Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Informática

Portal web para consulta de información y trámites de socios de asociaciones sin ánimo de lucro

Trabajo fin de estudio presentado por:	José Antonio Otero Martín
Director/a:	Pablo Ortiz García
Fecha:	Mayo 2024
Repositorio del código fuente:	https://github.com/JossTero/UNIR-GII-TFG-Portal-Socio.git
Vídeos de demostración:	Funcional: https://youtu.be/JXoG5r9Jnd8 Técnica: https://youtu.be/GOFxKuEdisM

Resumen

El objetivo de este proyecto es especificar, diseñar, desarrollar y validar un portal web para que los socios de asociaciones sin ánimo de lucro estén informados de las noticias y eventos de interés que organiza la asociación y les permita realizar ciertas gestiones y ciertos trámites como la justificación del pago de una cuota de socio.

Tras realizar un análisis previo del estado de las soluciones informáticas orientadas a las organizaciones del tercer sector, y observar que no existe consenso sobre una solución genérica que ayude a las asociaciones sin ánimo de lucro a poner a disposición de sus socios una herramienta que les permita interactuar con la asociación y ejercer ciertos derechos y obligaciones que marcan los estatutos de las organizaciones.

El portal web cuenta con un sistema de autenticación para que sólo usuarios registrados en la asociación puedan acceder, enfocando actualmente a los usuarios de tipo socio, y con funcionalidades de consulta de perfil del socio, noticias publicadas por la asociación, consulta y justificación de pago de cuotas de socio y consulta y descarga de documentos del socio. También permite consultar la información de la asociación y el órgano de gobierno actual.

Para llevar a cabo el desarrollo del sistema, se ha utilizado una metodología de desarrollo basada en el modelo de desarrollo en espiral, realizando el proyecto de desarrollo en distintas etapas, permitiendo general la versión inicial del producto.

Tras la especificación de las funcionalidades del sistema a incluir en la primera versión, realizar el diseño del producto, implementar la solución y probar su funcionamiento, se obtiene la primera entrega del portal web.

Se concluye que la aplicación cumple el objetivo marcado en el trabajo permitiendo a socios de asociaciones estar informados y realizar el trámite del pago de una cuota de socio.

Palabras clave: Portal web del socio, asociaciones sin ánimo de lucro, economía social, entidades sin ánimo de lucro, tercer sector, desarrollo web, *MERN Stack*, *MongoDB*, *Express*, *React*, *Node.js*, tramitación de pago cuota de socio, código abierto.

Abstract

The objective of this project is to specify, design, develop and validate a web portal so that members of non-profit associations are informed of the news and events of interest organized by the association and allows them to carry out certain procedures and certain procedures such as justification of payment of a membership fee.

After carrying out a prior analysis of the state of IT solutions aimed at third sector organizations, and observing that there is no consensus on a generic solution that helps non-profit associations to make a tool available to their partners that allows them to interact with the association and exercise certain rights and obligations established by the organizations' statutes.

The web portal has an authentication system so that only users registered in the association can access it, currently focusing on member-type users, and with functions to consult the member's profile, news published by the association, consultation and payment justification of membership fees and consultation and downloading of member documents. It also allows you to consult information about the association and the current governing body.

To carry out the development of the system, a development methodology based on the spiral development model has been used, carrying out the development project in different stages, allowing the initial version of the product to be generalized.

After specifying the system functionalities to be included in the first version, designing the product, implementing the solution and testing its operation, the first delivery of the web portal is obtained.

It is concluded that the application meets the objective set in the work, allowing association members to be informed and carry out the process of paying a membership fee.

Keywords: Partner web portal, non-profit associations, social economy, non-profit entities, third sector, web development, *MERN Stack*, *MongoDB*, *Express*, *React*, *Node.js*, membership fee payment processing, open source.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

El presente trabajo **se enfoca en las asociaciones sin ánimo de lucros**, siendo una parte de las entidades privadas que pertenecen al Tercer sector las cuales no tienen fines lucrativos. Estas entidades tienen diferentes objetos sociales, pero con un factor común y es que están gestionadas por y para sus socios. En este contexto, es imprescindible contar con soluciones informáticas que permitan a los socios interactuar con la asociación para realizar ciertas gestiones.

Partiendo de que **las organizaciones sin ánimo de lucro tienen la figura del socio** y que estos al inscribirse en una asociación **adquieren una serie de derechos**, como el de ser informado, **y tienen una serie de obligaciones**, como el pagar una cuota si así se requiere, siendo normas de obligado cumplimiento al estar establecidas en los Estatutos de la asociación, son estos derechos y obligaciones una de las bases de este trabajo.

Con esta situación, este Trabajo de Fin de Grado se centra en el desarrollo de un portal web cuyo propósito principal es acercar a los socios a las asociaciones a las que están inscritos para poder realizar ciertas gestiones de manera sencilla y directa para cubrir algunos de los derechos y obligaciones que tienen los socios.

1.1. Motivación

Una de las principales motivaciones que ha llevado enfocar este proyecto de fin de grado a las asociaciones sin ánimo de lucro es ser miembro de una asociación cultural que fue fundada en 2021 y que actualmente tiene unos 150 socios, teniendo una capacidad económica limitada para realizar una inversión en soluciones informáticas que facilite la comunicación y el cumplimiento de los derechos y obligaciones de los socios y la asociación.

El análisis del estado del arte en el sector no lucrativo ha revelado una falta de consenso en el sector a nivel tecnológico, con una solución reutilizable que permita a las asociaciones sin ánimo de lucro acercar al socio a la asociación de una manera directa, sencilla y especializada.

La posibilidad de desarrollar una solución que cubra esta necesidad, que siga la línea de las soluciones actuales con una clara orientación web, pudiendo poner en práctica todo el conocimiento adquirido durante el grado a nivel de documentación académica y de ingeniería

de software, junto con el aprender nuevas tecnologías y herramientas hasta ahora desconocidas, es otra razón importante para realizar este proyecto.

1.2. Planteamiento del trabajo

Desde el punto de vista de las entidades no lucrativas, el socio es de las figuras más importantes en este tipo de entidades porque sin socios no existirían las asociaciones ya que es una fuente económica importante (Plus Contacto, 2022) y según lo investigado**, no hay consenso en el sector sobre solución única reutilizable** que ayude a las asociaciones sin ánimo de lucro a poner a disposición de sus socios una solución que permita a los socios estar informados y tramitar el pago de las cuotas, entre otras gestiones. Esta variedad de soluciones se puede ver en la tabla 4 del apartado de Análisis del estado del arte, al comparar algunas de las herramientas que hay para la administración de las asociaciones, existiendo algunas herramientas con una solución propia para cubrir lo expuesto y otras que no lo tienen cubierto.

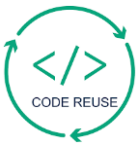
La propuesta es implementar una solución online que dé **solución a este tipo de necesidades a través de un portal web**, el cual se pueda integrar con las herramientas de gestión que tenga la asociación o mediante la carga de información a tiempo real, permitiendo al socio acceder al servicio desde cualquier dispositivo con internet y un navegador (ordenadores, móviles, tabletas, televisores inteligentes, etc.). Se parte de unas funcionalidades básicas como la de poder consultar la información del socio, poder consultar los órganos de gobierno y el órgano de representación de la asociación, poder consultar la documentación del socio y de la asociación, poder consultar y tramitar las cuotas como principales funcionalidades iniciales. **Esto les permitirá ejercer los derechos y deberes que le proporcionan los estatutos** de la asociación a la que está asociado desde cualquier lugar.

Al ser una solución tecnológica con una orientación de portal web, es importante tener manejar una de las definiciones de portal online que aclare por qué se ha dado este enfoque:

“Un portal es un sitio web que permite a un usuario acceder a diversos servicios, recursos, aplicaciones o posibilidades desde un mismo lugar.” (Bembibre, 2009)

Esta solución software tiene un enfoque de portal web para dar solución a la problemática de interacción del socio con la asociación, permitiendo ser reutilizado entre las distintas asociaciones, independientemente del software de gestión que utilicen. Este concepto de reutilización es importante porque permite ahorrar tiempo y costes a las organizaciones al usar soluciones genéricas ya existentes que cubren una de las necesidades de las asociaciones frente a sus socios.

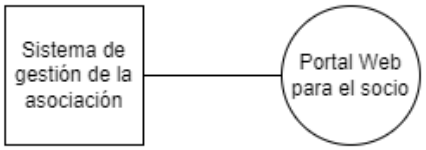
Figura 1. *Logo de reutilización de código.*



Fuente: Artículo Faster time-to-market (Seluxit, 2022)

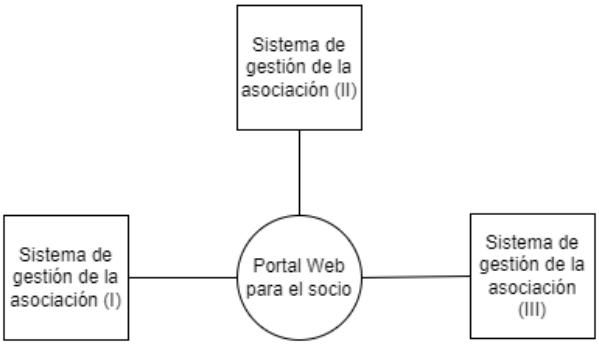
Esta propuesta permite varias orientaciones de puesta en marcha del sistema, la primera es que **puede ser integrada como una solución más dentro del sistema de información** de una asociación, como se aprecia en la figura 2, y la segunda es que **puede ser una solución en forma de servicio SaaS[i] en la que las asociaciones se conectan de forma externa** y mediante este sistema ofrecen el servicio comunicación y gestiones de socios evitando ser un sistema, interno tal como se muestra en la figura 3.

Figura 2. *Diagrama de contexto que representa el caso de un portal web integrado dentro del sistema de información de una asociación.*



Fuente: Elaboración propia

Figura 3. *Diagrama de contexto que representa el caso de un portal web en forma de servicio SaaS en el que varios sistemas de gestión consumen el servicio. Ejemplo con 3 sistemas de gestión.*



Fuente: Elaboración propia

Independientemente de la solución implementada, **cada asociación tendrá un dominio web que dará acceso al portal del socio para esa asociación** en concreto. Para acceder al portal, el socio se posicionará en la página principal de acceso al portal y tendrá que introducir los datos de acceso que le haya proporcionado la asociación y una vez identificado podrá poder realizar las acciones que le permita la asociación.

Aunque en el apartado de Tecnologías utilizadas se detalla más este aspecto**, la tecnología escogida** para el desarrollo completo del portal web es un ***MERN*** [ii] ***Stack***.

La metodología utilizada para realizar el desarrollo es el modelo de proceso evolutivo poque permite ir desarrollando y evolucionando el producto en distintas iteraciones trabajando en un ambiente con un nivel de incertidumbre controlado, pero que admite ir adaptándose a las necesidades de las organizaciones y usuarios de la aplicación. Este aspecto se detalla más en el apartado de Metodología de trabajo.

1.3. Estructura del trabajo

Para explicar el proceso que se ha seguido para realizar este proyecto, el trabajo se estructura en capítulos.

El capítulo 1 detalla el contexto y el estado del arte de las soluciones informáticas de organizaciones del tercer sector, contextualizando las asociaciones sin ánimo de lucro, categorizando las asociaciones a través una posible clasificación, se detalla la figura del socio y se describe el estado del arte de las diferentes soluciones para las asociaciones del cual se obtienen las conclusiones que impulsan este proyecto.

El capítulo 2 explica los objetivos de este trabajo y detalla la metodología utilizada para el desarrollo del portal web, que es lo que marca la estructura de los siguientes capítulos.

El capítulo 3 muestra la especificación del producto a implementar, partiendo de los requisitos funcionales, no funcionales, la definición del alcance de la primera versión, y se profundiza en las funciones del sistema a través de las historias de usuario y de los distintos casos de uso que se quieren tener en el primer paquete de funcionales del producto.

El capítulo 4 explica el diseño del producto realizado en función de la especificación realizada en el capítulo anterior, explicando la arquitectura del sistema y la estructura de datos utilizando distintas técnicas del proceso de ingeniería del software.

El capítulo 5 describe el proceso de implementación del portal web, explicando las tecnologías utilizadas y por qué se han utilizado, el modelo de datos del sistema y las distintas colecciones de datos y profundizando en el desarrollo de los distintos niveles del sistema, explicando la estructura y elementos programamos en la parte del servidor y en la parte del cliente.

En el capítulo 6 se explica las pruebas realizadas, realizando pruebas unitarias y pruebas funcionales y se muestra un ejemplo de cómo se han realizado las pruebas.

En el capítulo 7 se obtienen las conclusiones tras la realización del proyecto y se describe algunos de los posibles trabajos futuros que se pueden realizar en la aplicación, siendo alguno de ellos funcionalidades si indicadas en el alcance de la primera versión, pero no implementadas.

2. Contexto y Estado del Arte

En este apartado **se detallan los aspectos generales y tecnológicos de las asociaciones sin ánimo de lucro** y de la **figura del socio**. Aspectos que son la base y fundamentos del proyecto de ingeniería de desarrollado software descrito en este trabajo.

En el apartado relativo al Análisis de contexto se describe, desde una visión general, la actividad de las asociaciones sin ánimo de lucro, comenzando por explicar qué son las asociaciones sin ánimo de lucro, los tipos de asociaciones que hay junto con los objetivos que persiguen y la importancia del socio en la asociación siendo una parte a destacar del trabajo, al ser la figura a la que se le ha detectado ciertas carencias funcionales de gestión en un una parte importante de las soluciones de software de gestión de asociaciones analizadas.

En la parte del Estado del arte se explica la situación actual de las soluciones tecnológicas en el sector de las asociaciones sin ánimo de lucro, tanto desde el punto de vista de la administración de la asociación, como desde el punto del socio que pertenece a una asociación. El recorrido que se da en este documento sobre el estado del arte actual a nivel tecnológico pretende ayudar a comprender la problemática encontrada y servir como base de la solución propuesta.

2.1. Análisis de Contexto

Las asociaciones, como las organizaciones sin ánimo de lucro pueden pasar desapercibidas en la sociedad actual e incluso se puede ser socio de una de ellas sin haberse percatado de ello, simplemente por la inercia social. Al profundizar en los diferentes puntos de este apartado se podrá razonar esta afirmación con mayor claridad, sobre todo en el punto de Clasificación de las asociaciones según su objetivo.

Si se mira desde un prisma económico y social, **las asociaciones pertenecen al Tercer sector que es en el dedicado a realizar las acciones que hacen frente a las situaciones de apoyo social y de mejora, teniendo un fin NO lucrativo** (Fundación lealtad, 2022), ocupando un papel esencial en la sociedad actual porque realizan una labor social al promover actividades de ámbito cultural, deportivo, de ayuda social y humanitaria, medioambiental, educativo, entre otros e incentivando entre sus socios valores de desarrollo personal y social (Ayuda en

acción, 2020). Todo esto impulsado por el sentimiento de pertenencia a un grupo de personas que comparten un mismo objetivo.

Para visualizar la importancia que tiene las asociaciones en la sociedad, **el número de asociaciones sin ánimo de lucro** que se han inscrito en Registro Nacional de Asociaciones **desde su inicio como denominación legal** en 1965 hasta el 31/12/2022 **es de 61.864** (Anuario Estadístico del Ministerio del Interior, 2022), y **en el último ejercicio publicado se inscribieron 1.971** (Anuario Estadístico del Ministerio del Interior, 2022) tal y como se muestra la tabla de la figura 4.

Figura 4. *Altas en el RNA [iii]por grupo de actividad. Evolución 2005 a 2022.*

Año de inscripción	Grupo I. Ideológicas, culturales, educativas y de comunicación	Grupo II. Mujer, igualdad de trato y no discriminación	Grupo III. Infancia, jóvenes, personas mayores, familia y bienestar	Grupo IV. Medio ambiente y salud	Grupo V. Discapacidad y dependencia	Grupo VI. Víctimas, afectados y perjudicados	Grupo VII. Solidaridad, integración social, ayuda humanitaria y cooperación al desarrollo	Grupo VIII. Económicas, tecnológicas, de profesionales y de intereses	Grupo IX. Deportivas y recreativas	Grupo X. Varias	Total
1965-1969	224	4	14	18	11	1	17	232	67	93	**681**
1970-1979	380	9	33	67	31	8	37	229	73	67	**934**
1980-1989	2.100	38	123	386	125	42	136	648	248	358	**4.204**
1990-1999	4.914	145	373	914	215	198	798	1.848	981	580	**10.966**
2000	526	23	43	98	34	30	179	311	151	9	**1.404**
2001	513	25	38	108	43	37	194	323	138	13	**1.432**
2002	412	26	45	74	36	40	162	238	141	21	**1.195**
2003	467	21	42	86	40	26	198	322	143	27	**1.372**
2004	429	20	42	88	35	36	205	290	137	27	**1.309**
2005	578	24	56	104	43	47	259	326	195	12	**1.644**
2006	567	24	63	143	42	47	285	300	192	12	**1.675**
2007	518	38	60	118	21	28	296	268	174	24	**1.545**
2008	636	34	49	130	55	26	288	281	231	14	**1.744**
2009	753	41	55	157	40	52	379	340	233	20	**2.070**
2010	814	49	70	192	35	29	417	454	188	37	**2.285**
2011	851	45	89	196	40	30	325	378	249	24	**2.227**
2012	830	38	75	245	55	36	255	464	242	29	**2.269**
2013	1.035	27	82	331	58	50	264	553	250	27	**2.677**
2014	751	34	85	302	51	42	210	469	235	40	**2.219**
2015	634	20	38	274	55	45	188	425	211	11	**1.901**
2016	930	51	87	396	43	36	330	493	221	73	**2.660**
2017	944	69	107	381	34	42	248	456	242	280	**2.803**
2018	712	62	71	242	20	24	160	297	191	243	**2.022**
2019	769	45	83	322	18	19	169	309	166	299	**2.199**
2020	635	44	78	288	30	33	145	246	134	343	**1.976**
2021	728	59	108	308	45	12	180	316	167	557	**2.480**
2022	642	63	103	336	22	18	190	259	177	161	**1.971**
Total	**23.292**	**1.078**	**2.112**	**6.304**	**1.277**	**1.034**	**6.514**	**11.075**	**5.777**	**3.401**	**61.864**

Fuente: Anuario Estadístico del ejercicio 2022. (Ministerio del Interior, 2023)

De la columna de total de inscripciones anuales que se obtiene de la figura 4, se puede observar que, **en los últimos 14 años, el número de inscripciones anuales ronda las 2000 asociaciones**. Si se asume que no todas tendrán al comiendo de su actividad un sistema de gestión de asociaciones, pero sí podrán tener la necesidad de comunicación y trámites de los socios, **serán asociaciones candidatas a necesitar un sistema de comunicación y gestión de socios** desde cualquier sitio y es en este caso donde encaja la solución propuesta.

2.1.1. Las asociaciones sin ánimo de lucro

Para una comprensión más profunda de la estructura jurídica de la asociación y poder entender mejor el contexto de este proyecto, primero se tiene que definir el concepto de asociación:

"Una asociación sin ánimo de lucro es una **agrupación de personas** que se organizan para realizar una actividad colectiva. A diferencia de otras formas de organizarse y actuar, la asociación **goza de personalidad jurídica**, lo que la hace capaz de adquirir derechos y contraer obligaciones. Se establece así una **diferenciación entre el patrimonio de la asociación y el de las personas asociadas**." (Fundación Gestión y Participación Social, 2023)

Como características principales, para crear una asociación sin ánimo de lucro como entidad jurídica en España **debe haber un mínimo de 3 personas** las **cuales tienen que tener objetivos y/o actividades comunes**, con un **funcionamiento democrático**, teniendo **independencia de otras organizaciones** y como principal característica es el que **no buscan un fin lucrativo**, este último concepto es muy importante porque es la base de estas entidades, al tener un fin social y no lucrativo, aunque lo ideal es que generen beneficios para mantener la sostenibilidad de la asociación, teniendo que reinvertir estos beneficios en la actividad de la propia asociación.

Como parte de los requisitos para formalizar una asociación en el Registro Nacional de Asociaciones que pone a disposición del Ministerio del Interior del Gobierno de España para inscribir las asociaciones, se deben adjuntar los **Estatutos** que como se menciona en la publicación de Cuestiones básicas del funcionamiento asociativo publicado por el Ayuntamiento de Cáceres en 2018, **son las reglas fundamentales del funcionamiento de la asociación** delimitando los objetos y el modelo de gobierno y siendo vinculantes para los socios de forma voluntaria al ingresar en la asociación.

Para ejercer el base del funcionamiento democrático, se realizan a través de los **órganos de gobierno** y **del órgano de representación**.

El órgano de gobierno es donde se toman las decisiones de forma democrática sobre las acciones y actividades de la asociación, se realiza mediante **asamblea general la cual se realiza al menos una vez al año**, aunque existen asambleas extraordinarias que puede convocar la junta directiva para tratar temas con cierta urgencia. Como figura principal las asambleas a las están los socios que estarán presentes de manera presencial o telemática (según los medios

que disponga la asociación), y ella **se tratan temas de diferente índole** como la renovación de la junta, la aprobación de cuentas, modificación de Estatutos, actividades a realizar, etc., **los cuales son aprobados o no por los socios mediante un sistema de votación**, quedando patente en este tipo de actos el **poder soberano de los socios de la asociación**.

La asociación se gestiona a través del órgano de representación denominado normalmente como **junta directiva**, teniendo un periodo de vigencia y estando compuesta por socios con diferentes cargos dentro de la junta, como pueden ser el de presidente, vicepresidente, secretario, tesoro o vocal, asumiendo la **misión de gestionar y representar la asociación en actos oficiales en base a los Estatutos**. Esta junta se debe renovar al final del tiempo de vigencia estipulado en los estatus pudiendo seguir la actual por un nuevo periodo si así lo determinan los socios.

Este tipo de asociaciones no lucrativas tienen un **marco regulatorio específico**, teniendo **leyes de carácter estatal** y de **leyes de carácter autonómico** que regula cada autonomía, siendo un tipo de entidades reguladas y que necesitan una serie de formalizadas para su constitución y administración.

2.1.2. Clasificación de las asociaciones según su objetivo

Las asociaciones sin ánimo de lucro **se pueden clasificar según los objetivos que tenga y las necesidades que intentan satisfacer**. Según la fuente de información que se consulte, la clasificación de asociaciones varia algo en ciertos términos, en este documento se muestra una propuesta de clasificación en la tabla 1 (Ayudatpymes, 2024):

Tabla 1. *Clasificación de tipos de asociaciones según los objetivos y necesidades que tengan.*

Tipo de asociación	Objetivos
✓ **Asociaciones culturales**	Este tipo de asociaciones tienen como objetivo el **fomentar y promocionar la cultura y el arte** entre sus socios.
	Ejemplos: Asociaciones de literatura, cine, música, teatro, etc.

✓ **Asociaciones deportivas**	Este tipo de asociaciones tienen como objetivo el **promover la realización de actividades deportivas**.
	Ejemplos: Clubes deportivos, asociaciones de fútbol, tenis, natación, etc.
✓ **Asociaciones de ayuda social**	Este tipo de asociaciones tienen como objetivo el **ayudar a personas en situación de vulnerabilidad o necesidad.**
	Ejemplos: Asaciones de ayuda a personas mayores, de ayuda a personas con enfermedades raras, personas sin hogar, personas con discapacidad, etc.
✓ **Asociaciones medioambientales**	Este tipo de asociaciones tienen como objetivo el **promover la conservación y protección medioambiental**.
	Ejemplos: Asociaciones de conservación de la biodiversidad, de protección del medio ambiente, de protección de las aves, etc.
✓ **Asociaciones de voluntarios**	Este tipo de asociaciones tienen como objetivo el **fomentar el voluntariado enfocado a las actividades sociales**.
	Ejemplos: Asociaciones de voluntarios para ayuda en catástrofes, para situaciones de emergencia, ayuda humanitaria, etc.

✓ **Asociaciones educativas**	Este tipo de asociaciones tienen como objetivo **ser dedicadas a conceptos relacionados con la educación**.
	Ejemplos: Asociaciones de madres y padres, de profesores, de estudiantes, de profesionales, etc.
✓ **Otro tipo de asociaciones**	En este tipo de asociaciones irán las que **cuyos objetivos no están indicados en las anteriores**.
	Ejemplos: Asociaciones tecnológicas, de mujeres, de salud, políticas, contra la discriminación, etc.

Fuente: Elaboración propia basada en el artículo web Tipos de asociaciones sin ánimo de lucro con ejemplos reales. (Ayudatpymes, 2024)

Tras ver la clasificación anterior, puede que se confirme la afirmación realizada en el primer párrafo del apartado de Análisis de contexto de que una persona puede ser socio de una asociación sin haberse percatado de ello.

2.1.3. La importancia de la figura del socio en la asociación

Socio es un individuo de una sociedad, o agrupación de individuos (Real Academia Española, 2023) y sociedad como agrupación natural o pactada de personas, organizada para cooperar en la consecución de determinados fines (Real Academia Española, 2023), partiendo de estas definiciones se puede profundizar en el término de socio de una asociación.

Para que una persona se pueda consideran socio de una asociación sin ánimo de lucro se **debe inscribir** a través de una determinada forma y rellenado unos determinados datos, todo según dictamine la asociación, **aceptando la normativa** que tiene la asociación a través de los **Estatutos**, y en la mayoría de las ocasiones **irá acompañado del pago de una cuota** en el que la cuantía y la periodicidad de pago estará determinada por la asociación.

Esta inscripción, junto con el pago de una cuota si así se especifica en la asociación, **concederá al socio una serie de derechos y obligaciones dentro de la asociación** que vendrán determinado por los Estatutos de la asociación de forma concreta.

