UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una APP móvil para compartir actividades de ocio.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Bellot Pérez, Javier

Tutor/a: Albert Albiol, Manuela

Cotutor/a: Torres Bosch, María Victoria

CURSO ACADÉMICO: 2021/2022

Por y para mi familia por darme la oportunidad de hacer lo que me gusta

Agradecimientos

Primero de todo quisiera dar las gracias a todas aquellas personas que indirectamente han hecho posible que yo esté haciendo este trabajo de fin de grado (TFG). Es un honor intentar culminar todos los conocimientos que he adquirido en esta pieza. Considero que he puesto todo de mi parte para sacar el proyecto adelante y estoy orgulloso de ello. Este trabajo ha sido complicado de elaborar, y hay ciertas personas que me han dado ese pequeño empujón en ciertos momentos para poder acabarlo con un toque de calidad. A mis compañeros, especialmente a mi amigo Tomas Montalvo, quien me apoyó en todo momento durante este largo recorrido de la carrera y el TFG, a mis amigos, quienes confiaron en mi en todo momento y, por último, a mi familia, por darme las alas necesarias para volar; *Os quiero*.

Resumen

El presente TFG consiste en el desarrollo de una APP móvil orientada a fomentar las relaciones sociales de los usuarios a través de actividades de ocio. La aplicación permitirá que los usuarios puedan escoger actividades para realizar con otros usuarios. Los usuarios serán los que creen las actividades (por ejemplo, ir al cine el domingo) y determinen el tipo de acompañante/s que buscan. El diseño de la aplicación se realizará buscando maximizar la experiencia de usuario ofrecida. Para ello, se aplicarán algunos de los patrones de interacción más populares que se utilizan en aplicaciones de carácter social (como Tinder). El fin final del trabajo es lanzar la aplicación al mercado, por lo que el TFG incluirá la definición de un plan de negocio. La tecnología que se va a usar será Angular Js con Ionic para el *frontend*, Spring para el *backend*, y PostgreSQL para la BD.

Palabras clave: Aplicación móvil; relaciones sociales; patrones de interacción; experiencia de usuario; compartir planes; Angular Js; Ionic; Spring; PostgreSQL

Abstract

The present TFG consists of the development of a mobile APP oriented to promote the social relationships of the users through leisure activities. The application will allow users to choose activities to do with other users. The users will be the ones to create the activities (for example, going to the movies on Sunday) and choose the type of companion/s they are looking for. The application will be designed to maximize the user experience offered. To this end, some of the most popular interaction patterns used in social applications (such as Tinder) will be applied. The final goal of the work is to launch the application to the market, so the TFG will include the definition of a business plan. The technology to be used will be Angular Js with Ionic for the *frontend*, Spring for the *backend*, and PostgreSQL for the DB.

Keywords : Mobile application; social relationships; interaction patterns; user experience; sharing plans; Angular Js; Ionic; Spring; PostgreSQL; PostgreSQL.

Índice de contenidos

Índice de figuras

Índice de tablas

1 Introducción

A lo largo de esta sección se presenta la motivación que ha llevado a desarrollar el proyecto que se presenta en esta memoria, se introducen los objetivos que se quieren cumplir en el proyecto, y por último se explica la metodología que se ha seguido para la realización del mismo.

1.1 Motivación

Desde que comencé a estudiar en el ámbito del desarrollo de software, siempre había tenido la inquietud de desarrollar un proyecto tangible para poder demostrar las habilidades aprendidas en todo este camino. El mercado de las aplicaciones móviles siempre me ha interesado por el potencial que hay detrás. Al fin y al cabo, todos tenemos un teléfono móvil con el cual interactuamos todos los días con un gran número de aplicaciones. Por esto tomé la decisión de realizar una App móvil, con la visión de sacar algún día una aplicación al mercado.

¿Cómo surgió la idea que se va a desarrollar en este Trabajo Final de Grado (TFG)?

Casi todas las personas están acostumbrados a la interacción social con amigos y conocidos mediante aplicaciones móviles, en concreto, a través de redes sociales. Pero estas redes sociales están hechas de tal manera que por mucho que se tenga un número de “amigos” en ella, no fomentan la interacción en persona con los mismos. De hecho, todo lo contrario, están hechas para que la personas sean casi más un número o como mucho tan solo un escaparate [1].

Con la motivación de ayudar a la gente, y con el objetivo de paliar los problemas mencionados surge la aplicación que se desarrolla en este TFG, ***Join***. Esta aplicación está concebida como una red social que persigue el objetivo de permitir a las personas, empresas u organizaciones compartir actividades de ocio en las que cualquier usuario pueda unirse y participar en ellas.

¿Por qué ***Join*** es una buena idea?

El objetivo de esta aplicación era suplir una carencia que existía en el mercado. Existen ciertos espacios que intentan conseguir lo que buscamos con esta aplicación, sin embargo, no existe ninguno que abarque todo lo que ***Join*** pretende abarcar. Todos los espacios existentes han sido creados para un objetivo mucho más específico. La idea inicial de ***Join*** consistía en que los usuarios pudieran compartir una actividad que quisieran hacer con otras personas. Al publicar una actividad, a ella se podrían unir todas aquellas personas que estuvieran interesadas en realizar esa actividad. ¿Cuál podría ser un caso real?, pues realmente un usuario podría publicar todas aquellas actividades que imaginara, el límite está en su imaginación. Un ejemplo podría ser publicar una actividad para hacer una ruta en bicicleta, donde el resto de los usuarios interesados en la actividad se unirían y podrían realizar esta actividad juntos, conociéndose en el acto y pudiendo forjar una futura amistad.

También había una oportunidad de que las empresas se promocionasen en una aplicación de este estilo. Existen muchos negocios que organizan actividades dirigidas a las personas. Estos negocios podrían ofrecer estas actividades a aquellos usuarios a los que les interesase visitar o acudir a las actividades de estas empresas, y no acudir a las más informales creadas por otros usuarios. La unión de estas dos ideas combinadas me llamaba la atención y es por eso por lo que me decidí desarrollarla una aplicación que las abordara.

1.2 Objetivos

El objetivo principal de este trabajo es desarrollar ***Join***, una red social accesible desde una App móvil para que usuarios, empresas u organizaciones publiquen actividades a las que se pueden unir otros usuarios. Así pues, la aplicación pretende ayudar a cualquier persona a realizar sus actividades favoritas con otras personas, permitiendo que se conozcan entre sí de forma presencial y desarrollen una amistad 'real' entorno a la actividad.

Además, también se desarrollará un modelo de negocio que permita que la aplicación se sustente a lo largo de los años en el mercado.

Para cumplir estos objetivos se va a hacer una segmentación de pequeños objetivos más específicos que se deberán cumplir durante el desarrollo de la App.

Estos objetivos específicos son los siguientes:

- Obtener un primer MVP (Mínimo Producto Viable) para el testeo temprano de la aplicación.
- Que la aplicación sea sencilla, intuitiva y gratificante de utilizar para el usuario, es decir, que tenga un buen nivel de usabilidad, y que a su vez cuente con un aspecto gráfico atractivo para el usuario.
- Establecer un modelo de negocio orientado a largo plazo (5 años).
- Atraer a empresas y entidades que quieran publicar sus actividades en la App.
- Mantener una fidelidad de estas entidades.
- Determinar futuros modelos de negocio o caminos que pueda seguir la App si estuviese establecida unos años en el mercado.

1.3 Impacto esperado

El impacto que se quiere obtener con esta App está enfocado, 100%, al beneficio de sus usuarios y al de las organizaciones/entidades que quieran publicar sus actividades en dicha red social. A continuación, se desgrana este impacto según el perfil de usuario.

Impacto esperado para un usuario particular:

- Ayudar a resolver un problema que puede aparecer cuando se quiere realizar alguna actividad que requiera de más gente para llevarla a cabo. Por ejemplo, deportes de equipo donde se necesita un mínimo de personas para poder jugar un partido.

- Permitir al usuario conocer a gente nueva con intereses de ocio comunes. Es habitual que una persona se encuentre en una situación donde le resulte difícil conocer gente nueva debido, por ejemplo, a que ha cambiado de residencia u otra situación personal. Con la funcionalidad de esta App puede tener una "excusa" para conocer gente nueva que esté predispuesta a compartir una actividad y así conocerse mientras disfrutan de sus aficiones.

- Dar la oportunidad al usuario de experimentar actividades nuevas donde quizá encuentre una nueva afición.

- Ayudar a encontrar pareja, siempre y cuando la actividad esté descrita con tales intenciones y haya una persona dispuesta a realizarla contigo.

Impacto esperado para las empresas u organizaciones

- Dar visibilidad a actividades ofrecidas por la organización con el objetivo de un ofrecer un beneficio económicamente ventajoso para los usuarios. Por ejemplo: Una actividad en grupo de visita al Oceanogràfic de Valencia creada por la propia entidad.

- Conseguir que estas organizaciones obtengan un beneficio al publicar las actividades de manera que quieran seguir publicándolas y se vuelvan clientes fieles de la App.

1.4 Metodología

El desarrollo de ***Join*** se ha realizado siguiendo el enfoque "waterfall" o "cascada". En este enfoque se sigue un proceso secuencial, donde detrás de una fase se realiza otra sin volver atrás [2].

Principalmente el proceso llevado a cabo consta de las 5 fases típicas de desarrollo de un producto software, donde cada una de estas fases va a corresponder a un apartado de la memoria, exceptuando mantenimiento, que se definirá como un trabajo futuro de este TFG.

1. Análisis. En esta fase será donde se identifiquen todos los requisitos necesarios para empezar a diseñar y desarrollar la aplicación.

2. Diseño. En esta fase se definirá la arquitectura, tecnologías a usar y el apartado visual de ***Join***.

3. Implementación. En esta fase se mostrará como ha sido parte del proceso de desarrollar el primer MVP de ***Join*** mostrando unos ejemplos de *frontend* y *backend.*

4. Pruebas. En esta fase se mostrará como se han diseñado las pruebas unitarias, de integración y dirigidas a usuarios, así como los resultados de estas exponiendo ejemplos en cada una de ellas.

5. Mantenimiento. Esta fase está queda fuera del ámbito del TFG ya que se llevará a cabo cuando la App de ***Join*** se ponga en marcha.

2 Evaluación de la idea de negocio

A la hora de desarrollar cualquier proyecto uno de los aspectos más importantes es hacer un estudio del mercado. En este estudio se analizan todos los competidores que se medirán directamente con la idea del proyecto, ya sea por la similitud o temática que abarca. Sin este estudio nuestra App podría no ser tan competente como las otras, y, por ende, no tener éxito en su lanzamiento al mercado.

En esta sección se va a realizar un análisis de todas aquellas aplicaciones que compiten directamente con la desarrollada en este TFG. También se va a explicar y resumir ***Join*** y su modelo de negocio, para finalmente comparar en una tabla las prestaciones de ***Join*** con las otras aplicaciones analizadas y obtener una conclusión sobre la idea de negocio y estado del mercado en el que se sitúa la App. Estas Apps se basarán todas en la creación de eventos, actividades o situaciones de cierta temática con el objetivo de compartir esa afición con otras personas y socializar con ellas.

2.1 Idea de negocio

2.1.1 Análisis de competidores

A continuación, se van a analizar las aplicaciones que compiten directamente con ***Join*** para obtener una visión objetiva de la competencia.

Meetup[1]

Meetup es una red social móvil creada por 3 emprendedores informáticos en el año 2002. La aplicación permite a sus usuarios reunirse en persona mediante unos grupos de interés como el deporte, la música, la cultura, entre otros más de treinta temas. Un usuario puede tanto crear un grupo de interés como unirse, y el objetivo de estos grupos es compartir eventos u actividades de ese mismo interés juntos. Hoy en día cuenta con más de 59 millones de miembros en todo el mundo repartidos en más de 193 países. Esta aplicación se puede considerar uno de los competidores principales debido a su consolidación en el mercado y aceptación por la gente.

