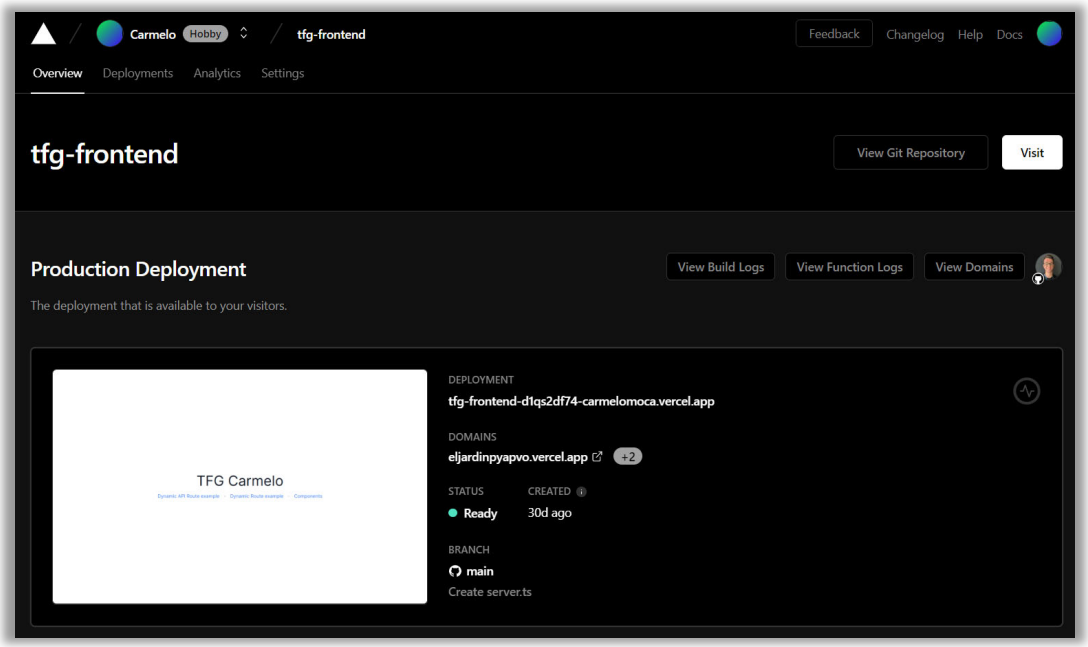


Figura 18: Dashboard Vercel, frontend

La conexión al *backend* se hará mediante el enlace: https://eljardin.up.railway.app/

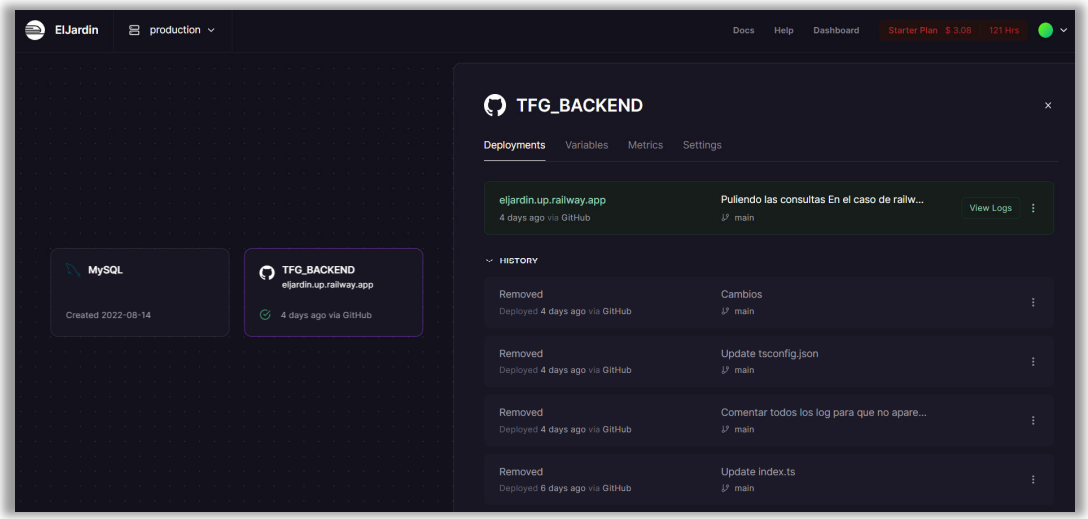


Figura 19: Dashboard Railway, backend

La conexión entre *backend* y persistencia se hace mediante internamente mediante variables de entorno.

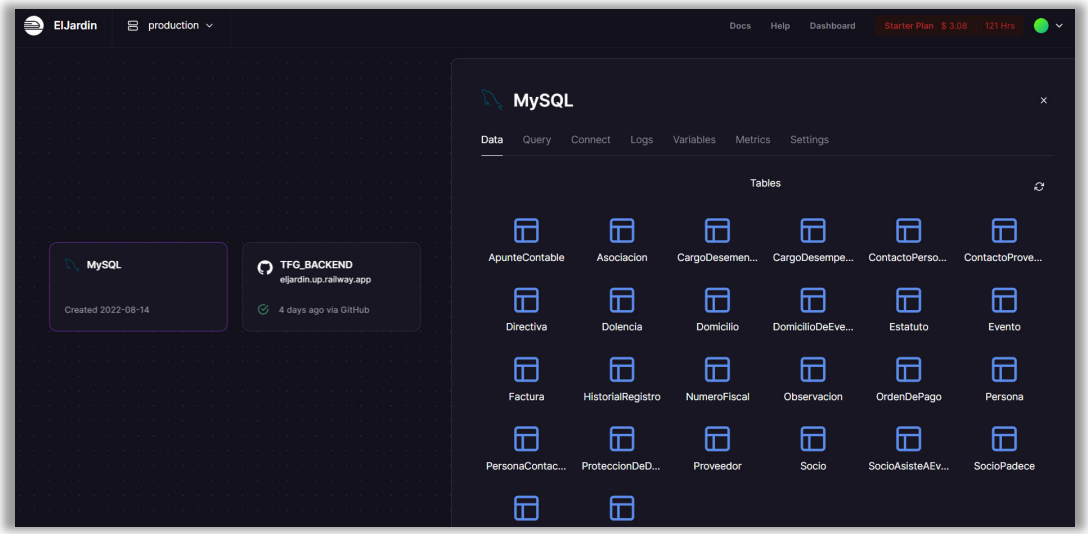


Figura 20: Dashboard Railway, Base de Datos

La dependencia que gestiona mediante código las conexiones *frontend-backend* y *backend-BD* se hace a través de la librería *mysql2* para *javascript*.

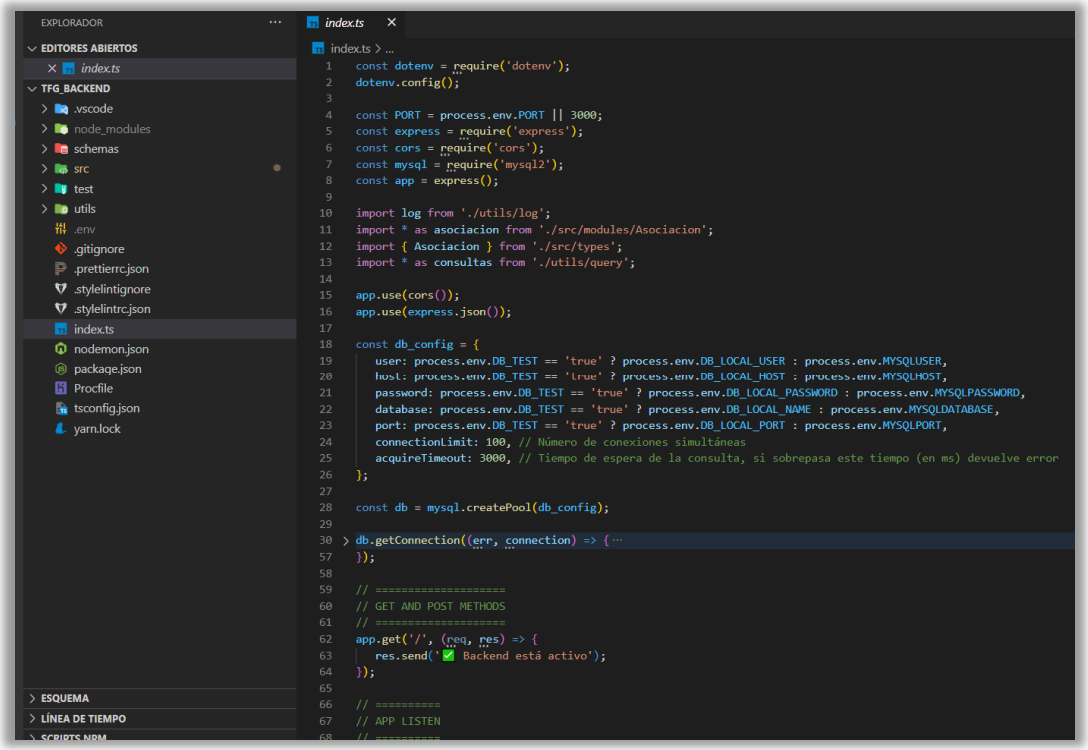


Figura 21: Configuración de comunicaciones entre distintos procesos de la aplicación.

4.4 Migración de datos

El programa actual tiene un proceso que crea un archivo '*.xlsx*' que sirve de copia de seguridad a la aplicación. Este archivo está organizado en hojas que simulan ser tablas de una BD.

Figura 22: Migración de datos. Backup, página resumen.

Por tanto, para migrar los datos tenemos dos opciones:

Opción A. Crear una función en *vba* (nativo de Excel) para que pase los datos del archivo *backup* en cuestión a un archivo *SQL* y que podamos inyectar así los datos en la *BD* mediante un script.

Opción B. Integrar una funcionalidad mediante funciones de *Javascript*, que lea documentos *Excel*, para que en cualquier momento (siempre que el usuario tenga los permisos adecuados), la aplicación, permita leer un archivo de *backup* antiguo y restaurarlo.

La ventaja de la opción A es que es más sencilla, pero no está integrada en nuestro aplicativo, y dependemos de otro lenguaje de programación.

La ventaja de la opción B es que, aun siendo más compleja, se podrá hacer en cualquier momento, estará integrada y formará parte como una funcionalidad más.

Teniendo en cuenta estas opciones, nos decantamos por la opción B, que aun siendo más compleja, queda mucho mejor integrada.

4.4.1 Modelo Entidad-Relación actual

El actual modelo Entidad-Relación del programa existente (Figura 23), está formado por un total de 7 tablas que contienen toda la información relativa a los socios.

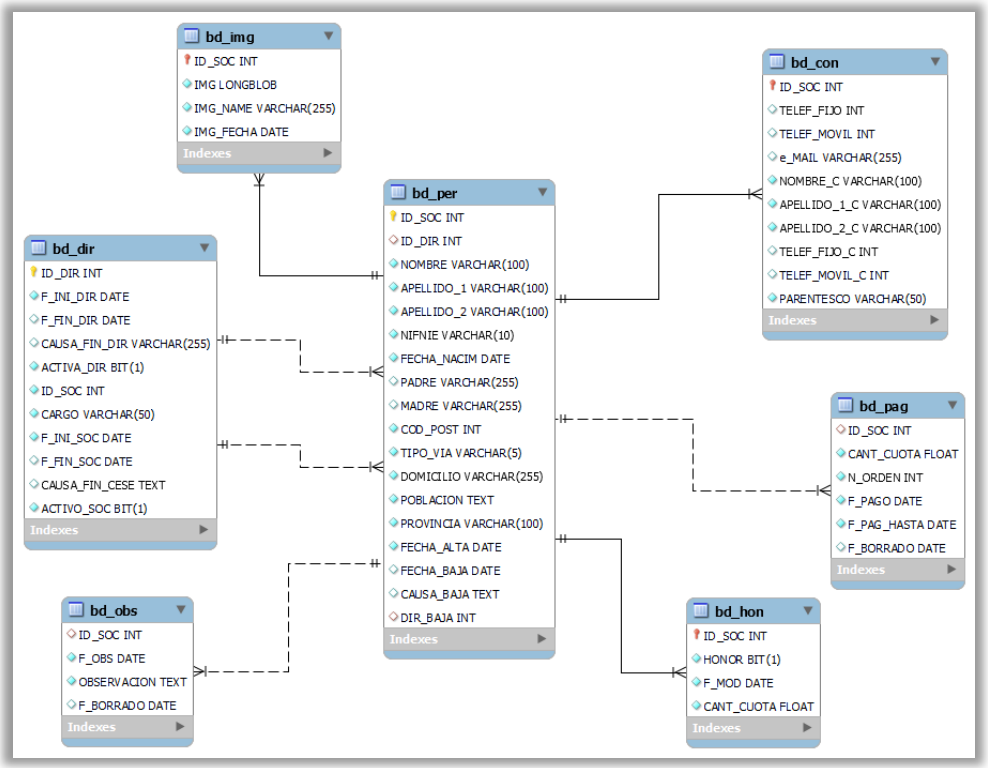


Figura 23: Diagrama -Entidad-Relación actual

En el *ANEXO II: Especificación software* apartado *7.1 Descripción de tablas en la BD actual* se hace una descripción detallada sobre el contenido de cada tabla.

4.4.2 Creación de *Wrappers*

Un *wrapper* consiste en un programa que nos permite extraer información de una fuente de datos concreta. En nuestro caso, esa fuente se trata del archivo de *backup* que el programa actual genera.

Este procedimiento constará de tres funcionalidades.

- Primero procesaremos el archivo *Excel* origen para generar un archivo *JSON* con la misma estructura que él.
- Después trataremos ese archivo *JSON* para transformarlo en un nuevo *JSON* con la estructura de la nueva BD.
- Por último pasaremos del segundo *JSON* a un *script sql* que nos servirá para inyectar los datos en la BD.

Hemos usado un *JSON* auxiliar por que pasar el *Excel* al *script sql* no es viable, ya que la estructura de ambas bases de datos es completamente distinta y necesitamos datos con los que completar algunas tablas, mientras que se leen otras.

Este método, aunque más laborioso, permite estandarizar el proceso de poblado inicial de la BD, y por tanto mejora la adaptabilidad a otros entornos.