En la tabla 2 se describen de forma general, algunos ejemplos más destacados de derechos y obligaciones de socios dentro de las asaciones (Gâbilos Software, 2024):

Tabla 2. *Listado genérico de derechos y obligaciones de los socios en su relación con una asociación sin ánimo de lucro.*

Derechos
1. **Poder participar en los órganos de gobiernos y órganos de representación**, teniendo **voto** en los temas tratados.
2. **Ser informado** del órgano de representación, de la realización de órganos de gobierno y de otro tipo de eventos.
3. **Poder defenderse antes de aplicarse medidas disciplinarias** por las malas prácticas que se le acusan.
4. **Poder impugnar acuerdos** que se realicen en los distintos órganos de la asociación.
5. **Poder desvincularse de la asociación en cualquier momento** siguiendo las pautas establecidas por la asociación.
Obligaciones
1. **Tener que compartir la finalidad de la asociación** y participar en la consecución de las mismas.
2. **Pagar las cuotas, derramas y otras participaciones** según definido en los Estatutos.
3. **Cumplir el resto de obligaciones indicadas en los Estatutos**.
4. **Cumplir los acuerdos aceptados** por en los órganos de gobierno y por los órganos de representación.

Fuente: Elaboración propia basada en el artículo web de Derechos y deberes de los asociados. (Gâbilos Software, 2024)

En la lista anterior de derechos y obligaciones que tienen los socios es donde está una de las claves del presente proyecto, en la facilidad que dan las asociaciones para poder ejercer estos derechos y obligaciones por parte de los socios como marcan los Estatutos.

De forma general, las asociaciones sin ánimo de lucro tienen 2 tipos de socios, aunque pueden aparecer otros según se definan en los estatutos (Blog Planes de futuro MAPFRE, 2022):

Tabla 3. *Tipos de socios de asociaciones sin ánimo de lucro.*

Socios numerarios	Socios de honor
Son socios a los que **se les asigna un número de socio**, son iguales frente los derechos y las obligaciones que marcan los estatutos, pueden formar parte de la junta directiva y pueden estar sujetos al pago de una cuota lo que les convertiría en socio.	Son una categoría especial que se utiliza como **persona de influencia de la asociación** para realizar actos sociales o recaudar fondos. No suelen tener que pagar una cuota ni tener los mismos derechos y obligaciones, aunque esto estará indicado en los estatutos.

Fuente: Elaboración propia basada en el artículo web de Tipos de socios según el tipo de asociación. (Blog Planes de futuro MAPFRE, 2022)

2.2. Estado del Arte

Para este apartado, se lleva a cabo un estudio de mercado sobre las distintas herramientas en idioma español que tenga un fin similar al que tiene planteo cubrir el portal del socio presentado en este trabajo.

Aunque la problemática detallada en este proyecto puede ser compartida en otras áreas como con por ejemplo en la formación a través plataformas de e-learning al gestionar alumnos, matrículas, pagos y otra información. También en el área la financiación a través del *crowdfunding*, teniendo una gestión de pagos, este trabajo se centra de forma concreta en aportar una solución específica en los socios de las asociaciones sin ánimo de lucro, proporcionando un factor de especialización con lógica y datos de negocio propios de las asociaciones sin ánimo de lucro, aportando un valor añadido al no ser una solución genérica.

En la actualidad, **existen múltiples herramientas para las organizaciones** donde la mayoría están enfocadas en la propia administración de la organización, con funcionalidades como el control del inventario, la gestión económica de subvenciones, ingresos y gastos, la organización de actividades para los socios, comunicaciones, cuotas, etc.

La investigación de las soluciones propias del sector se ha centrado en conocer si de alguna forma cubren la necesidad de ofrecer una solución que permita al socio interactuar con la asociación, consultar información y realizar trámites.

2.2.1. Análisis del estado del arte

En este apartado se describe algunas de las herramientas tecnológicas existentes en el mercado para las organizaciones sin ánimo de lucro y que se han considerado analizado porque **son herramientas enfocadas a cubrir las distintas necesidades de las organizaciones del tercer sector**, **permitiendo conocer el estado actual de las capacidades que ofrecen las herramientas del sector**.

También se ha sido un factor clave que sean **soluciones promocionadas en internet, permitiendo conocer y consultar las características y funciones** que ofrecen gracias a la información que han publicados en sus respectivas páginas web.

Otro criterio que se ha utilizado es el idioma la solución, que sea una **plataforma en español** hace pensar que apuestan por el mercado de países hispanohablante como posibles áreas de mercado y lo que hace presuponer que tendrán **soporte en la región y el sistema adaptado a la legislación** del país donde se use.

Las soluciones tecnológicas analizadas son:

- ***Ekamat*** es un ERP[iv] basado en *Microsoft* para fundaciones, asociaciones y entidades sin ánimo de lucro.
    - Solución para la gestión de las organizaciones centrada principalmente en optimizar procesos financieros, gestión de proyectos, la relación con los socios, patronos y donantes.

Figura 5. *Solución ERP Ekamat para fundaciones, asociaciones y entidades sin ánimo de lucro.*

Fuente: Página Web https://cuatroochenta.com/microsoft-dynamics/erp-fundaciones-ong/

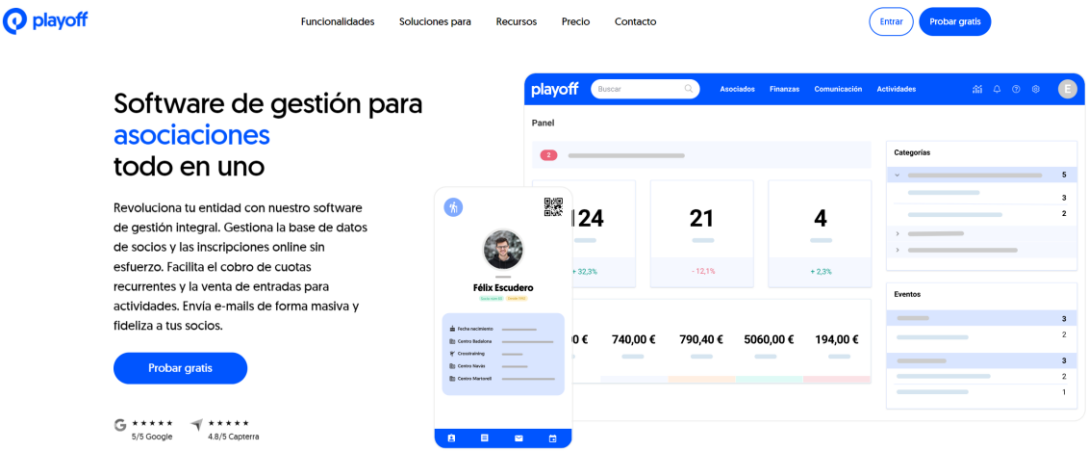
- ***Timtul*** es un software para asociaciones, partidos políticos, federaciones, fundaciones, colegios profesionales, gremios, clubs, organizaciones y comunidades.
    - Sistema para la gestión de miembros que te ayuda a gestionar la comunicación, organización y pagos con tus clientes, miembros o asociados.

Figura 6. *Solución Timtul para organizaciones del tercer sector.*

Fuente: Página Web https://timtul.com/

- ***Playoff*** es un software para clubes deportivos, asociaciones, academias y federaciones.
    - Software para la gestión integral de la organización, gestionando datos de los socios, facilita el cobro de las cuotas, gestión de actividades y otras funciones.

Figura 7. *Solución Playoff para clubes deportivos, asociaciones, academias y federaciones.*



Fuente: Página Web https://playoffinformatica.com/

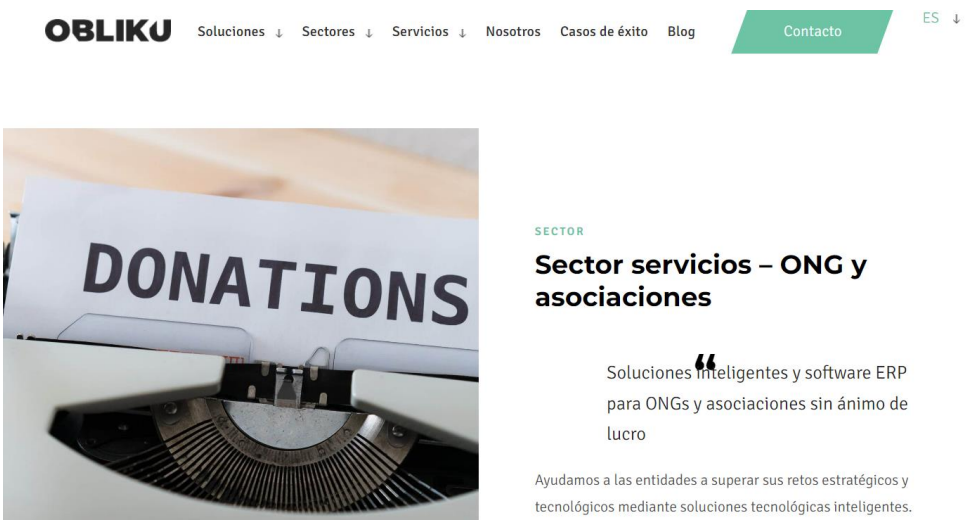
- ***Cucunver*** es un programa de gestión para asociaciones sin ánimo de lucro.
  - Solución que permite la gestión de tareas y el control global de lo que ocurren en la asociación.

Figura 8. *Solución Cucunver para asociaciones sin ánimo de lucro.*

Fuente: Página Web https://cucunver.com/

- ***Obliku*** es sistema EPR para las organizaciones y asociaciones sin ánimo de lucro.
  - Software enfocado a los procesos de gestión de la entidad, gestión fiscal y de contabilidad.

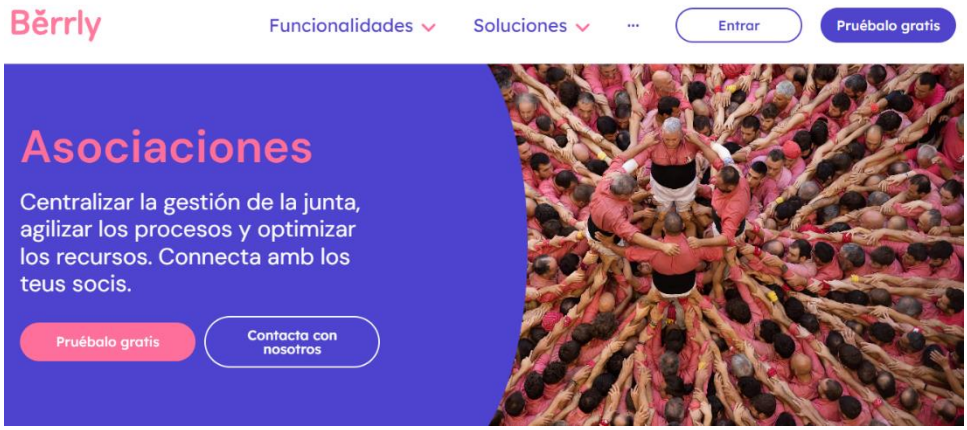
Figura 9. *Solución Obliku para organizaciones y asociaciones sin ánimo de lucro.*



Fuente: Página Web https://obliku.com/sectores/erp-para-ong/

- ***Berrly*** es un software para la gestión de socios ERP para las organizaciones sin ánimo de lucro.
    - Herramienta para la gestión de socios cumpliendo la normativa RGPD[v], gestión de comunicaciones, cobro de cuotas y organización de eventos.

Figura 10. *Solución Berrly para organizaciones sin ánimo de lucro.*



Fuente: Página Web https://www.berrly.com/es/solucion/asociaciones/

- ***Gong*** es sistema de gestión para organizaciones de desarrollo y cooperación.
    - Solución para la gestión económica, generación de informes, gestión técnica y de proyectos.

Figura 11. *Solución Gong para organizaciones de desarrollo y cooperación.*

Fuente: Página Web https://gong.es/

- ***Essenzial*** es una plataforma digital para fundaciones y asociaciones.
  - Sistema para gestionar la colaboración e interacción con sus socios, atención al profesional, gestión de campañas de marketing y otras funciones.

Figura 12. *Solución Essenzial como plataforma digital para fundaciones y asociaciones.*

Fuente: Página Web https://www.essenzial.com/solucion/nube-sectores/marketing-digital-fundaciones-y-sociedades-cientificas

- ***Cei*** es un sistema ERP para la gestión de organizaciones, obra social, fundaciones y asociaciones sin ánimo de lucro.
  - Software para la gestión de las áreas de negocio, de los proyectos, la integración de los colaboradores, reclutamiento de voluntarios y más funciones.

Figura 13. *Solución Cei para organizaciones, obra social, fundaciones y asociaciones sin ánimo de lucro.*

Fuente: Página Web https://www.ceieu.com/programa-software-gestion-erp-servicios/ong-fundaciones-asociaciones-obra-social/

- ***e-plan*** es un software para asociaciones desplegado en la nube.
    - Este sistema tiene funciones para la gestión de socios, de la agenda, tareas, tiene capacidades de gestor documental, gestión de actividades y asambleas entre otras funcionalidades.

Figura 14. *Solución e-plan como solución cloud para asociaciones.*

Fuente: Página Web https://www.teampyme.com/software-asociaciones

La función principal de las herramientas presentadas anteriormente es proporcionar capacidades de administración total o parcial a la organización desde el punto de vista de los dirigentes de la asociación y no desde el punto de vista del socio, aunque algunas de ellas cubren esta función de diferentes formas.

A continuación, se muestra la **tabla 4** **donde se comparan las soluciones expuestas, centrándose en las características que tienen relación con la solución de portal del socio propuesta**, al ser una solución que puede integrarse con el software de gestión propio de la entidad, aunque como futuras funcionalidades se propone crear un área de administración que permitirá cargar la información a través de formularios o mediante carga de ficheros, proporcionado independencia del sistema de gestión.

De las características de la tabla comparativa, **la característica clave es "Incluye Portal / App pasa socios"** que indica si la herramienta de gestión de la asociación incluye funciones para que el socio pueda realizar gestiones y trámites, además de la forma de resolver esta cuestión.

Tabla 4. *Tabla comparativa de características a destacar para la integración del portal del socio.*

	EKAMAT	TIMTUL	playoff	cucunver	OBLIKU	Berrly	GONG	essenzial	cei	TEAMPYME
Tipo de solución	ERP de Gestión adaptado	Plataforma adaptada (CRM[vi] + Eventos)	Software específico para asociaciones	Software específico para asociaciones	*ERP* de Gestión adaptado	Software específico para asociaciones	Software adaptado para asociaciones	Software específico para sociedades científicas	*ERP* de Gestión adaptado	Software adaptado para asociaciones
Específico para organizaciones sin ánimo de lucro	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Tipo de licencia	Desconocido	Pago por uso	Pago por uso	Pago por uso	Pago por uso	Pago por uso	Pago por uso	Desconocido	Desconocido	Pago por uso
Precio mínimo	Desconocido	100€/mes (con limitación de emails, hosting, etc.)	25€/mes (Hasta 150 socios y otras limitaciones)	35€/mes (Hasta 200 socios y otras limitaciones)	59€/mes (por usuario)	35€/mes (ilimitado, límite en envíos y tamaño en la nube)	Desconocido	Desconocido	Desconocido	0€/mes (100 socios con limitaciones)
Precio máximo	Desconocido	250€/mes (con menos limitación de emails, hosting, etc.)	A medida	65€/mes (ilimitado)	84,30€/mes (por usuario con función fabricación)	179€/mes (ilimitado, más envíos y tamaño en la nube)	Desconocido	Desconocido	Desconocido	79,95€/mes (ilimitado)
Gestión de cobros	Sí	Sí	Sí	Sí	No	Sí	No	No	Sí	Sí
Incluye Portal / App para socios	**No**	**No**	**Sí (Ambos)**	**Sí (App)**	**No**	**Sí (Ambos)**	**No**	**Sí (Portal sin gestión de cuotas)**	**No**	**Sí (Acceso al sistema limitado)**

Fuente: Elaboración propia basada en la información de las páginas web con la información de las soluciones comparadas

Las características que se han considerado interesantes junto con su motivo son:

- **Tipo de solución**: Se considera interesante el saber si es una solución específica para asociaciones o si hay sido adaptada porque puede influir en la evolución de las funcionalidades del software y que nunca se tenga en cuenta poner un portal como si parece que se hace en las soluciones específicas.
- **Específica para organizaciones sin ánimo de lucro**: Se considera interesante saber si es específica para organizaciones sin ánimo de lucro o no por la evolución que puede tener la herramienta y por el nivel de adaptabilidad a las necesidades de la asociación.
- **Tipo de licencia**: Se incluye para saber si el sistema es en propiedad o se paga en forma de servicio, para tener en cuenta el nivel de vinculación a la solución con la asociación y el desarrollo que tendrá. Al ser un pago por uso se entiende que la solución seguirá ampliando funcionalidades, en cambio en un pago *On-premise*[1] si no se tiene un soporte, la solución tendrá una serie de funcionalidades y el evolucionar el sistema dependerá del fabricante y de las versiones que saque, teniendo que volver a pagar por ellas.
- **Precio mínimo y precio máximo**: Indica el precio del conste del software o de la licencia, porque puede influir en las funcionalidades que tiene contratada la asociación según el precio que esté pagando (Pago básico con funcionalidades básicas y limitadas, o un pago premium con todo el catálogo de funcionalidades y menos limitaciones).
- **Gestión de cobros**: Si tiene la funcionalidad de gestionar cobros ya que el portal del socio se plantea con la gestión de la cuota del socio a un nivel de informar y revisar el pago por la administración, para poder aprovechar esta funcionalidad.
- **Portal / App para socios**: Si la solución dispone de alguna funcionalidad que permita al socio, a través de un portal o una aplicación móvil, consultar la información publicada por la asociación y realizar algún tipo de trámite. **Este punto es el más importante**, ya que este proyecto viene a cubrir la carencia que existe en determinadas asociaciones cuyas herramientas de gestión no disponen de este servicio para el socio. Además, permite que aquellas soluciones que no cuentan con esta funcionalidad la puedan incorporar de forma rápida, sin necesidad de un desarrollo a medida.

[1] *On-premise*: Modelo de licencia y de uso de software basado en el servidor.

2.3. Conclusión del análisis

Según lo expuesto en el apartado de contexto y estado del arte sobre las soluciones tecnológicas del tercer sector, **se observa que no hay consenso sobre una solución web reutilizable que esté dirigida a los socios de asociaciones sin ánimo de lucro** y que proporcione a las organizaciones un software que permita que sus socios puedan realizar consultas y trámites para cumplir algunos de los derechos y obligaciones que marcan los estatutos de la organización.

Es cierto que **algunos sistemas de gestión han implementado su propia solución para cubrir las necesidades desde la perspectiva del socio, pero son componentes acoplados al sistema de administración de la organización y no son soluciones reutilizables entre otros sistemas, ni se pueden utilizar de forma independiente al sistema de gestión**, estando ligadas a la adquisición de los servicios del sistema de administración y no pudiendo utilizar sólo las funciones para el socio.

La afirmación de que no se observa consenso se basa en la información descrita en la tabla 4 comparativa de solución, partiendo de la última característica clave donde **la mitad de las soluciones para las asociaciones si disponen de una opción propia e integrada** para permitir a los socios realizar consultas y gestiones, **pero la otra mitad no**, **por este motivo se llega a la conclusión de que no hay consenso tecnológico en el sector** y que haya cambia en el sector para usa solución dedicada para cubrir estas funciones que pueda ser utilizada por cualquier asociación que lo considere.

El portal web de este proyecto se plantea como una solución genérica que permite cubrir la necesidad de gestión desde el punto de vista del socio para aquellas asociaciones que no disponen algunos de los sistema de administración de forma nativa y que quieren que sus socios tengan de una plataforma para realizar ciertas gestiones, aunque no excluyente esta opción a las asociaciones que sí disponen de un sistema de administración propio y quieren implantar una solución reutilizable especializada en socios de asociaciones.

3. Objetivos y metodología de trabajo

En este apartado se definen los objetivos generales del trabajo y la metodología utilizada para el desarrollo de la solución planteada.

El objetivo principal es analizar, especificar, diseñar e implementar un portal web que permita a los socios de asociaciones sin ánimo de lucro realizar algunas de las gestiones necesarias por pertenecer a una asociación.

La metodología de trabajo es la metodología de desarrollo de software modelo de proceso evolutivo en espiral que es un tipo de metodología tradicional basada en los conceptos predictivos de desarrollo y no en conceptos ágiles como son otras metodologías actuales.

3.1. Objetivo general

El objetivo general es **proporcionar al sector una solución de software reutilizable que permita a las asociaciones, de manera rápida, poner a disposición de sus socios un sitio web donde puedan realizar las gestiones y trámites** necesarios para ejercer sus derechos y cumplir con sus obligaciones según lo establecido en los estatutos.

Para lograr este objetivo, se va a **implementar un portal web funcional** que permita a los socios de las asociaciones sin ánimo de lucro realizar gestiones y trámites en la asociación de la que es miembro.

Uno de los beneficios que aporta el implementar este portal es que es una solución enfocada al negocio de las asociaciones y no es un software genérico que resuelve problemas generales (Como puede ser soluciones *CRM*, *ERP* u otro tipo de soluciones genéricas), **teniendo las estructuras de datos y funcionalidades propias del sector**. Esto permitirá a las asociaciones incluir esta **herramienta como elemento de interacción entre el socio y la asociación, beneficiando a ambas partes de tener este enfoque específico**.

3.2. Objetivos específicos

Los objetivos específicos que permiten alcanzar el objetivo general están ligados a **las etapas de desarrollo del proceso de ingeniería de software para conseguir un producto utilizable** que cubra una serie de funcionalidades básicas:

1. **Analizar el contexto y estado del arte del sector**; Se describe el contexto real de las asociaciones sin ánimo de lucro, las necesidades existentes en el sector, el análisis de si las soluciones actuales cubren la problemática planteada y las conclusiones obtenidas.
2. **Desarrollar las especificaciones del producto**; Se recopila la información y se define el alcance de la primera entrega. Se especifican los requisitos funcionales, no funcionales y el alcance de la entrega inicial.
3. **Diseñar el producto**; Se realiza la definición de los objetos de negocio del sistema, se diseña la arquitectura de peticiones del sistema y se detalla la tecnología más adecuada para implementar la solución y por qué se ha escogido.
4. **Implementar el producto**; Se desarrolla el paquete de características iniciales del sistema, teniendo en cuenta cada una de sus capas.
5. **Realizar pruebas**; Se realizan las pruebas para asegurar que el producto funciona correctamente. Las pruebas realizadas son pruebas unitarias y pruebas funcionales.
6. **Detallar las conclusiones**; Se detallan las conclusiones obtenidas durante el proceso de ingeniería de software y en la elaboración de este proyecto.
7. **Proponer funcionalidades futuras**; Identificar y proponer nuevas funcionalidades que podrían incluirse en el producto en futuras iteraciones**.**

3.3. Metodología de trabajo

Se decide realizar el proceso de ingeniería del software utilizando la metodología de trabajo basado en un **modelo de proceso evolutivo en espiral** teniendo en cuenta que el nivel de incertidumbre inicial del producto es limitado ya que se comprende el producto básico, pero faltan detalles de algunas de las necesidades.

3.3.1. Modelo de proceso evolutivo en espiral

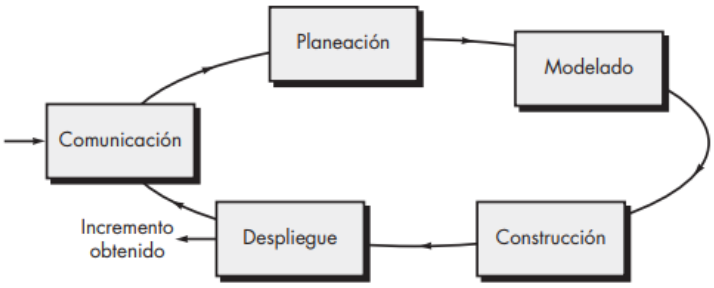
Con el objetivo de adaptar los requerimientos de negocio a la realidad operativa y las necesidades de los usuarios finales, se emplea el **modelo de proceso evolutivo en espiral**. Este enfoque facilita la creación de un producto inicial que luego se irá mejorando y ajustando en iteraciones sucesivas conforme se vayan identificando y especificando las necesidades específicas.

El modelo de proceso evolutivo en espiral se puede definir como:

"El modelo espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas." (Pressman, 2010).

Como indica Pressman en la definición del modelo en espiral, el proceso de ingeniería se realiza en iteraciones y cada iteración tiene una serie de actividades ordenadas que deben realizarse de manera rigurosa (Comunicación, planificación, modelado, implementación, despliegue) las cuales permiten conseguir un incremento del producto a través de un paquete de funcionalidades para los usuarios, los cuales tienen que proporcionan retroalimentación del paquete recibo para guiar las futuras entregas (Pressman, 2010).

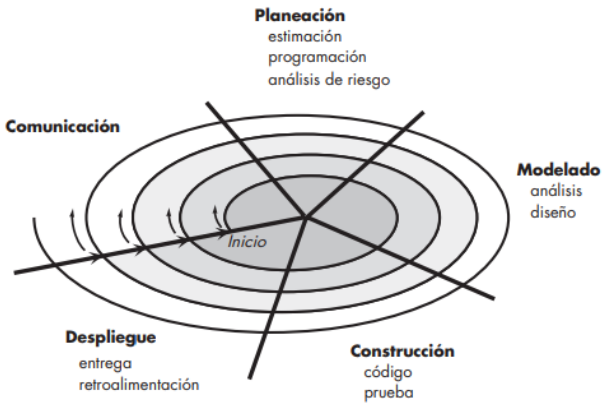
Figura 15. *Flujo de proceso evolutivo.*



Fuente: Libro de Ingeniería del software. Un enfoque práctico. (Pressman, 2010, p. 28)

En las primeras iteraciones, el producto obtenido será un prototipo o producto básico y según se vayan realizando iteraciones, se producirán versiones cada vez más cercanas a la funcionalidad concreta que vayan demandando los usuarios directos e indirectos del sistema.