¿Qué ofrece meetup?

Resulta ser que es una aplicación bastante sencilla y quizá esa sea la razón y clave de su éxito. Primero de todo tiene una pestaña donde aparecen todos los eventos y grupos en los que un usuario

[1] **Meetup:** https://www.meetup.com/es-ES

está unido y se va a organizar una quedada pronto. En esa misma pestaña hay un botón de crear grupo donde si te suscribes puedes crear tus propios grupos (ver Figura 1).

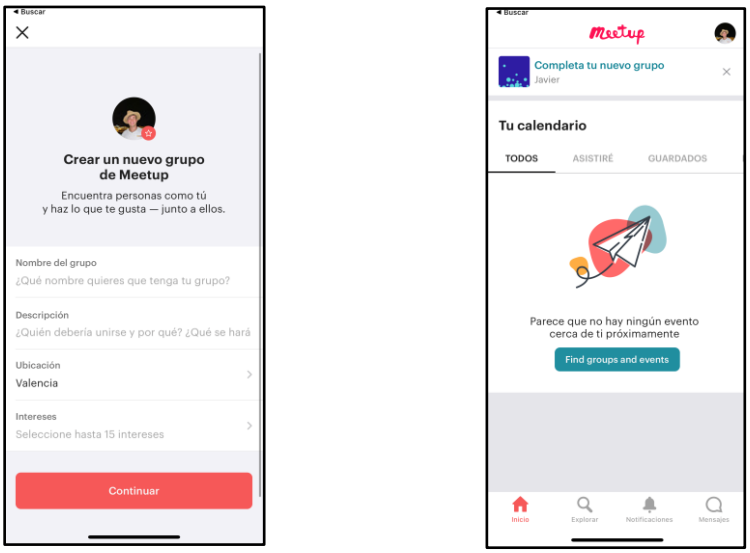


Figura 1-Meetup, Interfaz de creación de grupo (izquierda) y calendario de eventos (derecha)

La interfaz de buscar grupo de interés (ver Figura 2), es aquí donde mediante una serie de filtros más avanzados el usuario puede encontrar aquellos grupos que más le interesen. Estos filtros son:

- Fecha de la actividad a realizar
- Tipo de lugar
- Distancia
- Categoría

Son filtros bastante importantes porque siempre habrá gente que busque grupos de su edad y de pocas personas con el fin de tener más afinidad y cercanía con todos ellos.

Si el usuario decide unirse a un grupo, este podrá ver toda la información interna de dicho grupo, es decir, podrá ver los organizadores, los asistentes, las fotos, los comentarios, así como los siguientes eventos que se van a realizar dentro de ese grupo.

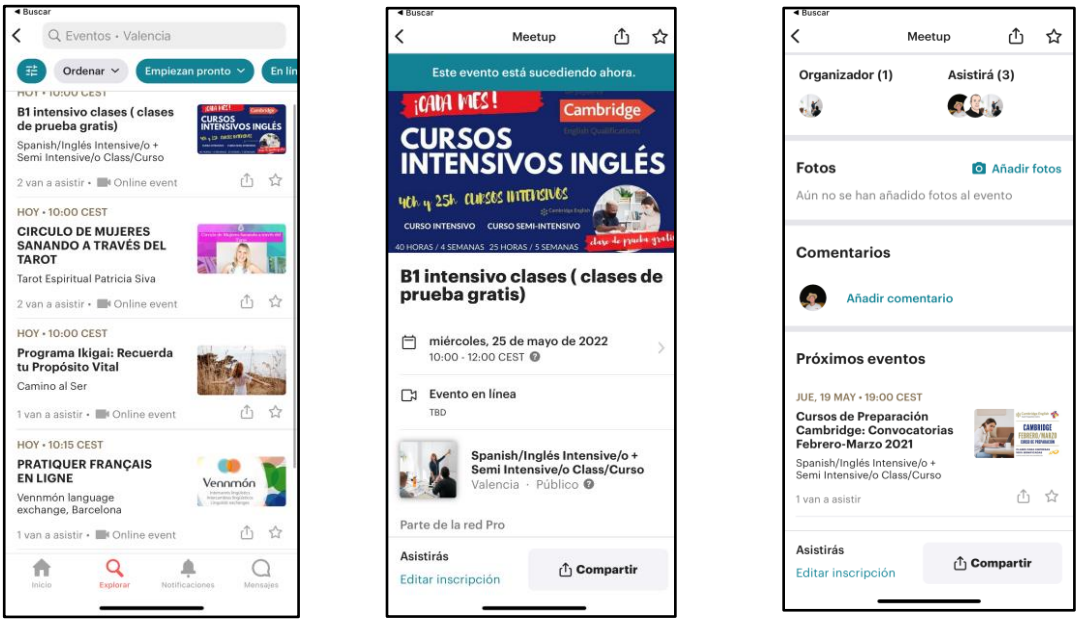


Figura 2 - Meetup, Interfaz de buscar evento y de visualizar evento

Por último, ofrece una pestaña de notificaciones y otra de mensajes (ver Figura 3). En la primera le llegarán al usuario todas las actualizaciones y noticias de los eventos que se realizarán en el grupo, así como información que deseen compartir los dueños de los grupos. En la pestaña de mensajes el usuario puede visualizar un chat interno del grupo, con opción de enviar mensaje directo a otros usuarios.

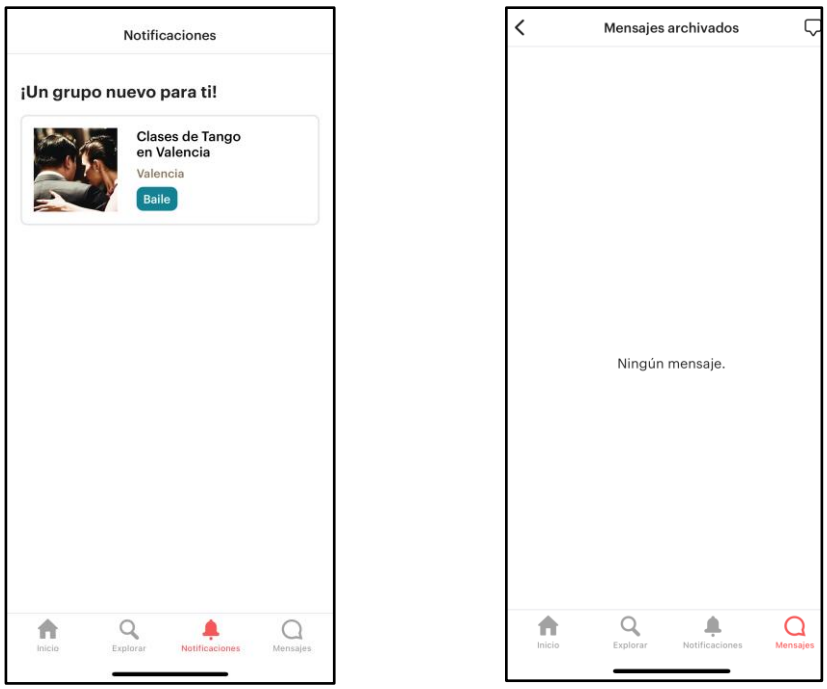


Figura 3 - Meetup, Interfaz de notificaciones y mensajes

Todas estas funcionalidades básicas están realmente pulidas y enfocadas a la usabilidad, esto es parte de por qué ha tenido tanto éxito esta red social en los últimos 10 años.

Sharify[2]

Sharify es una red social orientada a acudir a eventos y planes que hayan organizado los usuarios de su misma ciudad en tiempo real. Es decir, todas estas actividades se ven reflejadas en un mapa que se actualiza en vivo y con el cual el usuario puede interactuar. Fue fundada por la catalana Gemma Prenafeta, quien decidió crear esta aplicación cuando se fue a Londres y se dio cuenta que era complicado conocer a personas de otro país sin una plataforma que te incentivase a ello.

¿Cómo funciona Sharify?

Ofrece la posibilidad de crear y unirte a actividades de distinta índole en tu misma ciudad. También permite la creación de comunidades con la cual poder realizar ciertas actividades los usuarios de esta.

Lo más destacable de la aplicación es su funcionalidad de ver las actividades en vivo, esto lo hace a través de una interfaz de mapa donde el usuario puede ver estos planes en directo y sus respectivas horas y temáticas (ver Figura 4). El usuario puede interactuar con cada una de las actividades que tiene en el mapa pudiendo unirse a ellas y ver toda la información al respecto.

[2] **Sharify**: https://sharifyapp.com/es/

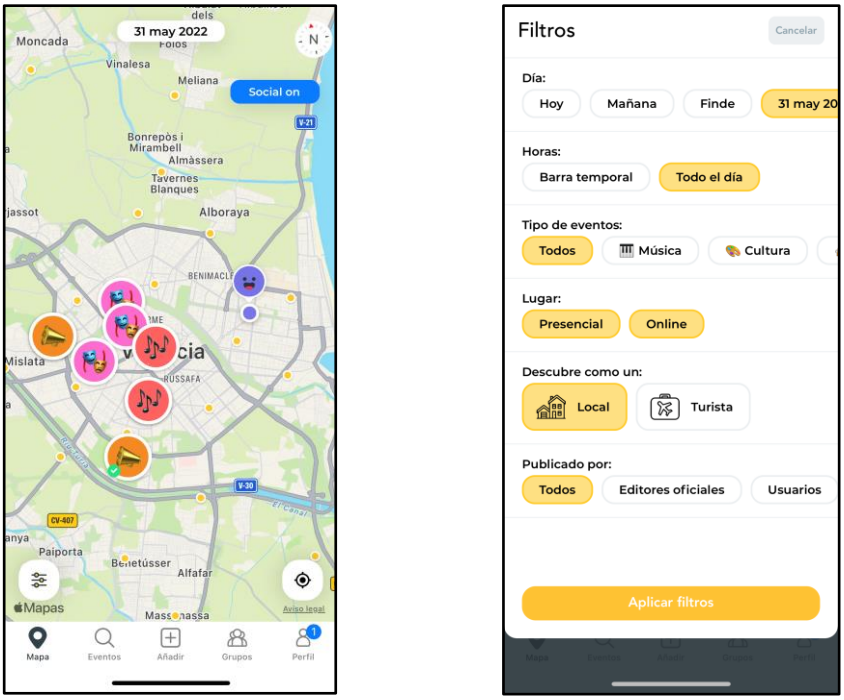


Figura 4 - Sharifiy, Interfaz de buscar y filtrar actividades

A parte del mapa, el usuario puede encontrar actividades en la pestaña de buscar (ver Figura 5). Es aquí donde la búsqueda es más exhaustiva. Esto es porque incluye todo los filtros anteriores y una barra de búsqueda que permite al usuario realizar búsquedas por nombre de evento o plan.