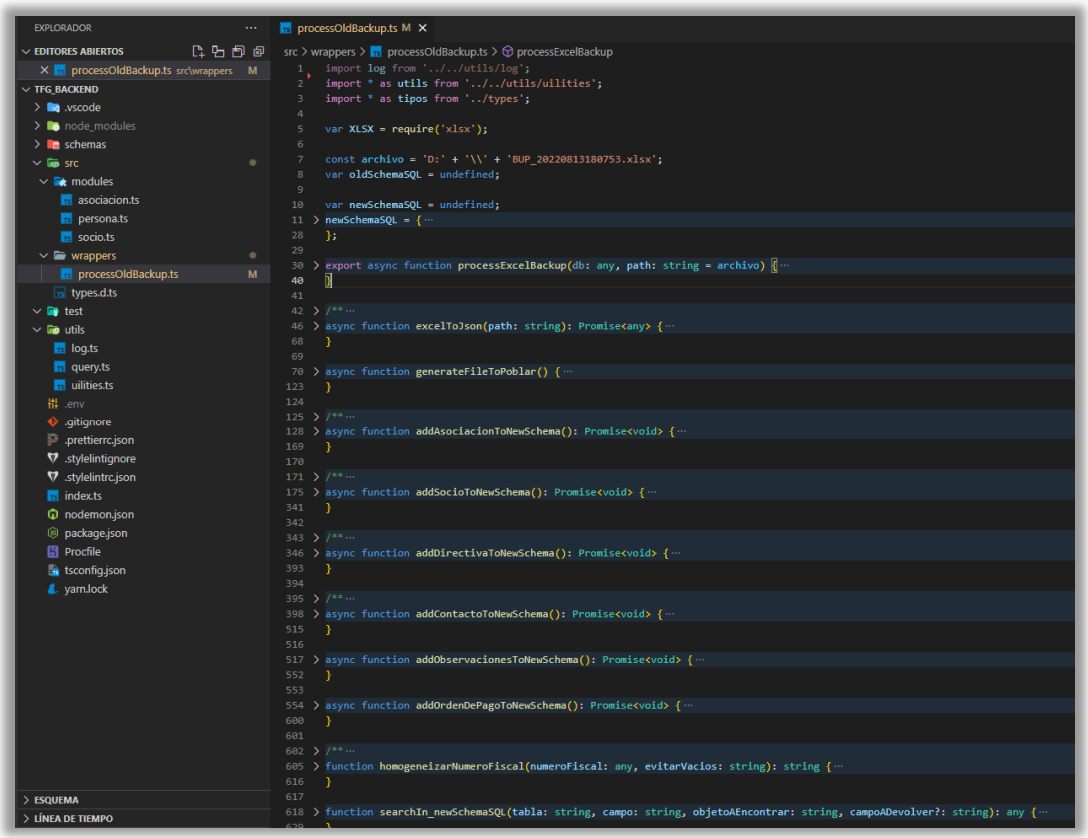


Figura 24: Código. Estructura del wrapper

5 Pruebas y Calidad del Software

El desarrollo del software de calidad necesita el uso de *test* para reducir significativamente los errores producidos durante el desarrollo, ya que estos suelen pasar desapercibidos durante la fase de desarrollo, debido a la vorágine creativa. Estos test no solo ayudan a la detección de errores, sino que también aseguran que el código funciona de acuerdo con las especificaciones. O ayudan en las labores de refactorización.

La elección de *Typescript* como lenguaje de desarrollo de la aplicación facilita bastante la terea de escribir código, ya que al ser un lenguaje tipado, el propio editor nos avisa de posibles incongruencias. Cosa que con *Javascript* no sucede, pudiendo corregir fallos durante la edición del código, sin necesidad de pasar por la compilación ni por la ejecución.

Aunque el lenguaje ayude en la minimización de errores, debemos diseñar unas pruebas que aseguren el funcionamiento de nuestra aplicación, así como asegurar la escalabilidad, sin sacrificar la funcionalidad.

En nuestro proyecto, hemos usado un *Framework* de pruebas llamado *Jest*, que es una librería abierta para pruebas en *Javascript* desarrollada por *Facebook* (junto a *React*) [29].

5.1 Pruebas unitarias

En el ámbito de la programación, los test unitarios son la forma de comprobar el funcionamiento de las unidades individuales, como pueden ser métodos y funciones concretas. De este modo aseguramos que cada unidad funcione de manera eficiente y correcta por separado, además de comprobar que el código funciona de acuerdo con las especificaciones [28]

Figura 25: Implementación de un test en Jest

Aunque para que los test funcionen, debemos asegurar que tenemos un entorno controlado, donde todas las variables externas las tenemos fijadas, para asegurar que la prueba solo verifica el método que estamos testeando. Para ello, y debido a que no solo testeamos funcionalidades matemáticas o algoritmos, sino que también comprobamos conexión a base de datos y extracción

de datos de ella, debemos tener un entorno de trabajo cerrado y controlado. Para ello hemos poblado una base de datos, con datos (valga la redundancia) específico para las pruebas.

Una vez creados los test, se pueden ejecutar mediante el comando *yarn run test*, mostrándonos el siguiente resultado.

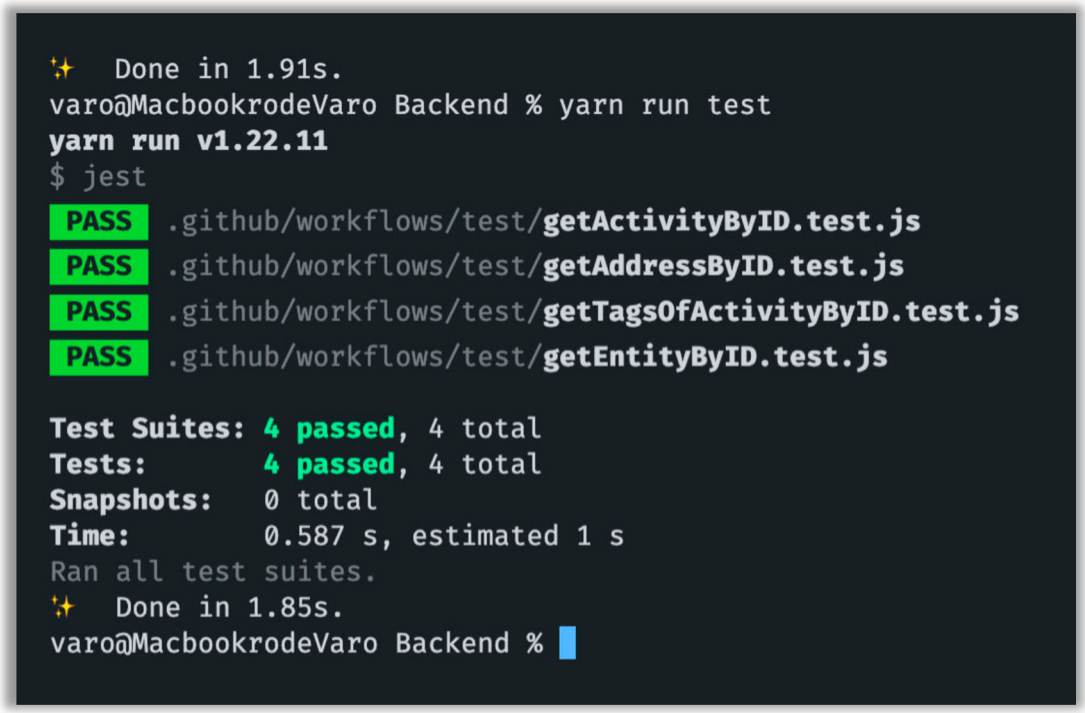


Figura 26: Resultado test unitarios

5.2 Pruebas de integración

Una vez realizadas las pruebas unitarias, y habiendo asegurado que cada método o función trabajan correctamente por su cuenta, llega la hora de probar que el funcionamiento en conjunto es el esperado.

En este tipo de pruebas nos centramos en comprobar la comunicación (hardware o software) entre los componentes, así como en los diferentes flujos que un socio puede tener dentro de aplicación (desde que inicia sesión hasta que realiza cualquier tipo de acción, como ver eventos, ver pagos, corregir información de su ficha de datos, etc.).

Este tipo de pruebas tiene un coste de personal elevado por lo que es necesario el uso de herramientas que automaticen estas comprobaciones. Pero también ayudan a asegurar a que la aplicación funciona de acuerdo con las especificaciones después de la resolución de algún bug o de la implementación de alguna funcionalidad extra. Ya que solo habrá que volver a ejecutar el test.

Para esta labor hemos decidido implementar el *framework Cypress*, ya que se integra fácilmente con *nextjs*, ofrece una documentación de calidad y es sencillo tanto de configurar como de escribir los test. Además cuenta con otras ventajas como log o la grabación en vídeo de las pruebas para su posterior revisión.

```
describe('search-available-courts', () => {
  it('calls center', () => {
    cy.visit('http://localhost:8100/search-available-courts');
    cy.wait(500);
    // eslint-disable-next-line max-len
    cy.get(
      '#main > app-search-available-courts > ion-content > div:nth-child(2) > app-court-card'
    ).click();
    cy.wait(500);
    cy.get(
      // eslint-disable-next-line max-len
      '#main > app-center-preview > ion-content > div > div.info > ion-grid > ion-row:nth-child(6) > ion-col > div > ion-button:nth-child(1)'
    ).click();
  });
```

Figura 27: Estructura de un test con Cupress

El procedimiento consiste en indicar a *Cypress* hacia que *url* debe navegar para que mediante selectores *css* indicar a que elementos le debe mandar el evento de clic para llevar a cabo el flujo que queremos testear.

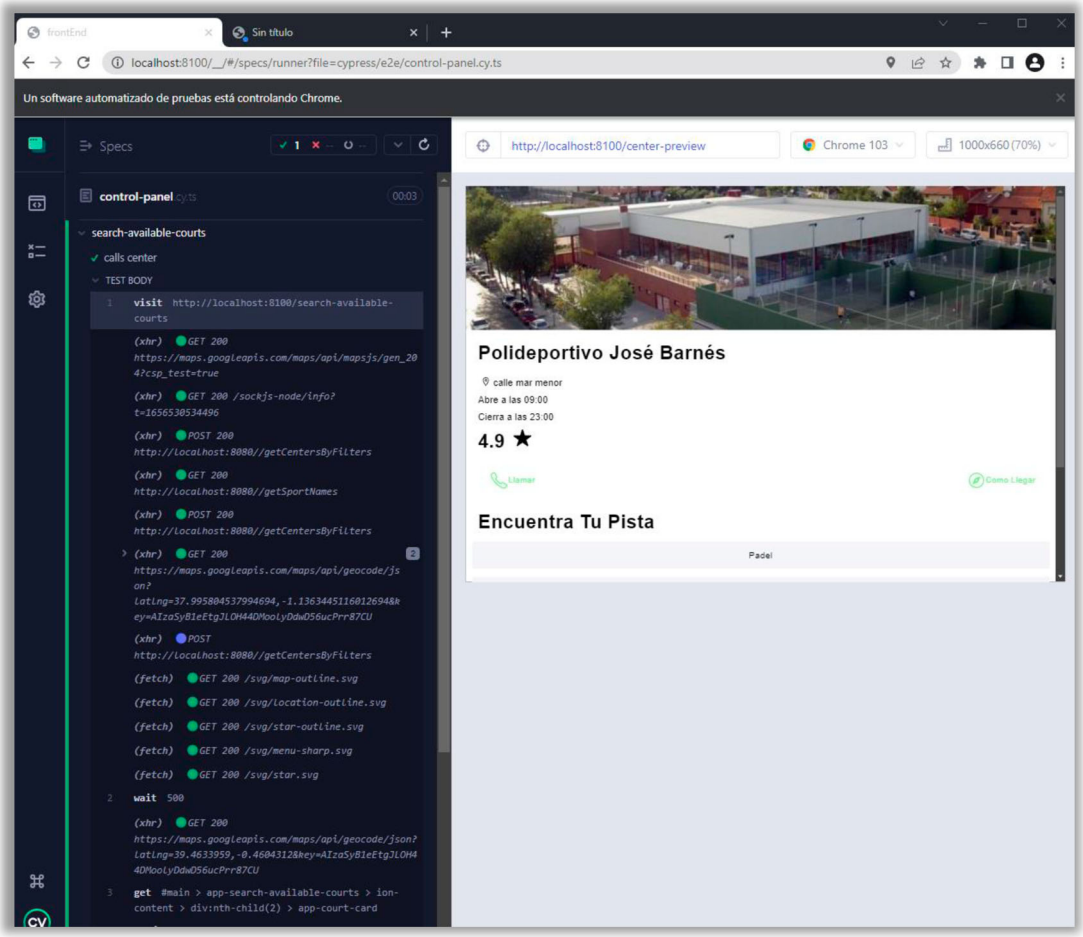


Figura 28: Captura de pantalla de una ejecución de test en Cypress[1]

En la Figura 28 podemos ver la interfaz gráfica de una ejecución de Cypress donde se muestra el navegador y el sistema de logs con todas las acciones que se están realizando.

[1] Imagen extraída del TFG:
Desarrollo de una aplicación híbrida para la reserva y gestión de instalaciones deportivas: desarrollo del *frontend*.
Grado en Ingeniería Informática. Curso 2021/2022
Rubén Gordo Gil (Tutor: Vicente Pelechano Ferragud.)

5.3 Evaluación heurística de la interfaz del usuario

Con el uso del análisis heurístico pretendemos identificar problemas comunes de usabilidad asociados a un producto, con el fin de poder resolverlos y mejorar tanto el grado de satisfacción del usuario como su experiencia. Pero también aumentamos la posibilidad de éxito de nuestro producto.

Si nos centramos en la usabilidad, el análisis heurístico es el método de evaluación en el que uno, o más expertos, hacen una comparativa entre el diseño del producto con una lista de principios predefinidos (heurísticos), para identificar donde no se sigue estos principios [30].

Aunque lo óptimo es que se lleve a cabo por 3-7 evaluadores (según *Nielsen Norman Group*), lo normal es que sea llevado a cabo por 2 personas [31].

Los 10 principios de Jakob Nielsen[2]:

1. ***Visibilidad del estado del sistema:*** El sistema debe mantener informado a los usuarios lo que está pasando en cada momento (debe haber *feedback* constante); el usuario no puede sentirse perdido.
2. ***Coincidencia entre el sistema y el mundo real:*** Se trata de hablar el lenguaje del usuario (que le sea familiar), y no términos orientados al sistema.
3. ***Control y libertad del usuario:*** El usuario debe poder deshacer y rehacer acciones; debe sentir que tiene el control en todo momento.
4. ***Consistencia y estándares:*** Es importante mantener la misma consistencia a lo largo de todo el sistema (color, lenguaje, flujo de navegación, etc.). La consistencia es clave a la hora de diseñar interfaces usables. Cuando el esquema de navegación está estructurado de acuerdo con el modelo mental de la mayoría de los usuarios, se tiene el 80% de éxito.
5. ***Prevención de errores:*** Se debe ayudar al usuario a equivocarse antes de que cometa el error; lo ideal sería validar el error antes de que realice la acción.
6. ***Mostrar en lugar de recordar:*** Se trata de minimizar la carga cognitiva del usuario, es decir, aliviar la carga de memoria del usuario haciendo visible objetos, acciones y opciones. Siempre se deben indicar los campos por los cuales se ha realizado la búsqueda.
7. ***Flexibilidad y eficiencia de uso:*** Aceleradores, atajos (visuales o no) y recomendaciones que faciliten la navegación, tanto para los usuarios sin experiencia como con experiencia. Es importante que el sistema permita personalizar acciones frecuentes.
8. ***Diseño estético y minimalista:*** Se trata de mostrar solo lo relevante; eliminar el ruido visual, información de menor importancia.
9. ***Ayudar a reconocer, diagnosticar y recuperarse de errores:*** Comunicar errores con facilidad, proveer una solución o camino a seguir y proponer una alternativa.
10. ***Ayuda y documentación:*** Por más fácil o simple que sea, es importante que el sistema ofrezca ayuda relevante al contexto del usuario, como las cajas de búsqueda, *FAQs*, contacto, etc. La misma debe ser fácil de encontrar.