Figura 16. *Modelo de proceso evolutivo en espiral.*



Fuente: Libro de Ingeniería del software. Un enfoque práctico. (Pressman, 2010, p. 161)

4. Especificación del producto

En este apartado se especifican y describen los requisitos funcionales, los requisitos funcionales y los casos de uso para la primera entrega.

A continuación, se definen algunos de los conceptos claves en la especificación de software utilizando el libro de Requerimientos del software de Wiegers y Beatty de 2013:

"Los requisitos funcionales especifican los comportamientos que exhibirá el producto en condiciones específicas. Describen lo que los desarrolladores deben implementar para permitir a los usuarios realizar sus tareas." (p. 9).

"Los requisitos no funcionales describen las características del producto en varias dimensiones que son importantes para los usuarios, los desarrolladores y mantenedores, como el rendimiento, la seguridad, la disponibilidad, y portabilidad." (p. 10).

"Un caso de uso describe una secuencia de interacciones entre un sistema y un actor externo que da como resultado que el actor pueda lograr algún resultado de valor." (p. 144).

4.1. Requisitos funcionales

Los requisitos funcionales que se han identificado y especificado para el primer paquete de funcionalidades del portal son los siguientes:

1. Para la **gestión de acceso al sistema**;
    a. RF0001 - El sistema deberá controlar el acceso y sólo lo permitirá a los usuarios registrados de tipo socio y administrador. Los usuarios deben ingresar al sistema a través de su nombre de usuario y contraseña.
    b. RF0002 - El sistema deberá validar que el usuario existe, los datos de autenticación son correctos y el usuario está activo.
    c. RF0003 - El sistema deberá controlar que los usuarios de tipo socio y de tipo administrador sólo acceden a la asociación de la que son miembros.
    d. RF0004 - El sistema deberá permitir a un usuario autenticado que pueda cerrar sesión salir del sistema.
2. Para la **gestión de perfil del socio**;
    a. RF0005 - El sistema deberá permitir a un usuario de tipo socio registrado ver su perfil.

b. RF0006 - El sistema deberá permitir a un usuario de tipo socio registrado ver los anuncios publicados por la asociación.
c. RF0007 - El sistema deberá mostrar en cada anuncio su título, la descripción del anuncio, el nivel de importancia (Alta, media o baja), el autor y documentación relaciona si lo tiene.
d. RF0008 - El sistema deberá mostrar los anuncios ordenados siendo el primero el último publicado y el último el primero publicado.
e. RF0009 - El sistema deberá permitir a un usuario de tipo socio registrado descargar cada documento anexado al anuncio.
f. RF0010 - El sistema deberá permitir a un usuario de tipo socio registrado ver sus cuotas de socio.
g. RF0011 - El sistema deberá mostrar por cada cuota de socio con su información más reciente.
h. RF0012 - El sistema deberá mostrar por cada cuota de socio el tipo (ordinaria o extraordinaria), el periodo (anual, semestral, trimestral, mensual o bimensual), la fecha de efecto, la fecha de vencimiento, el importe, el medio de pago y el estado (pendiente de cobro, pendiente de validación, cobrado).
i. RF0013 - El sistema deberá mostrar las cuotas de socio ordenadas siendo la primera la más reciente a partir de la fecha de efecto y la última la más antigua según la fecha de efecto.
j. RF0014 - Cuando una cuota del socio esté pendiente de cobro, el sistema deberá permitir justificar que se ha pasado indicando un título al justificante, una descripción y el documento que justifique el pago indicando al usuario que está pendiente de validación.
k. RF0015 - El sistema deberá permitir a un usuario de tipo socio registrado ver los documentos que tiene como socio de la asociación.
l. RF0016 - El sistema deberá mostrar en cada documento del socio su categoría (cuotas de socio, socio, asociación o anuncio), el nombre del fichero, la extensión, el tamaño, fecha de anexado).
m. RF0017 - El sistema deberá mostrar los documentos ordenados siendo el primero el último anexado y el último el primero anexado.

n. RF0018 - El sistema deberá permitir a un usuario de tipo socio registrado descargar cada documento del socio.

3. Para la **gestión de la información de la asociación**;
    a. RF0019 - El sistema deberá permitir a un usuario de tipo socio registrado ver la información de la asociación.
    b. RF0020 - El sistema deberá permitir a un usuario de tipo socio registrado descargar cada documento de la asociación.
    c. RF0021 - El sistema deberá permitir a un usuario de tipo socio registrado ver los integrantes del órgano de gobierno actual.
    d. RF0022 - El sistema deberá mostrar la fecha desde que lleva en vigor el órgano de gobierno actual.
    e. RF0023 - El sistema deberá mostrar por cada directivo su nombre, fecha desde que lleva en el cargo y cargo (presidente, secretario, tesorero o vocal).
    f. RF0024 - El sistema deberá mostrar los directivos según el orden de importancia del cargo, siendo primero el presidente, segundo el secretario, tercero el tesorero u último el vocal.
4. Para la **administración del sistema**;
    a. RF0025 - El sistema deberá registrar las trazas de arranque y parada del sistema.
    b. RF0026 - El sistema deberá registrar las trazas de las solicitudes de inicio y cierre de sesión de los usuarios.
    c. RF0027 - El sistema deberá registrar las trazas de cada petición realizada al sistema.
    d. RF0028 - El sistema deberá permitir desactivar el registro de las trazas de auditoría.
    e. RF0029 - El sistema deberá permitir a un usuario de tipo administrador consultar las solicitudes de verificación de pago de cuotas pendientes de los socios de la asociación.
    f. RF0030 - El sistema deberá permitir a un usuario de tipo administrador consultar las solicitudes de verificación de pago de cuotas pendientes de los socios de la asociación.

g. RF0031 - El sistema deberá permitir a un usuario de tipo administrador verificar el pago de una cuota de socio pendiente.

4.1.1. Historias de usuario

Los requisitos funcionales especificados para el primer paquete de funcionalidades se detallan en forma de historia de usuario.

"Una historia de usuario describe una funcionalidad que será valiosa para un usuario o comprador de un sistema o software." (Cohn, 2004, p. 4).

Las historias de usuarios a menudo se escriben según la siguiente plantilla, aunque también se pueden utilizar otros estilos:

Como <*rol*> **necesito** <*funcionalidad*> **para** <*beneficio*>.

(Wiegers & Beatty, 2013, p. 145)

En la tabla 5 se detallan las historias de usuario recogidas:

Tabla 5. *Historias de usuario con su prioridad de implementación.*

ID	COMO	<rol>	NECESITO	<funcionalidad>	PARA	<beneficio>	PRIORIDAD
UC-001	Como	socio	necesito	poder autenticarme en el sistema	para	poder acceder a la información relevante que publica la asociación.	Alta
UC-002	Como	socio	necesito	poder desconectarme del sistema	para	no dejar la sesión abierta en el dispositivo desde donde me haya autenticado.	Baja
UC-003	Como	socio	necesito	poder consultar mi información de socio	para	revisar que toda la información que tiene la asociación es correcta.	Alta
UC-004	Como	socio	necesito	poder consultar los anuncios publicados por los gestores de la asociación	para	enterarme de los eventos, asambleas y actividades que realiza la asociación.	Alta
UC-005	Como	socio	necesito	poder consultar las cuotas de socio	para	poder saber el estado de las cuotas y estar día en el pago de la cuota.	Alta
UC-006	Como	socio	necesito	poder justificar el pago de una cuota de socio	para	que la asociación pueda verificar el pago y no me den de baja de la asociación por impago.	Alta
UC-007	Como	socio	necesito	poder consultar mis documentos de cuotas de socios y otros documentos	para	revisar que lo adjunta está correcto y tener un listado de documentos enviados a la asociación.	Alta
UC-008	Como	socio	necesito	poder descargar un documento de mi perfil	para	revisar su contenido.	Media
UC-009	Como	socio	necesito	poder consultar la información de la asociación para tener la información disponible	para	toda tramitación e interacción con la asociación.	Media
UC-010	Como	socio	necesito	poder consultar los miembros de la junta directiva	para	saber los socios directivos que están gestionando la asociación.	Media
UC-011	Como	administrador del sistema	necesito	poder consultar toda la información de auditoría disponible de los arranques y paradas del sistema, de las solicitudes de inicio y cierre de sesión y del resto de peticiones realizadas al servidor.	para	detectar accesos no autorizados, caídas del sistema, incidencias y gestión de la seguridad.	Baja

Fuente: Elaboración propia

4.2. Requisitos no funcionales

Los requisitos no funcionales están enfocados a los atributos de calidad de un sistema para organizaciones del tercer sector, teniendo presente su labor social y su fin no lucrativo:

- **RNF001 - Web**: El sistema debe permitir al usuario usarlo desde cualquier lugar.
- **RNF002 - Seguro**: El sistema debe estar protegido contra el acceso a personal no autorizado.
- **RNF003 - Disponible**: El sistema debe estar disponible en todo momento para que el usuario pueda usarlo.
- **RNF004 - Compatible**: El sistema debe poder integrarse con los sistemas de gestión de las asociaciones existentes en el mercado para poder ofrecer las distintas funcionalidades en tiempo real.
- **RNF005 - Genérico**: Debe poder ser utilizado por cualquier asociación que necesite dar este servicio a sus socios.
- **RNF006 - Mantenible**: El sistema debe permitir un mantenimiento sencillo y ágil para que un perfil de desarrollador no especialista pueda mantenerlo, sin necesitar un equipo complejo o altamente cualificado porque no todas las organizaciones sin ánimo de lucro disponen de medios para tener un equipo de soporte.
- **RNF007 - Escalable**: El sistema debe adaptarse y crecer eficientemente ante un aumento de carga de trabajo.
- **RNF008 - Actuable**: El sistema debe poder manejar el número necesario de usuarios concurrentes sin limitaciones el número de usuarios conectados ni degradación en el rendimiento.
- **RNF009 - Portable**: El sistema debe poder usarse desde cualquier dispositivo.
- **RNF010 - Usable**: El sistema debe utilizar patrones de diseño de interfaz y utilizar los estándares y buenas prácticas que marca la experiencia de usuario, teniendo una interfaz clara, usable y homogénea.
- **RNF011 - Sostenible**: El sistema debe ser una solución sostenible a nivel económica, que tenerla en producción y realizar evolutivos no suponga un gasto excesivo o con una gran complejidad, que permite a las asociaciones centrar sus recursos en otros temas.

- **RNF012 - Código abierto**: El sistema debe ser una solución basad en código abierto para que todas las asociaciones que lo consideren, puedan acceder a esta solución y adaptarla a su medida sin gasto en licencia.
- **RNF013 - Compliance**: El sistema debe cumplir con todas las leyes y reglamentos aplicables al tercer sector.

4.3. Alcance de la primera versión

El alcance para la primera versión es realizar un portal funcional que cubra las funciones básicas de un portal web desde la perspectiva del socio, Teniendo que iniciar sesión para acceder a las funciones del sistema y una vez autenticado que le permita consultar su información personal, que pueda consultar las noticias y eventos que publique la asociación, poder consultar y descargar la documentación del socio y que pueda consultar las cuotas y su estado, además de poder justificar el pago de estas.

También se considera una funcionalidad inicial el poder consultar la información de la asociación y de la junta directiva actual.

4.4. Casos de uso

Las funcionalidades a implementar descritas en los siguientes casos de uso se han basado en las historias de usuario que se han definido, están enfocados desde la perspectiva del socio, que es el usuario al que se han orientado las primeras funcionalidades del sistema.

Los casos de uso se han agrupado en los siguientes sistemas:

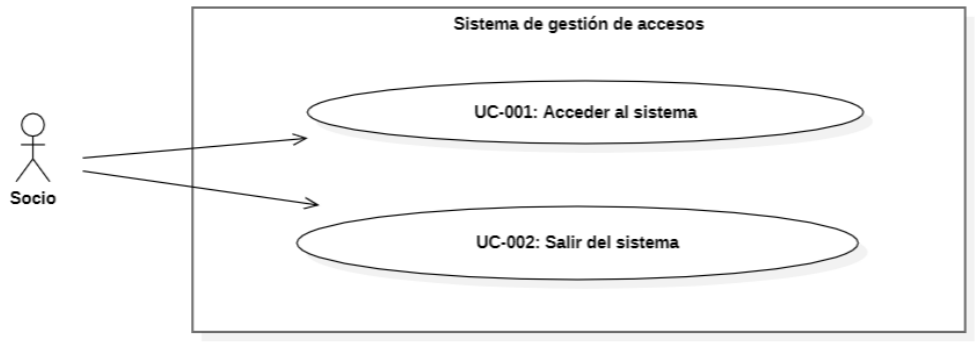
1. **Sistema de gestión de accesos**; Para la gestión de la autenticación y salida de usuarios del sistema.
2. **Sistema de gestión de perfil de socio**; Para la gestión de la consulta de la información actual del socio, de las publicaciones de la asociación y gestión de la cuota del socio.
3. **Sistema de gestión de información de asociación**; Para la gestión de la consulta de información actual de la asociación y de la junta directiva.
4. **Sistema de administración del sistema**; Para la administración del sistema.

4.4.1. Sistema de gestión de accesos

Este sistema se encarga de la autenticación y la salida de los usuarios del sistema, en este caso del usuario socio.

Los casos de uso detectados para este sistema son los siguientes:

Figura 17. Diagrama de casos de uso del Sistema de gestión de accesos.



Fuente: Elaboración propia con *StartUML*[2]

4.4.1.1. UC-001 Acceder al sistema

Tabla 6. *Especificación de caso de uso UC-001 Acceder al sistema.*

Identificador	UC-001
Nombre del caso de uso	Acceder al sistema
Actores	S3 – Socios
Requisitos relacionados	- RF0001, RF0002, RF0003, RF0026
Precondiciones	- PRE01: Ser socio de la asociación - PRE02: Tener un usuario en la asociación.
Escenario principal: CU-001-EP-001 Flujo principal de autenticación	
1. El usuario inicia el flujo de inicio de sesión 2. El sistema solicita introducir el usuario y la contraseña. 3. El usuario introduce el nombre de usuario y la contraseña. 4. El sistema comprueba si la información de autenticación es correcta. 5. Según el resultado de la comprobación a. Si los datos de autenticación son correctos; i. El sistema audita el acceso al sistema. ii. El sistema redirecciona a la página principal del portal de usuario. b. Si se produce un error en la autenticación → ver el flujo excepción CU-001-EE-001	
Escenario excepción: CU-001-EE-001 Flujo de excepción de autenticación	
1. El sistema muestra que hay un error en el proceso de autenticación 2. Continua → Paso 1 del flujo CU-001-EP-001	
Resultado	- RESU01: El usuario queda autenticado en el sistema.

Fuente: Elaboración propia

[2] *StartUML*: https://staruml.io/

4.4.1.2. UC-002 Salir del sistema

Tabla 7. *Especificación de caso de uso UC-002 Salir del sistema.*

Identificador	UC-002
Nombre del caso de uso	Salir del sistema
Actores	S3 – Socios
Requisitos relacionados	- RF0004, RF0026
Precondiciones	- PRE03: Estar autenticado en el sistema.
Escenario principal: CU-002-EP-001 Flujo principal de salir del sistema	
1. El usuario indica que quiere cerrar sesión. 2. El sistema audita la salida del sistema del usuario. 3. El sistema cierra elimina la sesión del usuario. 4. El sistema redirecciona al usuario la página de autenticación.	
Escenario excepción: CU-002-EE-001 Flujo de excepción de salir del sistema	
1. El sistema muestra por consola que hay un error en la salida de sesión. 2. Continua → Paso 3 del flujo CU-002-EP-001	
Resultado	- RESU02: El usuario ya no está autenticado en el sistema.

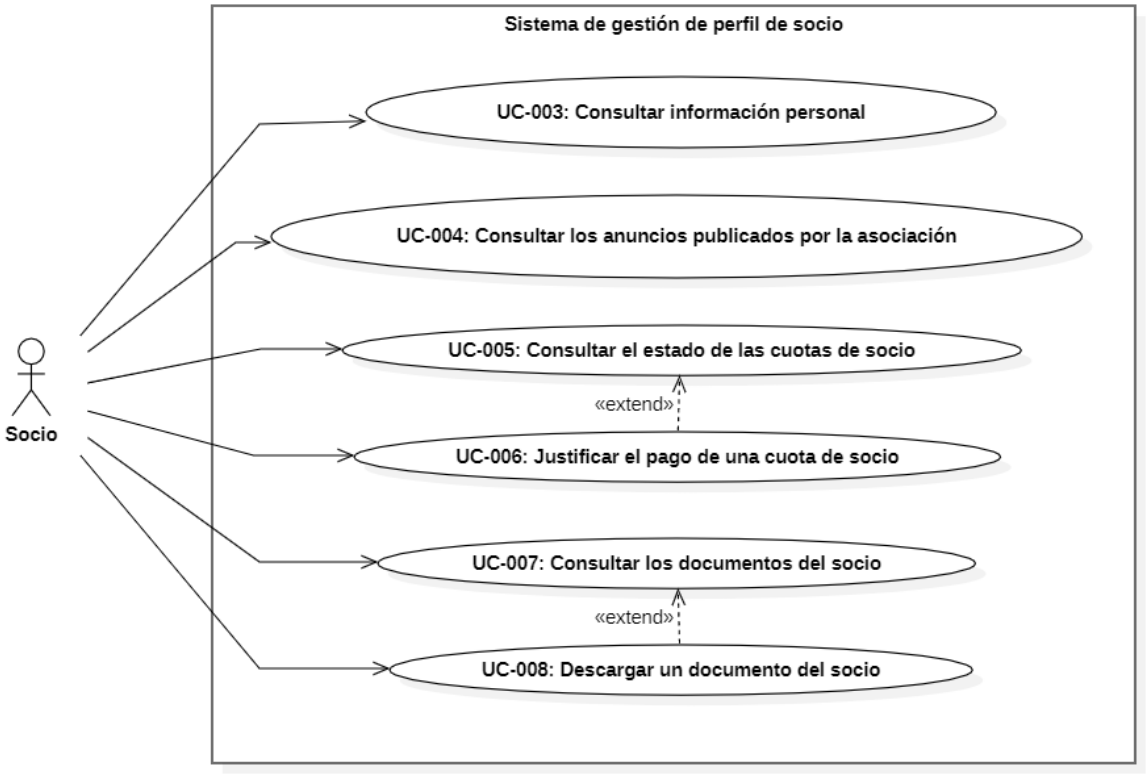
Fuente: Elaboración propia

4.4.2. Sistema de gestión de perfil del socio

Este sistema se permite al socio consultar su información actual, de las publicaciones de la asociación, gestionar la cuota del socio y consultar los documentos del socio.

Los casos de uso detectados para este sistema son los siguientes:

Figura 18. Diagrama de casos de uso del Sistema de gestión de perfil de socio.



Fuente: Elaboración propia con *StartUML*

4.4.2.1. UC-003 Consultar información personal

Tabla 8. *Especificación de caso de uso UC-003 Consultar información personal.*

Identificador	UC-003
Nombre del caso de uso	Consultar información personal
Requisitos relacionados	- RF0005, RF0027
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema.
Escenario principal: CU-003-EP-001 Flujo principal de consultar información personal	
1. El usuario indica que quiere ir a la consulta de información personal. 2. El sistema audita la petición de consulta de información personal. 3. El sistema muestra la información personal del socio.	
Escenario excepción: CU-003-EE-001 Flujo de excepción de consulta de información personal	
1. El sistema muestra por consola que hay un error al consultar la información. 2. Continua → Paso 1 del flujo CU-003-EP-001	
Resultado	- RESU03: El socio consulta su información personal.

Fuente: Elaboración propia

4.4.2.2. UC-004 Consultar los anuncios publicados por la asociación

Tabla 9. *Especificación de caso de uso UC-004 Consultar los anuncios publicados por la asociación.*

Identificador	UC-004
Nombre del caso de uso	Consultar los anuncios publicados por la asociación
Requisitos relacionados	- RF0006, RF0007, RF0008, RF0009, RF0027
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema. - PRE04: Estar en la consulta de información personal.
Escenario principal: CU-004-EP-001 Flujo principal de consulta de anuncios publicados por la asociación	
1. El usuario indica que quiere ir a la consultar los anuncios publicados por la asociación. 2. El sistema audita la petición de consultar los anuncios publicados por la asociación. 3. El sistema muestra los anuncios publicados por la asociación teniendo en cuenta el número de página y numero de publicaciones indicadas en la paginación.	
Escenario alternativo: CU-004-EA-001 Flujo alternativo de selección de número de publicaciones por página de la paginación	
1. Paso 1 del flujo CU-004-EP-001 2. El sistema permite seleccionar el número de publicaciones por página a mostrar. 3. El usuario indica el número de publicaciones a mostrar por página. 4. Continua → Paso 4 del flujo CU-004-EP-001	
Escenario alternativo: CU-004-EA-002 Flujo alternativo de selección de número de página en la paginación	
1. Paso 1 del flujo CU-004-EP-001 2. El sistema permite seleccionar el número de página en la paginación. 3. El usuario indica el número de página siguiente si no es la última o la anterior si no es la primera. 4. Continua → Paso 4 del flujo CU-004-EP-001	
Escenario alternativo: CU-004-EA-003 Flujo alternativo de descarga de documento	
1. Paso 1 del flujo CU-004-EP-001 2. El sistema permite descargar un documento del anuncio. 3. El usuario indica que quiere descargar un documento del anuncio. 4. El sistema descarga en el equipo local del usuario el documento indicado.	
Escenario excepción: CU-004-EE-001 Flujo de excepción de consulta de anuncios publicados por la asociación	
1. El sistema muestra por consola que hay un error al consultar los anuncios publicados por la asociación. 2. Continua → Paso 1 del flujo CU-004-EP-001	
Resultado	- RESU04: El socio consulta las publicaciones de la asociación.

Fuente: Elaboración propia

4.4.2.3. UC-005 Consultar el estado de las cuotas de socio

Tabla 10. *Especificación de caso de uso UC-005 Consultar el estado de las cuotas de socio.*

Identificador	UC-005
Nombre del caso de uso	Consultar el estado de las cuotas de socio
Requisitos relacionados	- RF0010, RF0011, RF0013, RF0027
Casos de uso relacionados	- UC-006
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema. - PRE04: Estar en la consulta de información personal.
Escenario principal: CU-005-EP-001 Flujo principal de consulta de estado de cuotas del socio	
1. El usuario indica que quiere ir a la consultar el estado sus cuotas de socio. 2. El sistema audita la petición de consultar el estado de cuotas de socio. 3. El sistema muestra los estados de cuotas de socio teniendo en cuenta el número de página y numero de estados de cuota indicados en la paginación.	
Escenario alternativo: CU-005-EA-001 Flujo alternativo de selección de número de estados de cuotas de socio por página de la paginación	
1. Paso 1 del flujo CU-005-EP-001 2. El sistema permite seleccionar el número de estados de cuotas de socio por página a mostrar. 3. El usuario indica el número de estados de cuotas de socio a mostrar por página. 4. Continua → Paso 4 del flujo CU-005-EP-001	
Escenario alternativo: CU-005-EA-002 Flujo alternativo de selección de número de página en la paginación	
1. Paso 1 del flujo CU-005-EP-001 2. El sistema permite seleccionar el número de página en la paginación. 3. El usuario indica el número de página siguiente si no es la última o la anterior si no es la primera. 4. Continua → Paso 4 del flujo CU-005-EP-001	
Escenario excepción: CU-005-EE-001 Flujo de excepción de consulta de estados de cuotas de socios	
1. El sistema muestra por consola que hay un error al consultar los estados de cuotas de socio. 2. Continua → Paso 1 del flujo CU-005-EP-001	
Resultado	- RESU05: El socio consulta sus estados de cuotas de socio.

Fuente: Elaboración propia

4.4.2.4. UC-006 Justificar el pago de una cuota de socio

Tabla 11. *Especificación de caso de uso UC-006 Justificar el pago de una cuota de socio.*

Identificador	UC-006
Nombre del caso de uso	Justificar el pago de una cuota de socio
Requisitos relacionados	- RF0014, RF0027
Casos de uso relacionados	- UC-005
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema. - PRE05: Estar en la consulta de estados de cuotas de socio. - PRE09: Esta la cuota de socio pendiente de cobro.
Escenario principal: CU-006-EP-001 Flujo principal de justificación de pago de una cuota de socio	
1. El usuario indica que quiere justificar el pago de una cuota del socio. 2. El sistema audita la petición de rellenar la justificación de la cuota de socio. 3. El sistema solicita rellenar la información necesaria para justificar el pago de la cuota de socio. 4. El usuario rellena la información solicitada e indica que quiere realizar la justificación con la información introducida. 5. El sistema audita la petición de justificar la cuota de socio. 6. El sistema actualiza el estado de la cuota de socio a pendiente de verificación. 7. Continua → Paso 1 del flujo CU-006-EP-001	
Escenario alternativo: CU-006-EA-001 Flujo alternativo de salir de justificación de pago de cuota	
1. Paso 5 del flujo CU-006-EP-001 2. El usuario indica que quiere salir del flujo de justificación de pago de cuota de socio. 3. El sistema redirecciona a la consulta de estados de cuota de socio → Paso 4 del flujo CU-005-EP-001	
Escenario excepción: CU-005-EE-001 Flujo de excepción de consulta de estados de cuotas de socios	
1. El sistema muestra por consola que hay un error al justificar el pago de la cuota de socio. 2. Continua → Paso 4 del flujo CU-005-EP-001	
Resultado	- RESU06: El socio justifica el pago de una cuota de socio.