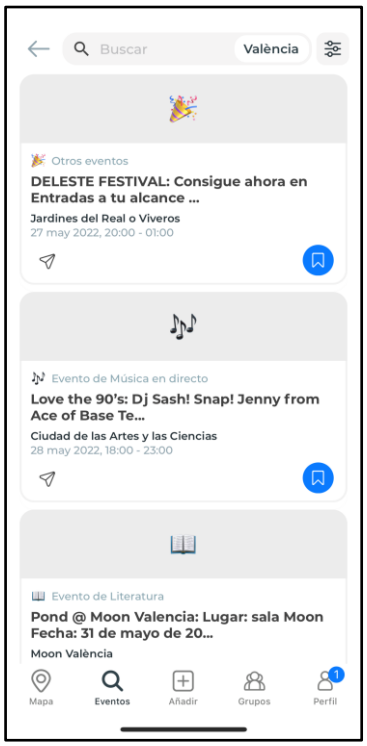


Figura 5 - Sharify, Interfaz de buscar actividad

La base de esta aplicación está en la creación de los planes (ver Figura 6). Sin un apartado de creación sencillo e intuitivo los usuarios estarían menos incentivados a crear actividades y por lo tanto la aplicación no se sustentaría. Cuando un usuario se dispone a crear la actividad podrá elegir la temática del plan, la fecha, el lugar y una breve descripción de esta. La navegación esta guiada por pasos, cuando el usuario elige una temática, se le dirige a la fecha, y usa este método hasta el final de la creación del plan. Es sin duda una de las mejores maneras de mantener la atención del usuario sin bombardearle de información.

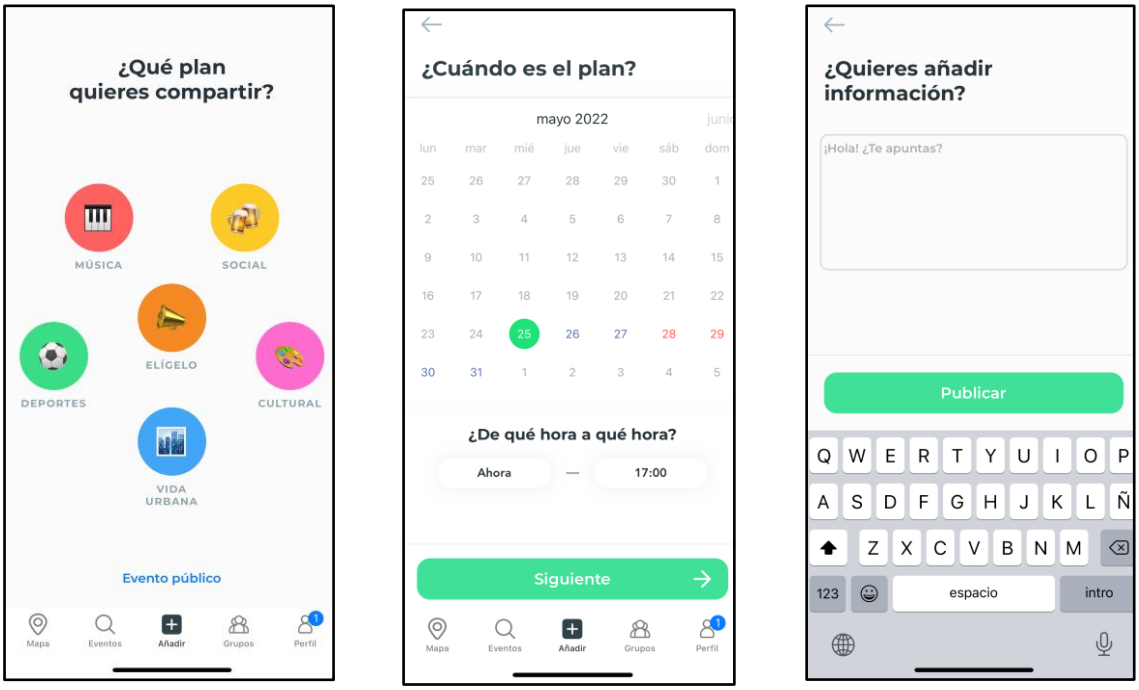


Figura 6 - Sharify, Interfaz de crear actividad

Una funcionalidad extra de la aplicación es la posibilidad de buscar o crear grupos de interés (ver Figura 7), donde por nombre el usuario puede buscar los grupos, así como unirse a ellos e interactuar mediante un chat. Esto permitirá al usuario encontrar otras personas con los mismos gustos que él y sin la necesidad de crear un plan para poder llevar a cabo ese interés juntos. Esto es algo fundamental en este tipo de aplicaciones donde el objetivo es encontrar a gente con la que compartir aficiones.

Por último, dispone de una pestaña perfil donde el usuario puede ver las actividades que él mismo ha creado y a las que se ha unido, así como también actualizar la información de su perfil público (visible para el resto de los usuarios), así como los intereses que él quiera establecer como principales.

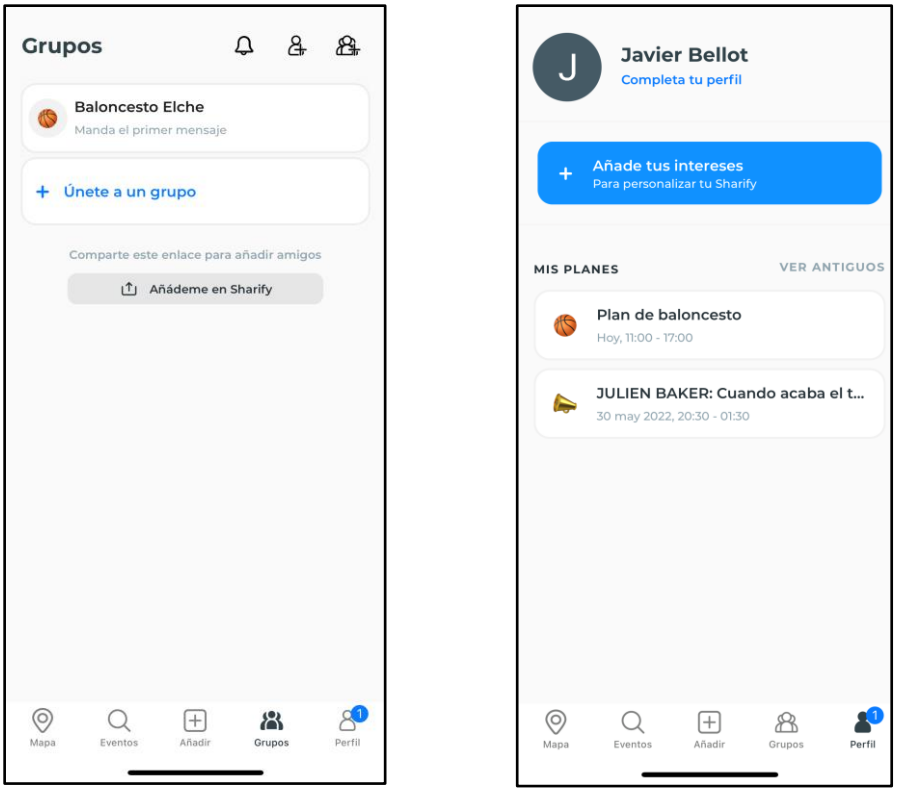


Figura 7 - Sharify, Interfaz de gestión de grupos y Mi Perfil

Eventbrite[3]

Eventbrite es una web y aplicación móvil fundada en 2006 en Estados Unidos. Está orientada a fomentar eventos locales. En esta aplicación ciertos negocios exponen sus eventos tanto de manera gratuita como de pago, con el objetivo de que la gente se apunte y acuda a ellos.

Eventbrite solo se enfoca en eventos, es decir, no existe la interacción entre usuario-usuario a la hora de crear actividades, sino que se enfoca en la interacción entre negocios locales-usuario.

Analizando las funcionalidades que se encuentran nada más entrar en la aplicación, se le recomienda al usuario varios eventos según los intereses que éste ha seleccionado (ver Figura 8). Al clicar el evento el usuario podrá ver la información interna de dicho evento y pagar la entrada para acudir a él.

[3] **Eventbrite:** https://www.eventbrite.es/

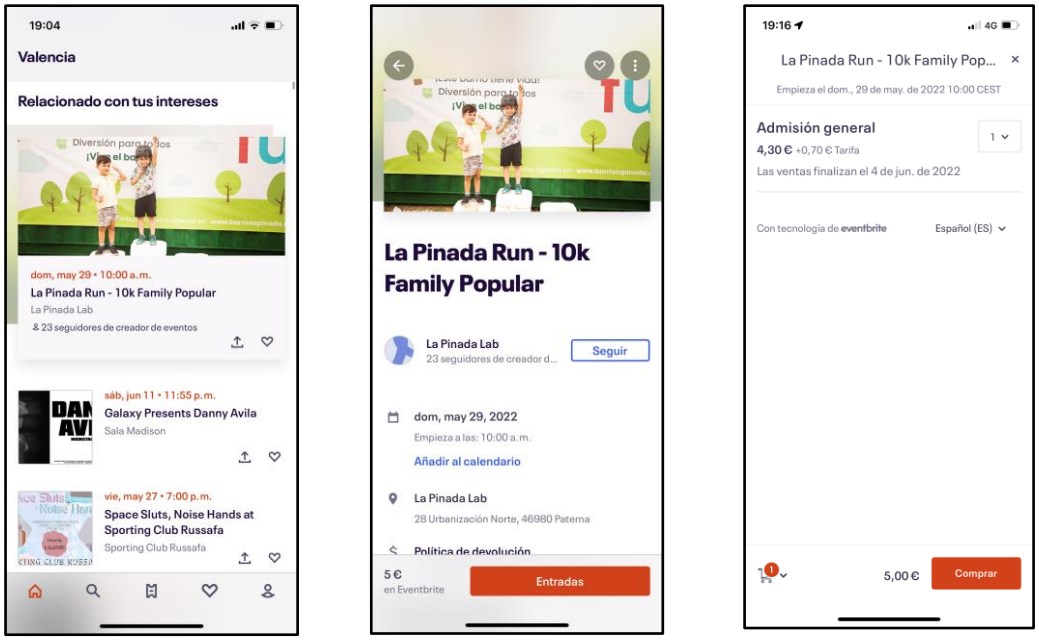


Figura 8 - Eventbrite, Interfaz de Eventos Recomendados, y Unirse a un evento

Como todas las aplicaciones anteriores que se han visto anteriormente, esta cuenta con un apartado de buscar, en este caso, buscar eventos según unos filtros (ver Figura 9). Sin embargo, los filtros que tienen son un tanto limitados, donde el usuario solo puede buscar por fecha e interés.

Figura 9 - Eventbrite, Interfaz de buscar evento

Algo característico de esta App es que tiene un apartado de tickets, donde se almacenan digitalmente las entradas de cada evento, esto es algo que no se ha encontrado en otras aplicaciones y puede ser bastante útil (ver Figura 10).

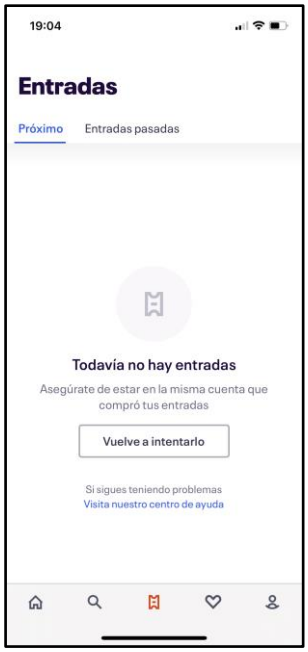


Figura 10 - Eventbrite, Interfaz de Entradas a eventos

Por último, se tiene un apartado de favoritos donde aparecen todos aquellos eventos que el usuario ha marcado como que le interesan, y un apartado de perfil donde el usuario puede seleccionar sus gustos y editar las configuraciones básicas de la aplicación (ver Figura 11).