[2] La información sobre los 10 principios están extraídos de Medium
https://blog.prototypr.io/evaluacion-heuristica-9c8fce655759

Análisis heurístico

	Web	Competidor 1	Competidor 2	Competidor 3
	Evaluador 1	Evaluador 1	Evaluador 1	Evaluador 1
Relevancia ¿Resuelve esta página mi problema?				
¿Es esta la página que estaba buscando? (analizar las fuentes de tráfico, revisar si está alineada la Landing page con las expectativas del usuario, revisar los Ads copys)	5	3	0	0
¿Es para gente como yo? (analizar ***personas*** para comprender las motivaciones del usuario)	3	4	0	0
Contenido: ¿Concuerda el vocabulario con el propósito, especialmente los títulos (headlines)?	3	2	0	0
¿Produce resonancia emocional, la *sensación* de estar en el sitio correcto? ¿Está enfocado y es claro?	2	5	0	0
¿Muestra la landing page la gama de productos?	5	2	0	0
¿Está claro el rango de precios ofrecido (bajo/medio/alto)?	4	1	0	0
¿Existe una propuesta de valor que encaje con los valores emocionales que pretende transmitir?	3	3	0	0
¿Es la experiencia coherente con la del resto de páginas?	5	5	0	0
Opcional: ¿Se usa el storytelling para crear resonancia?	4	2	0	0
Total Relevancia	75.6%	60.0%	0.0%	0.0%
Confianza ¿Puedo confiar en esta empresa?				
¿Se basa el sitio en los principios de credibilidad?	2	3	0	0
¿Es la arquitectura de la información clara y lo que se espera?	5	5	0	0
¿Se muestra la imagen de forma clara y transmite confiabilidad?	3	2	0	0
¿Usa sellos u otros generadores de confianza?	3	4	0	0
¿Se muestran opiniones de compradores (social proof)?	1	2	0	0
¿Se usan celebridades u otros famosos para promocionarlo?	2	3	0	0
¿Se muestran marcas conocidas que se vendan en la tienda?	4	5	0	0
Total Confianza	57.1%	68.6%	0.0%	0.0%
Orientación ¿Dónde debo ir (hacer click)?				
¿Dónde debo hacer click? ¿Existe un CTA claramente visible?	4	5	0	0
¿Indican los elementos CTA claramente las consecuencias de accionarlos?	5	4	0	0
¿Ayuda el sitio al usuario a sobreponerse a la paradoja de la elección?	4	2	0	0
¿Es fácil comparar opciones?	2	3	0	0
¿Se usan CTAs secundarios para trabajar las posibles objeciones del usuario?	3	1	0	0
Total Orientación	72.0%	60.0%	0.0%	0.0%
Estímulo ¿Por qué debería comprar aquí?				
¿Ofrece el sitio una propuesta de valor clara?	3	3	0	0
¿Es de verdad relevante la propuesta de valor?	5	2	0	0
¿Son únicas las propuetas de valor? ¿Realmente?	4	5	0	0
¿Está la percepción coste / beneficio bien diseñada?	3	3	0	0

Análisis heurístico

	Web	Competidor 1	Competidor 2	Competidor 3
	Evaluador 1	Evaluador 1	Evaluador 1	Evaluador 1
¿Crea el sitio una experiencia de usuario divertida, agradable?	5	4	0	0
¿Se usan regalos, obsequios u otros elementos de reciprocidad?	4	1	0	0
¿Se crea sentido de urgencia / escasez?	2	3	0	0
¿Se ofrece envoltorio / envío gratis?	3	2	0	0
Total Estímulo	82.9%	65.7%	0.0%	0.0%
Seguridad				
¿Responde el sitio a las preguntas más habituales en páginas transaccionales?	3	5	0	0
¿Se anticipan las dudas de los clientes y se resuelven correctamente?	5	4	0	0
¿Existe una FAQ?	5	5	0	0
¿Existe alguna sección que responda preguntas abiertas? ¿Hay enlaces a ella(s)?	4	4	0	0
¿Hay chat disponible? ¿o un teléfono de atención al cliente? ¿Son siempre visibles?	3	2	0	0
¿Es posible añadir algún tipo de "seguro" para reducir el riesgo?	4	3	0	0
Formularios y generación de Leads: ¿Se explica por qué se necesita cierta información?	3	2	0	0
¿Existe una prueba gratuita?	4	4	0	0
Total Seguridad	77.1%	71.4%	0.0%	0.0%
Facilidad				
¿Se **perciben** los formularios como sencillos?	3	2	0	0
Diseño limpio y agradable	4	4	0	0
¿Está orientado a la máxima simplicidad?	5	1	0	0
Interacción positiva UX (e.g. inline validation, cheering)	2	5	0	0
Fragmentado + Tunneling ¿Las cuestiones más fáciles están al principio?	4	2	0	0
¿Son realmente sencillos?	3	1	0	0
¿La funcionalidad sigue los modelos mentales?	4	3	0	0
Explicación clara de la funcionalidad	2	5	0	0
Se ofrece ayuda	3	4	0	0
Total Facilidad	66.7%	60.0%	0.0%	0.0%
Confirmación				
Mostrar buenas razones (racionales) para hacer la compra	3	4	0	0
Usar micro feedback en páginas y elementos (ej: inline validation)	2	5	0	0
Presentar buenas razones en Página final (Thank Page, Email de confirmación)	4	2	0	0
Usar tono amigable e interacciones positivas + feedback positivo	5	3	0	0
Total Confirmación	70.0%	70.0%	0.0%	0.0%

Análisis heurístico

	Web	Competidor 1	Competidor 2	Competidor 3
	Evaluador 1	Evaluador 1	Evaluador 1	Evaluador 1
Nota media	**71.6%**	**65.1%**	**0.0%**	**0.0%**

Resumen de resultados

	Web	Competidor 1	Competidor 2	Competidor 3
Relevancia	75.6%	60.0%	0.0%	0.0%
Confianza	57.1%	68.6%	0.0%	0.0%
Orientación	72.0%	60.0%	0.0%	0.0%
Estímulo	82.9%	65.7%	0.0%	0.0%
Seguridad	77.1%	71.4%	0.0%	0.0%
Facilidad	66.7%	60.0%	0.0%	0.0%
Confirmación	70.0%	70.0%	0.0%	0.0%

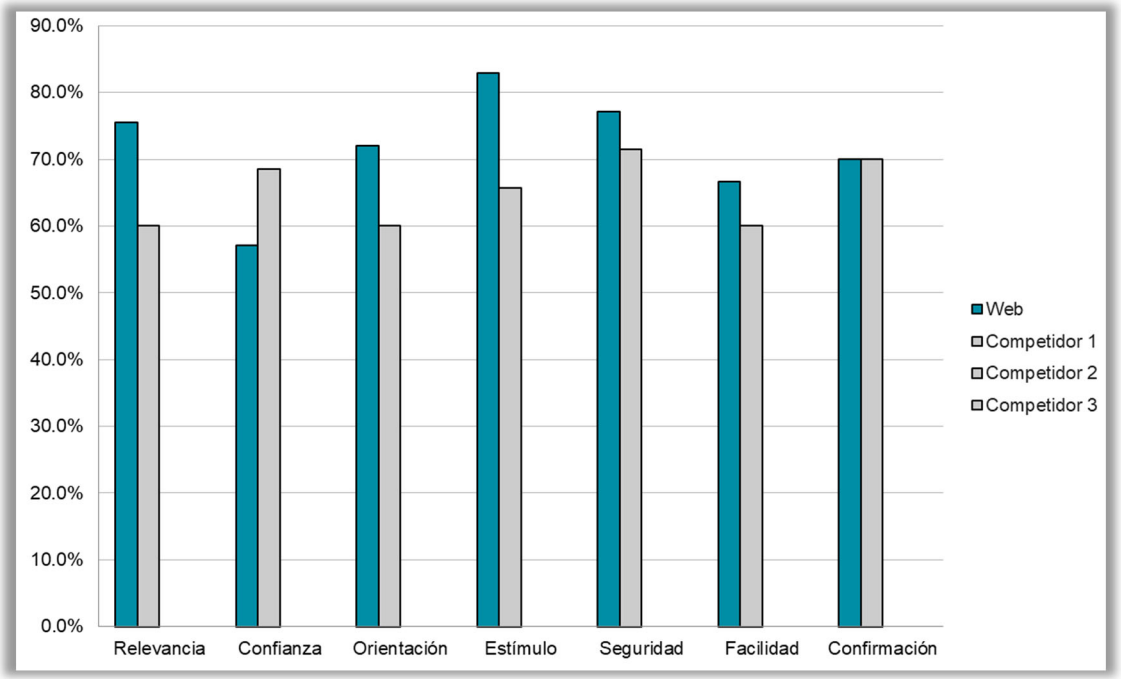


Figura 29: Formulario 'Evaluación Heurística'

5.3.1 ¿Cómo se lleva a cabo?

En primer lugar hay que conseguir reunir a varios expertos (entre 3 y 7 como comentamos en los párrafos anteriores), aunque esto suele disparar el coste.

Los especialistas llevan a cabo la evaluación individualmente para que ningún compañero les influyan. Además, cada experto revisa la interfaz dos veces, evaluando cada elemento de esta, de acuerdo con la lista de heurísticas.

5.3.2 ¿Cuándo usar esta técnica?

Aunque se puede utilizar en, prácticamente, cualquier momento del ciclo de desarrollo, se adapta mejor a etapas tempranas, cuando no hay material lo suficientemente firme para efectuar un test. Se puede proporcionar maquetas de papel o incluso especificaciones de diseño a los expertos y detectar una buena cantidad de problemas de usabilidad antes de que el trabajo real de producción de comienzo.

6 Mantenimiento y Gestión de Versiones

Para poder elegir un mejor flujo a la hora de controlar versiones, hemos decidido analizar entre dos conocidos métodos de trabajo, "*Gitflow*" [28] y "*Desarrollo basado en troncos*" [28].

En resumidas cuentas, "*Desarrollo basado en troncos*" tiene un uso prácticamente obligatorio en CI/CD (Integración y Distribución Continuas [29])

6.1 Comparación *Gitflow* y Desarrollo basado en troncos [28]

Mientras que *Gitflow* es un modelo de creación de ramas en Git, que utiliza estas ramas de larga duración y varias ramas principales, con duraciones y confirmaciones grandes ya que retrasan su fusión con la rama principal hasta que la función está completa. En el Desarrollo basado en troncos, el proceso está más simplificado, ya que se centra en una rama principal como fuente de correcciones y publicaciones. En este tipo de desarrollo la rama principal permanece estable sin incidencias y siempre lista para la implementación.

6.2 Metodología utilizada.

En nuestro proyecto usaremos dos ramas principales:

- **Main:** será nuestra rama estable, la que será publicada y con la que funcionará nuestro aplicativo. No está permitido subir nada que no se haya testeado correctamente en las ramas anteriores.
- **Beta:** corresponde a nuestra rama de testeo e integración, en ella se pasarán los test y de ella se extraerán todas las ramas hijas para integrar las funcionalidades. Prácticamente es un clon de la rama '*Main*'.
- **Funcionalidad_X:** Estar ramas se crean para implementar pequeñas funcionalidades. Aparecen y desaparecen en el momento en que se ha integrado en '*Beta*' y han pasado los test.

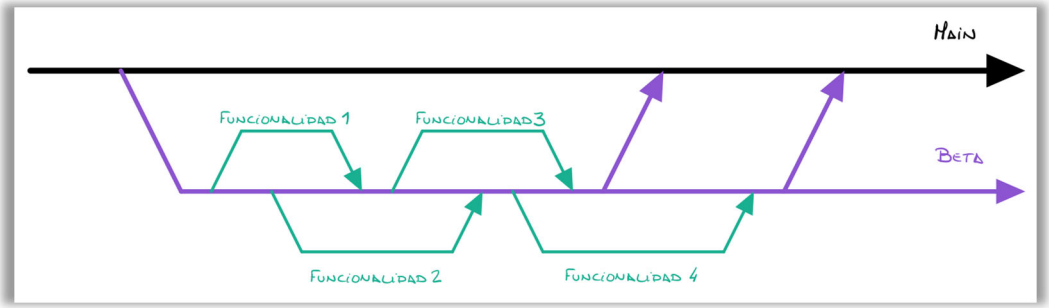


Figura 30: Flujo de ramas en git

6.3 Mantenimiento software

Conforme se va escribiendo el código de la aplicación, y este se va haciendo más grande y es más difícil seguir la traza cuando aparece un bug. Lo normal en esos casos es hacer que las funciones devuelvan el típico *-1* o *false*, pero con esto, la traza que seguimos en la depuración queda incompleta y difícil de seguir. A esto se le añade lo tedioso que es repetir la frase con el código de error en los test unitarios, aumentando la dificultad si cambiamos alguna mayúscula por minúscula, o si acentuamos o no alguna palabra.

Por lo expresado en el párrafo anterior, se ha decidido que la aplicación devuelva códigos de error para hacer más fácil la identificación de en el lugar del código donde ha sucedido, además de la descripción del error y la posible solución asociada.