Fuente: Elaboración propia

4.4.2.5. UC-007 Consultar los documentos del socio

Tabla 12. *Especificación de caso de uso UC-007 Consultar los documentos del socio.*

Identificador	UC-007
Nombre del caso de uso	Consultar los documentos del socio
Requisitos relacionados	- RF0015, RF0016, RF0017, RF0027
Casos de uso relacionados	- UC-008
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema. - PRE04: Estar en la consulta de información personal.
Escenario principal: CU-007-EP-001 Flujo principal de consulta de documentos del socio	
1. El usuario indica que quiere ir a la consultar los documentos del socio. 2. El sistema audita la petición de consultar los documentos del socio. 3. El sistema muestra los documentos del socio teniendo en cuenta el número de página y numero de documentos indicados en la paginación.	
Escenario alternativo: CU-007-EA-001 Flujo alternativo de selección de número de documentos del socio por página de la paginación	
1. Paso 1 del flujo CU-007-EP-001 2. El sistema permite seleccionar el número de documentos del socio por página a mostrar. 3. El usuario indica el número de documentos del socio a mostrar por página. 4. Continua → Paso 4 del flujo CU-007-EP-001	
Escenario alternativo: CU-007-EA-002 Flujo alternativo de selección de número de página en la paginación	
1. Paso 1 del flujo CU-007-EP-001 2. El sistema permite seleccionar el número de página en la paginación. 3. El usuario indica el número de página siguiente si no es la última o la anterior si no es la primera. 4. Continua → Paso 4 del flujo CU-007-EP-001	
Escenario excepción: CU-007-EE-001 Flujo de excepción de consulta de estados de cuotas de socios	
1. El sistema muestra por consola que hay un error al consultar los documentos del socio. 2. Continua → Paso 1 del flujo CU-007-EP-001	
Resultado	- RESU07: El socio consulta sus documentos.

Fuente: Elaboración propia

4.4.2.6. UC-008 Descargar un documento del socio

Tabla 13. *Especificación de caso de uso UC-008 Descargar un documento del socio.*

Identificador	UC-008
Nombre del caso de uso	Descargar un documento del socio
Requisitos relacionados	- RF0018, RF0027
Casos de uso relacionados	- UC-007
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema. - PRE06: Estar en la consulta de documentos del socio.
Escenario principal: CU-006-EP-001 Flujo principal de justificación de pago de una cuota de socio	
1. El usuario indica que quiere descargar un documento del socio. 2. El sistema audita la petición de descargar el documento del socio. 3. El sistema descarga en el equipo local el documento de la asociación. 4. Continua → Paso 4 del flujo CU-007-EP-001	
Escenario excepción: CU-005-EE-001 Flujo de excepción de consulta de estados de cuotas de socios	
1. El sistema muestra por consola que hay un error al descargar el documento del socio. 2. Continua → Paso 4 del flujo CU-007-EP-001	
Resultado	- RESU08: El socio descarga un documento.

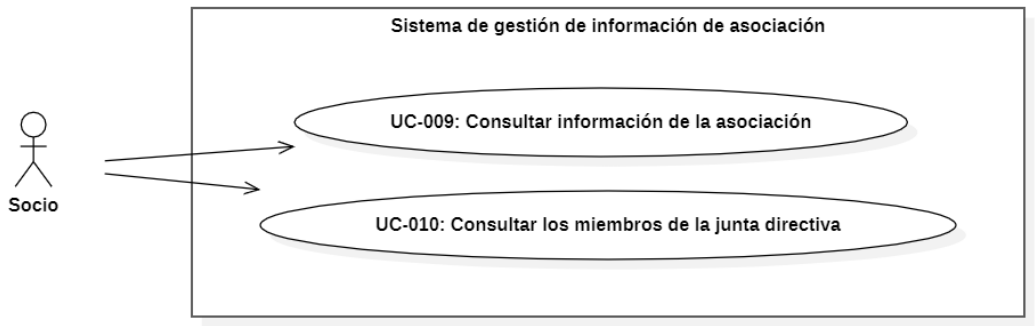
Fuente: Elaboración propia

4.4.3. Sistema de gestión de información de asociación

Este sistema se encarga de la autenticación y la salida de los usuarios del sistema, en este caso del usuario socio.

Los casos de uso detectados para este sistema son los siguientes:

Figura 19. Diagrama de casos de uso del Sistema de gestión de perfil de socio.



Fuente: Elaboración propia con *StartUML*

4.4.3.1. UC-009 Consultar información de la asociación

Tabla 14. *Especificación de caso de uso UC-009 Consultar información de la asociación.*

Identificador	UC-009
Nombre del caso de uso	Consultar información de la asociación
Requisitos relacionados	- RF0019, RF0020, RF0027
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema.
Escenario principal: CU-009-EP-001 Flujo principal de consulta de información de la asociación	
1. El usuario indica que quiere ir a la consulta de información de la asociación. 2. El sistema audita la petición de consulta de información de la asociación. 3. El sistema muestra la información de la asociación.	
Escenario alternativo: CU-009-EA-001 Flujo alternativo de descarga de documento de asociación	
1. Paso 1 del flujo CU-009-EP-001 2. El sistema permite descargar los documentos adjuntos de la asociación. 3. El usuario indica que quiere descargar un documento de la asociación. 4. El sistema audita la petición de descarga de documento de asociación. 5. El sistema descarga en el equipo local el documento de la asociación.	
Escenario alternativo: CU-009-EA-002 Flujo alternativo de descarga de documento	
5. Paso 1 del flujo CU-009-EP-001 6. El sistema permite descargar un documento de la asociación. 7. El usuario indica que quiere descargar un documento de la asociación. 8. El sistema descarga en el equipo local del usuario el documento indicado.	
Escenario excepción: CU-009-EE-001 Flujo de excepción de consulta de información de la asociación	
1. El sistema muestra por consola que hay un error al consultar la información de la asociación. 2. Continua → Paso 1 del flujo CU-003-EP-001	
Resultado	- RESU09: El socio consulta la información de la asociación.

Fuente: Elaboración propia

4.4.3.2. UC-010 Consultar los miembros de la junta directiva

Tabla 15. *Especificación de caso de uso UC-010 Consultar los miembros de la junta directiva.*

Identificador	UC-010
Nombre del caso de uso	Consultar los miembros de la junta directiva
Requisitos relacionados	- RF0021, RF0022, RF0023, RF0024, RF0027
Actores	S3 – Socios
Precondiciones	- PRE03: Estar autenticado en el sistema. - PRE07: Estar en la consulta de la asociación.
Escenario principal: CU-010-EP-001 Flujo principal de consulta de miembros de la junta directiva actual	
1. El usuario indica que quiere ir a la consulta la información de la junta directiva actual. 2. El sistema audita la petición de consulta de información de la junta directiva actual. 3. El sistema muestra los miembros de la junta directiva actual.	
Escenario excepción: CU-010-EE-001 Flujo de excepción de consulta de miembros de la junta directiva actual	
1. El sistema muestra por consola que hay un error al consultar los miembros de la junta directiva actual. 2. Continua → Paso 1 del flujo CU-003-EP-001	
Resultado	- RESU10: El socio consulta los miembros de la junta directiva.

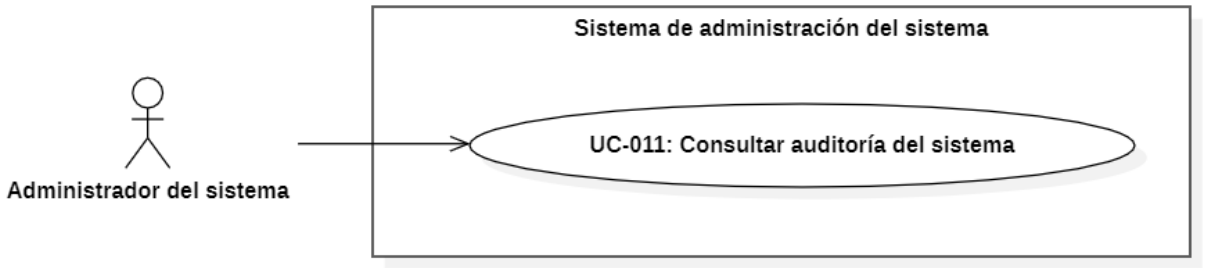
Fuente: Elaboración propia

4.4.4. Sistema de administración del sistema

Este sistema se encarga permitir al administrador del sistema poder consultar la auditoría de interacción con el sistema.

Los casos de uso detectados para este sistema son los siguientes:

Figura 20. Diagrama de casos de uso del Sistema de gestión de perfil de socio.



Fuente: Elaboración propia con *StartUML*

4.4.4.1. UC-011 Consultar auditoría del sistema

Tabla 16. *Especificación de caso de uso UC-011 Consultar auditoría del sistema.*

Identificador	UC-011
Nombre del caso de uso	Consultar auditoría del sistema
Requisitos relacionados	- RF0029, RF0030, RF0031
Actores	S4 – Administradores
Precondiciones	- PRE08: Estar autenticado en el sistema como administrador.
Escenario principal: CU-011-EP-001 Flujo principal de consulta de información de la asociación	
1. El usuario indica que quiere ir a la consulta de información de la auditoría. 2. El sistema audita la petición de consulta de información de la auditoría. 3. El sistema muestra la información de la auditoría a nivel de arranque, de inicio y cierre de sesión y las peticiones realizadas al sistema.	
Escenario excepción: CU-011-EE-001 Flujo de excepción de consulta de información de la asociación	
1. El sistema muestra por consola que hay un error al consultar la información de la auditoría. 2. Continua → Paso 1 del flujo CU-010-EP-001	
Resultado	- RESU11: El administrador consulta la información de la auditoría.

Fuente: Elaboración propia

5. Diseño del producto

Tras el desarrollo de las especificaciones del producto, en este apartado se aborda la fase de diseño de la solución desde una perspectiva más técnica, en el que se detalla la arquitectura del sistema y se define la estructura de los datos.

5.1. Arquitectura del sistema

Al ser una aplicación cliente-servidor, se define una **arquitectura del sistema en tres niveles**, permitiendo organizar la parte física de la parte lógica, el nivel de presentación, el nivel de negocio y el nivel de datos.

Los principales beneficios de esta arquitectura es que cada nivel se ejecuta en su propia infraestructura, se puede actualizar y escalar sin afectar al resto de capas (IBM, 2023), siendo este un último un factor clave en el desarrollo.

Figura 21. *Arquitectura del portal web en tres niveles*.

Fuente: Elaboración propia

5.2. Estructura de los datos

Para guardar la información con la que va a trabajar el sistema, es necesario utilizar una base de datos, permitiendo almacenar, consultar, actualizar y eliminar la información de una manera sistemática y eficiente.

Para que el sistema pueda mantener los estándares de calidad, rendimiento y e integridad, es imprescindible un buen diseño de la base de datos, con una relación apropiada entre las estructuras, y para esta solución se han definido unas colecciones de datos acordes al negocio y la información que maneja, siendo específicas de las asociaciones sin ánimo de lucro, lo que permite tener un software especializado.

5.2.1. Diagrama de Entidad-Relación

El diseño de la base de datos se realiza acorde al negocio de las asociaciones sin ánimo de lucro, teniendo en cuenta que el sistema puede manejar más de una asociación.

Se tiene en cuenta la información propia de un sistema informático que necesita que los usuarios se autentiquen para acceder, como es el caso del portal web y así se ha especificado.

Para cumplir con los estándares de seguridad y requisitos de administración, se registra la información de los arranques y parados del entorno, los accesos y salidas de los usuarios y las peticiones y respuestas realizadas al servidor en las tablas de auditoría.

Para almacenar toda la información, se han definido las siguientes colecciones de datos:

- *Associations*; Contiene la información general de la asociación sin ánimo de lucro a la que el socio pertenece. Todas las *asociaciones* deben disponer de una *junta directiva* actual que gestione la *asociación*.
- *Boardofdirectors*; Contiene la información de la junta directiva actual que gestiona la asociación sin ánimo de lucro.
- *Executives*; Contiene la información del directivo miembro de la junta directiva actual que gestiona la asociación sin ánimo de lucro.
- *Announcements*; Contiene la información del anuncio publicado por la asociación para notificar las noticias a los socios, avisando de eventos, asambleas y cualquier tipo de información que quiera notificar la asociación a sus socios.
- *Partners*; Contiene la información del socio asociado a la asociación sin ánimo de lucro.
- *Membershipfees*; Contiene la información de la cuota de socio que paga el socio por estar afiliado a la asociación sin ánimo de lucro. Cada cuota de socio tiene al menos una situación de la cuota de socio y tiene vinculado la última situación de la cuota de socio.
- *Membershipfeesituations*; Contiene la información de la situación de la cuota de socio que indica el estado del pago en que está la cuota de socio. Cada movimiento en la cuota de socio genera una situación de la cuota de socio indicando el estado del pago.
- *Documents*; Contiene la información del documento de la asociación, pudiendo ser este documento propio de la asociación, del socio, de la cuota de socio, de la situación de la cuota de socio o del anuncio de la asociación.

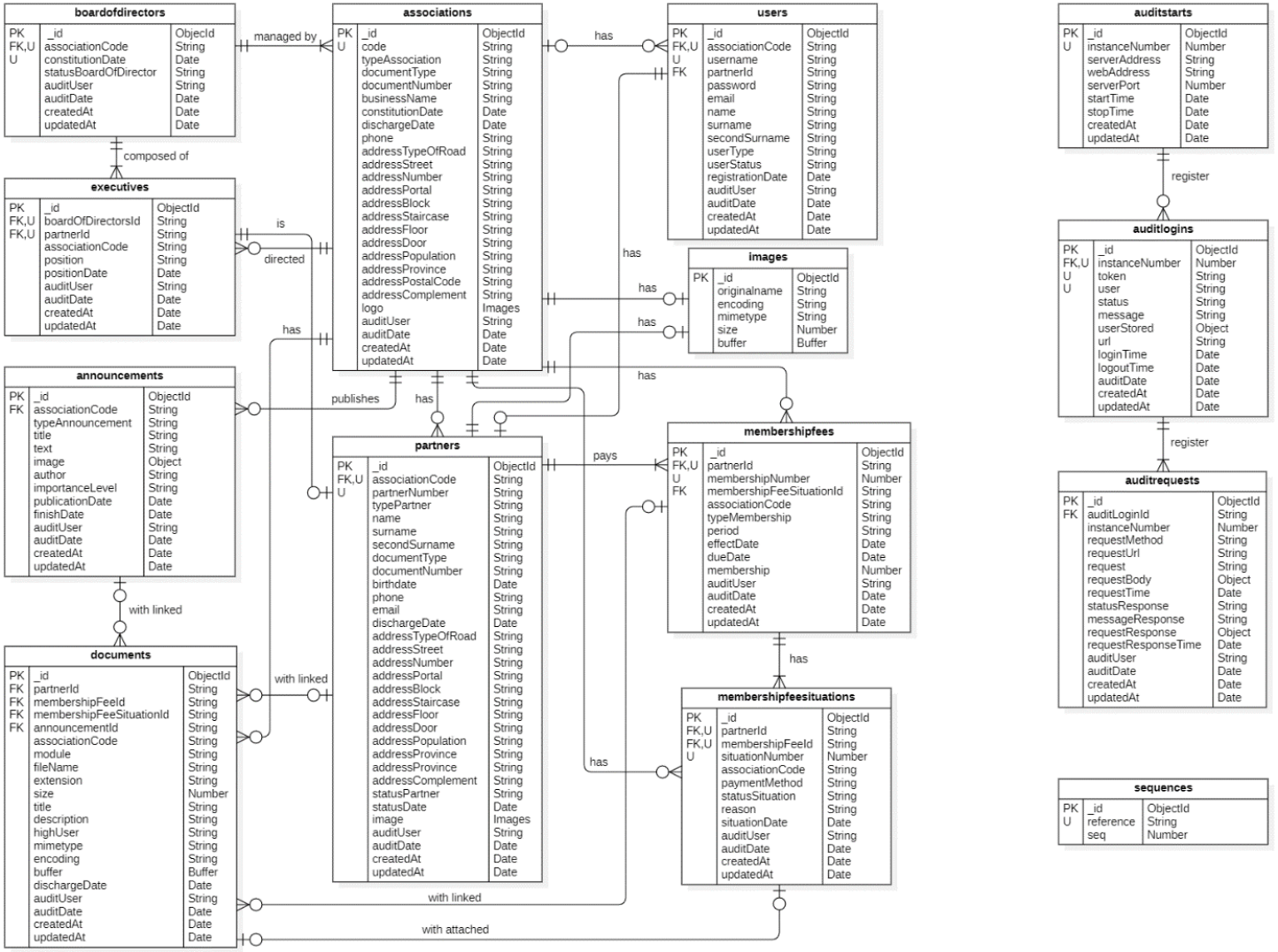
- *Images*; Contiene la información de la imagen del elemento.
- *Users*; Contiene la información del usuario que permite acceder al sistema para consultar la información. Según el tipo de usuario, se accede con un rol al sistema, se puede acceder como rol de socio o como rol de administrador.
- *Sequences*; Contiene la información que representa la auditoría de arranque del sistema donde se registran los arranques y paradas del sistema.
- *Auditstarts*; Contiene la información que representa la auditoría de arranque del sistema donde se registran los arranques y paradas del sistema.
- *Auditlogins*; Contiene la información que representa la auditoría de autenticación de usuario en el sistema donde se registran los accesos y las salidas de los usuarios.
- *Auditrequests*; Contiene la información que representa la auditoría de peticiones al sistema donde se registran la información de la petición y la respuesta del sistema.

Para ver la información concreta que almacena cada colección con el tipo de dato que guarda, se puede ver en el diagrama de entidad-relación del portal, la figura 22.

La clasificación de las colecciones, según la tipología de datos que maneja el sistema, se puede consultar en el apartado de Modelado de la base de datos.

También se puede consultar el Diccionario de datos para ampliar información al respecto de cada colección, el tipo de información que almacena, el tamaño que soporta y los valores aceptados en caso de haber algún tipo de restricción.

Figura 22. Diagrama entidad-relación del modelo de datos del portal del socio.



Fuente: Elaboración propia con *StartUML*

5.2.2. Diccionario de datos

Un diccionario de datos es “una colección de información detallada sobre las entidades de datos que utiliza una aplicación.” (Wiegers & Beatty, 2013).

El diccionario de datos contiene la información sobre las estructuras de negocio y del sistema con una descripción, composición de los atributos, longitud, valores y unidades permitidas, permitiendo al sistema tener un diccionario de las variables que favorece no se repita información y que se nombren a los conceptos de la misma forma.

La tabla con el diccionario de datos se encuentra en el Anexo A. Diccionario de datos.

6. Implementación del producto

Una vez diseñado el producto, en este apartado se presenta la implementación de la solución de portal web para el socio de asociaciones sin ánimo de lucro, detallando las diferentes tecnologías utilizadas y por qué, siguiendo con la explicación de la implementación de la base de datos, el desarrollo de la parte *Back-End* y el desarrollo de la parte *Front-End*.

6.1. Tecnologías utilizadas

Partiendo de los requisitos funcionales y sobre todo los requisitos no funcionales, que priorizan en todo momento el objeto social de las asociaciones sobre tener una tecnología compleja y costosa, buscan que la solución sea accesible para el mayor número de asociaciones, centrándose más en aquellas asociaciones con recursos más limitados que no se pueden permitir soluciones tecnológicas más adaptadas, pero con un mayor coste. Esta solución les permitirá cubrir el servicio de gestión y trámites con los socios de la asociación.

Con este enfoque, se investiga las posibles tecnologías de desarrollo web y se explora la gama de *Stacks* de desarrollo porque permiten construir y ejecutar una única aplicación y este concepto tiene buen encaje con lo que se busca implementar.

Un *Stack* es un conjunto de tecnologías, herramientas y lenguajes de programación que tienen buena interacción entre sí y que se utilizan para crear una aplicación o un sitio web, aportando la base de datos, el servidor web y el lenguaje de programación del lado servidor y del lado cliente (Kubo Mobile, 2023).

La implementación de una aplicación web utilizando un *Stack* tiene los siguientes factores que favorecen su elección (Kubo Mobile, 2023):

- **Eficiencia en el desarrollo** porque acelera el proceso de desarrollo al proporcionar librerías y marco de trabajo adecuados.
- **Buen rendimiento** del software porque son tecnologías optimizadas y preparadas para trabajar entre sí.
- **Escalabilidad** del sistema adaptándose al aumento de usuarios y datos.
- **Facilidad en el mantenimiento** porque los *Stacks* están son soluciones que están de moda, teniendo una comunidad de desarrolladores muy activa.
- Al cumplir los requisitos anteriores, mejora la **satisfacción del cliente**.

Se investiga la lista de *Stacks* para escoger el más adecuado. En el mundo de los *Stacks* de desarrollo, existe una lista amplia y en este trabajo se deduce a los que se ha considerado que actualmente son los más utilizados, según la información consultada en *stackoverflow.com*:

- ***Stack LAMP***: *Linux + Apache + MySQL + PHP / Pearl / Python*
- ***Stack JAMstack***: *JavaScript + API*[vii] *+ Markup*
- ***Stack Python Django***: *Python + Django + Apache + MySQL*
- ***Stack MEAN***: *MongoDB + Express + Angulas + Node.js*
- ***Stack MERN***: *MongoDB + Express + React + Node.js*

De los *Stacks* anteriores, los más populares y que más se adaptan para el desarrollo web son *MEAN Stack* estando más enfocado al desarrollo en la nube y *MERN Stack* estando más enfocado al desarrollo de aplicaciones avanzadas y dinámicas (UNIR, 2022), sobre estos dos se profundiza en los datos del uso de las tecnologías actuales para verificar que las tendencias en el desarrollo abalan la decisión tomada.

Según una encuesta realizada por el portal web *stackoverflow.com* en 2023 a 90.000 desarrolladores, el **lenguaje de programación más utilizado es *JavaScript* con un 63,61%** de los encuestados, siendo un lenguaje que permite programar tanto en cliente como en servidor, los cuales abalan el uso de algunos de los dos *Stack* a usar.

En esta misma encuesta, **los marcos y tecnologías web más utilizadas son *Node.js* siendo la primera con un 42.65%, *React* la segunda con un 40.58%** y ***Angular* la cuarta con un 17.46%**, lo cual hace pensar que son tecnologías en auge, teniendo una serie de componentes y funcionalidades ya construidas que aceleran el desarrollo y mantenimiento del portal, encajando perfectamente con en las necesidades tecnológicas buscadas.

También se obtiene de la encuesta sobre las bases de datos candidatas a utilizar y **la base de datos MongoDB está en la cuarta posición de las bases de datos más usadas con un 25,52%** de 33 que hay en la lista, teniendo una buena posición en el uso.

Teniendo en cuenta los factores favorables de usar un *Stack*, las ventajas que proporcionan en el desarrollo el uso de los *Stacks*, las tendencias en el desarrollo web y la orientación necesaria al desarrollo de aplicaciones avanzadas y dinámicas, se utiliza ***MERN Stack*** porque encaja con la necesidad del proyecto.

Figura 23. *MERN Stack*

Fuente: Artículo *Stack* tecnológicos: ¿Qué son los tech stack y cómo se construyen? (freelancermap, 2023)

6.1.1. MERS Stack

El encaje concreto de *MERN Stack* en los requisitos no funcionales y que confirma que la tecnología escogida es adecuada para el desarrollo:

- **Web**: *MERN Stack* está orientado a desarrollos webs.
- **Seguro**: El conjunto de tecnologías se utiliza para construir un sistema seguro además de implementar funcionalidades para tener un sistema seguro, como es el uso de la autenticación y autorización basada en Token.
- **Disponible**: Esta tecnología favorece la disponibilidad al tener la capacidad de escalar de forma eficiente, además de permitir usar herramientas de monitoreo y gestión para detectar posibles problemas de disponibilidad.
- **Compatible**: La tecnología que conecta el desarrollo cliente con el desarrollo en servidor es mediante la arquitectura *API RESTful*[3], esto es una tecnología que favorece la integración de sistemas, al exponer funciones *CRUD*[viii] sobre los objetos de negocio, como se necesita en los elementos de asociación, noticia, cuota, etc.
- **Genérico**: El sistema está pensado para hacer un desarrollo con su propio modelo de datos y puede desplegarse el sistema para una asociación o para varias, según la necesidad de organizaciones que lo utilicen.
- **Mantenible**: Al ser un *Stack* basado en el *framework*[4] *Express* tiene una arquitectura *API RESTful* que simplifica el desarrollo. También cuenta con una amplia comunidad de

[3] La API RESTful es una interfaz que dos sistemas de computación utilizan para intercambiar información de manera segura a través de Internet
[4] Un framework es una estructura organizada con un conjunto de reglas para el desarrollo software, cuyo objetivo es simplificar el trabajo.

desarrollares y de información en internet, que permite mantener y evolucionar la solución, pudiendo reutilizar componentes libres creados por la comunidad.

- **Escalable**: La utilización de una base de datos no relacional como MongoDB proporciona mucha flexibilidad en el manejo de datos no siendo un cuello de botella, como sucede en algunas de las soluciones con bases de datos relacionales. MongoDB permite escalar de manera eficiente de forma horizontal. También es un punto importante que la comunicación con el *Back-End* sea vía *API RESTful* teniendo como límite de peticiones lo que pueda soportar el servidor y si se despliega esta solución en un modelo Cloud, estos permiten escalar la capacidad de procesamiento para adaptarse a las necesidades del servicio.
- **Actuable**: La capacidad de escalabilidad que proporciona esta tecnología permiten que el sistema implemento pueda actuar sin problemas de rendimiento pudiendo tener un número elevado de usuarios concurrentes utilizando el sistema.
- **Portable**: Al ser una solución web permite ser utilizada en cualquier dispositivo que tenga un navegador y conexión a internet.
- **Usable**: El usar la biblioteca[5] *React*, basada en el lenguaje *JavaScript*, permite crear componentes e interfaces de usuario dinámicas, adaptables y reutilizables, siendo un punto importante que favorece la experiencia de usuario.
- **Sostenible**: Al ser una solución genérica, hará que este sistema sea sostenible, apoyado por la orientación de tener un mantenimiento sencillo que no se requiere un equipo de desarrolladores especialistas, apoyado en un gran catálogo de componentes reutilizables que favorecen esta filosofía.
- **Código abierto**: El código de esta solución está en un repositorio de código abierto accesible desde internet y está bajo la licencia de *Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International*, permitiendo ser utilizado libremente mientras no se realice un uso comercial.
- **Compliance**: Las capacidades de seguridad, gestión segura de datos que proporciona la base de datos, el cumplimiento de estándares en el desarrollo, unido con la

[5] Una biblioteca o librería es una herramienta que proporciona fragmentos de código reutilizables que permiten programar funciones de forma más rápida, sencilla y eficiente.

implementación de elementos de auditoría y registros favorecen el cumplimiento de este criterio.