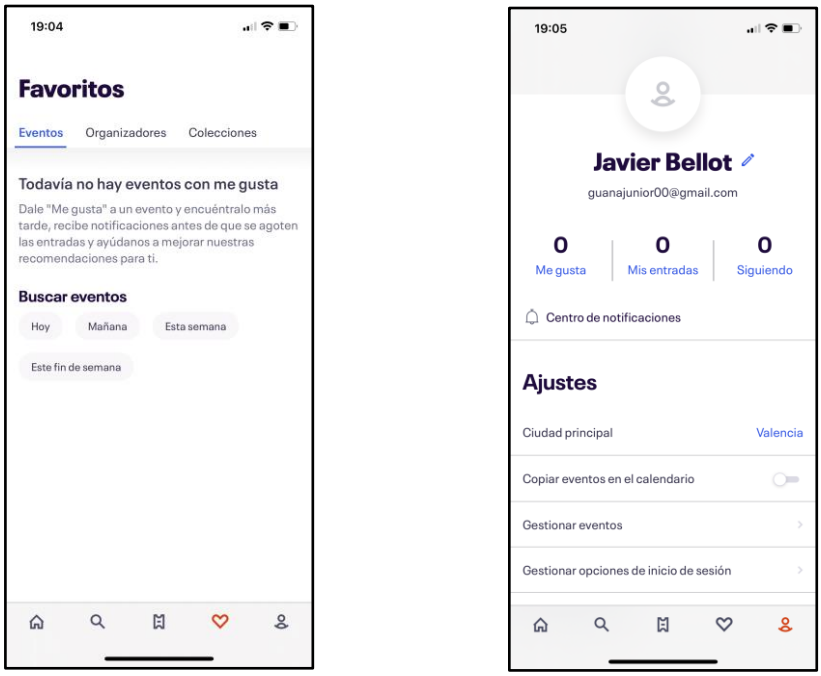


Figura 11 - Eventbrite, Interfaz de Eventos favoritos y Mi Perfil

En conclusión, lo que se puede sacar de esta App es que es una aplicación muy preparada para que los negocios locales se promocionen en ella y el usuario pueda llevar una gestión de aquellos eventos a los que se apuntan. Sin embargo, no se ve competencia en la interacción usuario-usuario ya que no hay una manera de crear actividades que pueda crear un usuario y estén dirigidas a que otros usuarios se unan.

2.1.1.1 Tabla comparativa

A continuación, se muestra una tabla comparativa de las tres Apps analizadas para así poder identificar las carencias que hay en cada una de ellas.

Funcionalidad analizada	**Meetup**	**Sharify**	**Eventbrite**
Actividades recomendadas según los gustos del perfil del usuario	✗	✓	✓
Búsqueda de actividad por mapa en vivo (integración Google Maps)	✗	✓	✗
Chat interno entre miembros de la actividad	✓	✓	✗
Creación de actividad gratuita	✗	✓	✗
Gestión de perfil de usuario	✓	✓	✓
Búsqueda de actividad por mapa en vivo (integración Google Maps)	✗	✓	✗
Buscar actividades por filtros dinámico	✗	✓	✓
Filtrar por fecha	✓	✓	✓
Filtrar por edad	✗	✗	✗
Filtrar por número de miembros/participantes	✗	✗	✗
Filtrar por género	✗	✗	✗
Filtrar por temática/interés	✓	✓	✓
Gestión y visualización de las actividades creadas por el usuario y a las que se ha unido	✓	✓	✓
Gestión y ajustes del perfil de usuario	✓	✓	✓
Foto de perfil	✓	✓	✓

Compartir actividad con otros usuarios	✓	✓	✗
Creación de grupos de interés y temática	✓	✓	✗
Descripción y categorías favoritas de usuario	✓	✓	✓
Eliminación de actividades	✓	✓	✗
Calendario de eventos/actividades	✓	✗	✓

Tabla 1 - Tabla comparativa de las aplicaciones de la competencia

La Tabla 1 tiene el objetivo de mostrar todos los "huecos" u oportunidades que ***Join*** puede abarcar y que no presentan algunas de estas aplicaciones. ***Join*** intentará abarcar todas estás funcionalidades con el objetivo de ser una App más completa que el resto de las aplicaciones analizadas.

2.2 Resumen de *Join*

Join es una App móvil que como funcionalidad principal tiene permitir a los usuarios compartir sus actividades favoritas para que otros usuarios se unan y puedan realizarla juntos. Además, también contempla que el usuario pueda unirse a actividades de pago creadas por organizaciones que se dedican a ello.

Cada usuario podrá buscar actividades de ciertas categorías dentro de su área y aplicar una serie de filtros para encontrar las que más se adecuen a él. Los mismos filtros los puede especificar el promotor de la actividad a la hora de la creación.

El usuario podrá encontrar estas actividades filtrando por rango de edad, número de participantes, categoría, km a la redonda, fecha, y género. El resultado de esta búsqueda puede ser o bien en forma de lista, o gráficamente en forma de mapa. Los filtros que aplica el usuario son dinámicos, permitiendo una búsqueda rápida y sencilla.

Una vez el usuario se haya unido a una actividad, este podrá chatear con el resto de los usuarios apuntados a dicha actividad para así poder especificar más detalles sobre la misma.

Independientemente de las actividades que se pueden publicar, los mismos usuarios podrán crear grupos de interés de ciertas temáticas. En estos grupos, de tamaño indefinido, el usuario encontrará gente que comparte sus gustos, y dentro de ellos se organizarán actividades internas que solo estarán disponibles para dichos grupos.

Como se mencionaba anteriormente, son las entidades u organizaciones las que podrán publicar sus actividades de pago, personalizando a detalle las mismas. Por ser una organización y publicar estas actividades privadas estas deberán pagar una cuota. Además, las entidades aportarán una pequeña comisión ínfima de los beneficios que reciben de las inscripciones de los usuarios a la App.

Ante esta situación se ha de plantear qué tipo de entidades querrían publicar y promocionarse en la App. Se ha considerado que cualquier entidad que ofrezca alguna actividad de ocio podría estar interesada en registrarse en ***Join***, esto incluye restaurantes, discotecas, cines, zoos, campamentos y todo tipo de organizaciones cuyo beneficio sea que las personas acudan a ellos.

2.2.1 Modelo de negocio

Existen varias maneras de monetizar una App móvil hoy en día. Algunas de estas maneras son menos intrusivas que otras, ya que, de normal, una aplicación donde se hagan pagos tenderá siempre a ser más intrusiva que una que no tiene ningún pago por parte del consumidor. Es por eso por lo que se le ha dado varias vueltas a la hora de pensar el modelo, ya que uno de los objetivos era que la aplicación fuese poco intrusiva y agradable de usar.

Lo primero que se tuvo en cuenta fue que un producto software es más una maratón que un sprint, esto significa que hay varias maneras de monetizar que suelen hacer que una aplicación dure poco. Los anuncios dentro de la aplicación es una ellas, ya que invaden completamente la pantalla del usuario sin su permiso y de normal no estará contento con esto. Por eso decidimos eliminar directamente esta opción.

Otro modelo que se consideró inviable para nuestro caso fue el premium, donde para usar la aplicación el usuario tiene que comprarla con un precio fijo. En las aplicaciones que involucran a personas como motor de funcionamiento no suele ser ideal ponerlas de pago, ya que es una gran barrera de entrada para utilizarla y el número de usuarios se reduciría en una cantidad muy grande.

Tras observar que ninguno de estos dos modelos encajaba correctamente en la aplicación, se estudió el modelo Freemium. Este modelo de negocio se basa en ofrecer la aplicación como un servicio gratuito y cobrar dinero por otras funcionalidades y servicios más avanzados [3]. Sin embargo, se le dio una vuelta de tuerca a este modelo, ya que no se quería que al usuario se le cobrase nada por obtener más funcionalidades, sino que de primeras las tuviese todas disponibles. Es por esto por lo que el modelo SaaS (Software as a service) se ofrece a organizaciones y entidades que quieran optar a publicar sus actividades y eventos de pago en nuestra aplicación. Estas tendrían que adquirir unas licencias en donde se les cobraría una trimestralidad de coste bajo por poder publicar y promocionar sus actividades.

Normalmente es posible tener varias maneras de generar ingresos dentro de una aplicación. Al tener decidido que nuestro software se iba a presentar como un servicio a estas entidades se pensó que se podría obtener algún otro ingreso por cada actividad de pago que se publicase. Este sería un pequeño beneficio en forma de porcentaje por cada persona que pagase por unirse a una actividad de una organización. Es decir, si un zoo organiza una actividad que cuesta 20€, la aplicación obtendría un pequeño porcentaje de esa cantidad por cada persona que se apuntase. Este porcentaje sería alrededor de un 3%, el porcentaje ínfimo que se ha mencionado anteriormente, una cantidad que no supusiese una molestia para la entidad.

Estos dos modelos serían la base económica del proyecto, pero se han considerado otros modelos que podrían implantarse en el futuro de la aplicación.

Una buena opción es que, si ***Join*** consigue funcionar bien y consigue tener un equipo grande detrás, se podrían publicar actividades de pago propias, donde todo el beneficio de estas inscripciones iría directamente para ***Join***. Las actividades podrían consistir en competiciones de ciertos deportes, viajes organizados por ***Join*** u otro tipo de actividad de ocio que pueda gestionar un equipo interno.

Por último, otra opción viable es que, si ***Join*** llegara a tener una base de usuarios fieles, quienes estén dispuestos a pagar un servicio mensual premium, estos podrían acceder a una serie de actividades gratuitas exclusivas para este tipo de usuarios.

Sin embargo, para poder implementar estas dos últimas se ha de tener una cantidad de usuarios base que estuviesen contentos con la aplicación y un equipo grande detrás de ***Join***, por lo que se ha considerado inviable para los primeros años de la aplicación.

2.2.2 Proyección económica

Es importante planear y estudiar los gastos y beneficios económicos que puede tener la aplicación en los próximos años. Sin un análisis de proyección económica un lanzamiento al mercado es sentenciar el fin de tu proyecto.

Ahora se comentará desde un punto de vista objetivo y sincero la proyección económica de nuestra idea.

En primer lugar, se debe decidir cuanto se les cobrará a las empresas/organizaciones por poder publicar una actividad en nuestra aplicación. Se ha decidido dividir este servicio en varios planes de distintos precios que permitan publicar un número máximo de actividades con un número máximo de participantes (Tabla 2).

	N.º de actividades a publicar permitidas	**N.º máximo de participantes por actividad**	**Precio trimestral**
Básico	10	10	180€
Estándar	20	20	240€
Premium	30	30	300€

Tabla 2 - Planes ofrecidos a las empresas u organizaciones en Join

Cálculo del beneficio de licencias

Se tiene que establecer un *precio medio de una licencia*, ya que en la tabla se trabajará con este valor. Se puede asumir que el porcentaje de licencias vendido es el mismo para los 3 planes de precios, pero esto no sería realista, ya que de primeras prácticamente todas las empresas cogerían el plan básico. Finalmente se ha estimado que el porcentaje de compras de licencias será: 60% de licencias serán Básicas, 30% serán estándar, y el último 10% serán premium.

Si se realizan los cálculos:

60% de 180€ = 108€

30% de 240€ = 72€

10% de 300€ = 30€

Precio medio por licencia = 108€ + 72€+ 30€

Total = 210€

Con estos valores y unas estimaciones de licencias vendidas por trimestre se puede calcular los beneficios relacionados a las licencias. Estos resultados se mostrarán en la Figura 12 donde se van a mostrar los ingresos que vienen de este modelo de negocio y del que se va a comentar ahora.

Cálculo del 3% de comisiones

Ahora hace falta calcular cuánto beneficio se va a sacar de las comisiones del 3% en cuanto a las inscripciones de pago por parte de los usuarios. Se pueden tener en cuenta los porcentajes anteriores, es decir, un 60% de las entidades clientes podrán publicar 10 actividades trimestrales, un 30%, 20 actividades, y por último un 10% podrán publicar 30 actividades. Es probable que algunas organizaciones no lleguen a publicar todas las actividades de su cupo, pero pagando por una suscripción intentarán publicar las máximas posibles, es decir, la mayoría de las entidades de normal publicará un 80-90% de las actividades máximas de su cupo.