A continuación se muestran, a modo de ejemplo, algunas de las claves de error que se han implementado en el código de la aplicación:

Código	***Descripción***	***Solución***
ER_BAD_TABLE_ERROR	La tabla donde se está consultando los datos no existe.	Hay dos opciones, que efectivamente no exista y estamos haciendo la consulta en una BD vacía, sin poblar. O puede ser que el nombre de la tabla esté mal escrito (*case sensitive*).
ERR_BD_0001	Error de ejecución del script de borrado de todas las tablas.	Se da en la inicialización del programa. Ejecutar el script de manera manual mediante *MySQLWorkbench* (o similar).
ERR_BD_0002	Error de ejecución del script de creación de todas las tablas.	Se da en la inicialización del programa. Ejecutar el script de manera manual mediante *MySQLWorkbench* (o similar).
ERR_BD_0003	Error de ejecución del script de creación de claves foráneas.	Se da en la inicialización del programa. Ejecutar el script de manera manual mediante *MySQLWorkbench* (o similar).
ERR_BD_0004	Ha fallado la inicialización de la base de datos	Comprobar que todos los archivos /schemas/scriptSql están generados y con los datos correctos

Tabla 4: Códigos de error que genera las consultas a la Base de Datos

7 Conclusiones

Desarrollar una aplicación desde el cero, teniendo en cuenta todos los procedimientos mostrados en el grado, es un trabajo laborioso y lleno de matices. El desarrollo, a título personal, ha sido muy enriquecedor, el tratar tan a fondo con nuevas tecnologías, el escuchar e intentar plasmar en un programa funcional las ideas que otras personas, me ha hecho entender lo importante que es la especificación de requisitos, en un proyecto software. También me ha mostrado la importancia de la mantenibilidad de la aplicación, ya que sin ella definida, cualquier cambio no garantizaría que el sistema sigue funcionando correctamente, incluso en etapas tempranas del desarrollo, es muy útil la implementación de este tipo de test para ganar eficiencia en el desarrollo.

De los objetivos generales que se describieron en el punto 1.2.1 creo que han quedado cubiertos todos los puntos, aunque sin poder profundizar todo lo que me hubiese gustado. Recordemos que el TFG son 12 créditos (o el equivalente a unas 300 horas de trabajo).

En cambio, de los objetivos específicos que se detallaban en el punto 1.2.2, la parte, quizás más innovadora, que es la de integración de la aplicación con redes sociales, es la que más atrás a quedado, bien por tiempo, bien porque después de reuniones con la asociación, era lo que más estaban posponiendo, ya que le urgen otras funcionalidades con uso más periódico. Aunque están encantados con integrar el correo electrónico en la propia web, ya que según ellos:

'Está todo organizado en un sitio y no nos tenemos que marear en el internet'.

7.1 Relación con los estudios cursados

Desde el principio, me ha movido el propósito de poder agrupar todo el conocimiento adquirido en la titulación de *Grado en Ingeniería Informática* y más específicamente, en *la rama de software* y poder desarrollar un proyecto que conectase, si no todas, las asignaturas que he creído más relevantes a la hora de desarrollar software de calidad.

Las asignaturas que durante todo este tiempo he tenido en mente son:

- **Interfaces Persona Computador (IPC)** para el diseño de una interfaz usable adaptada a personas de la 3ª edad, con conocimientos nulos en informática.
- **Ingeniería de Requisitos (AER)** para poder hacer un buen desarrollo de la aplicación, amoldándose a las necesidades de la asociación.
- **Aplicación de patrones de diseño, refactorización y *clean code* (DDS)** para tener un código ordenado, sin código muerto, fácil de mantener y escalar.
- **Mantenibilidad Software (MES)** para poder hacer un software mantenible, y ampliable en un futuro, ya sea para nuevas funcionalidades, o para adaptarlas fácilmente a otras asociaciones.
- **Diseño de pruebas (AVD)** para poder comprobar y adaptar el software rápidamente.
- **Interoperabilidad (IEI)** para poder integrar distintos tipos de mensajería (WhatsApp, Telegram, SMS, email, …).
- **Desarrollo ágil (PIN y PSW)** para poder gestionar el proyecto en distintos sprint, y que la asociación vaya viendo resultados que puedan ir usando lo antes posible.

7.2 Trabajo futuro

Debido a la envergadura de la aplicación y a que se trata de un entorno vivo que se está utilizando, quedan pendientes un sinfín de funcionalidades (descritas en los diagramas de casos de uso) que no han sido posible de implementar en un primer y único sprint. Por tanto, quedan pendientes su implementación, de acuerdo con el orden que establezca la asociación.

No obstante, hay una funcionalidad que se ha pasado por alto, y que sería interesante desarrollar de cara a poder implementar esta herramienta en otras asociaciones, y que es la creación de una interfaz de configuración del entorno, ya que actualmente se hace mediante script cargando los datos iniciales en la base de datos.

También queda por hacer un Anexo dedicado solo y exclusivamente al mantenimiento software, donde se recoja, entre otras cosas, todos los códigos de error implementados en la herramienta, en que funciones o métodos se genera y su posible solución.

Por último, nos falta redactar un manual del usuario que describa todas las funcionalidades de la aplicación.

8 Referencias

[1] Berrly Data Tools, SL, «Berrly,» [En línea]. Available: https://www.berrly.com/es/?utm_term=software%20asociaciones&utm_campaign=MF%20-%20Campa%C3%B1a%200&utm_source=adwords&utm_medium=ppc&hsa_acc=3338716397&hsa_cam=10271016949&hsa_grp=104987847720&hsa_ad=536051849247&hsa_src=g&hsa_tgt=kwd-334928716294&hsa_kw=. [Último acceso: 06 Agosto 2022].

[2] Playoff informática S.L., «Playoff,» [En línea]. Available: https://playoffinformatica.com/gestion-de-asociaciones/?utm_source=google&utm_medium=cpc&utm_campaign=164334724&utm_content=125523686800&utm_term=software%20gestion%20de%20asociaciones&gclid=CjwKCAjw5NqVBhAjEiwAeCa97UdDBVjqRhHAhJaBh3Y8TlViX2sSxvOz1VZIlQdZ. [Último acceso: 06 Agosto 2022].

[3] cucunver, «cucunver,» [En línea]. Available: https://cucunver.com/contacto. [Último acceso: 06 Agosto 2022].

[4] Taclia, «taclia,» [En línea]. Available: https://www.taclia.com/. [Último acceso: 06 Agosto 2022].

[5] A. B., «LinkedIn,» 07 01 2021. [En línea]. Available: https://es.linkedin.com/pulse/qu%C3%A9-es-la-t%C3%A9cnica-moscow-alexander-ballesteros-gonzalez#:~:text=El%20m%C3%A9todo%20MosCow%20es%20una,cada%20uno%20de%20los%20requerimientos.. [Último acceso: 03 07 2022].

[6] Agencia Estatal Boletín Oficial del Estado, «Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.,» [En línea]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2002-5852. [Último acceso: 06 Agosto 2022].

[7] Agencia Estatal Boletín Oficial del Estado, «Ley Orgánica 1/2002, de 22 de marzo, reguladora del Derecho de Asociación.,» [En línea]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673. [Último acceso: 06 Agosto 2022].

[8] pmoinformatica, «Requisitos funcionales,» 30 05 2018. [En línea]. Available: http://www.pmoinformatica.com/2018/05/que-es-requerimiento-funcional.html. [Último acceso: 14 07 2022].

[9] pmoinformatica, «Requisitos no funcionales,» 06 005 2015. [En línea]. Available: http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html. [Último acceso: 14 07 2022].

[10] Vercel Inc., «Vercel,» [En línea]. Available: https://vercel.com/. [Último acceso: 01 Julio 2022].

[11] railway, «railway,» [En línea]. Available: https://railway.app/. [Último acceso: 23 Agosto 2022].

[12] Microsoft, «Visual Studio Code,» [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 06 Agosto 2022].

[13] GitHub, Inc., «Github,» [En línea]. Available: https://github.com/. [Último acceso: 06 Agosto 2022].

[14] GitHub, Inc, «GitHub Desktop,» [En línea]. Available: https://desktop.github.com/. [Último acceso: 06 Agosto 2022].

[15] Microsoft, «Microsoft Word,» [En línea]. Available: https://www.microsoft.com/es-es/microsoft-365/word. [Último acceso: 06 Agosto 2022].

[16] Microsoft, «Microsoft Power Point,» [En línea]. Available: https://www.microsoft.com/es-es/microsoft-365/powerpoint. [Último acceso: 06 Agosto 2022].

[17] dia-installer.de, «Editor des diagramas Dia,» [En línea]. Available: http://dia-installer.de/index.html.es. [Último acceso: 06 Agosto 2022].

[18] draw.io, «draw.io,» [En línea]. Available: https://drawio-app.com/. [Último acceso: 06 Agosto 2022].

[19] Atlassian, «Trello,» [En línea]. Available: https://trello.com/?&aceid=&adposition=&adgroup=121044377259&campaign=12739109720&creative=514116061106&device=c&keyword=trello&matchtype=e&network=g&placement=&ds_kids=p62988466005&ds_e=GOOGLE&ds_eid=700000001557344&ds_e1=GOOGLE&gclid=EAIaIQobChMIt8z5orm. [Último acceso: 06 Agosto 2022].

[20] Oracle, «MySQL Workbench,» [En línea]. Available: https://www.mysql.com/products/workbench/. [Último acceso: 06 Agosto 2022].

[21] Cartero, Inc., «Postman,» [En línea]. Available: https://www.postman.com/. [Último acceso: 06 Agosto 2022].

[22] 3ymedia School, «Figma,» 19 Octubre 2022. [En línea]. Available: https://3ymedia.school/que-es-figma/. [Último acceso: 23 Agosto 2022].

[23] K. Tool, «Kanban Tool,» 01 01 2022. [En línea]. Available: https://kanbantool.com/es/metodologia-kanban. [Último acceso: 03 07 2022].

[24] Arquitectura Hexagonal con Typescript en APIs web con Nodejs, «Youtube,» DotNet 2021, 23 Junio 2021. [En línea]. Available: https://www.youtube.com/watch?v=ds7mHECHNj0. [Último acceso: 11 Agosto 2022].

[25] F. B. Galiano, «Arquitecturas multicapa,» [En línea]. Available: http://elvex.ugr.es/decsai/csharp/design/layers.xml. [Último acceso: 06 Agosto 2022].

[26] Wikipedia, «Diagrama de clases,» [En línea]. Available: https://es.wikipedia.org/wiki/Diagrama_de_clases. [Último acceso: 06 Agosto 2022].

[27] Wikipedia, «Modelo entidad-relacion,» [En línea]. Available: https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n. [Último acceso: 06 Agosto 2022].

[28] Atlassian, «Comparación de Gitflow y desarrollo basado en troncos,» [En línea]. Available: https://www.atlassian.com/es/continuous-delivery/continuous-integration/trunk-based-development. [Último acceso: 20 08 2022].

[29] Red Hat, «¿Qué son la integración y la distribución continuas (CI/CD)?,» [En línea]. Available: https://www.redhat.com/es/topics/devops/what-is-ci-cd. [Último acceso: 20 08 2022].

ANEXO I: Objetivos de desarrollo sostenible (ODS)

1. Grado de relación del trabajo con los ODS

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**	**X**			
ODS 4. **Educación de calidad.**				
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**		**X**		
ODS 10. **Reducción de las desigualdades**.			**X**	
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**		**X**		
ODS 17. **Alianzas para lograr objetivos.**			**X**	

2. Reflexión sobre la relación del TFG con los ODS[3]

Punto 3.- *Garantizar una vida sana y promover el bienestar en todas las edades es esencial para el desarrollo sostenible.*

Nuestra aplicación cuenta con un apartado de 'Dolencias' en el que cualquier socio puede registrar sus intolerancias, alergias, minusvalías y otras enfermedades que quiera compartir con la asociación. De esta manera, al reservar estancias en hoteles o viajes gastronómicos estas se contemplan en la reserva y así puede disfrutar de menús apropiados a su dieta o descargar aquellos lugares con barreras arquitectónicas.

Punto 9.- *La industrialización inclusiva y sostenible, junto con la innovación y la infraestructura, pueden dar rienda suelta a las fuerzas económicas dinámicas y competitivas que generan el empleo y los ingresos. Estas desempeñan un papel clave a la hora de introducir y promover nuevas tecnologías, facilitar el comercio internacional y permitir el uso eficiente de los recursos.*

Aunque se trata de una aplicación adaptada a una asociación, esta ha servido para promover nuevas tecnologías, ya que hasta hace escasamente unos años el método de gestión era totalmente manual, con fichas manuscritas.

De este proyecto salen dos mejoras:

1. Creación de una aplicación usando nuevas tecnologías y diciéndole adiós al papel, usando y llevando a la 3ª edad la digitalización.
2. Creación de una aplicación que se puede explotar en otras asociaciones, creando un servicio vendible.

Punto 10.- *Reducir las desigualdades y garantizar que nadie se queda atrás forma parte integral de la consecución de los Objetivos de Desarrollo Sostenible.*

Con esta aplicación, se lleva la tecnología a nuestro mayores, en su tiempo de ocio, para que les sea fácil el tránsito de 'lo rustico' y manual, a lo digital. Ya que la mayoría usan todos un smartphone pero pocos les enseñan las capacidades que dicho dispositivo tienen.

[3] Objetivos de Desarrollo Sostenible,
El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años.
Para alcanzar estas metas, todo el mundo tiene que hacer su parte: los gobiernos, el sector privado, la sociedad civil y personas como usted.
https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

ANEXO II: Especificación software

1. Glosario

- A -

Administrador del sistema: Usuario con permisos '*root*' con acceso a todo el sistema, sin restricción de ningún tipo (siempre que no ponga en riesgo la integridad de la aplicación).

Administrativo del Ayuntamiento: Usuario con acceso de consulta a datos de los socios de forma masiva (listado de direcciones para correo postal y otros informes que el integrante de la directiva haya configurado).

Asociación: Hace referencia a "Club Municipal de Pensionistas de la Tercera Edad".

- B -

BackEnd: Es la parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web, por eso se dice que corre del lado del servidor.

BackUp: Anglicismo que se refiere a una copia de seguridad de los datos.

BLOB: se trata del tipo de almacenamiento que sufren los archivos al guardarlos en una Base de Datos. Los datos son almacenados en cadenas de bytes y permite poder guardar archivos de múltiples formatos.

- C -

CI/CD: método para distribuir las aplicaciones a los clientes con frecuencia mediante el uso de la automatización en las etapas del desarrollo de aplicaciones.

Clientes de mensajería: API que conecta la aplicación con la mensajería elegida (Telegram o WhatsApp).

Clientes de eMail: API que conecta la aplicación con el cliente de correo de la asociación.

CRUD: En informática, es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: *Create*, *Read*, *Update* and *Delete*), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

Cuota: Cantidad de dinero que un socio paga en un periodo de tiempo, para formar parte de la asociación durante dicho periodo de tiempo.

- D -

Directiva: Conjunto de socios (en activo) que organizan la asociación durante un periodo de 4 años (o menos). Son elegidos mediante votación en una asamblea extraordinaria.

- E -

Evento: Actividad que se realizará en el club, como viajes, cenas, excursiones, etc.

- F -

Familiar: Usuario que solo puede tener acceso de solo lectura a los datos de un socio, si este lo ha autorizado.

FrontEnd: Es la parte de un sitio web que interactúa con los usuarios, por eso se dice que está del lado del cliente.

- G -

GPS: sistema que permite localizar cualquier objeto (una persona, un vehículo, etc.) sobre la Tierra con una precisión de hasta centímetros (si se utiliza GPS diferencial), aunque lo común son unos pocos metros.