6.1.2. JavaScript

El lenguaje de desarrollo de la tecnología *MERN Stack* tanto en el lado del cliente, en el servidor y en el uso de la base de datos es *JavaScript*.

"*JavaScript* es un lenguaje de programación o de secuencias de comandos que permite implementar funciones complejas en páginas web, permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes y prácticamente todo lo demás." (MDN Web Docs, 2024).

Figura 24. *Logo JavaScript*

Fuente: Página Web
https://upload.wikimedia.org/

Tener un único lenguaje favorece a poder tener un mantenimiento del sistema más homogéneo y más facilidad y compatibilidad entre componentes, mejorando la eficiencia de la aplicación y el rendimiento al ser una tecnología altamente integrable por su arquitectura basada en *API RESTful*, sin olvidar el beneficio que tener una comunidad muy activa como son las comunidades de los *Stacks* como este.

6.1.3. MongoDB

La base de datos para almacenar la información del portal web es *MongoBD*, que representa la "M" de *MERN Stack*.

"*MongoDB* es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado." (MongoDB, 2024).

Se almacena la información en colecciones de datos indexadas, lo que en un modelo relacional serían las tablas de base de datos, cada elemento de la colección es un documento flexible similar a JSON[ix]. Permite interactuar con la base de datos en tiempo real y el lenguaje de interacción es *JavaScript*, encajando perfectamente en el *Stack* de desarrollo.

6.1.4. MongoDB Compass

Para la consulta de la información en la base de datos, el desarrollo y las pruebas del portal, se utiliza la herramienta de *MongoDB Compass*.

"*Compass* es una herramienta interactiva gratuita para consultar, optimizar y analizar los datos en *MongoDB*. Es la interfaz gráfica de usuario desarrollado por y para *MongoDB*.

Compass proporciona de todo, desde el análisis de esquemas hasta la optimización de índices y pipelines de agregación en una única interfaz centralizada." (MongoDB, 2024).

6.1.5. Node.js + Express

Node.js es el entorno de ejecución para poder programar la parte del servidor del portal web en lenguaje *JavaScript*, porque *JavaScript* es un lenguaje orientado para la parte de la programación en cliente. Forma parte de las tecnologías de *Stack* de desarrollo, siendo la "N" de *MERN Stack*.

"*Node.js®* es un entorno de ejecución de *JavaScript* multiplataforma, de código abierto y gratuito que permite a los desarrolladores crear servidores, aplicaciones web, herramientas de línea de comandos y scripts." (Nodejs, 2024).

Express es el *framework* utilizado para el desarrollo del portal web y que forma parte de las tecnologías de *Stack* de desarrollo, siendo la "E" de *MERN Stack*.

"Express es un *framework* web transigente, escrito en *JavaScript* y alojado dentro del entorno de ejecución *NodeJS*. El módulo explica algunos de los beneficios clave de este *framework*, como configurar tu entorno de desarrollo y como realizar tareas comunes en desarrollo y publicación web." (MDN Web Docs, 2024).

La unión de *Express* con *Node.js* es muy popular en la actualidad, además de proporcionar bastantes ventajas, porque *Express* es un *framework* de desarrollo en *JavaScript*, el cual se puede correr en *Node.js* aprovechando muchas funcionalidades ya implementadas para construir la lógica del servidor, teniendo ya componentes para trabajar con bases de datos como *MongoDB* y otras muchas funcionalidades, lo que acelera el desarrollo, dota de seguridad al sistema al utilizar funciones ya probadas y validadas por una gran comunidad de desarrolladores, entre otras ventajas.

6.1.6. React

La programación en la parte de cliente del portal web se realiza a través de la librería de React, que permite crear componentes reutilizables para construir las partes de la página web. Forma parte de las tecnologías de *Stack* de desarrollo, siendo la "R" de *MERN Stack*.

"*React* es la biblioteca para interfaces de usuario web y nativas. Permite crear interfaces de usuario a partir de piezas individuales llamadas componentes escritas en *JavaScript*." (React, 2024).

Los componentes de React son funciones en *JavaScript* que se puede reutilizar en diferentes partes del portal web, teniendo una sintaxis llamada JSX[x], que permite crear, mantener y eliminar los componentes, además de proporcional la lógica de renderizado de las partes de las páginas web del portal.

6.1.7. Visual Studio Code

Para la programación del código del portal web, tanto de la parte servidor como de la parte cliente, se ha utilizado del IDE[xi] de desarrollo *Visual Studio Code*.

"*Visual Studio Code* es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Incluye compatibilidad integrada con *JavaScript*, *TypeScript* y *Node.js*, y cuenta con un amplio ecosistema de extensiones para otros lenguajes y entorno de ejecución (como *C++*, *C#*, *Java*, *Python*, *Go*, *.NET*)." (VisualStudio, 2024).

***Figura 25**. Logo VSC.*

Fuente: Página Web

https://www.pngegg.com/en/png-veytl

Visual Studio Code tiene compatibilidad integrada para desarrollar en *JavaScript* con *Node.js*, lo que lo hace un entorno de desarrollo idóneo para codificar la lógica del portal web.

6.1.8. Postman

El intercambio de información entre el nivel de presentación, la parte del cliente, con el nivel de aplicación, la parte del servidor donde está la lógica de negocio, es a través de la tecnología *API RESTful*.

Para realizar las pruebas de funcionalidad de la parte del servidor se ha utilizado la herramienta de *Postman*, esta herramienta permite probar la lógica de negocio en el servidor sin necesidad de tener implementado o necesitar la lógica de cliente.

Figura 26. *Logo Postman.*

Fuente: Página Web

https://www.postman.com/

"*Postman* es una plataforma API para crear y utilizar API. *Postman* simplifica cada paso del ciclo de vida de la API y agiliza la colaboración para que puedas crear mejores API y más rápido." (Postman, 2024).

6.1.9. GitHub

Para el desarrollo del portal web, es necesario crear un proyecto con base de código programado y para la gestión del código es recomendable utilizar un repositorio de código.

Utilizar un repositorio de código aporta al desarrollo un control de versiones y de cambios, lo que permitirá consultar todo el histórico de cambios lo cual es una gran ventaja. También favorece al desarrollo colaborativo si el repositorio de código está en la nube. Sirve como un respaldo por si el código actual se corrompe, pudiendo recuperar la versión que se necesite. Permite integrarse con herramientas de pruebas lo que favorece a la calidad del código y del sistema y permite ser compartido con otros miembros de la comunidad.

En este proyecto, el repositorio de código utilizado es GitHub en la versión gratuita y que permite subir y bajar el código de la nube. Se ha integrado con *Visual Studio Code* y esto ha permitido tener un entorno de desarrollo completo y seguro a nivel de código, y mediante comandos se ha podido trabajar con el repositorio.

Figura 27. *Logo GitHub.*

Fuente: Página Web
https://1000logos.net/

Para concretar, "*GitHub* es una plataforma donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código." (GitHub, 2024).

6.2. Modelado de la base de datos

Dentro de la tipología de datos que va a manejar el sistema, se pueden diferencias 3 bloques de tipos de datos, los datos propios del negocio de las asociaciones sin ánimo de lucro, los datos del sistema y los datos de auditoría.

Se utiliza el término de colecciones en lugar de tablas porque en *MongoDB*, que es la BBDD[xii] del portal web, es una base de datos no relacional orientada a colecciones de datos.

6.2.1. Colecciones de negocio

Las colecciones de datos de negocio son aquellas que van a almacenar los datos de negocio, como su propio nombre indica, y son los datos relacionados con las asociaciones sin ánimo de lucro, como son:

- *Associations*
- *Boardofdirectors*
- *Executives*
- *Partners*
- *Membershipfees*
- *Membershipfeesituations*
- *Announcements*
- *Documents*
- *Images*

6.2.2. Colecciones del sistema

Las colecciones de datos de sistema son aquellas que van a almacenar los sistemas de información, siendo colecciones que probablemente la mayoría de sistemas de información manejen, independientemente del negocio al que se dedique la aplicación:

- *Users*
- *Sequences*

6.2.3. Colecciones de auditoría

Las colecciones de datos de auditoría son aquellas las que se dedican a registras la interacción de los actores del sistema con el propio sistema, y estarán definidas en aquellos sistemas que quieran cubrir un mínimo de seguridad, como es el caso del portal web:

- Auditstarts
- Auditlogins
- Auditrequests

6.3. Desarrollo del Back-End

El *Back-End* es el nivel de aplicación donde está la lógica de negocio, dentro de la arquitectura de cliente-servidor, es el servidor.

La primera parte del desarrollo como tal ha sido realizar la parte *Back-End*, porque permite construir la estructura sobre lo que se asienta el proyecto. Para la implementación se han utilizado distintas herramientas como *Visual Studio Code* para la programación, *MongoBD Compass* para la consulta de información en la base de datos, *Postman* para las pruebas de funcionamiento de los puntos de acceso al sistema y *GitHub* para almacenar el código generado en la nube, todas ellas vistas en el apartado de Tecnología utilizadas.

En la preparación del entorno de desarrollo y del proyecto, se han instalado algunos módulos importantes para la consecución del proyecto, estando detallados de manera más específica en el *README.md* del repositorio.

Algunos de los módulos instalados y que han proporcionado diferentes funcionalidades necesarias para el desarrollo del sistema son por ejemplo "`mongoose`" para interactuar con la base de datos, "`body-parser`" para parsear[6] el cuerpo de una petición recibida,

[6] Parsear es el proceso de analizar una cadena de texto para identificar su estructura sintáctica y extraer información significativa de ella.

"`bcryptjs`" para encriptar las contraseñas de los usuarios, "`jsonwebtoken`" para generar *tokens*[7] de autenticación, "`cors`" para controlar el acceso al sistema, "`multer`" para tratar ficheros y uno de los más importantes es "`express`" que el framework de desarrollo que proporciona métodos y propiedades para manejar peticiones HTTP. Estos módulos han acelerado el desarrollo y facilitado la interacción entre distintos elementos del sistema.

El fichero principal de la parte del servidor es el "`index.js`" porque es donde se define la estructura del entorno. La configuración de valores dinámicos se hace a través de variables de entorno, pero si estás no está definidas, el sistema tiene unos valores por defecto definidos en un fichero de constantes.

- Se implementa la lógica del servidor a nivel de nombre del servidor, del nombre del host, del puerto de escucha, del protocolo, etc.
- También se implementa la lógica de actuación del *CORS*[xiii], siendo "un mecanismo basado en cabeceras HTTP que permite a un servidor indicar cualquier dominio, esquema o puerto con un origen distinto del suyo desde el que un navegador debería permitir la carga de recursos" (MDN Web Docs, 2024), para definir el protocolo, nombre del host configurado para el CORS y el puerto.
- Se incluye las rutas de acceso a los *endpoints*[8] de las API expuestas para ser consumidas por el *Front-End*.

```
./routes/administratorSession
./routes/session
./routes/user
./routes/association
./routes/partner
./routes/membershipFee
./routes/membershipFeeSituation
./routes/boardOfDirectors
./routes/executive
./routes/announcement
./routes/document
/
```

[7] Los tokens contienen información de autorización, pero no información de identidad. Se usan para autenticar y proporcionar información de autorización en sistemas de información.
[8] Endpoint de API es el lugar donde se realizan esas solicitudes para consumir la lógica del recurso expuesto.

- Por último, se implementa la lógica para detectar el arranque y parada del servidor para registrarlo en la colección de auditoría pertinente.

La lógica del servidor se ha divido en 3 partes principalmente, hay más funcionalidad, pero toda ellas son para acompañar a estos elementos:

- Modelos
- Controladores
- Enrutamiento

El sentido de la distribución anterior es que el servidor recibe una petición a través de las rutas expuestas. Cada ruta representa una función con una lógica determinada y esta es resuelta por el controlador encargado de resolver esa lógica. La petición demandará algún tipo de acción de consulta, creación, actualización o eliminación de información, esta acción se realizará a través del modelo pertinente que representa la colección de datos de la base de datos. Esto permite encapsular en distintas partes y elementos la lógica completa del servidor.

6.3.1. Modelos

Los modelos son representación de cada colección de la base de datos descritas anteriormente, habiendo un fichero de script por cada colección.

Por mantener organizado la jerarquía los scripts del repositorio, los ficheros que actúan como modelos están en la carpeta "*models*", y hay uno por cada colección de la base de datos.

- ✓ `server\src\models\announcement.js`
- ✓ `server\src\models\association.js`
- ✓ `server\src\models\boardOfDirectors.js`
- ✓ `server\src\models\document.js`
- ✓ `server\src\models\executive.js`
- ✓ `server\src\models\image.js`
- ✓ `server\src\models\membershipFee.js`
- ✓ `server\src\models\membershipFeeSituation.js`
- ✓ `server\src\models\partner.js`
- ✓ `server\src\models\sequence.js`
- ✓ `server\src\models\user.js`
- ✓ `server\src\models\audit\auditLogin.js`
- ✓ `server\src\models\audit\auditRequest.js`
- ✓ `server\src\models\audit\auditStart.js`

Cada uno de estos ficheros implementa un objeto de tipo "`Schema`", definiendo cada una de las variables que maneja a nivel de tipo, obligatoriedad, si forma parte de la clave única, relación con otra variable de otro modelo y si se quitan los espacios finales en el valor guardado.

Para mostrar un ejemplo de un fichero de entidad de colección, se adjunta el código de la entidad usuario "`server\src\models\user.js`" donde se muestra lo comentado.

```
const userScheme = new Schema(
  {
    username: {
      type: String,
      required: true,
      trim: true,
      unique: true,
    },
    password: {
      type: String,
      required: true,
    },
    email: {
      type: String,
      required: true,
      trim: true,
      unique: true,
    },
    name: {
      type: String,
      required: true,
      trim: true,
    },
    surname: {
      type: String,
      required: true,
      trim: true,
    },
    secondSurname: {
      type: String,
      trim: true,
    },
    registrationDate: {
      type: Date,
      trim: true,
      default: Date.now,
    },
    userType: {
```

```
      type: String,
      required: true,
      trim: true,
    },
    userStatus: {
      type: String,
      required: true,
      trim: true,
    },
    associationCode: {
      type: String,
      ref: DDBB_CONSTANTS.associationsCollection,
      trim: true,
    },
    partnerId: {
      type: String,
      ref: DDBB_CONSTANTS.partnersCollection,
      trim: true,
    },
    auditUser: {
      type: String,
      required: true,
      trim: true,
    },
    auditDate: {
      type: Date,
      required: true,
      trim: true,
      default: Date.now,
    },
  },
  {
    timestamps: true,
  }
);
```

6.3.2. Controladores

Los controladores contienen la lógica propia de cada elemento al que representan. Los controladores tienen funciones que realizan las acciones sobre el elemento, (como pueden ser el crear, consultar, actualizar o eliminar el elemento) y utilizan de los modelos de las colecciones afectadas para interactuar con la base de datos si es preciso.

Estos ficheros están en la carpeta "*controllers*".

- ✓ `server\src\controllers\announcement.js`
- ✓ `server\src\controllers\association.js`

- ✓ server\src\controllers\boardOfDirectors.js
- ✓ server\src\controllers\document.js
- ✓ server\src\controllers\executive.js
- ✓ server\src\controllers\membershipFee.js
- ✓ server\src\controllers\membershipFeeSituation.js
- ✓ server\src\controllers\partner.js
- ✓ server\src\controllers\session.js
- ✓ server\src\controllers\user.js
- ✓ server\src\controllers\audit\auditLogin.js
- ✓ server\src\controllers\audit\auditRequest.js
- ✓ server\src\controllers\audit\auditStart.js

Para mostrar un ejemplo de un controlador, se adjunta el código de la entidad usuario "server\src\controllers\user.js" donde se muestra las funciones expuestas.

```
// Método para crear un usuario
const create = async (req, res, next) => {
  //validateSchema(registerSchema, req, res, next);

  const { email } = req.body;
  const userFound = await UserModel.findOne({ email });
  if (userFound) {
    const response = {
      status: 'error',
      message: `El email ${email} ya está en uso`,
    };
    return sendRequestResponse(req, res, next, 400, response);
  }

  if (req.user) {
    req.body.auditUser = req.user.username;
  }
  const parms = req.body;
  const passwordHash = await bcrypt.hash(
    parms.password,
    USER_CONSTANTS.moduleHash
  );
  parms.password = passwordHash;
  const userModel = new UserModel(parms);
  userModel
    .save()
    .then(async (userStored) => {
      const token = await createAccessToken(userStored);
```

```
      const response = {
        status: 'success',
        message: 'Usuario creado correctamente',
        userStored: getStructureUserResponse(userStored),
      };
      res.cookie('token', token);
      return sendRequestResponse(req, res, next, '', response);
    })
    .catch((err) => {
      const message = `Se ha producido un error al guardar el usuario
${userModel.email}: ${err}`;
      const response = {
        status: 'error',
        message: message,
      };
      return sendRequestResponse(req, res, next, 404, response);
    });
};

// Método para consultar un usuario
const read = async (req, res, next) => {
  const userId = req.params.userId;
  try {
    const userFound = await UserModel.findById(userId);
    if (!userFound) {
      const response = {
        status: 'error',
        message: `No se ha encontrado el usuario: ${userId}`,
      };
      return sendRequestResponse(req, res, next, 404, response);
    }
    const response = {
      status: 'success',
      user: getStructureUserResponse(userFound),
    };
    return sendRequestResponse(req, res, next, '', response);
  } catch (error) {
    const response = {
      status: 'error',
      message: `Error al consultar el usuario ${userId}: ${error}`,
    };
    return sendRequestResponse(req, res, next, 500, response);
  }
};

// Método para actualizar un socio
const update = async (req, res, next) => {
  const userId = req.params.userId;
  try {
```

```
    req.body._id = userId;
    if (req.user) {
      req.body.auditUser = req.user.username;
    }
    const userModel = new UserModel(req.body);
    const userUpdated = await UserModel.findByIdAndUpdate(
      userId,
      userModel,
      {
        new: true,
      },
      {
        _id: userId,
        email: req.body.email,
        name: req.body.name,
        surname: req.body.surname,
        secondSurname: req.body.secondSurname,
        userType: req.body.userType,
        userStatus: req.body.userStatus,
      }
    );
    if (!userUpdated) {
      const response = {
        status: 'error',
        message: `No se ha encontrado el usuario: ${userId}`,
      };
      return sendRequestResponse(req, res, next, 404, response);
    }
    const response = {
      status: 'success',
      user: getStructureUserResponse(userUpdated),
    };
    return sendRequestResponse(req, res, next, '', response);
  } catch (error) {
    const response = {
      status: 'error',
      message: `Error al actualizar el usuario con identificador ${userId}:
${error}`,
    };
    return sendRequestResponse(req, res, next, 500, response);
  }
};

// Método para eliminar un usuario
const remove = async (req, res, next) => {
  const userId = req.params.userId;
  try {
    const userRemoved = await UserModel.findByIdAndDelete(userId);
    if (!userRemoved) {
```

```
      const response = {
        status: 'error',
        message: `No se han encontrado el usuario a eliminar: ${userId}`,
      };
      return sendRequestResponse(req, res, next, 404, response);
    }
    const response = {
      status: 'success',
      message: 'Usuario eliminado correctamente',
      user: getStructureUserResponse(userRemoved),
    };
    return sendRequestResponse(req, res, next, '', response);
  } catch (error) {
    const response = {
      status: 'error',
      message: `Error al eliminar el usuario con código ${userId}: ${error}`,
    };
    return sendRequestResponse(req, res, next, 500, response);
  }
};

// Método para devolver la lista de asociaciones
const getusers = async (req, res, next) => {
  const associationCode = req.user.associationCode;
  try {
    let users = null;
    if (associationCode) {
      users = await UserModel.find(
        { associationCode: associationCode },
        '_id username email name surname secondSurname registrationDate type
status associationCode'
      ).sort('-registrationDate');
    } else {
      users = await UserModel.find(
        {},
        '_id username email name surname secondSurname registrationDate type
status associationCode'
      ).sort('-registrationDate');
    }
    const response = {
      status: 'success',
      users: users,
    };
    return sendRequestResponse(req, res, next, '', response);
  } catch (error) {
    const response = {
      status: 'error',
      message: `Error al consultar la lista de usuarios: ${error}`,
    };
```

```
    return sendRequestResponse(req, res, next, 500, response);
  }
};
```

6.3.3. Enrutamiento

Los enrutadores son los puntos de acceso al sistema expuestos para permitir invocar la lógica requerida, están enfocados a cada una de las acciones que se puede realizar sobre el elemento que representan tenga un punto de acceso.

Estos ficheros están en la carpeta "*routes*".

- ✓ `server\src\routes\administratorSession.js`
- ✓ `server\src\routes\announcement.js`
- ✓ `server\src\routes\association.js`
- ✓ `server\src\routes\boardOfDirectors.js`
- ✓ `server\src\routes\document.js`
- ✓ `server\src\routes\executive.js`
- ✓ `server\src\routes\membershipFee.js`
- ✓ `server\src\routes\membershipFeeSituation.js`
- ✓ `server\src\routes\partner.js`
- ✓ `server\src\routes\session.js`
- ✓ `server\src\routes\user.js`

Cada uno de estos ficheros expone métodos como el crear, el consultar, el actualizar y el eliminar, que son las operaciones genéricas de los elementos que maneja el sistema (*CRUD*).

Para mostrar un ejemplo de enrutador, se adjunta el código de la entidad usuario "`server\src\routes\user.js`" donde se muestra las operaciones expuestas.

```
const userRouter = express.Router();
userRouter.post(
  "/usuarios/",
  authRequiered,
  auditRequestInput,
  create,
  auditRequestOutput
);
userRouter.get(
  "/usuarios/:userId",
  authRequiered,
  auditRequestInput,
```

```
  read,
  auditRequestOutput
);
userRouter.put(
  "/usuarios/:userId",
  authRequiered,
  auditRequestInput,
  update,
  auditRequestOutput
);
userRouter.delete(
  "/usuarios/:userId",
  authRequiered,
  auditRequestInput,
  remove,
  auditRequestOutput
);
userRouter.get(
  "/usuarios/",
  authRequiered,
  auditRequestInput,
  getusers,
  auditRequestOutput
);
```

6.3.4. Registro de auditoría

Para explicar cómo se realiza la validación de que el usuario está autenticado en el sistema y de cómo se registran las peticiones que llegan al servidor, se va a utilizar el elemento de sesión, siendo un elemento que maneja el sistema y que expone dos operaciones, la creación de la sesión que se invocará cuando un usuario quiere iniciar sesión en el sistema y la eliminación de sesión, que se invocará cuando un usuario quiere cerrar sesión en el sistema.

El siguiente código que se muestra es el fichero "`server\src\routes\session.js`" con los puntos de entrada al sistema para las solicitudes sobre la sesión.

```
sessionRouter = express.Router();
sessionRouter.post(
  "/:associationCode/login",
  login,
  auditRequestInput,
  auditRequestOutput
);
sessionRouter.delete(
  "/:associationCode/logout",
  authRequiered,
  auditRequestInput,
```

```
  logout,
  auditRequestOutput
);
```

Como se puede ver en el código anterior, en el método "POST" de creación de la sesión, se llama a la función de "`login`" que proporciona el controlador "`server\src\controllers\session.js`", teniendo la lógica de inicio de sesión.

La petición de autenticación tiene unas funciones *middleware*[9] que se ejecutarán después de la resolución de la autenticación y registrarán la información de entrada y la salida de la solicitud en la colección de auditoría de login.

Para la petición de cierre de sesión, hay una función *middleware* previa que validan que la petición viene desde un usuario autenticado en el sistema, otra función *middleware* previa que registra la entrada de la solicitud en la colección de la base de datos de auditoría de login y tras resolver la lógica de cierre de sesión, se invoca a la función *middleware* que registra el resultado de la petición de login actualizando el registro de la colección generado anteriormente.

A través de estas funciones middleware, se ha controlado el acceso a todos los recursos expuestos del sistema para que las peticiones se realicen por usuarios registrados y autenticados en el sistema y todas las peticiones queden registradas en las tablas de auditoría, como se puede ver en el código del enrutador del usuario.

Estas funciones *middleware* tienen la lógica pertinente para acometer su tarea en el momento que se le invoca dentro de la resolución de la petición.

6.3.5. Autenticación y token de sesión

Como se muestra en el código del apartado anterior, el sistema tiene la función de inicio de sesión en el sistema, y es importante detallar cómo se realiza es validación.

Primero se valida si el usuario existe y si existe, se comprueba el tipo de usuario que sea de tipo y que sea un usuario válido frente a la asociación actual, si no es así, no se puede acceder al sistema.

[9] Middleware en Express es una función que se puede ejecutar antes o después del manejo de una ruta.

Después se valida la contraseña que viaja encriptada en la petición y se desencripta en el servidor, para garantizar mayor seguridad.

Si la contraseña es correcta y el usuario está activo, quedará autenticado en el sistema, el cual le proporciona un token de acceso que se envía al cliente en forma de cookie de sesión.

Si no se cumple todo lo anterior, el usuario no quedará autorizado. Caso distinto es el usuario administrador, que no se valida que sea usuario de tipo socio ni que esté vinculado a la asociación, pero tiene un punto de entrada propio distinto al del usuario de tipo socio.