Con una simple fórmula es posible calcular el importe que se obtendrá de este modelo de negocio secundario todos los trimestres.

*(N.º de organizaciones activas en la plataforma * N.º de actividades medio que publica una organización * N.º de participantes medio en una actividad de pago * Precio medio de una actividad de pago) * Cantidad de comisión que recibe* ***Join*** *(3%)*

El número de organizaciones activas en la plataforma es un valor que se tiene que intuir respecto a nuestro plan de marketing y crecimiento de la aplicación. Sin este valor el cálculo de la proyección económica es imposible de obtener. El valor se representará en las figuras que se van a mostrar a continuación, donde se ve el número de licencias que hay nuevas por trimestre, el total de licencias, los gastos, y los beneficios trimestrales y totales, todo para poder obtener una visión económica predecible y futura.

En cuanto al precio medio de una actividad de pago, lo decide la propia empresa y es un valor variable. Pero para estimar se ha de tomar un valor fijo, y se ha considerado que 15€ es un valor aceptable que cualquier usuario estaría dispuesto a pagar.

Ahora vamos a calcular el resto de las variables de la fórmula.

Primero de todo calcular el valor del *N.º de actividades medio que publica una organización.*

Calculando:

Número de actividades disponibles para publicar en un plan de inscripción, entre el porcentaje de entidades que usarán ese plan de suscripción, de las cuales un aproximadamente un 85% serán publicadas.

Este último valor de 85% se ha estimado (se ha comentado anteriormente, se estima que las entidades publicarán un número total de actividades del 80-90% del máximo de su plan de suscripción), debido a que se tendría que testear la aplicación en un mercado real para obtener el valor real. Se ha considerado que es un valor optimo ya que una entidad que pague una suscripción de normal querrá publicar todas las actividades que tenga en su cupo disponibles, como se ha mencionado anteriormente, pero puede no ser que no llegue al máximo, es por eso por lo que el 85% de actividades publicada de su cupo, se ha considerado un valor que se daría en un caso real.

(60% de 10) * 0.85 = 5.1 ≈ 5

(30% de 20) * 0.85 = 5.1 ≈ 5

(10% de 30) * 0.85 = 2.55 ≈ 2

5+5+2 = **12 actividades al trimestre por organización / entidad**

Es posible obtener una estimación sobre el *N.º de participantes medio en una actividad de pago* utilizando la misma metodología. Solamente hay que tener en cuenta el porcentaje de llenado de las plazas de una actividad. Es decir, si hay 10 plazas, se puede llenar el 30%, 50%, 70% etc. Esto es muy difícil de estimar ya que depende de si la actividad gusta, si somos realistas e incluso conservadores se puede tener en cuenta que solo el 50% de las actividades se llenarán. Esto significa que los cálculos en principio podrían salir más altos en valores si las actividades se llenasen siempre.

Calculando:

Número de plazas disponibles en un plan de inscripción, entre el porcentaje de entidades que usarán el plan de suscripción que tiene asignado ese máximo, de las cuales un aproximadamente un 85% serán publicadas.

(60% de 10 plazas) * 0.50 = 3

(30% de 20 plazas) * 0.50 = 3

(10% de 30 plazas) * 0.50 = 1.5 ≈ 1

3+3+1 = **7 participantes de media en una actividad de pago**

Si se junta esta fórmula de la comisión sumado al beneficio que se obtiene por cada suscripción trimestral de las empresas es posible obtener los beneficios e ingresos trimestrales y totales (Figura 12).

Número de Trimestre	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Año	1				2				3				4				5			
Tipos de licencias acumuladas	T1	T2	T3	T4	T1	T2	T3	T4	T1	T2	T3	T4	T1	T2	T3	T4	T1	T2	T3	T4
Licencias nuevas vendidas	4	4	4	5	8	14	10	16	10	14	10	15	18	20	22	24	26	27	35	30
Licencias renovadas trimestrales	0	4	8	10	16	19	26	28	35	36	40	40	44	49	63	68	73	60	69	83
Numero de personas inscritas en actividades de pago	0	360	720	900	1440	1710	2340	2520	3150	3240	3600	3600	3960	4410	5670	6120	6570	5400	6210	7470
Total de licencias nuevas vendidas y renovadas	4	8	12	15	24	33	36	44	45	50	50	55	62	69	85	92	99	87	104	113
Ingresos Trimestrales																				
Ingresos trimestral	991 €	1.982 €	2.974 €	3.717 €	5.947 €	8.177 €	8.921 €	10.903 €	11.151 €	12.390 €	12.390 €	13.629 €	15.364 €	17.098 €	21.063 €	22.798 €	24.532 €	21.559 €	25.771 €	28.001 €
Total Ingresos	991 €	2.974 €	5.947 €	9.664 €	15.611 €	23.789 €	32.710 €	43.613 €	54.764 €	67.154 €	79.544 €	93.173 €	108.536 €	125.635 €	146.698 €	169.495 €	194.027 €	215.586 €	241.357 €	269.359 €

Figura 12- Beneficio esperado de Join a lo largo de 5 años

Tras calcular los ingresos hay que analizar cuáles van a ser los gastos trimestrales, para así poder obtener el balance. Se ha considerado todos los gastos básicos que se necesitan en para poder poner en marcha la App e ir mejorando con el tiempo. Hay gastos que no se obtienen hasta el comienzo de cierto trimestre, ya que se necesitaría un cierto capital para disponer de ellos.

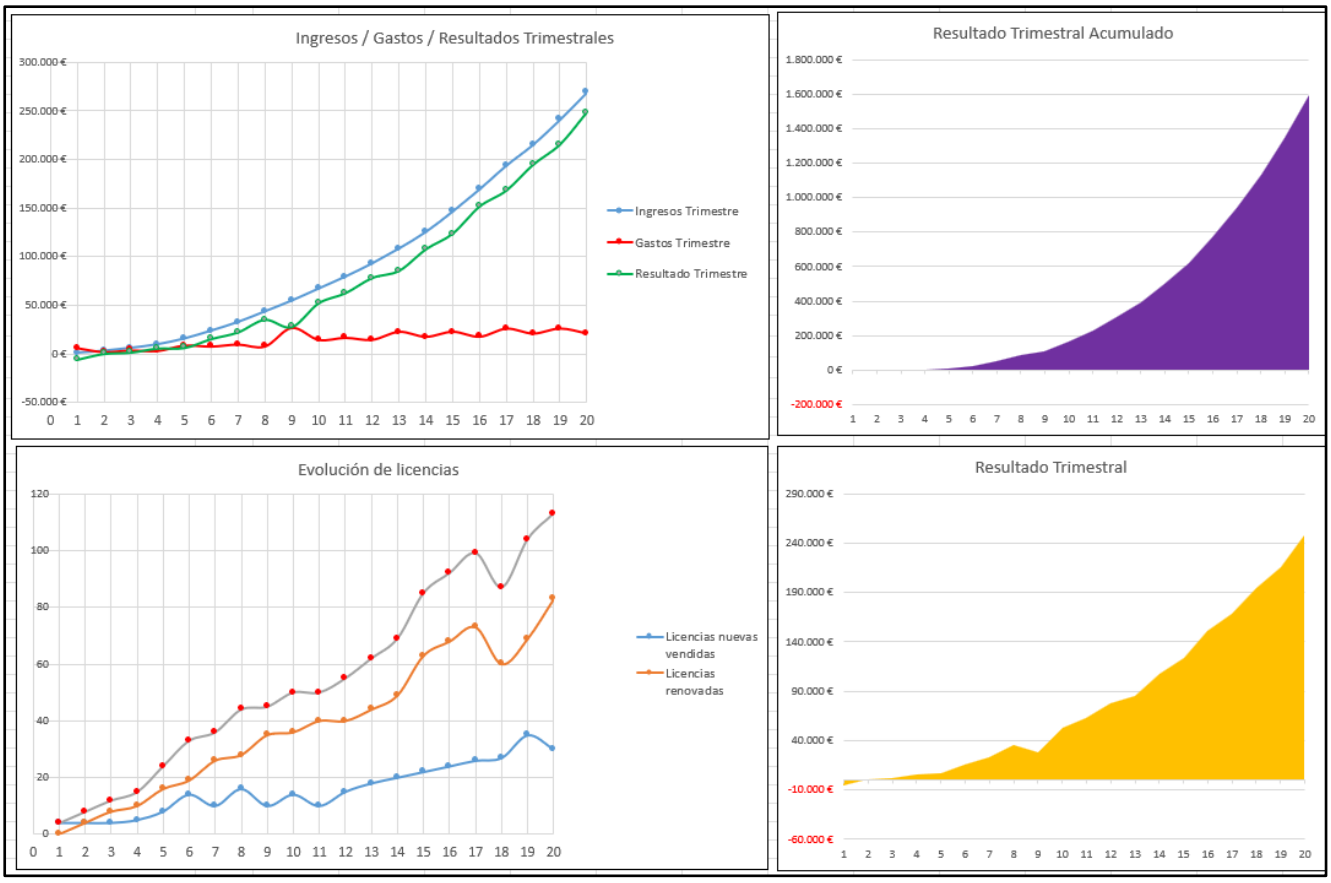


Figura 13 - Balance económico gráfico en los próximos 5 años

Gastos Anuales																				
Ordenadores	3.000 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	10.000 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €
Base de datos alojada en Amazon Web Services	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €	198 €
Sociedad Limitada	3.000 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €	0 €
Marketing (publicitario)		500 €	500 €	0 €	1.000 €	0 €	2.000 €	0 €	2.000 €	0 €	2.000 €	0 €	5.000 €	0 €	5.000 €	0 €	5.000 €	0 €	5.000 €	0 €
Desarrolladores	0 €	0 €	1.600 €	1.600 €	3.500 €	3.500 €	3.500 €	3.500 €	7.000 €	7.000 €	7.000 €	7.000 €	7.000 €	7.000 €	7.000 €	7.000 €	10.000 €	10.000 €	10.000 €	10.000 €
Gestoría	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €	100 €
Trabajadores de marketing	0 €	1.500 €	1.500 €	1.500 €	4.000 €	4.000 €	4.000 €	4.000 €	7.000 €	7.000 €	7.000 €	7.000 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €
Api de google	0 €	0 €	0 €	0 €	20 €	48 €	87 €	115 €	137 €	159 €	197 €	219 €	237 €	251 €	298 €	319 €	425 €	497 €	534 €	598 €
Proveedor del chat de Join- GetStream.io	200 €	200 €	200 €	200 €	200 €	200 €	200 €	200 €	300 €	300 €	300 €	300 €	300 €	300 €	300 €	300 €	300 €	300 €	300 €	300 €
Total Gastos	6.498 €	2.498 €	4.098 €	3.598 €	9.018 €	8.046 €	10.085 €	8.113 €	26.735 €	14.757 €	16.795 €	14.817 €	22.835 €	17.849 €	22.896 €	17.917 €	26.023 €	21.095 €	26.132 €	21.196 €
Resultado Trimestral	-5.507 €	476 €	1.849 €	6.066 €	6.593 €	15.743 €	22.625 €	35.500 €	28.029 €	52.397 €	62.749 €	78.356 €	85.701 €	107.786 €	123.802 €	151.578 €	168.004 €	194.491 €	215.225 €	248.163 €
Resultado Trimestral Acumulado	-5.507 €	-5.031 €	-3.182 €	2.884 €	9.478 €	25.220 €	47.845 €	83.345 €	111.374 €	163.770 €	226.519 €	304.875 €	390.576 €	498.362 €	622.164 €	773.742 €	941.746 €	1.136.237 €	1.351.462 €	1.599.625 €

Figura 14 - Balance económico numérico en los próximos 5 años

De los datos de las figuras Figura 13 y Figura 14 se puede concluir que se trata de un proyecto a largo plazo, donde los beneficios de los primeros trimestres serán muchos, pero serán justos para poder cubrir los gastos necesarios. Pero conforme pasan los años si se desarrolla un buen producto y se llevan buenas campañas de marketing, los ingresos incrementarán enormemente. Al incrementar los ingresos el equipo podrá crecer y lo mismo ocurrirá con el dinero dirigido al marketing y al desarrollo. Lo que marca la diferencia es probablemente las comisiones que se vaya a llevar ***Join*** por inscripción, ya que parece algo insignificante, pero al final del día trae unos ingresos tremendamente altos si la aplicación es muy activa.