- I -

Integrante directiva: Usuario estándar del aplicativo, es quien puede agregar/modificar socios, eventos, etc. en la asociación. Es un socio que se encarga de la gestión de la asociación mientras dura el mandato (normalmente 4 años).

- J -

Javadoc: Es el estándar de la industria para documentar clases de objetos.

- K -

Kanban: Sistema de visualización que coordina una cadena de montaje para la entrega a tiempo de la producción. Desarrollo ágil de software que emplea prácticas de gestión visual.

- N -

Numero de orden: Es un número incremental que, en el libro de cuentas de la asociación, registra el orden en que se efectúan los pagos de los socios, en un año en concreto. Este número se pone a 1 todos los día 01 de enero de cada nuevo año, por tanto, a lo largo de los años se repite, y no tiene por qué coincidir para el mismo socio.

Numero de socio: Es un número incremental que identifica al socio dentro de la asociación. Es personal e intransferible, no se reaprovecha. Un socio solo lo pierde por cuestiones de impago de cuotas, no se puede recuperar (salvo excepciones puntuales), al igual que la antigüedad.

- P -

Proveedor/Acreedor: Sociedad que vende un producto o servicio a la asociación.

- R -

Root: Usuario que en un sistema tiene todos los permisos y no tiene restringida ninguna acción, área o tarea.

Rama 'main': en un repositorio de *git*, es la rama principal, es decir, la rama que pertenece a producción (con la que el usuario final trabaja).

- S -

Script '.sql': Archivo ejecutable que contiene instrucciones enfocadas a bases de datos relacionales, como *PostgreSQL* o *MySQL*.

Socio: Forman parte de la asociación. A nivel del aplicativo solo pueden visualizar y apuntarse a eventos. También puede modificar sus datos personales (enfermedades asociadas incluidas) y familiares y personas de contacto.

- V -

VBA Excel (macros): *VBA* son las siglas de "*Visual Basic for Application*", un lenguaje de programación disponible para los usuarios de *Microsoft Office* en programas como *Excel*. *VBA* se desarrolló en los años noventa para unificar los distintos lenguajes de macros de cada uno de los programas.

- X -

xlsx: Extensión característica que hace referencia a que el archivo ha sido generado con el programa Microsoft Excel.