6.3.6. Multi asociación

Uno de los factores diferenciadores y que permite que el sistema versátil es poder tener cargada información de múltiples asociaciones, con su información relacionada de socios, junta directiva, etc., en un solo despliegue.

Para asegurar que la información consultada es propia de cada asociación y no hay cruce de datos, el código de la asociación forma parte de los parámetros de la ruta de acceso a la solicitud realizada y este código es utilizado en cada acceso a la base de datos para crear, consultar, modificar o eliminar información propia del sistema. Además de validar en cada petición que la asociación a la que pertenece el usuario autenticado es la que está llegando en la petición, teniendo un aseguramiento total de que no se produce un cruce de información ni un acceso a información no autorizado.

Para mostrar un ejemplo de cómo se ha implementado esto, en el controlador del socio "`server\src\controllers\partner.js`", el método "`getPartners`" que devuelve todos los socios obtiene el código de asociación que viene como parámetro en la petición. Este control se realiza en casi todas las peticiones del sistema, además de obtener el resto de parámetros para selecciona el elemento sobre el que realizar la acción.

```
// Método para consultar los socios de la asociación
const getPartners = async (req, res, next) => {
  try {
    const partners = await PartnerModel.find({
      associationCode: req.params.associationCode,
    }).sort('-dischargeDate');
    const response = {
      status: 'success',
      partners: partners,
    };
```

```
    return sendRequestResponse(req, res, next, '', response);
  } catch (error) {
    const response = {
      status: 'error',
      message: `Error al consultar los socios: ${error}`,
    };
    return sendRequestResponse(req, res, next, 500, response);
  }
};
```

6.3.7. Lista de puntos de acceso disponibles

La tabla el listado de puntos de acceso disponibles se encuentra en el Anexo B. Puntos de acceso disponibles.

6.4. Desarrollo del Front-End

El *Front-End* es el nivel de presentación donde está la lógica de interfaz, dentro de la arquitectura de cliente-servidor, es el cliente.

Esta parte del desarrollo se ha realizado después del *Back-End*, una vez está las los API de acceso a la lógica del sistema definidos e implementados, así ha permitido centrase en la lógica de la capa de presentación.

Para la implementación se han utilizado distintas herramientas como *Visual Studio Code* para la programación y *React* para implementar los componentes de las páginas del portal, vistas en el apartado de Tecnología utilizadas.

Como ha sucedido en la parte del servidor, en esta parte también se han instalado algunos módulos importantes para la consecución del proyecto y también están detallados de manera más específica en el *README.md* del repositorio.

Algunos de los módulos instalados son por ejemplo "`node-fetch`" realizar las llamadas HTTP (API) al servidor, "`js-cookie`" leer y trabajar con la cookie de sesión, "`react-hook-form`" para la gestión de estado de los componentes y "`react-router-dom`" para la navegación fluida del portal web en una sola página.

Importante destacar que **para implementar la lógica del cliente se ha adaptado una plantilla descargada desde el sitio web https://themeforest.net/**, concretamente la plantilla "*Dashra - React Admin Template*" https://themeforest.net/item/dashra-react-admin-

template/48837630, bajo la licencia regular la cuál no permite cobrar a los usuarios finales de la aplicación, no siendo impedimento esto porque este no es el objetivo del portal web.

Esta decisión ha permitido ahorrar tiempo en el desarrollo, tener una consistencia y estándar en el diseño de la interfaz gráfica, se ha reutilizado los componentes ya preconstruidos que traída la plantilla adaptados a la necesidad concreta del portal web del socio, siendo aprovechada la comunidad de desarrollo que aporta esta librería y, sobre todo, acelerar el proceso de desarrollo, garantizando la calidad y consistencia del código al ser una base sólida de desarrollo.

El componente principal de la parte del cliente es el "`main.jsx`", aunque dentro tiene el componte "`App.jsx`" que a su vez tiene el componente "`Router.jsx`" que define la estructura de pantallas de la presentación.

6.4.1. Enrutamiento

El enrutamiento en el lado del cliente es la navegación por los distintos componentes de la aplicación, y cuando se quiere acceder a un componente específico, el sistema de enrutamiento a través de la URL indicada, muestra la información del componente, esto significa que cada componente debe tener asignado una URL.

Las URL del portal web son las siguientes:

```
➢ /
➢ /login
➢ /logout
➢ '/' + association.code + '/login'
➢ '/' + association.code + '/logout'
➢ '/' + association.code + '/perfil-socio'
➢ '/' + association.code + '/perfil-socio' + '/noticias'
➢ '/' + association.code + '/perfil-socio' + '/cuotas'
➢ '/' + association.code + '/perfil-socio' + '/documentos'
➢ '/' + association.code + '/asociacion'
```

Es importante destacar que las URL del portal web son dinámicas en función de las asociaciones almacenadas en la base de datos. El sistema consulta la lista de asociaciones y genera de manera dinámica el árbol de navegación por cada asociación, permitiendo acceder a las diferentes asociaciones del portal web. Esta lógica está implementada en el componente "`Router.jsx`".

```
            <Routes>
              <Route path="/" element={<Home />} />
              <Route element={<LoginLayout />}>
                <Route index path={'/login'} element={<Login />} />
                <Route path={'/logout'} element={<Logout />} />
              </Route>
              {/* Asociaciones */}
              {associations.map((association, index) => (
                <Route key={index}>
                  <Route
                    key={index + 'LoginLayout'}
                    element={<LoginLayout association={association} />}
                  >
                    <Route
                      key={index + '/'}
                      path={'/' + association.code}
                      element={<Home />}
                    />
                    <Route
                      index
                      key={index + '/login'}
                      path={'/' + association.code + '/login'}
                      element={<Login />}
                    />
                    <Route
                      key={index + '/logout'}
                      path={'/' + association.code + '/logout'}
                      element={<Logout />}
                    />
                  </Route>
                  <Route key={index + 'Layout'} element={<Layout />}>
                    <Route
                      key={index + '/perfil-socio'}
                      path={'/' + association.code + '/perfil-socio'}
                      element={<ProfileOverview />}
                    >
                      <Route
                        index
                        key={index + '/noticias'}
                        path={
                          '/' + association.code + '/perfil-socio' +
'/noticias'
                        }
                        element={<Projects />}
                      />
                      <Route
                        key={index + '/cuotas'}
                        path={
                          '/' + association.code + '/perfil-socio' + '/cuotas'
```

```
                    }
                    element={<Activities />}
                  />
                  <Route
                    key={index + '/documetos'}
                    path={
                      '/' +
                      association.code +
                      '/perfil-socio' +
                      '/documetos'
                    }
                    element={<Documents />}
                  />
                </Route>
                <Route
                  key={index + '/asociacion'}
                  path={'/' + association.code + '/asociacion'}
                  element={<Users />}
                ></Route>
              </Route>
            </Route>
          ))}
          <Route key={'*'} path="*" element={<Error />} />
        </Routes>
```

6.4.2. Componentes

Como ya se ha comentado, *React* está basado en componentes que son uno de los conceptos esenciales y son los encargados de construir las distintas partes de la interfaz del portal web. (React.dev, 2024).

Un de componente de *React* es está formado por *JavaScript* para la crear lógica del componente, por JSX siendo una extensión de la sintaxis de *JavaScript* que permite escribir HTML de manera declarativa, y en algunas ocasiones CSS. (React.dev, 2024).

Los principales componentes reutilizados de la plantilla y que están al primer nivel son:

- ✓ `<LoginLayout />`: Componente de página de inicio del portal.
- ✓ `<Login />`: Componente de formulario de inicio de sesión.
- ✓ `<Logout />`: Componente de cierre de sesión.
- ✓ `<ProfileOverview />`: Componente de consulta del perfil del socio.
- ✓ `<Projects />`: Componente de consulta de noticias de la asociación.
- ✓ `<Activities />`: Componente de consulta de cuotas del socio.
- ✓ `<Documents />`: Componente de consulta de los documentos del socio.
- ✓ `<Users />`: Componente de consulta de la asociación y de la junta directiva.

Estos componentes a su vez se pueden dividir en subcomponentes, y si se considera profundizar en la lógica de los componentes, se puede acceder al repositorio de código y revisar el código. Esto proporciona la ventaja de poder reutilizar los distintos componentes en distintas partes del portal o incluso poner otros componentes implementados por terceros.

Una de las características que tienen los componentes es que se pueden comunicar entre sí e interactuar entre ellos, facilitando al desarrollador la integración de los componentes en el sistema, logrando que la aplicación sea más dinámica.

La comunicación de los componentes en React se hace a través de las "`props`" que son los datos que se pasan a un componente (React.dev, 2024). Gracias a este elemento se puede pasar información entre componentes y es la información que proporciona el dinamismo del componente.

A continuación, se muestra un ejemplo del uso de los "`props`" sobre el componente que se ha utiliza para mostrar la información de cada miembro de la junta directiva.

El componente "`UsersCom`" representa a la junta directiva y dentro se recorre la lista de directivos consultada por el servicio de consulta de directivos de la junta directiva, y la lista devuelta se recorre para mostrar la información de cada directivo lo que se hace a través del componente "`UserCard`" al que en cada iteración de la lista se le pasa como "`props`" el objeto de directivo y el identificador de directivo que hace único al componente, siendo esto una propiedad esencial en React por cada componente que puede aparecer varias veces en una pantalla.

```
{executives?.map((executive) => (
      <UserCard executive={executive} key={executive?._id} />
))}
```

En siguiente código muestra como es el componente que se encarga de mostrar en pantalla la información del directivo de la junta directiva y cómo recibe los "`props`", viendo un claro de ejemplo de esta propiedad y de cómo genera dinámicamente las pantallas según la información enviada por el servidor.

```
function UserCard({ executive }) {
  …
}
```

6.4.3. Servicios

Como se ha comentado al describir la tecnología *Postman*, el intercambio de comunicación entre el nivel de presentación y el nivel de aplicación se realiza a través de servicios denominados API web, concretamente del tipo de servicio API REST.

Un API es una interfaz de comunicación entre un servidor web y un navegador web y al ser de tipo REST define un conjunto de funciones como *POST*, *GET*, *PUT* y *DELETE* que los clientes del servicio pueden utilizar para acceder a la información del servidor vía HTTP (Amazon, 2024).

Para el desarrollo *Front-End* que representa la parte del cliente, no se ha utilizado *axios*[10] que es uno de los módulos más utilizadas para llamar a los servicios web expuestos por la parte del servidor, los comentados en el apartado de Desarrollo del *Front-End*, pero no se ha podido utilizar porque el sistema daba un problema de incompatibilidad entre las versiones de módulos que no se puedo solventar, y en su lugar se ha utilizado el módulo *fetch*[11], siendo este módulo la base de *axios*, y que ha permito implementar el fichero de utilidad "`client\src\helpers\requestAPI.js`" que contiene la lógica de la llamadas a los servicios del servidor. Tiene una función que recibe la URL del servicio a invocar y el objeto de la petición. En el siguiente código se puede ver que se usa *fetch*.

```
const request = async (url, objectRequest) => {
  try {
    const response = await fetch(url, objectRequest);
    return await response.json();
  } catch (error) {
    throw error;
  }
};
```

Un ejemplo de uso de esta funcionalidad es la consulta de información del socio del usuario autenticado de tipo socio que está implementada en el fichero "`client\src\apis\partnersAPI.js`":

```
export const getPartnerRequest = async (user, id) => {
  return await requestUserAPI(user, 'GET', '/socios/' + id);
};
```

[10] *Axios*: https://axios-http.com/es/
[11] *Fetch*: https://fetch.spec.whatwg.org/

Este ejemplo llama a la función "`requestUserAPI`" con el usuario que solicita la información, el método del servicio que se consulta y la URL relativa del servicio consultado la cual tiene el id de socio consultado (se obtiene del usuario autenticado de tipo socio).

La función "`requestUserAPI`" monta parte de la URL incluyendo la asociación en la que se está autenticado porque es un parámetro obligatorio en los recursos expuestos y este se obtiene del usuario o de la URL del navegador, y con esto se compone la URL de llamada al servidor incluyendo en primera parte de la cadena la URL del servidor que va a recibir la petición, pudiendo ser configurado este valor por una variable de entorno.

Si viene un mensaje de cuerpo para enviar en la petición, utilizando el método y el cuerpo se compone el objeto de petición y con esta información se llama a la función "`request`" que es la que se ha visto al principio del apartado y que utiliza el módulo de *fetch* para comunicarse con el servidor.

```
export const requestUserAPI = async (user, method, specificUrl, body) => {
  const path = getRouteUser(user, specificUrl);
  const url = '' + API_REST_CONSTANTS.url_api + path;
  const objectRequest = getRequestObject(method, body);
  return await request(url, objectRequest);
};
```

7. Pruebas

El realizar pruebas en el desarrollo de software son la mejor garantía de asegurarse de que el sistema no presenta errores y se comporta como se ha definido, permitiendo controlar la calidad y el correcto funcionamiento (UNIR, 2022).

En el desarrollo del portal web, se han realizado pruebas unitarias en los diferentes componentes y pruebas funcionales.

7.1. Pruebas unitarias

Las pruebas unitarias "Comprueban que cada una de las piezas o unidades más pequeñas del software en el que se está trabajando funcione correctamente. Estas pruebas se aplican de manera individual y son las primeras que deben realizarse durante todo el proceso de desarrollo." (UNIR, 2022), siendo estas un tipo de pruebas funcionales.

Estas pruebas se han realizado sobre los objetos de negocio que se han implementado en la aplicación utilizando la herramienta de *Postman*, la cual ha permitido probar de manera unitaria los diferentes *endpoints* expuestos.

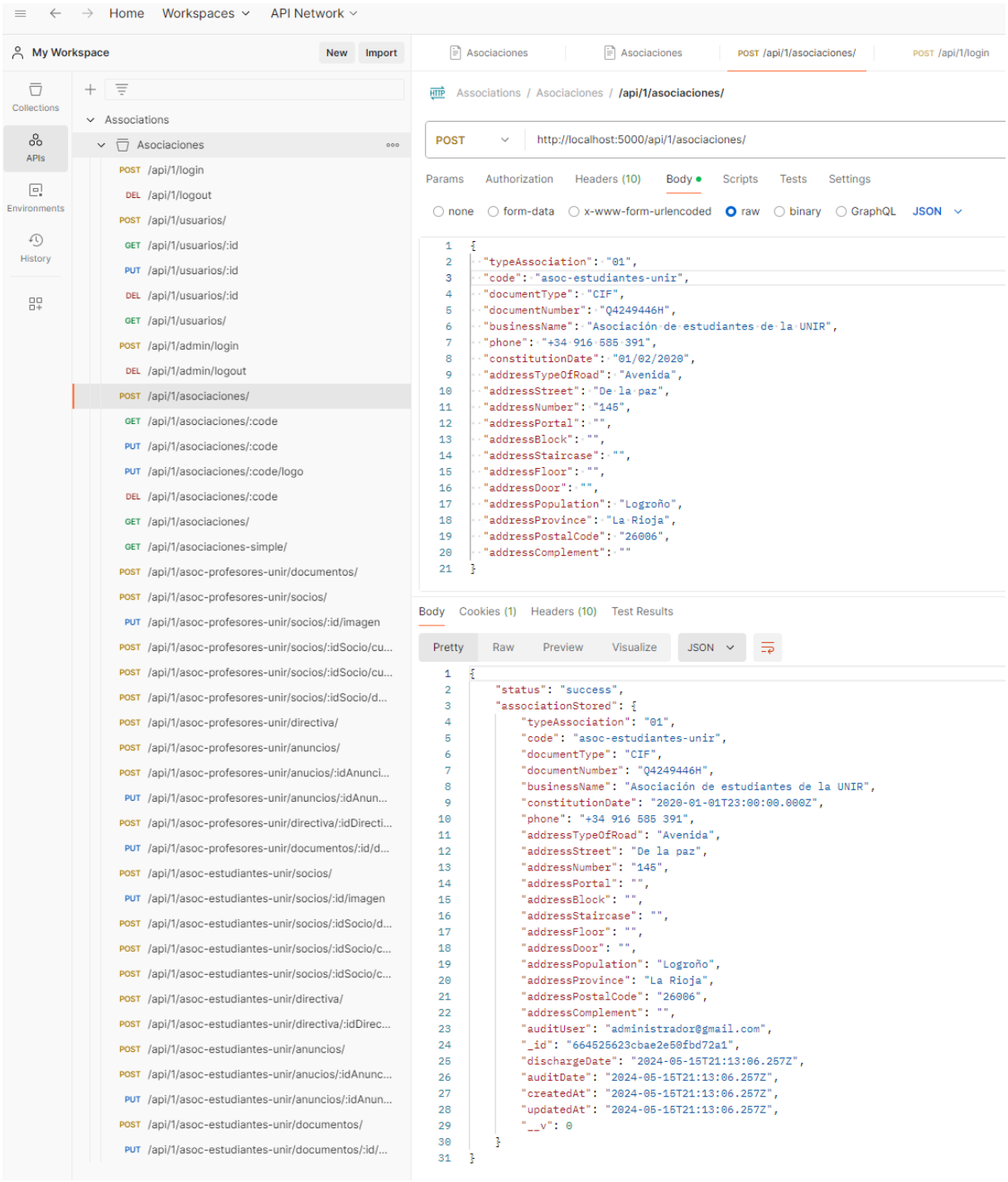
Cada uno de los puntos de acceso descritos en el Anexo B. Puntos de acceso disponibles se ha realizado una batería de pruebas con el *Postman* hasta que se ha obtenido el resultado esperado tanto en la estructura de información enviada como en la respuesta obtenida, además en las operaciones de POST, PUT y DELETE se ha verificado en base de datos a través de la herramienta *MongoDB Compass* que la información resultante era correcta:

A continuación, se muestran figuras con un ejemplo de cómo se han realizado las pruebas unitarias, teniendo que tener el servidor levantado y combinado la herramienta *Postman* con *MongoDB Compass*.

Lo primero es configurar en el *Postman* la información del *endpoint* a probar, indicando la URL y el objeto JSON del cuerpo, además para probar este *endpoints* en necesario estar autenticado en el sistema.

En la figura 28 se muestra un ejemplo de uso del Posta para la realización de una prueba de creación de una asociación.

Figura 28: *Ejemplo de resultado de llamada vía API al servicio de creación de asociación.*

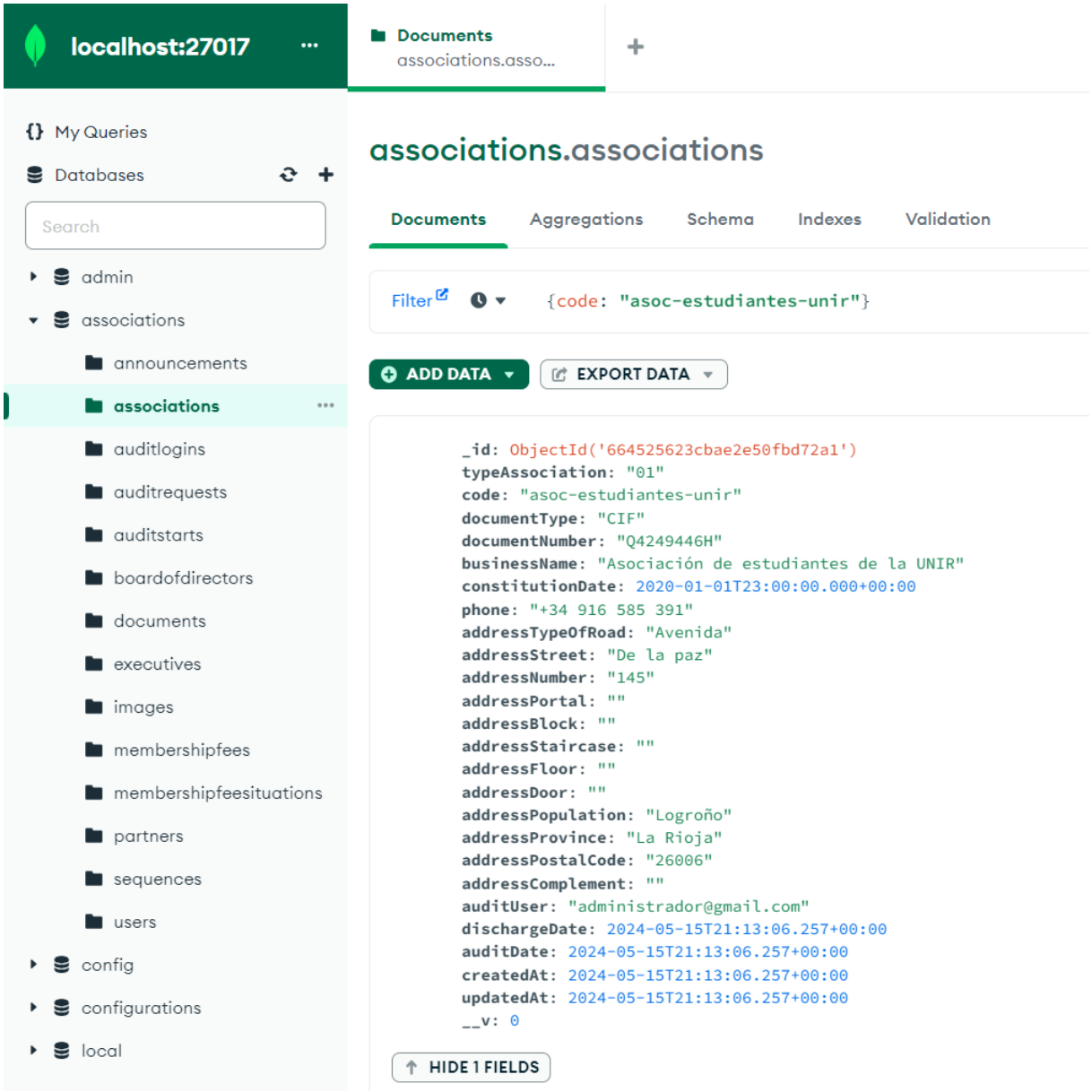


Fuente: Elaboración propia usando la herramienta Postman.

Como se puede ver en la figura anterior, se realiza la llamada al servidor y este devuelve el estado de la petición, siendo correcta y el objeto que ha creado en la base de datos.

La figura 29 muestra como es correcta la información creada y coincide con los datos que ha devuelve el servidor.

Figura 29: *Ejemplo de resultado de asociación registrada en base de datos desde Postman.*



Fuente: Elaboración propia usando la herramienta MongoDB Compass.

Este tipo de pruebas se ha realizado con cada uno de los objetos y métodos que implementa el portal web en el servidor.

En el Anexo C.Pruebas unitarias en puntos de acceso se detallan los puntos de acceso probados con el resultado de la prueba unitaria obtenida.

7.2. Pruebas funcionales

Las pruebas funcionales "comprueban las funciones del software creado para establecer la usabilidad y las características de cara al mercado. Son consideradas pruebas de caja negra o *black-box testing* porque lo que se verifica es el comportamiento del sistema, si todo funciona

tal y como está establecido en el documento *Software Requirement Specification* (SRS)." (UNIR, 2022), en este caso no son establecidas en el SRS si no en el apartado especificación del producto de este proyecto.

Estas pruebas se han realizado en función de los casos de uso definidos. **En el Anexo D. Pruebas funcionales se muestran los resultados de cada prueba funcional** realizada por cada caso de uso.

7.3. El sistema en funcionamiento

El funcionamiento del sistema se puede ver a través del vídeo funcional vinculado mediante un enlace de *YouTube* [12] en la portada del trabajo, donde se presenta funcionalmente el sistema.

[12] *Youtube*: https://www.youtube.com/

8. Conclusiones y trabajo futuro

En este apartado se presentan las conclusiones que se han obtenido tras la realización del trabajo de fin de grado junto con unas posibles líneas de trabajo futuras para evolucionar el sistema.

8.1. Conclusiones del trabajo

Como conclusión general del trabajo y partiendo del objetivo principal de proyecto definido en el apartado de objetivo general, que es proporcionar a las asociaciones sin ánimo de lucro una solución reutilizable a través de una herramienta para que el socio pueda consultar la información relevante de la asociación y realizar trámites como es la justificación del pago de la cuota del socio, permitiendo al socio ejercer una parte importante de sus derechos y obligaciones como marcan los estatutos, y como se puede ver en el video funcional vinculado a este trabajo, **se considera que el objetivo principal se han cumplido totalmente**.

Aunque **el sistema** inicialmente está enfocado al nicho de las asociaciones, **también puede ser utilizado en las diferentes organizaciones sin ánimo de lucro en general**, las cuales podrían **beneficiarse de esta solución y usarla como sistema de comunicación entre el socio y la organización**, lo que **impulsaría la participación de los socios en las actividades que realizan** al tener acceso más fácilmente a las noticias, eventos y otras publicaciones destacadas.

Se puede ver el portal web en funcionamiento a través del vídeo funcional vinculado mediante un enlace de *YouTube* en la portada del trabajo. En este vídeo se muestra y explica la funcionalidad que tiene el portal a través de varios ejemplos.

Partiendo de los objetivos específicos, se detallan las conclusiones obtenidas a nivel de estos objetivos concretos.

Respecto al **análisis del contexto y estado de arte** se ha descubierto posibles nichos de mercado, desarrollos disponibles y funcionalidades que se pueden incorporar en la solución que se ha desarrollado para que las entidades del tercer sector tengan otras alternativas para gestionar la relación del socio con la asociación, siendo el **resultado satisfactorio al permitir descubrir el posible nicho de mercado** que tiene la propuesta del portal web detallada en este trabajo.

Con el resultado del análisis de contexto y estado del arte, se realiza la **especificación del producto** desde el punto de vista del socio, lo que ha permitido dar un enfoque al portal a nivel funcional para poder consultar noticias, eventos, la información de la asociación, la junta directiva actual, documentos y tramitar el pago de las cuotas de socio, y desde el punto de vista el administrador en lo relacionado con la administración del uso del sistema, siendo **considerado un alcance adecuado para la primera versión del portal web**.