2.2.3 Análisis DAFO

Para la obtención de una mejor de idea sobre cuáles pueden ser los aspectos fuertes y débiles de ***Join*** se ha realizado un análisis DAFO [4], en este diagrama se analizará las debilidades, amenazas, fortalezas y oportunidades. Las debilidades y fortalezas las corresponden a los puntos positivos y negativos internos, y las amenazas y oportunidades a los respectivos externos.

Figura 15 - Análisis DAFO

Si se puede sacar algo en claro del diagrama de la (Figura 15) es que hay bastante fortalezas y oportunidades que sin duda el que más se beneficia de ellas es el usuario. Al fin y al cabo, la base de una aplicación social son los usuarios y se han de cuidar lo máximo posible.

Sin embargo, el aspecto negativo es que como aun nos encontramos en la etapa de estudiantes, la experiencia es lo que flaquea en el proyecto. Viendo esto sería ideal plantearse contratar a algún desarrollador con más experiencia en el sector para que ***Join*** no se quede atrás comparado con la competencia.

2.2.4 Lean Canvas

El modelo Lean Canvas [5]es una herramienta que sirve para esbozar y analizar el modelo de startups. Es muy útil para cualquier proyecto emprendedor que quiere saber si su idea es apta para el mercado, con ella se puede obtener una imagen general que ayudara a saber si la idea puede fracasar o no.

2 Problema

- RRSS de hoy en día son muy superficiales, en vez de priorizar la interacción humana natural
- Incapacidad de socializar en entornos desfavorables (estancia en otro país, o región)
- Dificultad en encontrar personas con las que compartes gustos e intereses
- Aplicaciones intrusivas y bombardeantes de información no relevante a lo que busca o necesita el usuario
- Mala usabilidad en las aplicaciones

4 Solución

- Creación de actividades de manera sencilla e intuitiva
- Resolución de la intención de búsqueda del usuario
- Grupos de interés y actividades recomendadas según tus gustos.
- Char directo con aquellas personas con las que quieras interactuar
- Permitir a entidades publicar sus actividades privadas

8 Métricas

- Aumento del número entidades suscritas al plan mensual
- Número de actividades generadas
- Número usuarios registrados y activos en la aplicación
- Alcance de Join a nivel de marketing

3 Proposición de valor

Aplicación completa que permite al usuario encontrar a personas con sus mismos gustos, compartir actividades con ellos, crear grupos de interés de distintas índoles e interactuar con sus miembros.

Se podrá acudir a ciertos eventos privados que hayan promocionado empresas suscritas y dar una a las personas una oportunidad de socializar en este mundo tan digital.

9 Ventaja competitiva

Tener una aplicación social que evita lo superficial e incentiva a las personas a quedar entre ellas y conocerse. Es difícil de encontrar algo así porque la mayoría priorizan el que el usuario se enganche al contenido digital de la misma sin buscar un objetivo final mas puro.

5 Canales

- Redes Sociales
- Plataformas de publicidad
- Web

1 Clientes

Clientes:

Usuarios:

- Cualquier persona que este interesada en compartir experiencias sin que nadie le obligue a pagar nada por ello.

Organizaciones:

- Locales/empresas o entidades que quieran promocionar sus actividades para promover su negocio.

Early adopters:

- Gente interesada en conocer nuevas personas
- Gente que busca compartir un hobby
- Personas que busquen debatir y organizar actividades en un grupo de interés.
- Entidades u organizaciones que tengan un negocio basado en que la gente acuda a su actividad/evento y quiera promocionarse pagando poco.

7 Costos

Costos iniciales:

- Creación de una empresa/sociedad : 3000€
- Material informático : 3000€
- Marketing publicitario inicial: 4000€
- Gestoría: 600€
- Chat interno GetStream.io: 200€

Inversión inicial: 10800€

6 Ingresos

- Servicio mensual dividido en 3 paquetes a las empresas u organizaciones que quieran publicar sus actividades
- 1% de cada inscripción de pago que haga un usuario a una actividad de pago publicada por una organización.

Figura 16 - Lean Canvas

La Figura 16 muestra el modelo Lean Canvas aplicado al proyecto ***Join***. Este modelo puede ayudar a visualizar las ventajas o desventajas del proyecto analizado, ya que hace reflexionar al

autor para que se dé cuenta de si la idea tiene potencial o no. En el caso de ***Join***, principalmente se pueden destacar varios aspectos:

- Son muchas las carencias que hay actualmente en las Apps analizadas y donde **Join** puede cubrir dicho nicho.
- Se diferencia bastante de la competencia.
- ***Join*** tendría un gran número de "early adopters".
- Hay mucho potencial de generación de ingresos.

2.2.5 Conclusiones de la evaluación

Tras haber hecho todo el análisis y evaluación de competidores de ***Join*** es posible sacar varias conclusiones en claro. En primer lugar, ***Join*** se ha comparado con aplicaciones que llevan años en el mercado y con un equipo muy grande detrás de ellas. Por mucho que el objetivo de nuestra aplicación sea abarcar todas las funcionalidades que a los otros competidores les falta, es difícil llegar, en los primeros años del proyecto, a la calidad de una empresa madura.

Es por eso por lo que, contra esas Apps, ***Join*** si puede ser competitiva si se instaura en el mercado a lo largo de los años, pero por el momento, las aplicaciones con las que puede competir son aquellas que su objetivo principal no sea hacer que las personas puedan compartir actividades con otros usuarios, con el objetivo de socializar y encontrar gente con la que realizar dichas.

Como se ha visto en el análisis todas las Apps revisadas flaquean en algún apartado, algunas solo buscan tener grupos de interés, otras solo mostrar eventos locales de la ciudad, y solo una de ellas (Sharify) se centra al 100% en emparejar a sus usuarios en una actividad para así compartir esa afición, y en esa en concreto donde las búsquedas son bastante limitadas, ya que no se tiene en cuenta aspectos tan fundamentales como la edad, género y número de personas con las que el usuario desea juntarse. Hay que mencionar también que, si el objetivo del usuario es encontrar a una persona del sexo opuesto, de su misma edad y de establecer una relación más íntima a través de una actividad con ella, una aplicación como Sharify no supliría su necesidad.

3 Análisis del problema

A lo largo de esta sección se va a realizar un análisis segmentado sobre el dominio de la aplicación a desarrollar. Con este análisis se especificarán los requisitos funcionales y no funcionales, el modelo del dominio, las características del sistema y, por último, los casos de uso que se dan, usando diagramas. Todos los diagramas que se han realizado en esta sección se han realizado usando la herramienta software para creación de diagramas LucidChart[4].

3.1 Especificación de requisitos

Para la obtención de los requisitos del sistema se ha usado la descripción del dominio o problema que se explica en el apartado 2.2 "Resumen de *Join*". De este apartado saldrán requisitos individuales a nivel funcional y no funcional, de los cuales, a partir de los funcionales, se podrá obtener casos de uso que engloben varios de estos requisitos.

3.1.1 Requisitos funcionales

Los requisitos se han obtenido mediante la técnica Brainstorming[5] cómo se aplicó en la asignatura de cuarto de IS *Análisis y especificación de requisitos*.

Los requisitos van a ser redactados en lenguaje natural cumpliendo las reglas de este tipo de especificación de ser claros, concisos, completos y consistentes [6].

A continuación, se identifican los requisitos con mayor prioridad.

1. **Creación de una actividad por un usuario estándar:**

- Un usuario debe ser capaz de definir un título de la actividad que desea crear.
- Un usuario debe ser capaz de definir una corta descripción de menos de 200 caracteres de la actividad que desea crear.
- Un usuario podrá escoger solo una temática para la actividad que desea crear. Por ejemplo, "montañismo".
- Un usuario podrá escoger el rango de edad de los participantes de su actividad, limitando este número entre los 18-75 años.
- Un usuario podrá escoger el número de participantes máximo que quiera que acudan a su actividad

[4] **LucidChart:** https://www.lucidchart.com/

[5] **Brainstorming**: Herramienta de trabajo que facilita el surgimiento de buenas nuevas ideas sobre un problema determinado.

- Un usuario podrá escoger si desea que a la actividad solo puedan unirse personas de un mismo sexo o de sexo mixto.
- Un usuario podrá escoger el día, fecha y hora de la actividad, siempre y cuando se cumpla que sea 1 hora posterior a la hora actual.
- El usuario tendrá la opción de definir un lugar geográfico para la actividad.
- El usuario tendrá acceso directo a la información y chat de la actividad que ha creado.

2. Búsqueda de actividad de usuario estándar:

- El usuario podrá filtrar actividades por categorías y subcategorías. Por ejemplo, categoría: deporte, subcategoría: montañismo.
- El usuario podrá filtrar actividades por género.
- El usuario podrá filtrar actividades por rango de edad.
- El usuario podrá filtrar actividades por número de participantes de la actividad.
- El usuario podrá filtrar actividades por fecha.
- El usuario podrá buscar actividades en un mapa geográfico.
- El usuario podrá buscar actividades por distancia en km a la redonda.
- Los filtros se deben aplicar dinámicamente en la aplicación sin tener que realizar ningún tipo de recarga.

3. Unirse a una actividad de usuario estándar:

- El usuario podrá unirse a cualquier actividad creada por otro usuario siempre y cuando cumpla los requisitos que se han especificado para ella.
- El usuario tendrá acceso directo a la información y chat de la actividad que ha creado (esto se hará mediante una pestaña “mis planes”).
- El usuario podrá interactuar mediante un chat con las personas que también se hayan unido a la actividad.
- El usuario podrá ver el perfil de los usuarios que pertenecen a esa actividad.
- El usuario podrá unirse a una actividad de pago creada por una entidad.
- El usuario pagará la cuota de la de la actividad de pago creada por la entidad mediante una pasarela de pago en la aplicación.

4. Perfil de usuario estándar y Perfil de entidad organizadora:

- El usuario tendrá acceso a una vista con toda la información de su perfil.
- El usuario podrá editar toda la información y parámetros que conlleva la creación de un perfil.

5. Creación de una actividad por una entidad organizadora:

- Una organización debe ser capaz de definir el título de la actividad que desea crear.

- Una organización debe ser capaz de definir una corta descripción de menos de 200 caracteres de la actividad que desea crear.
- Una organización podrá escoger solo una temática para la actividad que desea crear. Por ejemplo, “montañismo”.
- Una organización podrá escoger el rango de edad de los participantes de su actividad, limitando este número entre los 18-75 años.
- Una organización podrá escoger el número de participantes máximo que quiera que acudan a su actividad.
- Una organización podrá escoger si desea que a la actividad solo puedan unirse personas de un mismo sexo o de sexo mixto.
- Una organización podrá escoger el día, fecha y hora de la actividad, siempre y cuando se cumpla que sea 1 hora posterior a la hora actual.
- Una organización tendrá la opción de definir un lugar geográfico para la actividad.
- Una organización podrá especificar un precio para la actividad.