2. Modelo de dominio

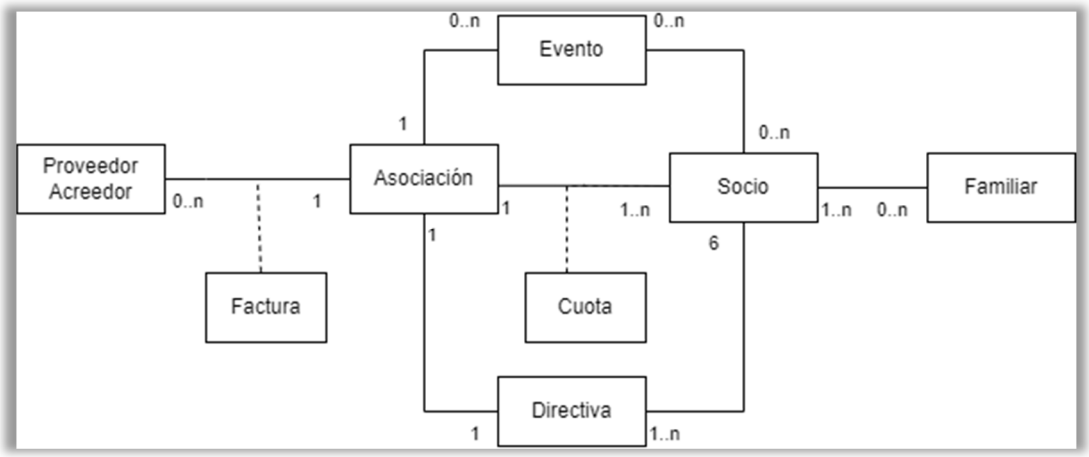


Figura 31: Anexo II. Modelo de dominio

3. Diagrama de contexto

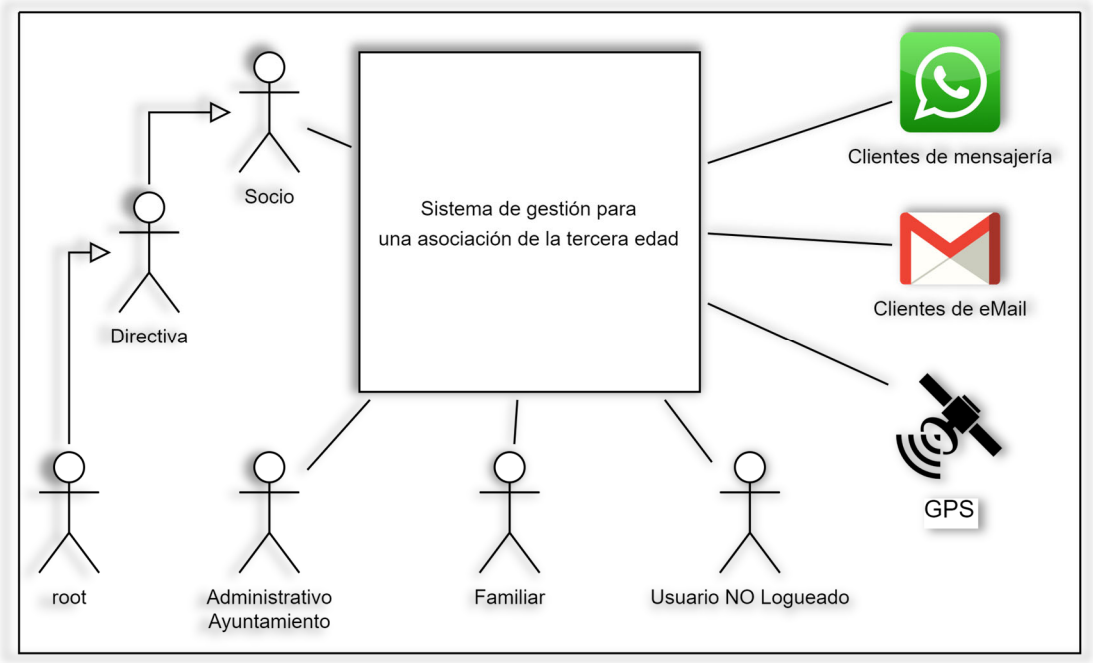


Figura 32: Anexo II. Diagrama de contexto

4. Diagrama de casos de uso

4.1. Esquema detalle por cada actor

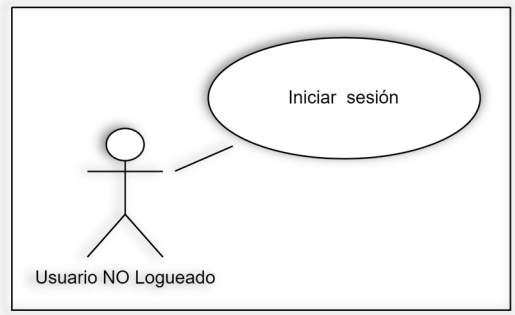


Figura 33: Anexo II. Diagrama de casos de uso. Usuario No Logueado

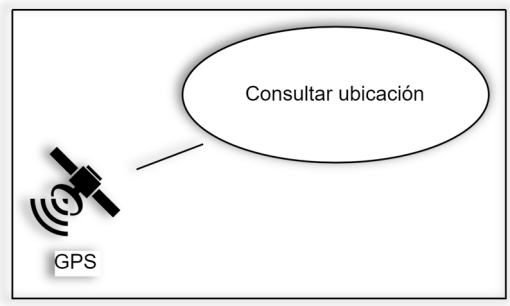


Figura 34: Anexo II. Diagrama de casos de uso. GPS

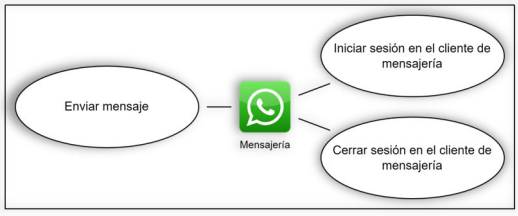


Figura 35: Anexo II. Diagrama de casos de uso. Cliente mensajería

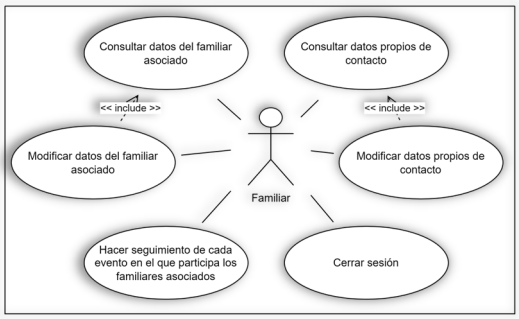


Figura 36: Anexo II. Diagrama de casos de uso. Familiar

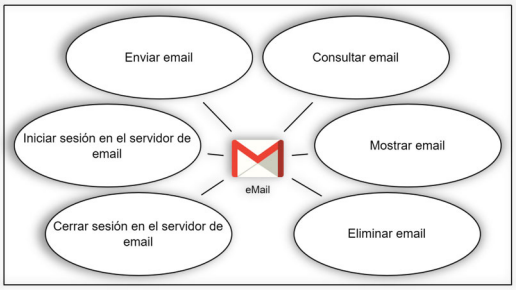


Figura 37: Anexo II. Diagrama de casos de uso. Cliente eMail

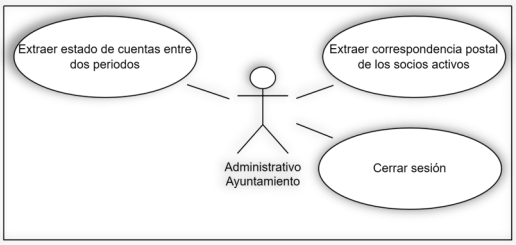


Figura 38: Anexo II. Diagrama de casos de uso. Administrativo del ayuntamiento

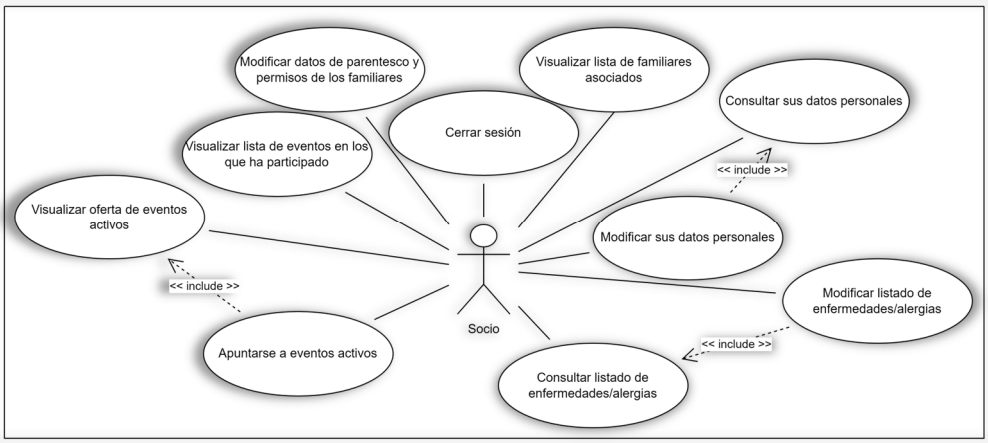


Figura 39: Anexo II. Diagrama de casos de uso. Socio

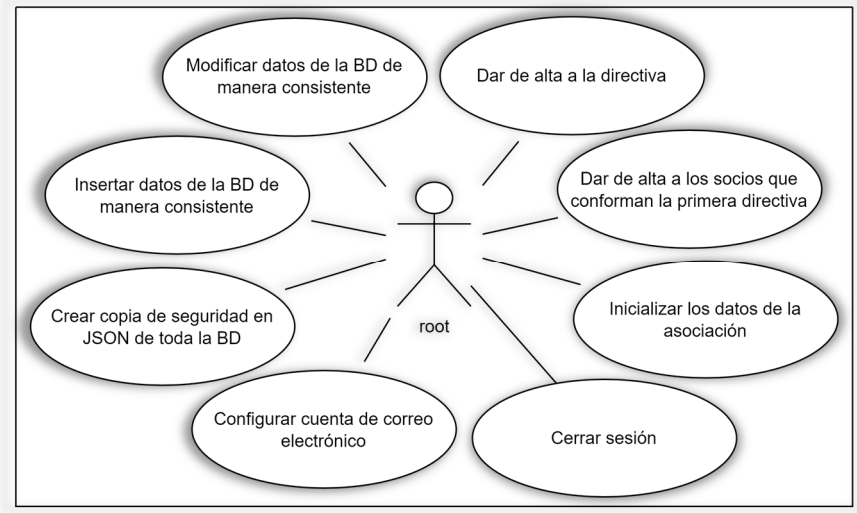


Figura 40: Anexo II. Diagrama de casos de uso. Root

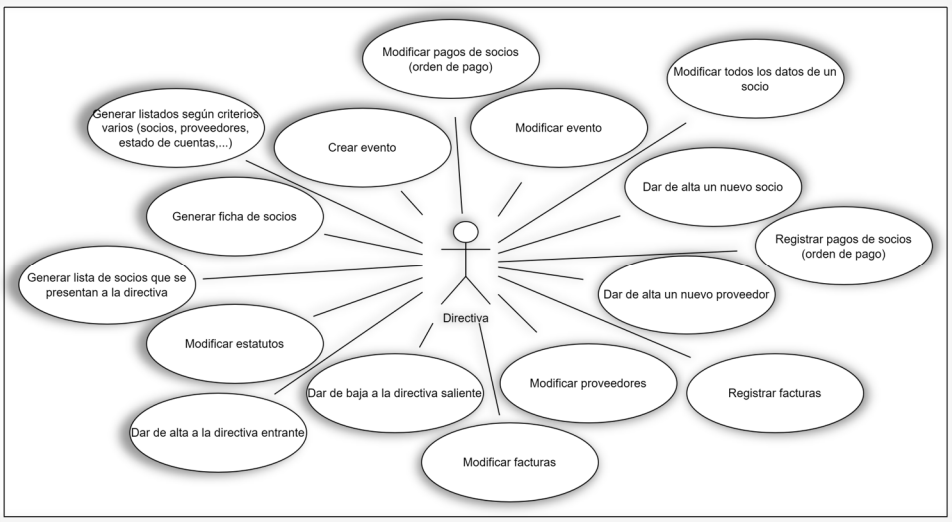


Figura 41: Anexo II. Diagrama de casos de uso. Directiva

4.2. Esquema global

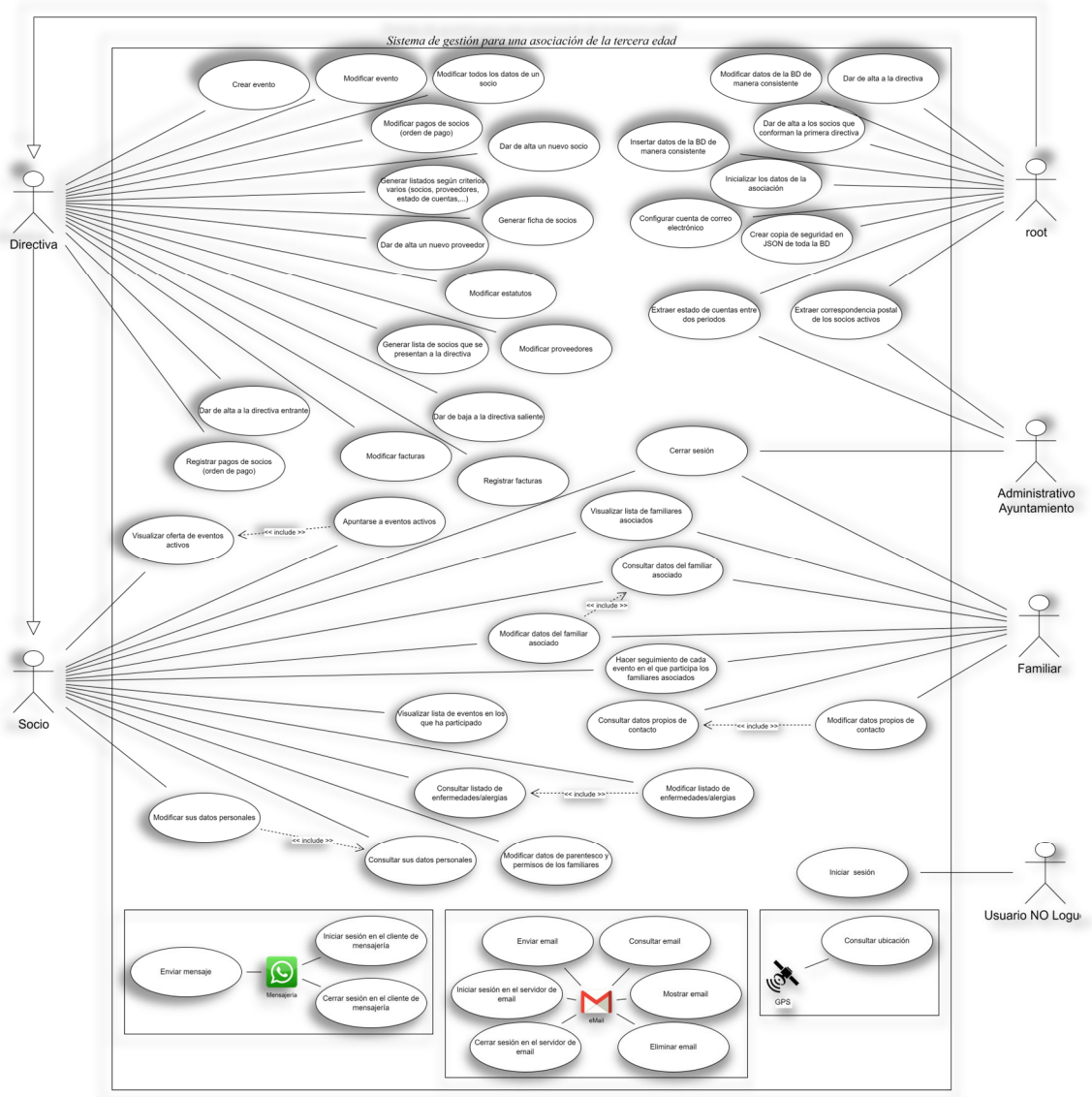


Figura 42: Anexo II. Diagrama de casos de uso.

4.3. Detalle de casos de uso

Para la codificación de los casos de uso y de los prototipos se ha decidido seguir la siguiente codificación XX_YY_ZZZ, donde cada parte puede tomar los siguientes valores:

- XX:
  - *CU: Casos de uso.*
  - *MC: Mockups.*
- YY:
  - *AA: Administrativo del ayuntamiento.*
  - *CO: Comunes:*
  - *DI: Integrantes de la directiva.*
  - *EM: Clientes de correo electrónico.*
  - *FA: Familiares de los socios.*
  - *NL: Usuario NO Logueado.*
  - *RO: Usuario Root.*
  - *SO: Socios.*
- ZZZ: Numeración secuencial que comienza en 1 con cada combinación de las anteriores.

Caso de uso		**Id**
Inicio de sesión		*CU_NL_001*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Usuario NO Logueado
2.	Tipo de asociación	
3.	Breve descripción	Cualquier usuario del sistema debe registrarse antes de poder hacer loguearse y hacer uso de la aplicación.
4.	Precondición	Los usuarios deben tener una cuenta para poder usar la aplicación.
5.	Flujo de eventos	1. El usuario accede al enlace de inicio de sesión. 2. El usuario introduce el *nick* y el *password* asociados a su cuenta. 3. El usuario cliquea sobre el botón de acceso. 4. El sistema verifica la veracidad de los datos introducidos, en caso de ser correctos, el usuario accede a la página asociada a sus datos y a su rol.
6.	Postcondición	El usuario ha iniciado sesión y ha accedido a su panel, de acuerdo con el rol asociado.
7.	Observaciones	
8.	Boceto relacionado	Figura 43: Anexo II. Mockup: Inicio de sesión

Tabla 5: Anexo II. CU_NL_001, Inicio de sesión

Caso de uso		Id
Cerrar Sesión		*CU_CO_001*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Cualquier usuario registrado.
2.	Tipo de asociación	
3.	Breve descripción	Desde cualquier lado de la aplicación, el usuario podrá cerrar la sesión en de la aplicación
4.	Precondición	El usuario debe estar logueado
5.	Flujo de eventos	1. Presionar el botón de "cerrar sesión"
6.	Postcondición	Volver a la pantalla de inicio de la aplicación
7.	Observaciones	
8.	Boceto relacionado	TODOS

Tabla 6: Anexo II. CU_CO_001, Cerrar sesión

Caso de uso		Id
Dar de alta nuevo socio		*CU_DI_001*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	Introducir a un nuevo socio en el sistema.
4.	Precondición	
5.	Flujo de eventos	1. Seleccionar si el socio ya ha estado dado de alta anteriormente en la asociación (para poder recuperar los datos que todavía pudieran quedar de él), en caso afirmativo, será necesario buscarlo por su antiguo número de socio o por su DNI. 2. Introducir / Revisar los datos personales del socio. 3. Introducir / Revisar los datos relativos a la asociación del socio. 4. Introducir / Revisar los datos de contacto del socio, incluido añadir, eliminar o modificar los datos de los familiares asignados. 5. Introducir / Revisar la foto del socio. 6. Introducir / Revisar estado de pagos. 7. Introducir / Revisar observaciones. 8. Introducir / Revisar dolencias del socio. 9. Aceptar el alta nueva. 10. Generar el documento pdf para que firme en papel el tratamiento informatizado de los datos (imprimir dos copias, uno con cabecera 'para socio' otro con cabecera 'para Asociación'
6.	Postcondición	Deberá aparecer un mensaje de que los datos se han insertado correctamente o si se ha producido un error.

		Deberá preguntar si se quiere volver a la pantalla principal de la directiva o insertar un nuevo socio
7.	Observaciones	
8.	Boceto relacionado	Figura 47: Anexo II. Mockup: Registrar nuevo socio, parte I Figura 48: Anexo II. Mockup: Registrar nuevo socio, parte II Figura 49: Anexo II. Mockup: Registrar nuevo socio, parte III

Tabla 7: Anexo II. CU_DI_001, Dar de alta a un nuevo socio

Caso de uso		**Id**
Modificar datos de un socio		*CU_DI_002*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	El formulario es el mismo que el de *CU_DI_001*
3.	Breve descripción	Modificar todos los datos de un socio.
4.	Precondición	El socio debe haber sido dado de alta con previamente
5.	Flujo de eventos	1. Introducir los datos del socio mediante su número de socio, mediante su DNI o mediante su nombre y apellidos (en este último caso, si se repite algún nombre, deberá aparecer una mini vista con los datos más relevantes de todos los coincidentes (nombre completo, domicilio y fotografía). 2. Introducir / Revisar los datos personales del socio. 3. Introducir / Revisar los datos relativos a la asociación del socio. 4. Introducir / Revisar los datos de contacto del socio, incluido añadir, eliminar o modificar los datos de los familiares asignados. 5. Introducir / Revisar la foto del socio. 6. Introducir / Revisar dolencias del socio. 7. Introducir / Revisar el documento firmado de protección de datos. 8. Introducir / Revisar estado de pagos. 9. Introducir / Revisar observaciones. 10. Aceptar los cambios (si lo hubiere) o salir sin guardar.
6.	Postcondición	Deberá aparecer un mensaje de si los datos se han actualizado correctamente o si se ha producido un error. Deberá preguntar si se quiere volver a la pantalla principal de la directiva o modificar los datos de otro socio
7.	Observaciones	
8.	Boceto relacionado	Figura 46: Anexo II. Mockup: Panel de socio

Tabla 8: Anexo II. CU_DI_002, Modificar datos de un socio

Caso de uso		Id
Insertar pagos de socio, masivamente		*CU_DI_003*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	Permite introducir rápidamente, órdenes de pago a varios socios de una vez.
4.	Precondición	Los socios tienen que estar dados de alta.
5.	Flujo de eventos	1. En el panel principal, seleccionar la opción 'Ordenes de pago masivas' 2. Seleccionar del desplegable (donde se mostrará la imagen, numero de socio y los nombres y apellidos) todos los socios a los que queremos asignar una orden de pago. 3. Asignar a cada socio el número de orden de pago, la cuota a ingresar y la fecha fin a la que corresponde ese pago (normalmente hasta el 31/12 del año en curso). 4. Aceptar.
6.	Postcondición	Deberá aparecer un mensaje de si los datos se han actualizado correctamente o si se ha producido un error. El sistema se redirigirá a la pantalla principal, si todo ha ido bien. Si ha ido mal, no se deberá de borrar los datos ya insertados.
7.	Observaciones	
8.	Boceto relacionado	MC_DI_002

Tabla 9: Anexo II. CU_DI_003, Insertar pagos de socio, masivamente

Caso de uso		Id
CRUD Evento		*CU_DI_004*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	Vista de creación, modificación y eliminación de eventos (o actividades) que organiza la asociación.
4.	Precondición	
5.	Flujo de eventos	1. En el panel principal, clicar sobre el botón correspondiente. 2. Insertar / Modificar la descripción del evento. 3. Insertar / Modificar los datos de la ubicación. 4. Insertar / Modificar las fechas, precio y cantidad máxima de socios. 5. Publicar el evento o guardar para modificarlo posteriormente. 6. Guardar para salir o salir sin guardar
6.	Postcondición	Deberá aparecer un mensaje de si los datos se han actualizado correctamente o si se ha producido un error.

		El sistema se redirigirá a la pantalla principal, si todo ha ido bien. Si ha ido mal, no se deberá de borrar los datos ya insertados.
7.	Observaciones	
8.	Boceto relacionado	Figura 50: Anexo II. Mockup: Registrar nuevo evento

Tabla 10: Anexo II. CU_DI_004, CRUD Evento

Caso de uso		**Id**
CRUD Directiva		*CU_DI_005*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	Se trata de dar de alta a la directiva que se encargará de gestionar el club durante los 4 años de mandato y de dar de baja a la directiva que acaba dicho mandato. Solo puede haber una directiva activa. Solo puede haber una directiva entrante. Solo puede haber una directiva saliente por lo que el día de alta de una debe
4.	Precondición	Los socios entrantes deben estar al día con el pago de las cuotas para poder ser elegidos miembros de la nueva directiva. Un socio puede desempeñar varios cargos.
5.	Flujo de eventos	1. En el panel principal, seleccionar el botón correspondiente. 2. Seleccionar a los socios a partir del desplegable (solo aparecerán los socios que estén al día del pago de cuotas), y se emparejará con el cargo que desempeñará. 3. Seleccionar la fecha en la que pasará como activa (un día antes se pondrá automáticamente como fecha fin de la actual).
6.	Postcondición	Deberá aparecer un mensaje de si los datos se han actualizado correctamente o si se ha producido un error. El sistema se redirigirá a la pantalla principal, si todo ha ido bien. Si ha ido mal, no se deberá de borrar los datos ya insertados.
7.	Observaciones	Los socios de la directiva saliente perderán el rol ‘DIR’ y recuperarán el rol ‘SOC’ por lo que dejarán de tener los privilegios de edición. En cambio, los socios de la directiva entrante promocionarán del rol ‘SOC’ al ‘DIR’ para poder tener los permisos adecuados de edición.
8.	Boceto relacionado	MC_DI_004

Tabla 11: Anexo II. CU_DI_005, CRUD Directiva

Caso de uso		Id
CRUD Proveedor		*CU_DI_006*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	Dar de alta, modificar o eliminar un proveedor.
4.	Precondición	Un proveedor no puede ser eliminado si tiene facturas asignadas
5.	Flujo de eventos	1. En el panel principal, seleccionar el botón correspondiente. 2. Introducir los datos fiscales y de contacto del proveedor.
6.	Postcondición	Deberá aparecer un mensaje de si los datos se han actualizado correctamente o si se ha producido un error. El sistema se redirigirá a la pantalla principal, si todo ha ido bien. Si ha ido mal, no se deberá de borrar los datos ya insertados.
7.	Observaciones	
8.	Boceto relacionado	MC_DI_005

Tabla 12: Anexo II. CU_DI_006, CRUD Proveedor

Caso de uso		Id
CRUD Factura		*CU_DI_007*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	Dar de alta, modificar o eliminar una factura.
4.	Precondición	El proveedor debe existir.
5.	Flujo de eventos	1. En el panel principal, seleccionar el botón correspondiente. 2. Elegir el proveedor. 3. Introducir los datos de la factura.
6.	Postcondición	Deberá aparecer un mensaje de si los datos se han actualizado correctamente o si se ha producido un error. El sistema se redirigirá a la pantalla principal, si todo ha ido bien. Si ha ido mal, no se deberá de borrar los datos ya insertados.
7.	Observaciones	
8.	Boceto relacionado	MC_DI_006

Tabla 13: Anexo II. CU_DI_007, CRUD Factura

Caso de uso		Id
CRUD Estatutos		*CU_DI_008*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	La directiva gestiona todas las actualizaciones de los estatutos de la asociación, por lo que estos deben quedar expuestos para que puedan ser consultados por los socios. Para ello, la directiva debe 'publicarlos' en la aplicación.
4.	Precondición	
5.	Flujo de eventos	1. En el panel principal, seleccionar el botón correspondiente. 2. Seleccionar el documento pdf con los estatutos.
6.	Postcondición	Deberá aparecer un mensaje de si los datos se han actualizado correctamente o si se ha producido un error. El sistema se redirigirá a la pantalla principal, si todo ha ido bien. Si ha ido mal, no se deberá de borrar los datos ya insertados.
7.	Observaciones	
8.	Boceto relacionado	MC_DI_007