Partiendo de la especificación del producto, se realiza el **diseño del producto** donde se define la arquitectura y la estructura de datos utilizando diversas técnicas para concretar el diseño. Gracias a esta fase del proyecto, **se han podido asentar las bases del proyecto a nivel tecnológico en función de los requisitos funcionales y no funcionales recogidos**, guiando los siguientes pasos del proyecto.

El seguir el diseño realizado del producto, ha permitido realizar una **implementación del producto** acorde a lo especificado en fases anteriores del proyecto, **cubriendo los atributos de calidad de la solución y los requisitos funcionales desde el punto de vista del socio**. **Las funciones definidas propias del administrador no han sido finalmente implementadas en esta primera versión**, lo que **hace concluir que este objetivo no ha quedado cubierto completamente**, si no de una forma parcial.

Para probar el correcto funcionamiento del sistema, se han **realizado pruebas** de integración de los componentes, lo que ha permitido validar cada componente y las pruebas funcionales han permitido validar que el funcionamiento a nivel funcional es correcto, como se puede ver en el vídeo funcional que está vinculado a través de un enlace de *YouTube* en la portada del trabajo. **Al realizar las pruebas indicadas y el ver el funcionamiento del sistema, este objetivo se da como cubierto**.

Los siguientes puntos de los objetivos son el **obtener las conclusiones**, estando detalladas en este punto y las **líneas de trabajo futuras** que están descritas en el siguiente punto, **ambos objetivos se consideran cubiertos**.

8.2. Líneas de trabajo futuro

Aunque las funcionalidades implementadas en el portal web del socio han cubierto una gran parte de las necesidades establecidas, ha quedado sin implementarse a nivel de administración del sistema:

1. Para la **administración del sistema**;
    a. El sistema permitirá desactivar el registro de las trazas de auditoría.
    b. El sistema permitirá a un usuario de tipo administrador consultar las solicitudes de verificación de pago de cuotas pendientes de los socios de la asociación.
    c. El sistema permitirá a un usuario de tipo administrador consultar las solicitudes de verificación de pago de cuotas pendientes de los socios de la asociación.

Siguiendo la línea anterior para los usuarios de tipo administrador del sistema, sería interesante para el sistema:

- ✓ El sistema permitirá a los usuarios administradores de la asociación cargar la información de la asociación de manera manual y masiva.
- ✓ El sistema permitirá a los usuarios administradores de la asociación poder sacar informes y estadísticas del uso del sistema.
- ✓ El sistema deberá avisar de intentos de accesos no autorizados tras 5 intentos.

Esta funcionalidad permitirá que el sistema se pueda utilizar completamente independiente a un sistema de administración de la asociación o sin tener que utilizar herramientas propias de un perfil técnico como es el *Postman* para cargar la información en la base de datos.

Otras funcionalidades interesantes para los usuarios de tipo administrador del sistema:

- ✓ El sistema permitirá a los usuarios administradores de la asociación poder sacar informes y estadísticas del uso del sistema.
- ✓ El sistema deberá avisar de intentos de accesos no autorizados tras 5 intentos.

También hay una vía de evolución en la interfaz de la aplicación, porque al utilizar una plantilla con un diseño ya predefinido, han quedado ciertos aspectos de experiencia de usuario sin

tener en cuenta, por lo que sería conveniente revisar la experiencia de usuario del usuario del portal.

Otra vía importante de desarrollo podría ser integrar en el sistema medios de pago que son utilizados en la actualidad por herramientas ERP para pagar por tarjeta de crédito, *bizum*[13] o *Paypal*[14], quitando de carga de trabajo a los administradores de la asociación ya que se automatiza esta función del sistema.

8.3. Valoración personal

A nivel personal, ha sido un proyecto muy enriquecedor y motivador que he disfrutado al realizarlo, ya que me ha permitido profundizar en un sector del que no tenía apenas conocimiento, más allá del vínculo de ser socio de una asociación cultural dedicada a la organización de las fiestas patronales de un pequeño pueblo de la provincia de Zamora. Esta experiencia ha siendo la fuente de inspiración de este proyecto, despertando en mí la curiosidad por conocer el estado tecnológico actual de las asociaciones sin ánimo de lucro.

También me ha permitido poner en práctica muchos de los aspectos vistos en las diferentes asignaturas del grado y aprender nuevas tecnologías que no conocía. Además, podido aplicar el conocimiento adquirido durante el grado en un proyecto que se ha implementado siguiendo las fases y prácticas del proceso de ingeniería de software, lo que me ha permitido completar el ciclo de vida del software desde un proyecto que parte de cero.

Uno de los retos más importantes que he afrontado en este proyecto ha sido el aprender desde cero las distintas tecnologías del *MERN Stack*, ya que hasta ahora mi experiencia concreta como desarrollador a nivel de lenguaje de programación era con JAVA[15] y JSP[xiv]. Sin embargo, este proyecto me ha permitido salir de mi zona de confort y conocer algunas de las tendencias y tecnologías actuales que están siendo demandadas en el mercado en los perfiles de programación.

[13] *Bizum*: https://bizum.es/

[14] *Paypal*: https://www.paypal.com/es/home

[15] *Java*: https://www.java.com/es/

Referencias bibliográficas

1000logos. (2023). GitHub Logo. https://1000logos.net/wp-content/uploads/2021/05/GitHub-logo.png

Asociaciones. (s. f.-a). Ministerio del interior. Recuperado 2 de invierno de 2024, de https://www.interior.gob.es/opencms/es/servicios-al-ciudadano/tramites-y-gestiones/asociaciones/

Asociaciones. (s. f.-b). Berrly. Recuperado 16 de mayo de 2024, de https://www.berrly.com/es/solucion/asociaciones/

Ayuntamiento de Murcia. (2019). ASOCIACIONISMO. Informajoven. Centro de Información Juvenil de Murcia. https://informajoven.org/info/participacion/J_5.asp

Bembibre., V. (2009, febrero). Definición de Portal. Significado.com. https://significado.com/portal/

Beneficios de utilizar un programa de gestión para asociaciones, cucunver. (2020, octubre 29). Blog de cucunver | Recursos gratis para la gestión de tu asociación. https://blog.cucunver.com/beneficios-de-utilizar-un-programa-de-gestion-para-asociaciones-cucunver/

Blog Planes de Futuro MAPFRE. (2022, octubre 15). Tipos de socios según el tipo de asociación. Planes de futuro BLOGS MAPFRE. https://planesdefuturo.mapfre.es/derechos-obligaciones/documentacion/tipos-de-socios-en-asociacion/

Codigo by Tecsup. (2023, junio 26). ¿Qué es MERN y por qué deberías aprenderlo? Edu.pe. https://codigo.edu.pe/blog/que-es-mern-y-por-que-deberias-aprenderlo/

Cohn, M. (2004). User Stories Applied for Agile Software Development. Addison Wesley Professional.

Comunicación Bados Duplá. (2022, junio 3). 13 preguntas frecuentes sobre las asociaciones sin ánimo de lucro. Fundaciones y Asociaciones. https://fundacionesyasociaciones.com/13-preguntas-frecuentes-sobre-las-asociaciones-sin-animo-de-lucro/

Comunicación, & Contenidos. (2020, enero 16). El papel de las ONG en la sociedad actual y su función social. Ayuda en Acción. https://ayudaenaccion.org/blog/solidaridad/papel-ong-sociedad-actual/

¿Cuál es el stack tecnológico más popular? (2019, junio 18). Syntonize. https://www.syntonize.com/stack-tecnologico-mas-popular/

cucunver. (s. f.). Cucunver.com. Recuperado 13 de mayo de 2024, de https://cucunver.com/

Cuestiones básicas de funcionamiento asociativo. (2018, mayo 2). Ayuntamiento de Cáceres. https://www.ayto-caceres.es/ciudadania/participacion/asociaciones/cuestiones-basicas-de-funcionamiento-asociativo/

Dashra - React Admin Template. (s. f.). ThemeForest. Recuperado 12 de mayo de 2024, de https://themeforest.net/item/dashra-react-admin-template/48837630

Digitalización de Fundaciones, Sociedades Científicas ESSENZIAL. (2023, febrero 28). Essenzial.com; Essenzial Spain, S.L. https://www.essenzial.com/solucion/nube-sectores/marketing-digital-fundaciones-y-sociedades-cientificas

Equipo de Estrategia y Operaciones de PwC. (s. f.). Estudio sobre el presente y futuro del Tercer Sector social en un entorno de crisis. https://www.pwc.es/es/fundacion/assets/presente-futuro-3sector.pdf

ERP para ONG y asociaciones. (2021, septiembre 6). Obliku. https://obliku.com/sectores/erp-para-ong/

Express Web Framework (Node.js/JavaScript). (s. f.). MDN Web Docs. Recuperado 12 de mayo de 2024, de https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs

Faster time-to-market. (2022, marzo 7). Seluxit. https://www.seluxit.com/faster-time-to-market/

Featured creator. (s. f.). WordPress themes & website templates from. ThemeForest. Recuperado 12 de mayo de 2024, de https://themeforest.net/

freelancermap. (s. f.). MEAN Stack. Recuperado 1 de abril de 2024, de https://lh5.googleusercontent.com/Oytwmb_lfafA2EUFibYFQFoLeeFZobLk85ZwnbNH-_BWXku2rU3qKPe2OxIHYCYoJMWF9Yx9o89PBzioj5uHhw00t063YycPqNMzPrcbvGrSH4ov

RYIhBQt_aDEZQKFOUJkwl14N6Vfg-w_XTCjXy-w06A5IWCzUSILt4oJd93Tz6qeEoPDNhB0X637cfg

Gábilos - Asociaciones, derechos y deberes de los asociados. (s. f.). Gabilos.com. Recuperado 17 de marzo de 2024, de https://www.gabilos.com/comosehace/asociaciones/textoasociacionderechos.htm

Gestron. (2024, enero 10). Tipos de asociaciones sin ánimo de lucro con ejemplos reales. Ayudatpymes. https://ayudatpymes.com/gestron/tipos-de-asociaciones-sin-animo-de-lucro/

GitHub. (s. f.). Acerca de GitHub y Git. Recuperado 12 de mayo de 2024, de https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

Gong. (s. f.). Gong.es. Recuperado 19 de marzo de 2024, de https://gong.es/

In, L. (s. f.). Soluciones Bitrix24. Bitrix24.es. Recuperado 13 de mayo de 2024, de https://www.bitrix24.es/solutions/

Intercambio de recursos de origen cruzado (CORS). (s. f.). MDN Web Docs. Recuperado 12 de mayo de 2024, de https://developer.mozilla.org/es/docs/Web/HTTP/CORS

Jorge, C. (2023, mayo 22). La importancia de la presencia en línea: maximiza tu éxito digital. Linkedin.com. https://es.linkedin.com/pulse/la-importancia-de-presencia-en-l%C3%ADnea-maximiza-tu-%C3%A9xito-jorge-cruz

Juárez, J. R. R. (2017). GUÍA DE ASOCIACIONES (2a ed.) (Ministerio del interior, Gobierno de España, Ed.). https://www.interior.gob.es/opencms/pdf/servicios-al-ciudadano/asociaciones/Guia-de-asociaciones-2-edicion.pdf

Kubo Mobile. (2023, noviembre 20). Cómo defino mi stack de desarrollo. Linkedin.com. https://es.linkedin.com/pulse/c%C3%B3mo-defino-mi-stack-de-desarrollo-kubosas-blp8f

Librestado. (2019, diciembre 16). Asociaciones – una forma jurídica habitual con ventajas poco conocidas. Librestado. https://librestado.com/blog/asociaciones/

Llanos, A. A. (2023, diciembre 27). Introducción a las asociaciones. Asociaciones.org. https://asociaciones.org/introduccion-a-las-asociaciones/

MERN stack explained. (s. f.). MongoDB. Recuperado 11 de mayo de 2024, de https://www.mongodb.com/resources/languages/mern-stack

Ministerio del interior, Gobierno de España (Ed.). (2023). 2022 ANUARIO ESTADÍSTICO del Ministerio del Interior. https://www.interior.gob.es/opencms/pdf/archivos-y-documentacion/documentacion-y-publicaciones/anuarios-y-estadisticas/ultimo-anuario-estadistico/Anuario_estadistico_2022_126150729.pdf

Ministerio del Interior, Gobierno de España. (2023, noviembre 13). Anuario Estadístico del Ministerio del Interior 2022. Gob.es. https://www.interior.gob.es/opencms/es/archivos-y-documentacion/documentacion-y-publicaciones/anuarios-y-estadisticas/anuarios-estadisticos-anteriores/anuario-estadistico-de-2022/

MongoDB Compass. (s. f.). MongoDB. Recuperado 12 de mayo de 2024, de https://www.mongodb.com/es/products/tools/compass

Nikotaf. (2016). Javascript badge. https://upload.wikimedia.org/wikipedia/commons/thumb/b/ba/Javascript_badge.svg/946px-Javascript_badge.svg.png

Number of internet users worldwide 2023. (s. f.). Statista. Recuperado 11 de mayo de 2024, de https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/

ONG y Tercer Sector, preguntas frecuentes. (2022, enero 21). Fundación Lealtad. https://www.fundacionlealtad.org/ong-y-tercer-sector-preguntas-frecuentes/

Parada, M. (2020, octubre 21). MERN Stack: Qué es y qué ventajas ofrece. Openwebinars.net. https://openwebinars.net/blog/mern-stack-que-es-y-que-ventajas-ofrece/

Pasar props a un componente. (s. f.). React.dev. Recuperado 15 de mayo de 2024, de https://es.react.dev/learn/passing-props-to-a-component

Pettit, K. (2022, agosto 17). Por qué son importantes las asociaciones sin ánimo de lucro y cómo beneficiarán a su empresa. EarthShare. https://www.earthshare.org/es/why-nonprofit-partnerships-matter-and-how-they-will-benefit-your-business/

Plataforma de ONG de Acción Social (Ed.). (2023). BARÓMETRO del Tercer Sector de Acción Social en España 2022.

https://www.plataformaong.org/ARCHIVO/documentos/biblioteca/1676295113_resumen ejecutivo-barometro-3-sector.pdf

Plus Contacto. (2022, octubre 27). La importancia de los/as socios/as para las ONG. Plus Contacto. https://pluscontacto.com/importancia-socios-ong/

Postman API platform. (s. f.). Postman.com. Recuperado 12 de mayo de 2024, de https://www.postman.com/

Pressman, R. S. (2010). Ingeniería del software. Un enfoque práctico (V. C. Olguín & J. E. Brito, Trads.; Sétima edición). The McGraw-Hill Companies, Inc.

¿Qué es JavaScript? (s. f.). MDN Web Docs. Recuperado 12 de mayo de 2024, de https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

¿Qué es la arquitectura de tres niveles? (2023, marzo 21). Ibm.com. https://www.ibm.com/es-es/topics/three-tier-architecture

¿Qué Es MongoDB? (s. f.). MongoDB. Recuperado 12 de mayo de 2024, de https://www.mongodb.com/es/company/what-is-mongodb

¿Qué es una interfaz de programación de aplicaciones (API)? (s. f.). Amazon.com. Recuperado 15 de mayo de 2024, de https://aws.amazon.com/es/what-is/api/

React. (s. f.). React.dev. Recuperado 12 de mayo de 2024, de https://es.react.dev/

Real Academia Española. (2023a). Sociedad. Rae.es. https://dle.rae.es/sociedad

Real Academia Española. (2023b). Socio, cia. Rae.es. https://dle.rae.es/socio

Roldán, P. N. (2018, diciembre 6). Socio. Economipedia. https://economipedia.com/definiciones/socio.html

Run JavaScript everywhere. (s. f.). Nodejs.org. Recuperado 12 de mayo de 2024, de https://nodejs.org/en/

Sociedades sin ánimo de lucro. (2019, noviembre 26). Asesorías. https://asesorias.com/empresas/sociedades/sin-animo-lucro/

Software de gestión para entidades. (2023, abril 13). Playoff; Playoff Informática. https://playoffinformatica.com/

Software ERP para fundaciones, asociaciones y ONGs. (2023, agosto 18). Cuatroochenta. https://cuatroochenta.com/microsoft-dynamics/erp-fundaciones-ong/

Software para asociaciones. (2017, diciembre 26). The Team Pyme. https://www.teampyme.com/software-asociaciones

Software y programa de gestión ERP para ONG, Obra social, Fundaciones y asociaciones. (s. f.). Ceieu.com. Recuperado 19 de marzo de 2024, de https://www.ceieu.com/programa-software-gestion-erp-servicios/ong-fundaciones-asociaciones-obra-social/

Stack overflow developer survey 2023. (s. f.). Stack Overflow. Recuperado 1 de abril de 2024, de https://survey.stackoverflow.co/2023/

TimTul. (s. f.). Timtul.com. Recuperado 19 de marzo de 2024, de https://timtul.com/

Tu primer componente. (s. f.). React.dev. Recuperado 15 de mayo de 2024, de https://es.react.dev/learn/your-first-component

Visual Studio Code full logo, tech companies, png. (s. f.). Pngegg.com. Recuperado 16 de mayo de 2024, de https://www.pngegg.com/en/png-veytl

Visual Studio: IDE y Editor de código para desarrolladores de software y Teams. (2018, septiembre 26). Visual Studio. https://visualstudio.microsoft.com/es/

Vive. (2022a, junio 13). MERN y MEAN Stack, las agrupaciones de herramientas de software más populares. UNIR. https://www.unir.net/ingenieria/revista/mern-vs-mean-stack/

Vive. (2022b, agosto 26). La importancia de las pruebas de software. UNIR. https://www.unir.net/ingenieria/revista/pruebas-software/

What is A technology stack? Tech stacks explained. (s. f.). MongoDB. Recuperado 11 de mayo de 2024, de https://www.mongodb.com/resources/basics/technology-stack

Wiegers, K., & Beatty, J. (2013a). Software Requirements (Third Edition). Microsoft Press.

Wiegers, K., & Beatty, J. (2013b). Software Requirements, Third Edition (Tercera edición). Christian Holdener, S4Carlisle Publishing Services.

Anexo A. Diccionario de datos.

Tabla 17. *Diccionario de datos del portal del socio.*

Elemento de datos	Descripción	Composición o tipo	Longitud	Valores y unidades
Association	Información general de la *asociación* sin ánimo de lucro a la que el socio pertenece.	ID de la *asociación* + Code + TypeAssociation + DocumentType + DocumentNumber + BusinessName + ConstitutionDate + DischargeDate + Phone + AddressTypeOfRoad + AddressStreet + (AddressNumber) + (AddressPortal) + (AddressBlock) + (AddressStaircase)		

		+ (AddressFloor) + (AddressDoor) + (AddressPopulation) + (AddressProvince) + (AddressPostalCode) + (AddressComplement) + 0:1{Logo} + AuditUser + AuditDate + CreatedAt + UpdatedAt		
Board of director	Información de la *junta directiva* actual que gestiona la asociación sin ánimo de lucro. Esta *junta directiva* se relaciona con la *asociación*. Todas las *asociaciones* deben disponer de una *junta directiva* actual que gestione la *asociación*.	ID de la *junta directiva* + AssociationCode + ConstitutionDate + StatusBoardOfDirector + AuditUser + AuditDate + CreatedAt + UpdatedAt		

Executive	Información del directivo miembro de la junta directiva actual que gestiona la asociación sin ánimo de lucro.	ID del *directivo* + BoardOfDirectorsId + PartnerId + AssociationCode + Position + PositionDate + AuditUser + AuditDate + CreatedAt + UpdatedAt		
Partner	Información del *socio* asociado a la *asociación* sin ánimo de lucro.	ID del *socio* + AssociationCode + PartnerNumber + TypePartner + Name + Surname + (SecondSurname) + DocumentType + DocumentNumber		

		+ Birthdate + Phone + Email + DischargeDate + AddressTypeOfRoad + AddressStreet + (AddressNumber) + (AddressPortal) + (AddressBlock) + (AddressStaircase) + (AddressFloor) + (AddressDoor) + (AddressPopulation) + (AddressProvince) + (AddressPostalCode) + (AddressComplement) + StatusPartner + StatusDate + 0:1{Image}		

		+ AuditUser + AuditDate + CreatedAt + UpdatedAt		
Membership fee	Información de la *cuota de socio* que paga el *socio* por estar afiliado a la *asociación* sin ánimo de lucro. Cada *cuota de socio* tiene al menos una *situación de la cuota de socio* y tiene vinculado la última *situación de la cuota de socio*.	ID de *cuota de socio* + PartnerId + MembershipNumber + MembershipFeeSituationId + AssociationCode + TypeMembership + Period + EffectDate + DueDate + Membership + AuditUser + AuditDate + CreatedAt + UpdatedAt		

Membership fee situation	Información de la *situación de la cuota de socio* que indica el estado del pago en que está la *cuota de socio*. Cada movimiento en la *cuota de socio* genera una *situación de la cuota de socio* indicando el estado del pago.	ID de la *situación de la cuota de socio* + PartnerId + MembershipFeeId + SituationNumber + AssociationCode + PaymentMethod + StatusSituation + Reason + SituationDate + AuditUser + AuditDate + CreatedAt + UpdatedAt		
Announcement	Información del *anuncio* publicado por la *asociación* para notificar las noticias a los *socios*, avisando de eventos, asambleas y cualquier tipo de información que quiera notificar la *asociación* a sus *socios*.	ID del *anuncio* + AssociationCode + TypeAnnouncement + Title + Text + 0:1{Image}		

		+ Author + ImportanceLevel + PublicationDate + (FinishDate) + AuditUser + AuditDate + CreatedAt + UpdatedAt		
Document	Información del *documento* de la asociación, pudiendo ser este documento propio de la *asociación*, del *socio*, de la *cuota de socio*, de la *situación de la cuota de socio* o del *anuncio de la asociación*.	ID del *documento* + PartnerId + MembershipFeeId + MembershipFeeSituationId + AnnouncementId + AssociationCode + Module + FileName + Extension + Size + Title		

		+ Description + HighUser + Mimetype + Encoding + Buffer + DischargeDate + AuditUser + AuditDate + CreatedAt + UpdatedAt		
User	Información del *usuario* que permite acceder al sistema para consultar la información. Según el tipo de *usuario*, se accede con un rol al sistema, se puede acceder como rol de *socio* o como rol de administrador.	ID de *usuario* + AssociationCode + Username + (PartnerId) + Password + Email + Name + Surname + (SecondSurname)		

		+ UserType + UserStatus + RegistrationDate + AuditUser + AuditDate + CreatedAt + UpdatedAt		
Audit start	Información que representa la *auditoría de arranque* del sistema donde se registran los arranques y paradas del sistema.	ID de *auditoría de arranque* + InstanceNumber + ServerAddress + WebAddress + ServerPort + StartTime + StopTime + CreatedAt + UpdatedAt		
Audit login	Información que representa la *auditoría de autenticación de usuario* en el sistema	ID de *auditoría de acceso* + InstanceNumber + Token		

	donde se registran los accesos y las salidas de los usuarios.	+ User + Status + Message + UserStored + Url + LoginTime + LogoutTime + AuditDate + CreatedAt + UpdatedAt		
Audit request	Información que representa la *auditoría de peticiones* al sistema donde se registran la información de la petición y la respuesta del sistema.	ID de *auditoría de petición* + AuditLoginId + InstanceNumber + RequestMethod + RequestUrl + Request + RequestBody + RequestTime + StatusResponse		

		+ MessageResponse + RequestResponse + RequestResponseTime + AuditUser + AuditDate + CreatedAt + UpdatedAt		
Sequence	Información de las secuencias del sistema	ID de la *secuencia* + Reference + Seq		
Logo	Información del símbolo gráfico de la asociación.	ID del *logo* + Originalname + Encoding + Mimetype + Size + Buffer		
Image	Información de la imagen del elemento.	ID de la *imagen* + Originalname + Encoding		

		+ Mimetype + Size + Buffer		
ID de asociación (associationId)	Identificador único de asociación generado por el sistema.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de la junta directiva (boardOfDirectorsId)	Identificador único de la junta directiva.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID del socio (partnerId)	Identificador único del socio.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a

				partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID del ejecutivo (executiveId)	Identificador único del directivo.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de cuota de socio (membershipFeeId)	Identificador único de cuota de socio.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de la situación de la cuota de socio (membershipFeeSituati	Identificador único de situación de cuota de socio.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a

onId)				partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de documento (documentId)	Identificador único de documento.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de usuario (userId)	Identificador único de usuario.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID del logo (logoId)	Identificador único del logo de la asociación.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a

				partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de la imagen (imageId)	Identificador único de la imagen del elemento.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de auditoría de arranque (auditStartId)	Identificador único de la auditoría de arranque del sistema.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de auditoría de acceso (auditLoginId)	Identificador único de la auditoría de acceso del sistema.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a

				partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de auditoría de petición (auditRequestId)	Identificador único de la auditoría de petición del sistema.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.
ID de la secuencia (sequenceId)	Identificador único de la secuencia de datos.	Caracteres alfanuméricos	24	Caracteres generados por la base de datos para hacer único al objeto generado los 8 primeros caracteres a partir de una marca de tiempo, los 10 siguientes de un valor aleatorio y los 6 últimos es un contador incremental.