6. Inicio de sesión:

- Cualquier tipo de usuario, organizador o estándar debe poder iniciar sesión con un email y contraseña si se ha registrado previamente.

7. Registro en la aplicación:

- Un usuario estándar deberá introducir su nombre y apellidos, con el objetivo de que se muestre en el perfil de dicho usuario.
- El usuario estándar deberá introducir su fecha de nacimiento, para que esa edad se utilice a la hora de poder unirte a ciertas actividades.
- El usuario estándar deberá definir su sexo.
- El usuario estándar deberá definir el rango de edad que quiere en las actividades que se le mostrarán como recomendadas.
- El usuario estándar deberá definir la distancia máxima a la redonda de las actividades que se le mostrarán como recomendadas.
- El usuario estándar deberá elegir sus temáticas y subtemáticas favoritas, siendo el mínimo 3 a escoger, con el objetivo de usar estas como base para las actividades que se le recomienden.
- El usuario estándar podrá introducir una descripción que le describa como persona para introducirse a los demás.
- El usuario estándar tendrá la opción de añadir una foto de perfil personal que se mostrará de escaparate en su perfil.
- La entidad organizadora deberá establecer un nombre para su organización.
- La entidad organizadora deberá establecer una descripción para su organización.
- La entidad organizadora deberá establecer una imagen de perfil para su organización.

A continuación, se identifican los requisitos con menor prioridad

8. **Creación de grupos de interés:**

- Cualquier usuario podrá definir una temática para el grupo de interés.
- Cualquier usuario podrá escoger una geolocalización opcional para un grupo de interés.

9. **Búsqueda de grupos de interés:**

- Un usuario debe poder buscar grupos de interés por geolocalización.
- Un usuario debe poder buscar grupos de interés por temática.

3.1.2 Requisitos no funcionales

Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación. Son propiedades o cualidades que el producto debe tener. A continuación, se identifican los requisitos no funcionales para ***Join***:

1. Todas las pantallas / componentes se deben cargar en menos de 5 segundos con el objetivo de no perder la atención del usuario.
2. El sistema debe relacionar los errores con los conceptos que el usuario pueda entender, es decir, mensajes claros de error que no desorienten al usuario.
3. La aplicación se debe poder usar en los sistemas operativos más usados, Android e IOS.
4. Diseñar las interfaces con una sintonía de color que sea agradable a la vista.

3.2 Modelo de dominio

En un análisis del problema es necesario un modelo de dominio que plasme las entidades o conceptos que están envueltos en el dominio de este. En la Figura 17 muestra el modelo de dominio de ***Join*** que se ha realizado usando el lenguaje UML. El modelo especifica las distintas relaciones entre las entidades del dominio.

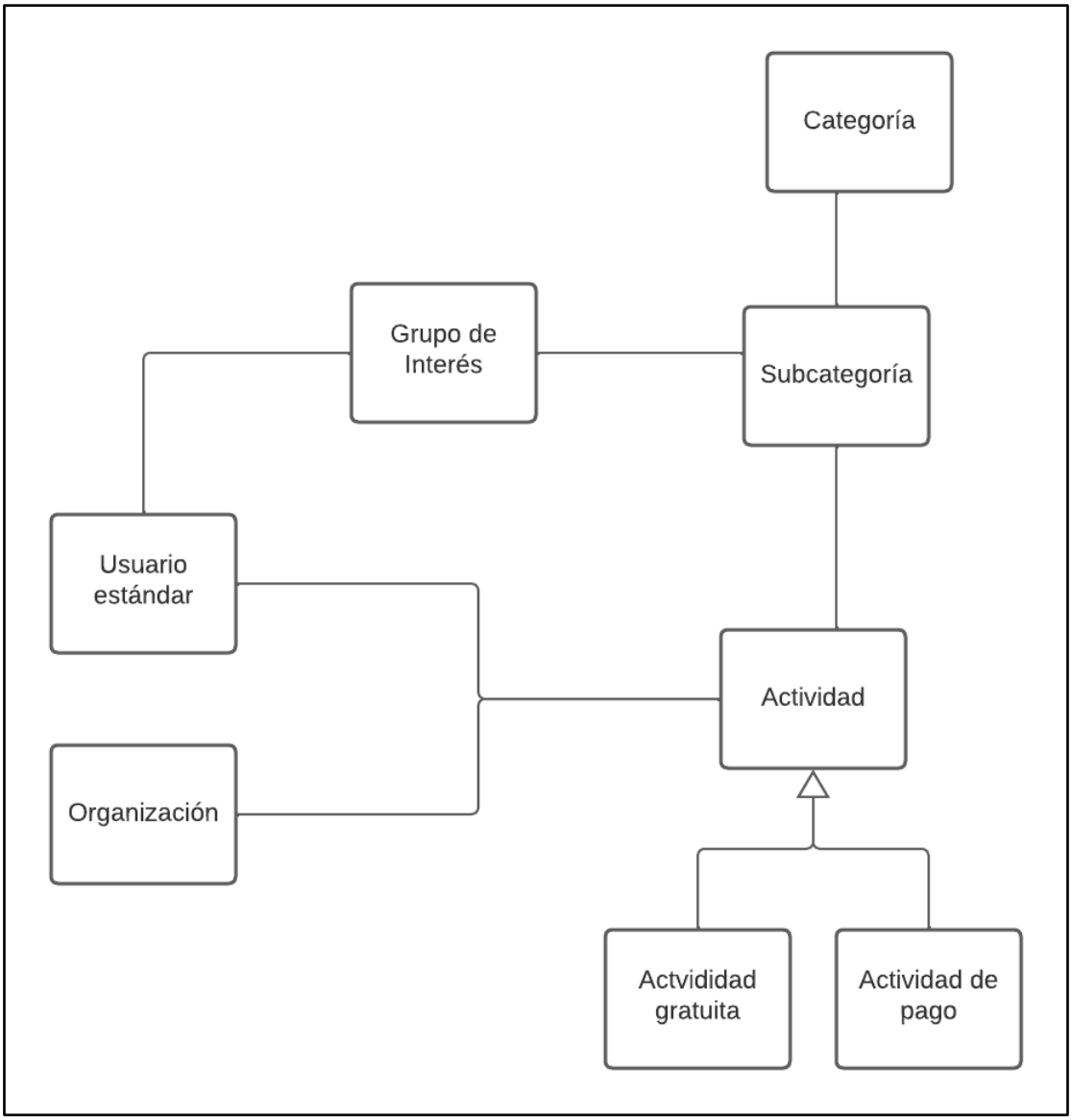


Figura 17 - Modelo de dominio

La base de la aplicación son las actividades, los usuarios pueden crearlas y unirse a ellas, por ello se ve una relación entre los dos tipos de usuarios y la propia actividad. Todo grupo de interés y actividad está relacionado a una subcategoría o categoría, esto facilita que el usuario estándar pueda buscar los grupos que pertenecen a las categorías que más disfruta. Esto se representa mediante las relaciones que se han dibujado entre Usuario estándar y Grupo de interés, así como entre grupo de interés y actividad con categoría y subcategoría.

3.3 Características del sistema

Las características de un sistema son las funcionalidades del mismo vistas desde un alto nivel. Estas características van a permitir identificar las agrupaciones de casos de uso desde la perspectiva y contexto de estas funcionalidades generales.

A continuación, se muestran las características del sistema de ***Join***:

Autenticación y sesiones

Inicio de sesión / autenticación y registro de usuarios estándares y entidades organizadoras.

1. **Gestión de actividades**

   Creación y eliminación de actividades por parte de usuarios estándares y entidades organizadoras.

2. **Búsquedas de actividades**

   Sistema de búsqueda de actividades involucrando los filtros dinámicos y las actividades recomendadas por la aplicación.

3. **Gestión de grupos de interés**

   Creación y eliminación de grupos de interés.

4. **Configuración de perfil**

   Gestión de parámetros del perfil de usuario.

3.3.1 Diagrama de contexto

El diagrama de contexto marca las limitaciones del sistema desde una vista de alto nivel. Realizar el diagrama de contexto permite observar gráficamente los dos subsistemas en los que se va a dividir la aplicación. La Figura 18 muestra el diagrama de contexto de ***Join***, en donde se identifican dos subsistemas, el de usuarios estándar y el de entidades organizadoras. Estos dos subsistemas serán los que más adelante serán usados para dividir los casos de uso del subsistema de usuario estándar, y el de las entidades organizadoras (Figura 18).

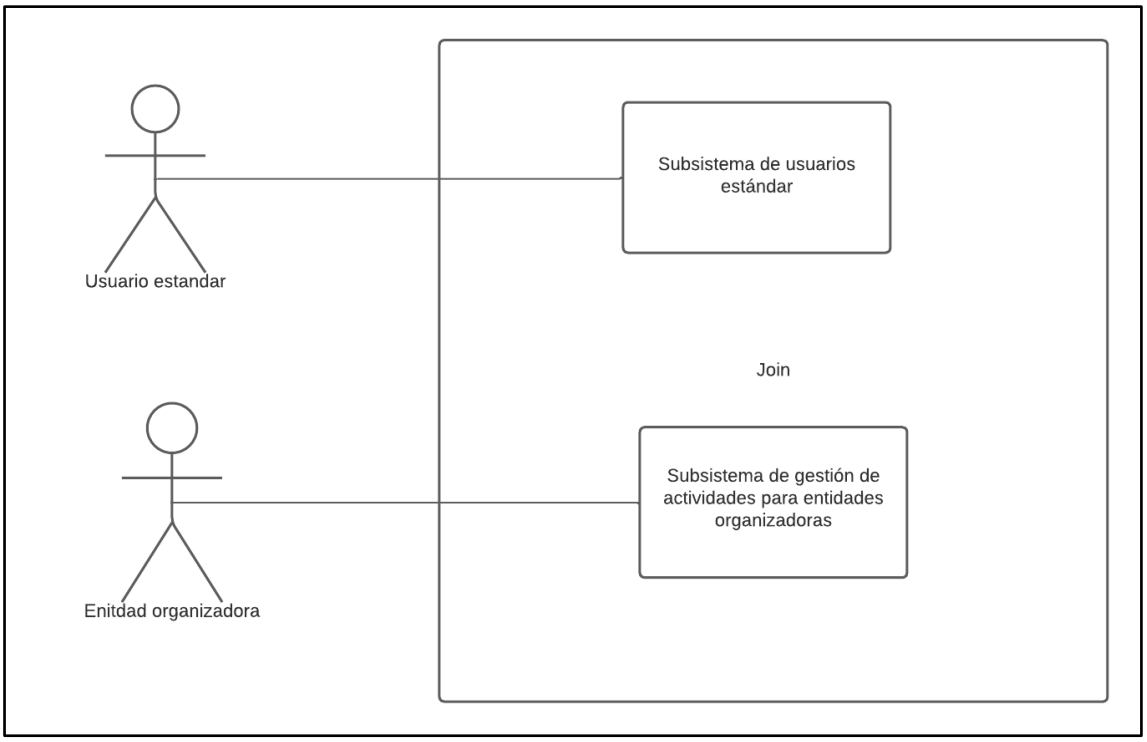


Figura 18 - Diagrama de context de Join

3.4 Casos de uso

Los casos de uso describen acciones que realizan los actores de una aplicación. Llevan a cabo algún proceso o funcionalidad dentro de esta [7]. Es importante definir los diagramas de casos de uso para saber qué hace cada usuario del sistema, especialmente para la fase del desarrollo.

A continuación, se muestran los diagramas correspondientes a los casos de uso que se dan en ***Join***.