Tabla 14: Anexo II. CU_DI_008, CRUD Estatutos

Caso de uso		Id
Generar ficha del socio		*CU_DI_009*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Directiva
2.	Tipo de asociación	
3.	Breve descripción	Aunque este documento tiene los días contados gracias al uso de la aplicación. La directiva desea poder generar (e imprimir posteriormente) una ficha (igual que la que poseen actualmente) con los datos del socio seleccionado
4.	Precondición	
5.	Flujo de eventos	1. En el panel principal, seleccionar el botón correspondiente. 2. Seleccionar del desplegable (donde se mostrará la imagen, numero de socio y los nombres y apellidos) todos los socios a de los que queremos generar la ficha.
6.	Postcondición	El sistema se redirigirá a la pantalla principal, si todo ha ido bien. Si ha ido mal, no se deberá de borrar los datos ya insertados.
7.	Observaciones	
8.	Boceto relacionado	Figura 45: Anexo II. Mockup: Gestor de socios

Tabla 15: Anexo II. CU_DI_009, Generar ficha del socio

Caso de uso		Id
CRUD email		*CU_EM_001*
N.º	***Nombre del campo***	***Descripción***
1.	Actor	Cliente de correo electrónico
2.	Tipo de asociación	
3.	Breve descripción	Añade funcionalidad para poder configurar e integrar la cuenta de correo electrónico oficial de la asociación en la aplicación. De esta manera, al tenerla embebida en el mismo sistema, facilita la recepción, lectura y envío de correos electrónicos, sin necesidad de tener abierto otra página del navegador web.
4.	Precondición	
5.	Flujo de eventos	1. En el panel principal, seleccionar el botón correspondiente.
6.	Postcondición	El sistema se redirigirá a la pantalla de gestión de correo electrónico.
7.	Observaciones	
8.	Boceto relacionado	Figura 44: Anexo II. Mockup: Correo