Code	Código único de la asociación. No puede existir en el sistema varias asociaciones con el mismo código.	Caracteres alfanuméricos		Código sin espacios. Es parte de la URL del portal del socio para la asociación.
TypeAssociation	Código de tipo de asociación.	Caracteres alfanuméricos	4	Valores: -01: Asociación cultural
DocumentType	Tipo de documento de identificación de persona física.	Caracteres alfanuméricos	3	Valores: -NIF: Número de Identificación Fiscal -CIF: Código de Identificación Fiscal -NIE: Número de Identidad de Extranjero -PAS: Número de pasaporte
DocumentNumber	Número del documento de identificación de persona física.	Caracteres alfanuméricos	9	Número de identificación de persona válido según el tipo de documento.
BusinessName	Nombre de la asociación sin ánimo de lucro.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
ConstitutionDate	Fecha de constitución de la asociación sin ánimo de lucro.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS

DischargeDate	Fecha de alta.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
Phone	Número de teléfono sin prefijo.	Entero positivo	9	Formato de teléfono español, tanto teléfono fijo o teléfono móvil.
AddressTypeOfRoad	Tipo de vía de la dirección.	Caracteres alfanuméricos		Tipos de vías recogidas en el registro de la seguridad social.
AddressStreet	Nombre de la calle de la dirección.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
AddressNumber	Número de la dirección.	Caracteres alfanuméricos		Caracteres numéricos enteros o el valor S/N.
AddressPortal	Portal de la dirección.	Caracteres alfanuméricos		Números enteros o letras.
AddressBlock	Bloque de la dirección.	Caracteres alfanuméricos		Números enteros o letras.
AddressStaircase	Escalera de la dirección.	Caracteres alfanuméricos		Números enteros o letras.
AddressFloor	Piso de la dirección.	Enteros positivos		

AddressDoor	Puerta de la dirección.	Caracteres alfanuméricos		Números enteros o letras.
AddressPopulation	Población de la dirección.	Caracteres alfanuméricos		Listado oficial del INE de poblaciones.
AddressProvince	Provincia de la dirección.	Caracteres alfanuméricos		Listado oficial del INE de provincias.
AddressPostalCode	Código postal de la dirección.	Enteros positivos.		Listado oficial del INE de códigos postales.
AddressComplement	Complemento adicional de la dirección.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
AuditUser	Usuario que genera la información en el sistema. Información de auditoría.	Caracteres alfanuméricos		Código de usuario de la colección de usuarios.
AuditDate	Fecha en la que genera la información en el sistema. Información de auditoría.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
CreatedAt	Fecha de alta de la información en la base de datos.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS

UpdatedAt	Fecha de última modificación de la información en la base de datos.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
AssociationCode	Código de la asociación sin ánimo de lucro al que pertenece el elemento.	Caracteres alfanuméricos		Código sin espacios.
StatusBoardOfDirector	Código de estado de la junta directiva.	Caracteres alfanuméricos	4	Valores: -ACTI: Activo -INAC: Inactivo
StatusDate	Fecha de actualización estado del elemento.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
Position	Código de cargo del miembro de la junta directiva.	Caracteres alfanuméricos	4	Valores: -PRES: Presidente -SECR: Secretario -VOCA: Vocal -TESO: Tesorero
PositionDate	Fecha de alta en el cargo de como directivo de la junta directiva.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS

PartnerNumber	Número de socio. Identificador de negocio del socio dentro de la asociación sin ánimo de lucro.	Entero positivo.		Número secuencial entero generado por el sistema con cada socio de una asociación, comenzando por 1.
TypePartner	Código de tipo de socio.	Caracteres alfanuméricos	4	Valores: -NUME: Numerario -HONO: Honor
StatusPartner	Código de estado de socio.	Caracteres alfanuméricos	4	Valores: -ACTI: Activo -INAC: Inactivo
Name	Nombre de persona física.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
Surname	Primer apellido de persona física.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
SecondSurname	Segundo apellido de persona física.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
Birthdate	Fecha de nacimiento de persona física.	Fecha	29	Fecha en estándar ISO 8601 en formato:

				YYYY-MM-DDTHH:MM:SS
Email	Dirección de correo electrónico.	Caracteres alfanuméricos		Formato de correo electrónico: nombredeusuario@dominio.com
MembershipNumber	Número de cuota de socio. Identificador de negocio de la cuota de socio del socio dentro de la asociación sin ánimo de lucro.	Entero positivo.		Número secuencial entero generado por el sistema con cada cuota por socio de una asociación, comenzando por 1.
TypeMembership	Código de tipo de cuota de socio.	Caracteres alfanuméricos	4	Valores: -EXTR: Ordinaria -ORDI: Extraordinaria
Period	Código de periodo de cobro de cuota de socio.	Caracteres alfanuméricos	4	Valores: -ANUA: Anual -SEME: Semestral -TRIM: Trimestral -MENS: Mensual -BIME: Bimensual
EffectDate	Fecha de efecto.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS

DueDate	Fecha de vencimiento.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
Membership	Valor económico de la cuota de socio.	Numérico con decimales		Numérico con 2 decimales.
SituationNumber	Número de situación de cuota de socio. Identificador de negocio de la situación de cuota de socio del socio dentro de la asociación sin ánimo de lucro.	Entero positivo.		Número secuencial entero generado por el sistema con cada situación de cuota de socio de una cuota del socio de una asociación, comenzando por 1.
PaymentMethod	Código de medio de pago de cuota de socio.	Caracteres alfanuméricos	4	Valores: -EFEC: Efectivo -BANC: Bancario
StatusSituation	Código de estado de situación de pago de cuota de socio.	Caracteres alfanuméricos	4	Valores: -PECO: Pendiente de cobro -COBR: Cobrado -ANUL: Anulado
Reason	Código de motivo de situación de pago de cuota de socio.	Caracteres alfanuméricos	4	Valores: -PECO: Pendiente de cobro -PEVA: Pendiente de validación

				-COBR: Cobrado -ANUL: Anulado
SituationDate	Fecha de situación.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
TypeAnnouncement	Código de tipo de anuncio de la asociación sin ánimo de lucro.	Caracteres alfanuméricos	4	Valores: -NOTI: Noticia -ASOR: Asamblea ordinaria -ASEX: Asamblea extraordinaria -EVEN: Evento
Title	Título del elemento.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
Text	Texto de detalle del elemento.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
Author	Autor de la publicación.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
ImportanceLevel	Código de nivel de importancia.	Caracteres alfanuméricos	4	Valores: -ALTA: Alta

				-MEDI: Media -BAJA: Baja
PublicationDate	Fecha de publicación.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
FinishDate	Fecha de finalización.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
Module	Código de modulo.	Caracteres alfanuméricos		Valores: -membershipfees: Cuotas de socio -partners: Socio -associations: Asociación -announcements: Anuncio
FileName	Nombre del archivo.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
Extension	Extensión del archivo.	Caracteres alfanuméricos	5	.pdf, .txt, .png, .jpef, .gif
Description	Descripción del elemento.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.

HighUser	Usuario de alta del elemento.	Caracteres alfanuméricos		
Originalname	Nombre original del archivo.	Caracteres alfanuméricos		Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes.
Encoding	Codificación del archivo.	Caracteres alfanuméricos		7bit
Mimetype	Tipo de archivo.	Caracteres alfanuméricos		application/pdf, image/jpeg, text/plain
Size	Tamaño máximo en bytes del archivo.	Entero positivo		Puede contenedor el 0.
Buffer	Datos binarios en formato hexadecimal en base 64 del archivo.	Caracteres alfanuméricos		Hexadecimal en base 64.
Username	Nombre de usuario.	Caracteres alfanuméricos		Formato de correo electrónico.
Password	Contraseña del usuario.	Caracteres alfanuméricos	6	Puede contener espacios en blanco, tildes, guiones, puntos y apóstrofes. Mínimo 6 caracteres.
UserType	Tipo de usuario.	Caracteres alfanuméricos	8	Valores: -ADMIN: Administrador -PARTNER: Socio

UserStatus	Código de estado del usuario.	Caracteres alfanuméricos	4	Valores: -ACTI: Activo -INAC: Inactivo -BLOC: Bloqueado
RegistrationDate	Fecha de registro.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
InstanceNumber	Número de arranque del sistema. Identificador del sistema que identifica el arranque del sistema.	Entero positivo.		Número secuencial entero generado por el sistema por cada arranque del sistema, comenzando por 1.
ServerAddress	Dirección del servidor de la aplicación.	Caracteres alfanuméricos		No puede contener espacios en blanco, tildes o apóstrofes.
WebAddress	Dirección de la página web.	Caracteres alfanuméricos		No puede contener espacios en blanco, tildes o apóstrofes.
ServerPort	Número de puerto del servidor de aplicaciones.	Entero positivo.		Puertos desde 1024 hasta 49151.
StartTime	Fecha de arranque del sistema.	Fecha	29	Fecha en estándar ISO 8601 en formato:

				YYYY-MM-DDTHH:MM:SS
StopTime	Fecha de parada del sistema.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
Token	Token web de acceso que proporciona el sistema al usuario.	Caracteres alfanuméricos		No puede contener espacios en blanco, tildes, guiones o apóstrofes.
Status	Código de estado del acceso al sistema.	Caracteres alfanuméricos		success, error
Message	Mensaje del acceso al sistema.	Caracteres alfanuméricos		
UserStored	Información del *usuario autenticado* que se ha autorizado el acceso al sistema	ID de *usuario autenticado* + AssociationCode + Username + (PartnerId) + Email + Name + Surname + (SecondSurname) + UserType + UserStatus		

		+ RegistrationDate		
Url	Url del API que recibe la petición.	Caracteres alfanuméricos		No puede contener espacios en blanco, tildes o apóstrofes.
LoginTime	Fecha de autenticación y acceso al sistema.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
LogoutTime	Fecha de desconexión y salida del sistema.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
RequestMethod	Método de la petición.	Caracteres alfanuméricos	6	POST, PUT, GET, DELETE
RequestUrl	Url de la petición del navegador al recurso API expuesto por el sistema.	Caracteres alfanuméricos		No puede contener espacios en blanco, tildes o apóstrofes.
Request	Url de la petición del navegador web sin la parte del servidor.	Caracteres alfanuméricos		No puede contener espacios en blanco, tildes o apóstrofes.
RequestBody	Objeto JSON que es el cuerpo de respuesta del sistema.	Objecto JSON		

RequestTime	Fecha de la petición al sistema.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
StatusResponse	Código de estado de respuesta de la petición al sistema.	Caracteres alfanuméricos		success, error
MessageResponse	Mensaje de respuesta de la petición al sistema.	Caracteres alfanuméricos		
RequestResponse	Objeto JSON con la respuesta completa.	Caracteres alfanuméricos		
RequestResponseTime	Fecha de respuesta del sistema a la petición.	Fecha	29	Fecha en estándar ISO 8601 en formato: YYYY-MM-DDTHH:MM:SS
Reference	Código de referencia de la secuencia.	Caracteres alfanuméricos		No puede contener espacios en blanco, tildes o apóstrofes.
Seq	Valor de la secuencia.	Entero positivo.		Número secuencial generado por el sistema por cada código de referencia.

Fuente: Elaboración propia

Anexo B. Puntos de acceso disponibles

Tabla 18. *Tabla con la lista de API expuestas desde la parte del servidor.*

Elemento	Método	Ruta relativa	Descripción
Administradores	`POST`	/admin/login	Inicio de sesión de un usuario de tipo administrador del sistema.
	`DELETE`	/admin/logout	Cierre de sesión de un usuario de tipo administrador del sistema.
Usuarios	`POST`	/usuarios/	Creación de un usuario.
	`GET`	/usuarios/	Consulta de los usuarios del sistema.
	`GET`	/usuarios/:userId	Consulta de un usuario por su identificador.
	`PUT`	/usuarios/:userId	Actualización de un usuario por su identificador.
	`DELETE`	/usuarios/:userId	Eliminación de un usuario por su identificador.
Asociaciones	`POST`	/asociaciones/	Creación de una asociación.
	`GET`	/asociaciones/	Consulta de las asociaciones del sistema.
	`GET`	/asociaciones-simple/	Consulta de las asociaciones del sistema con información reducida.

	GET	/asociaciones/:associationCode	Consulta de una asociación por código de asociación.
	PUT	/asociaciones/:associationCode	Actualización de una asociación por código de asociación.
	PUT	/asociaciones/:associationCode/logo	Actualización del logo de una asociación por código de asociación.
	DELETE	/asociaciones/:associationCode	Eliminación de una asociación por código de asociación.
Usuarios de una asociación	POST	/:associationCode/login	Inicio de sesión de un usuario de tipo socio para una asociación en concreto.
	DELETE	/:associationCode/logout	Cierre de sesión de un usuario de tipo socio para una asociación en concreto.
Juntas directivas de una asociación	POST	/:associationCode/directiva/	Creación de una junta directiva de la asociación seleccionada.
	GET	/:associationCode/directiva-activa/	Consulta de la junta directiva actual de la asociación seleccionada.
	GET	/:associationCode/directiva/:boardId	Consulta de las juntas directiva por identificador de la junta de la asociación seleccionada.

	PUT	/:associationCode/directiva/:boardId	Actualización de la junta directiva por identificador de la junta de la asociación seleccionada.
	DELETE	/:associationCode/directiva/:boardId	Eliminación de la junta directiva por identificador de la junta de la asociación seleccionada.
Directivos de la junta directiva de una asociación	POST	/:associationCode/directiva/:boardId/directivos/	Creación de un directivo de junta directiva indicada de la asociación seleccionada.
	GET	/:associationCode/directiva/:boardId/directivos/	Consulta de los directivos de junta directiva indicada de la asociación seleccionada.
	GET	/:associationCode/directiva/:boardId/directivos/:executiveId	Consulta de un directivo por identificador de junta directiva indicada de la asociación seleccionada.
	PUT	/:associationCode/directiva/:boardId/directivos/:executiveId	Actualización de un directivo por identificador de junta directiva indicada de la asociación seleccionada.
	DELETE	/:associationCode/directiva/:boardId/directivos/:executiveId	Eliminación de un directivo por identificador de junta directiva indicada de la asociación seleccionada.
Socios de una asociación	POST	/:associationCode/socios/	Creación de un socio de la asociación seleccionada.

	GET	/:associationCode/socios/	Consulta de los socios de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId	Consulta de un socio por identificador de la asociación seleccionada.
	PUT	/:associationCode/socios/:partnerId	Actualización de un socio de la asociación seleccionada.
	PUT	/:associationCode/socios/:partnerId/imagen	Actualización de la imagen de perfil de un socio por identificador de la asociación seleccionada.
	DELETE	/:associationCode/socios/:partnerId	Eliminación de un socio por identificador de la asociación seleccionada.
Cuotas de un socio de una asociación	POST	/:associationCode/socios/:partnerId/cuotas/	Creación de una cuota de un socio indicado de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId	Consulta de las cuotas de un socio indicado de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId	Consulta de una cuota por identificador de un socio indicado de la asociación seleccionada.
	PUT	/:associationCode/socios/:partnerId/cuotas/:membershipId	Actualización de una cuota por identificador de un socio indicado de la asociación seleccionada.

	DELETE	/:associationCode/socios/:partnerId/cuotas/	Eliminación de una cuota por identificador de un socio indicado de la asociación seleccionada.
Documentos de la cuota de un socio de una asociación	POST	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/	Creación de un documento de una cuota de un socio indicado de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/:documentId	Consulta de los documentos de una cuota de un socio indicado de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/:documentId	Consulta de un documento de una cuota de un socio indicado de la asociación seleccionada.
	PUT	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/:documentId	Actualización de un documento de una cuota de un socio indicado de la asociación seleccionada.
	DELETE	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/	Eliminación de un documento de una cuota de un socio indicado de la asociación seleccionada.
Situaciones de una cuota de un socio de una asociación	POST	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/	Creación de una situación de una cuota de un socio indicado de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/:situationId	Consulta de las situaciones de una cuota de un socio indicado de la asociación seleccionada.

	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/:situationId	Consulta de una situación de una cuota de un socio indicado de la asociación seleccionada.
	PUT	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/:situationId	Actualización de una situación de una cuota de un socio indicado de la asociación seleccionada.
	DELETE	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/	Eliminación de una situación de una cuota de un socio indicado de la asociación seleccionada.
Documentos de un socio de una asociación	POST	/:associationCode/socios/:partnerId/documentos/	Creación de un documento de un socio indicado de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId/documentos/:documentId	Consulta de los documentos de un socio indicado de la asociación seleccionada.
	GET	/:associationCode/socios/:partnerId/documentos/:documentId	Consulta de un documento por identificador de un socio indicado de la asociación seleccionada.
	PUT	/:associationCode/socios/:partnerId/documentos/:documentId	Actualización de un documento por identificador de un socio indicado de la asociación seleccionada.
	DELETE	/:associationCode/socios/:partnerId/documentos/	Eliminación de un documento por identificador de un socio indicado de la asociación seleccionada.

Anuncios de una asociación	POST	/:associationCode/anuncios/	Creación de un anuncio de la asociación seleccionada.
	GET	/:associationCode/anuncios/	Consulta de los anuncios de la asociación seleccionada.
	GET	/:associationCode/anuncios/:announcementId	Consulta de un anuncio por identificador de la asociación seleccionada.
	PUT	/:associationCode/anuncios/:announcementId	Actualización de un anuncio de la asociación seleccionada.
	PUT	/:associationCode/anuncios/:announcementId/imagen	Actualización de la imagen de un anuncio por identificador de la asociación seleccionada.
	DELETE	/:associationCode/anuncios/:announcementId	Eliminación de un anuncio por identificador de la asociación seleccionada.
Documentos de un anuncio de una asociación	POST	/:associationCode/anuncios/:announcementId/documentos/	Creación de un documento de un anuncio indicado de la asociación seleccionada.
	GET	/:associationCode/anuncios/:announcementId/documentos/:documentId	Consulta de los documentos de un anuncio indicado de la asociación seleccionada.
	GET	/:associationCode/anuncios/:announcementId/documentos/:documentId	Consulta de un documento por identificador de un anuncio indicado de la asociación seleccionada.

	PUT	/:associationCode/anuncios/:announcementId/documentos/:documentId	Actualización de un documento por identificador de un anuncio indicado de la asociación seleccionada.
	DELETE	/:associationCode/anuncios/:announcementId/documentos/	Eliminación de un documento por identificador de un anuncio indicado de la asociación seleccionada.
Documentos de una asociación	POST	/:associationCode/documentos/	Creación de un documento de la asociación seleccionada.
	GET	/:associationCode/documentos/	Consulta de los documentos de la asociación seleccionada.
	GET	/:associationCode/documentos/:documentId	Consulta de un documento por identificador de la asociación seleccionada.
	PUT	/:associationCode/documentos/:documentId	Actualización de un documento por identificador de la asociación seleccionada.
	PUT	/:associationCode/documentos/:documentId/documento	Actualización de un documento por identificador para anexar el documento de la asociación seleccionada.
	DELETE	/:associationCode/documentos/:documentId	Eliminación de un documento por identificador de la asociación seleccionada.

Fuente: Elaboración propia

Anexo C. Pruebas unitarias en puntos de acceso

Tabla 19: *Tabla con las pruebas unitarias realizadas sobre los puntos de acceso expuestos a través de Postman.*

Elemento	Método	Ruta relativa	Resultado de la prueba
Administradores	POST	/admin/login	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/admin/logout	OK - Sin errores. Recibida respuesta esperada.
Usuarios	POST	/usuarios/	OK - Sin errores. Recibida respuesta esperada.
	GET	/usuarios/	OK - Sin errores. Recibida respuesta esperada.
	GET	/usuarios/:userId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/usuarios/:userId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/usuarios/:userId	OK - Sin errores. Recibida respuesta esperada.
Asociaciones	POST	/asociaciones/	OK - Sin errores. Recibida respuesta esperada.
	GET	/asociaciones/	OK - Sin errores. Recibida respuesta esperada.
	GET	/asociaciones-simple/	OK - Sin errores. Recibida respuesta esperada.
	GET	/asociaciones/:associationCode	OK - Sin errores. Recibida respuesta esperada.
	PUT	/asociaciones/:associationCode	OK - Sin errores. Recibida respuesta esperada.

	PUT	/asociaciones/:associationCode/logo	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/asociaciones/:associationCode	OK - Sin errores. Recibida respuesta esperada.
Usuarios de una asociación	POST	/:associationCode/login	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/logout	OK - Sin errores. Recibida respuesta esperada.
Juntas directivas de una asociación	POST	/:associationCode/directiva/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/directiva-activa/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/directiva/:boardId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/directiva/:boardId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/directiva/:boardId	OK - Sin errores. Recibida respuesta esperada.
Directivos de la junta directiva de una asociación	POST	/:associationCode/directiva/:boardId/directivos/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/directiva/:boardId/directivos/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/directiva/:boardId/directivos/:executiveId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/directiva/:boardId/directivos/:executiveId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/directiva/:boardId/directivos/:executiveId	OK - Sin errores. Recibida respuesta esperada.
Socios de una asociación	POST	/:associationCode/socios/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/	OK - Sin errores. Recibida respuesta esperada.

	GET	/:associationCode/socios/:partnerId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/socios/:partnerId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/socios/:partnerId/imagen	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/socios/:partnerId	OK - Sin errores. Recibida respuesta esperada.
Cuotas de un socio de una asociación	POST	/:associationCode/socios/:partnerId/cuotas/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/socios/:partnerId/cuotas/:membershipId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/socios/:partnerId/cuotas/	OK - Sin errores. Recibida respuesta esperada.
Documentos de la cuota de un socio de una asociación	POST	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/socios/:partnerId/cuotas/:membershipId/documentos/	OK - Sin errores. Recibida respuesta esperada.
Situaciones de una cuota de un	POST	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/:situationId	OK - Sin errores. Recibida respuesta esperada.

socio de una asociación	GET	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/:situationId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/:situationId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/socios/:partnerId/cuotas/:membershipId/situaciones/	OK - Sin errores. Recibida respuesta esperada.
Documentos de un socio de una asociación	POST	/:associationCode/socios/:partnerId/documentos/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/:partnerId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/socios/:partnerId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/socios/:partnerId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/socios/:partnerId/documentos/	OK - Sin errores. Recibida respuesta esperada.
Anuncios de una asociación	POST	/:associationCode/anuncios/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/anuncios/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/anuncios/:announcementId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/anuncios/:announcementId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/anuncios/:announcementId/imagen	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/anuncios/:announcementId	OK - Sin errores. Recibida respuesta esperada.
	POST	/:associationCode/anuncios/:announcementId/documentos/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/anuncios/:announcementId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.

Documentos de un anuncio de una asociación	GET	/:associationCode/anuncios/:announcementId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/anuncios/:announcementId/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/anuncios/:announcementId/documentos/	OK - Sin errores. Recibida respuesta esperada.
Documentos de una asociación	POST	/:associationCode/documentos/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/documentos/	OK - Sin errores. Recibida respuesta esperada.
	GET	/:associationCode/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.
	PUT	/:associationCode/documentos/:documentId/documento	OK - Sin errores. Recibida respuesta esperada.
	DELETE	/:associationCode/documentos/:documentId	OK - Sin errores. Recibida respuesta esperada.

Fuente: Elaboración propia

Anexo D. Pruebas funcionales

Tabla 20: *Tabla de prueba funcional del caso de uso UC-001*

Identificador	PF0001
Caso de uso	UC-001 Acceder al sistema
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU01: El usuario queda autenticado en el sistema.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 21: *Tabla de prueba funcional del caso de uso UC-002*

Identificador	PF0002
Caso de uso	UC-002 Salir del sistema
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU02: El usuario ya no está autenticado en el sistema.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 22: *Tabla de prueba funcional del caso de uso UC-003*

Identificador	PF0003
Caso de uso	UC-003 Consultar información personal
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU03: El socio consulta su información personal.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 23: *Tabla de prueba funcional del caso de uso UC-004*

Identificador	PF0004
Caso de uso	UC-004 Consultar los anuncios publicados por la asociación
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU04: El socio consulta las publicaciones de la asociación.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 24: *Tabla de prueba funcional del caso de uso UC-005*

Identificador	PF0005
Caso de uso	UC-005 Consultar el estado de las cuotas de socio
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU05: El socio consulta sus estados de cuotas de socio.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 25*: Tabla de prueba funcional del caso de uso UC-006*

Identificador	PF0006
Caso de uso	UC-006 Justificar el pago de una cuota de socio
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU06: El socio justifica el pago de una cuota de socio.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 26*: Tabla de prueba funcional del caso de uso UC-007*

Identificador	PF0007
Caso de uso	UC-007 Consultar los documentos del socio
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU07: El socio consulta sus documentos.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 27*: Tabla de prueba funcional del caso de uso UC-008*

Identificador	PF0008
Caso de uso	UC-008 Descargar un documento del socio
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU01: El usuario queda autenticado en el sistema.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 28*: Tabla de prueba funcional del caso de uso UC-009*

Identificador	PF0009
Caso de uso	UC-009 Consultar información de la asociación
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU09: El socio consulta la información de la asociación.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Tabla 29*: Tabla de prueba funcional del caso de uso UC-010*

Identificador	PF0010
Caso de uso	UC-010 Consultar los miembros de la junta directiva
Pasos por seguir	Los flujos descritos en el caso de uso.
Resultado esperado	RESU10: El socio consulta los miembros de la junta directiva.
Resultado obtenido	Resultado esperado, sin errores.

Fuente: Elaboración propia

Índice de acrónimos

[i] *SaaS*: *Software as a Service*. Poner a disposición softwares y soluciones de tecnología por medio de la internet, como un servicio.
[ii] *MERN*: *MongoDB, Express, React y Node*.
[iii] *RNA*: Registro Nacional de Asociaciones.
[iv] *ERP*: Enterprise Resource Planning. Sistema de planificación de recursos empresariales.
[v] *RGPD*: Reglamento General de Protección de Datos.
[vi] *CRM*: *Customer Relationship Management*. Sistema de gestión integrada de ventas, marketing, atención al cliente y todos los puntos de contacto.
[vii] *API*: Application Programming Interface. Interfaz de programación de aplicaciones.
[viii] *CRUD*: Create, Read, Update, Delete. Crear, leer, actualizar, eliminar.
[ix] *JSON*: JavaScript Object Notation. Notación de objetos JavaScript.
[x] *JSX*: JavaScript XML.
[xi] *IDE*: Integrated Development Environment. Entorno de desarrollo integrado.
[xii] BBDD: Base de datos.
[xiii] *CORS*: Cross-origin Resource Sharing. Intercambio de Recursos de origen cruzado
[xiv] *JSP*: JavaServer Pages.