Tabla 16: Anexo II. CU_EM_001, CRUD email

4.4. Mockups

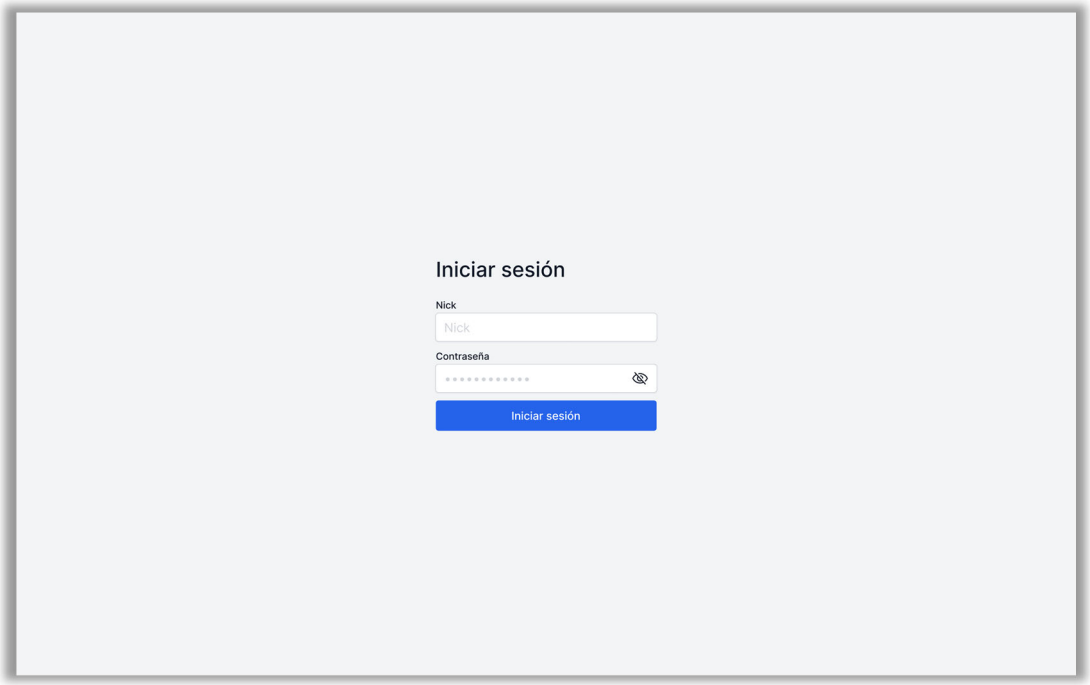


Figura 43: Anexo II. Mockup: Inicio de sesión

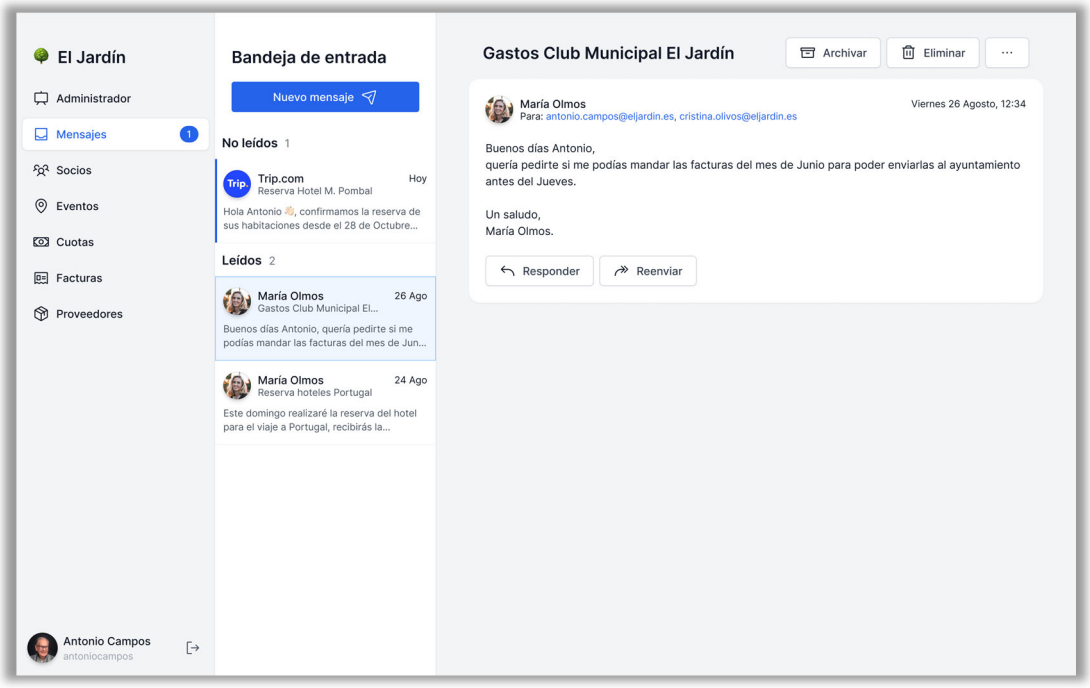


Figura 44: Anexo II. Mockup: Correo electrónico

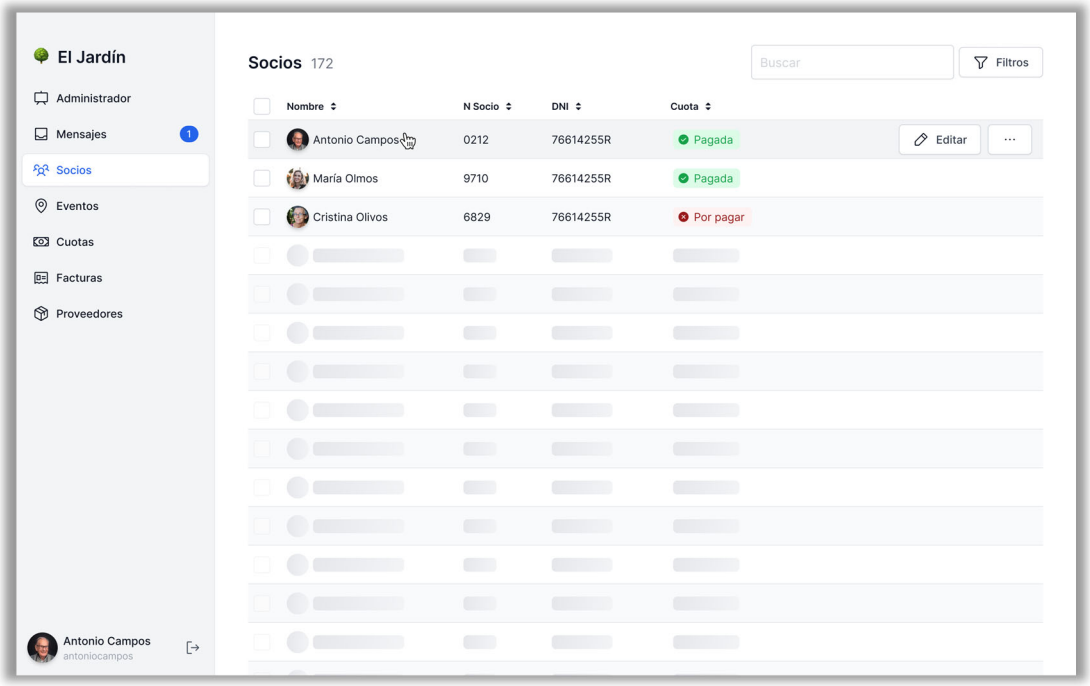


Figura 45: Anexo II. Mockup: Gestor de socios

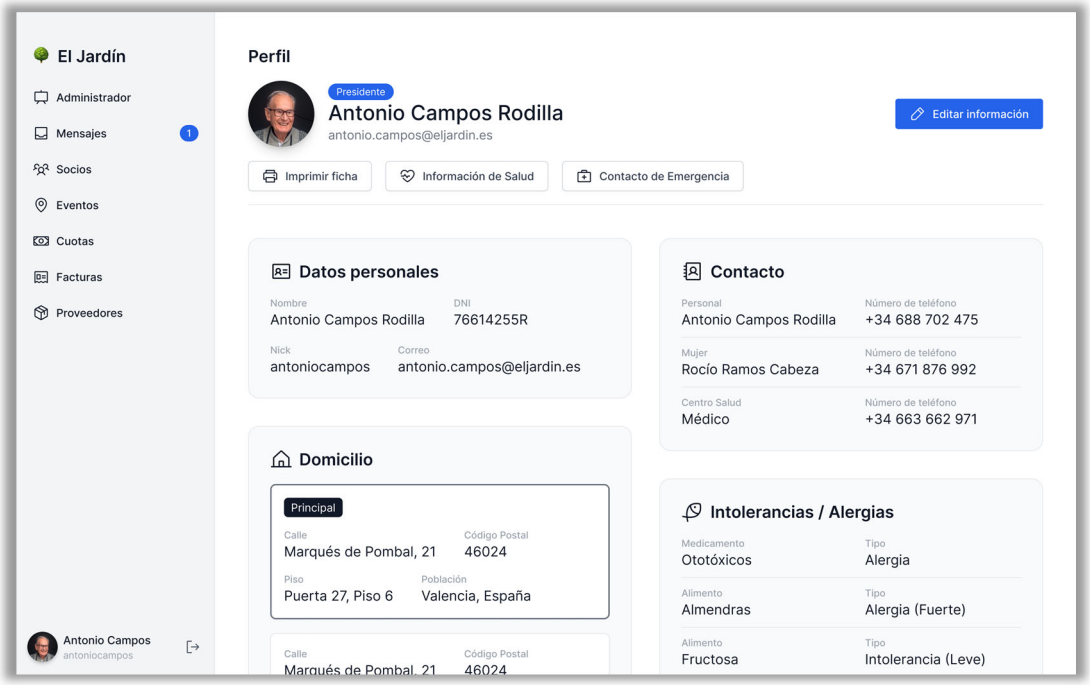


Figura 46: Anexo II. Mockup: Panel de socio

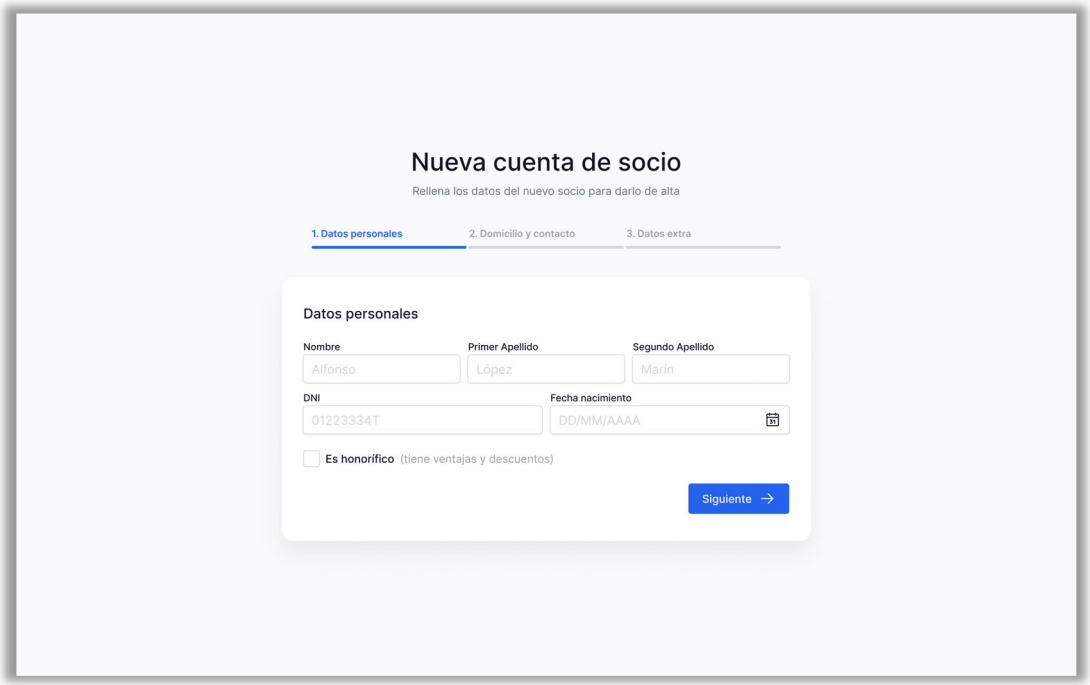


Figura 47: Anexo II. Mockup: Registrar nuevo socio, parte I

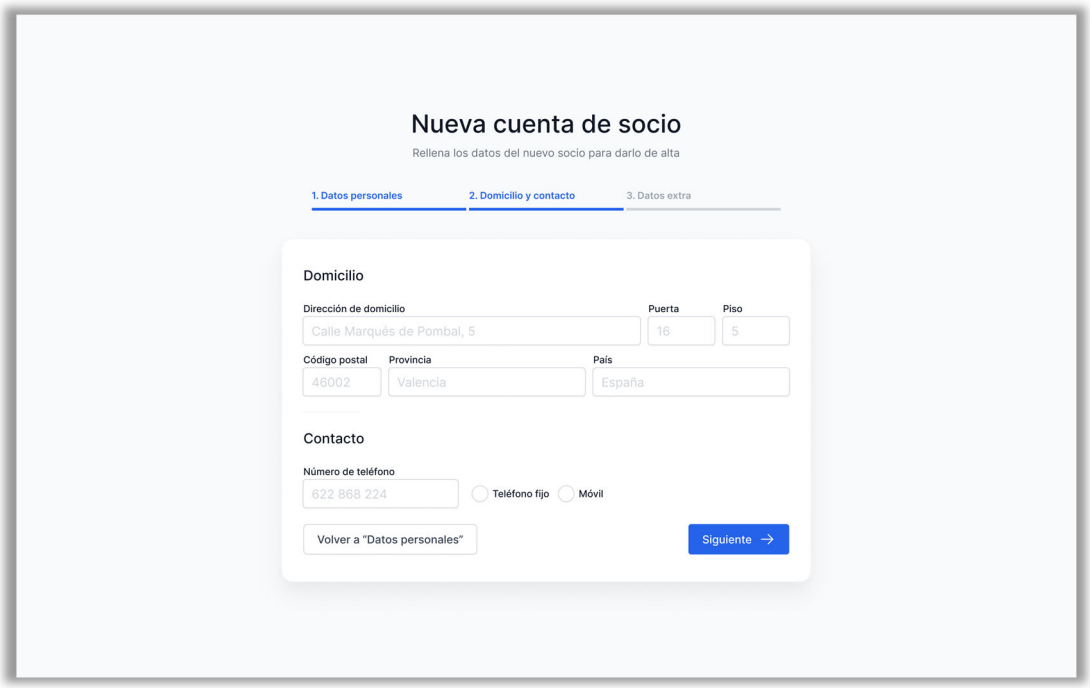


Figura 48: Anexo II. Mockup: Registrar nuevo socio, parte II

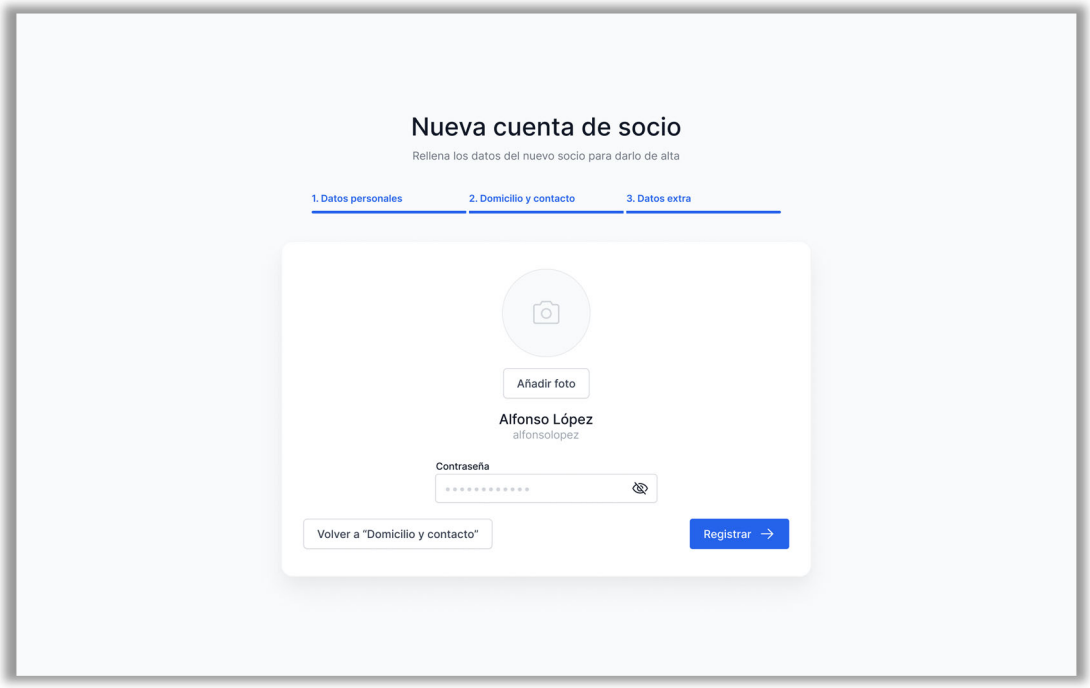


Figura 49: Anexo II. Mockup: Registrar nuevo socio, parte III

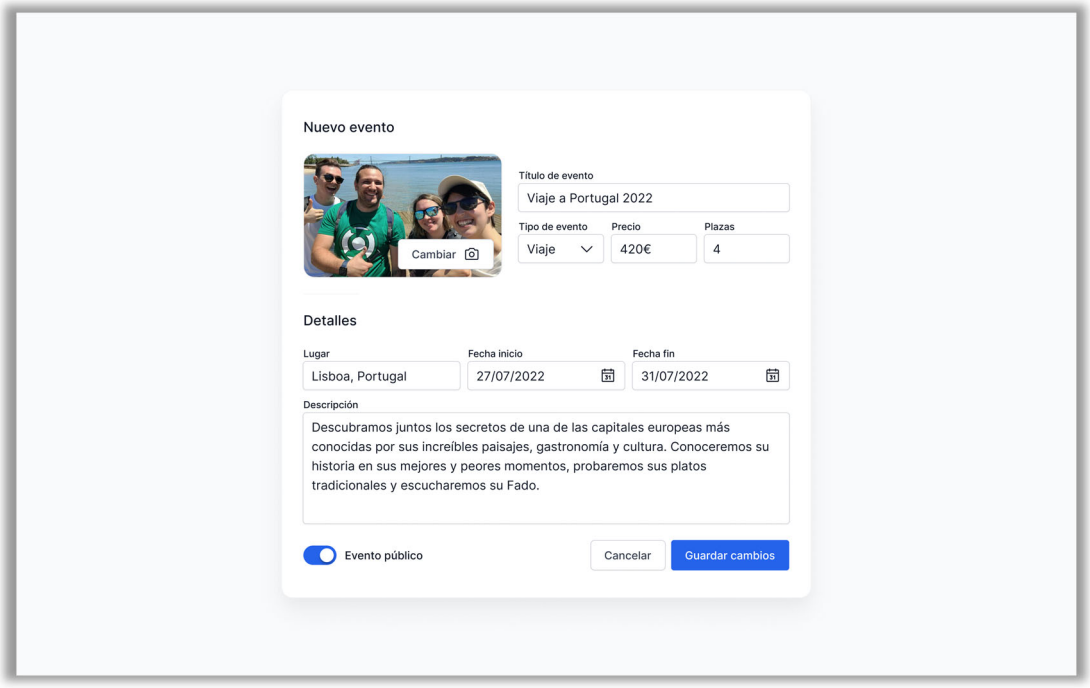


Figura 50: Anexo II. Mockup: Registrar nuevo evento

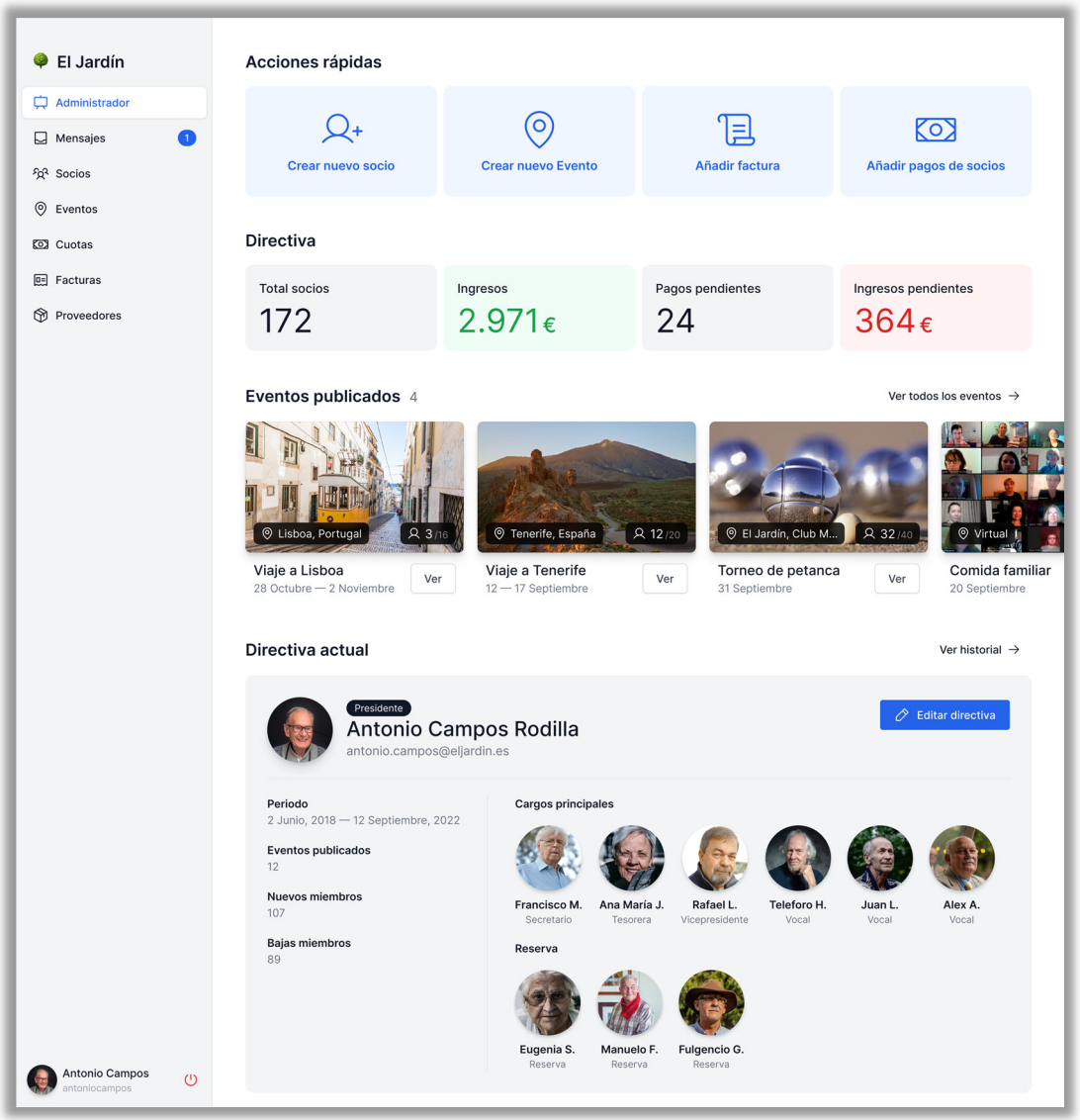


Figura 51: Anexo II. Mockup: Página principal

5. Diagrama de clases

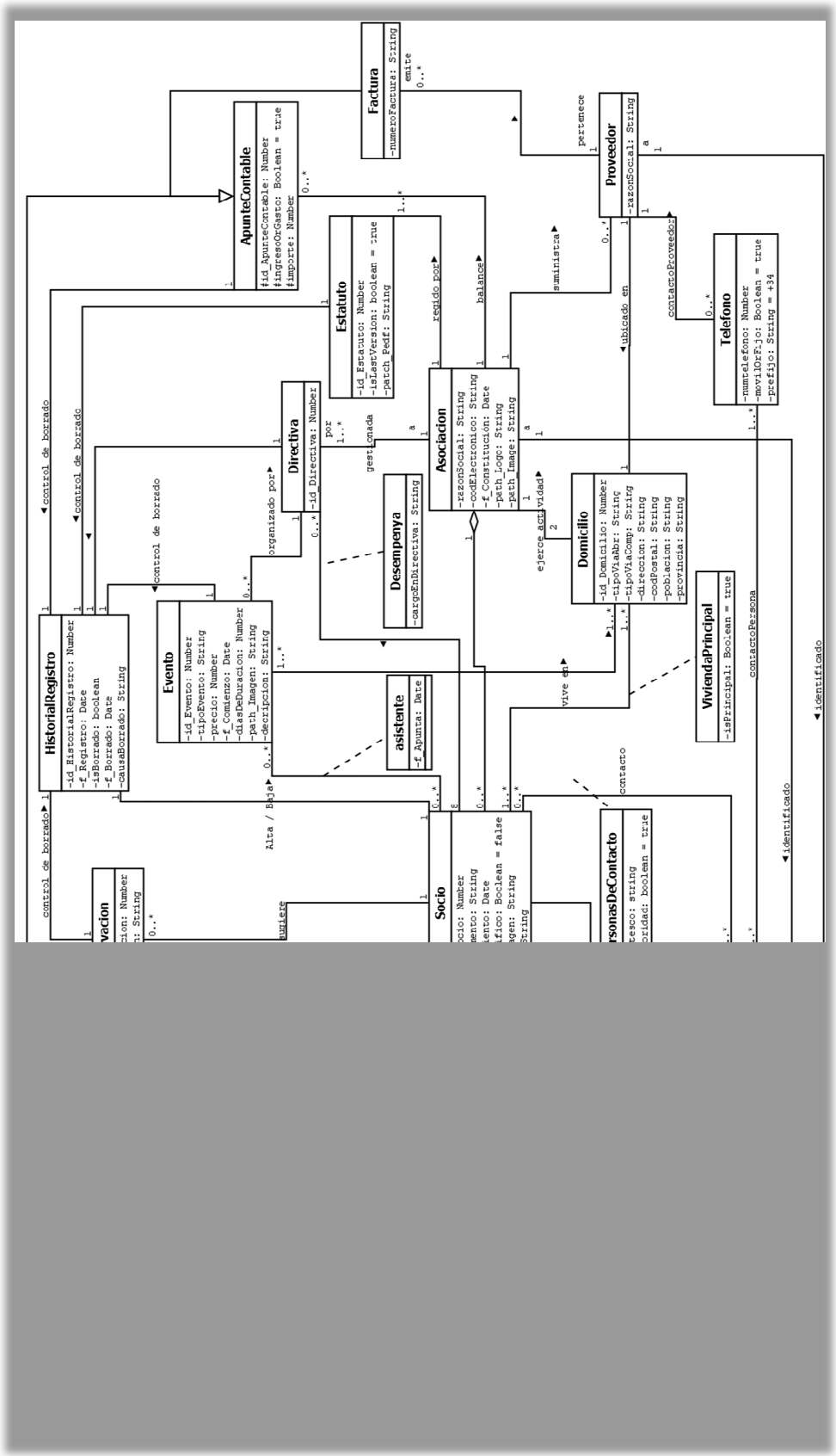


Figura 52: Anexo II. Diagrama de clases