DIAGRAMAS

Subsistema de usuarios estándar

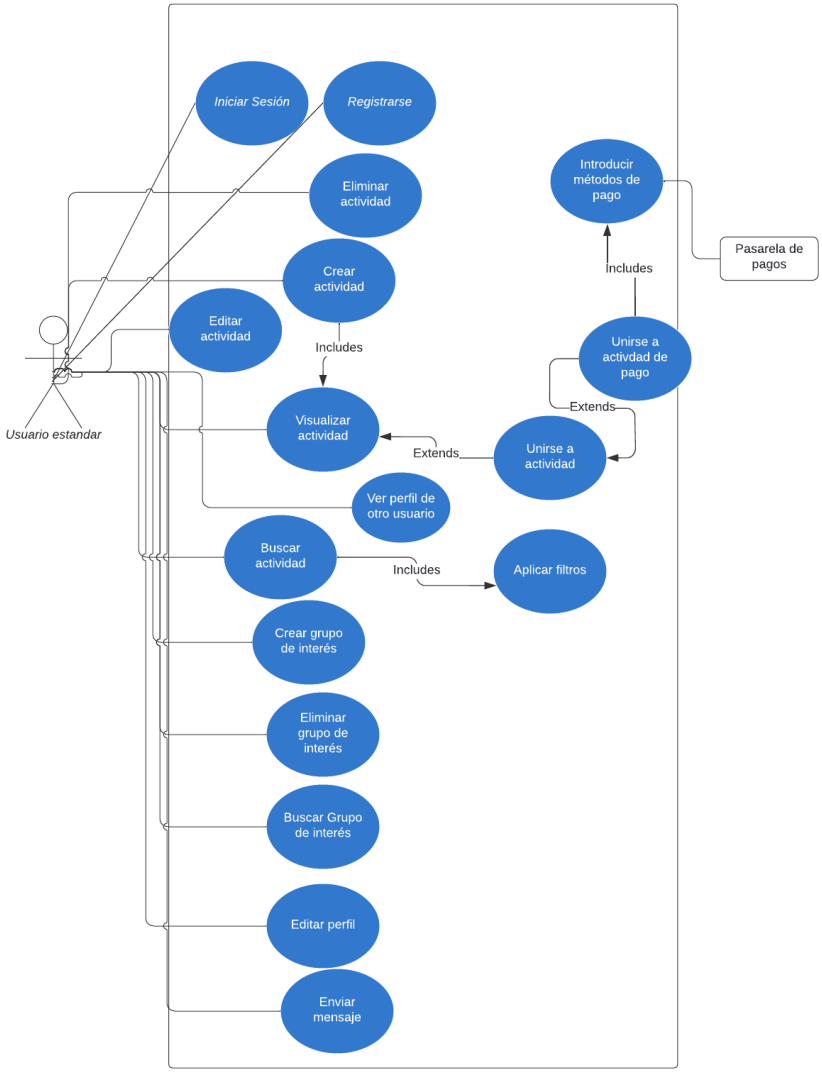


Figura 19 - Diagrama de casos de uso - Subsistema de usuario estándar

Subsistema de entidades organizadoras

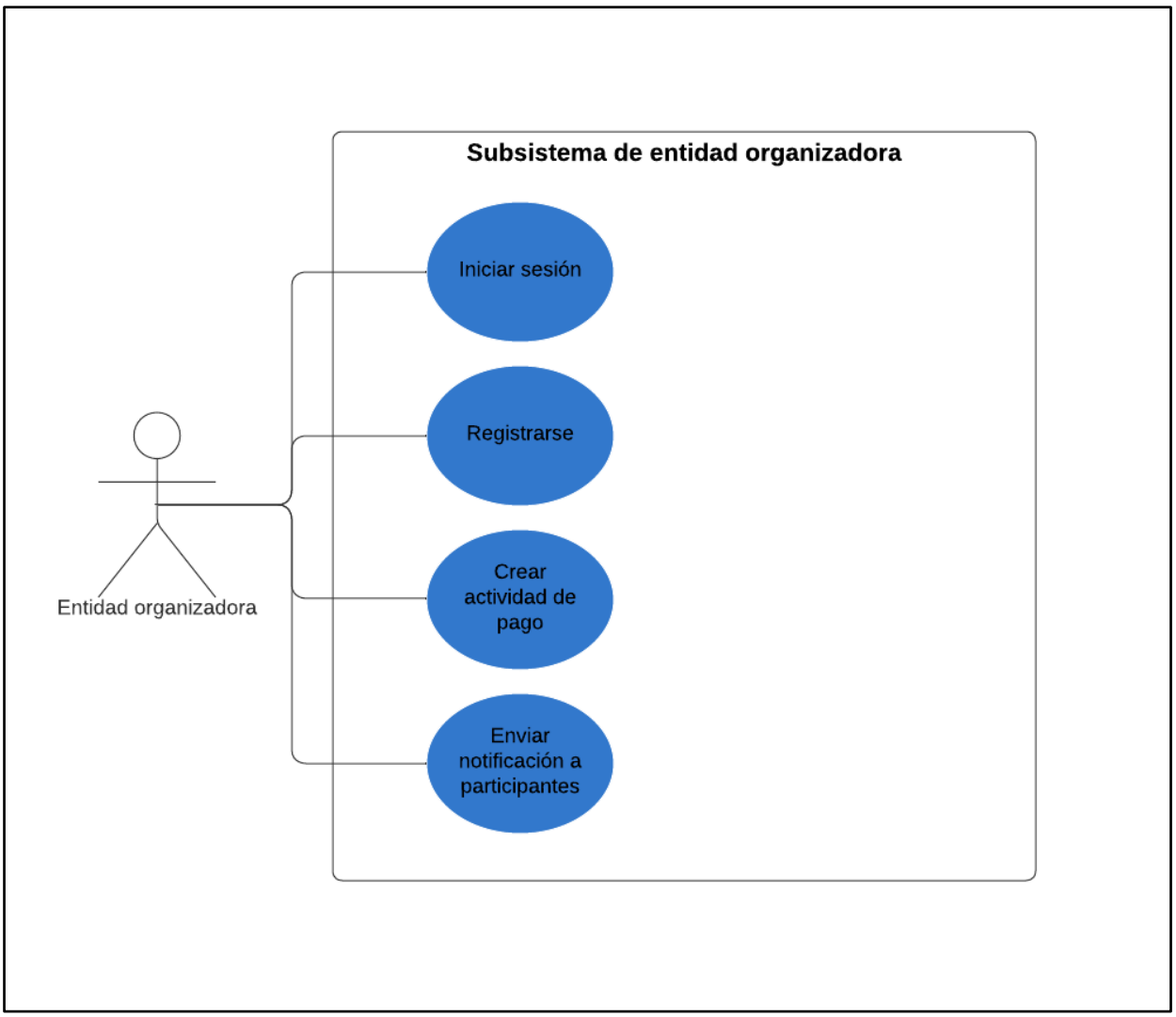


Figura 20 - Diagrama de casos de uso - Subsistema de entidades organizadoras

4 Diseño de la solución propuesta

Esta sección presenta el diseño de la solución propuesta organizada en varios bloques. En primer lugar, se presenta la arquitectura elegida para llevar a cabo el desarrollo (4.1), en segundo lugar, las tecnologías que se van a usar para desarrollar la aplicación (4.2), y por último el diseño y evaluación de las interfaces gráficas de usuario (4.3).

4.1 Arquitectura del sistema

Para llevar a cabo el desarrollo de ***Join*** se ha elegido una arquitectura por capas [8], una de las más utilizadas en el desarrollo por su simpleza. Esta arquitectura consiste en dividir la aplicación en capas, donde cada capa tiene un rol bastante definido. Realmente el número de capas suele variar dependiendo del contexto, pero suelen ser entre 3-4 capas las más comunes. En el caso de ***Join***, se divide la aplicación en un *frontend*, *backend* y una base de datos. Hay que tener en cuenta que cada uno de estos elementos puede tener una o más capas. El número total de capas en esta aplicación es de 4. Se pueden dividir en las siguientes: (1) Capa de presentación (2) Capa de lógica de negocio, (3) Capa de persistencia de datos y (4) Capa de base de datos (ver Figura 21). El *frontend* engloba la capa de presentación (1), y el *backend* la capa (2), (3) y (4)

Gráficamente se puede ver como se comunican estas capas para tener una idea de cómo funciona internamente la arquitectura.

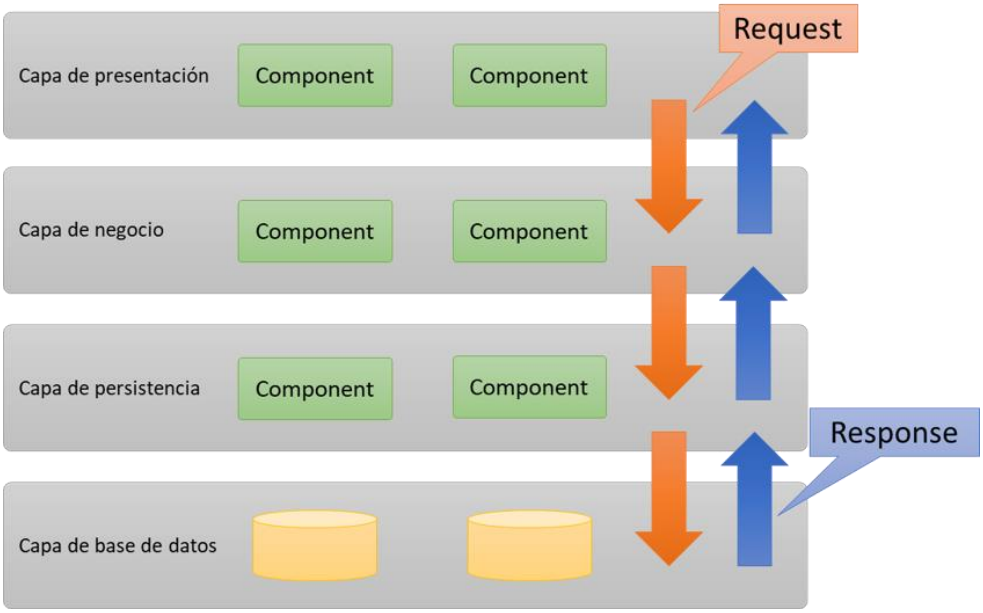


Figura 21 - Arquitectura en capas de Join

Visualmente es más sencillo de entender, cada capa solo es capaz de comunicarse con la que tiene inmediatamente por debajo, formando un flujo si una capa quiere comunicarse con la última. Cada capa se encarga de una tarea distinta, esto hace que se cumpla el principio de separación de preocupaciones [9].

Elegir esta arquitectura otorga ciertas ventajas con respecto a otras arquitecturas como son:

- Separación de responsabilidades
- Fácil de desarrollar
- Fácil de probar
- Fácil de mantener
- Seguridad

4.2 Tecnologías

A continuación, se enumeran las tecnologías usadas en ***Join***. Se han agrupado según han sido utilizadas en el *frontend, backend* o base de datos.

4.2.1 Tecnologías utilizadas en el *frontend*

Hay que destacar que al ser una aplicación móvil existen tecnologías muy distintas que se pueden utilizar. En el caso de ***Join*** se ha utilizado Ionic Framework [10], que se trata de un SDK de código abierto orientado al desarrollo de *webapps* o aplicaciones híbridas. Esto significa que utiliza tecnología web que se interpreta en el navegador para construir aplicaciones móviles, con posibilidad de que también funcionen en web. Por esta razón Ionic se ha de montar encima de un Framework[6] web.

El escogido ha sido Angular JS[7], un Framework de JavaScript que permite crear SPA's (Single page aplications, o aplicaciones de una sola página). Al ser tecnología web, el *frontend* se ha construido con HTML para el esqueleto de las páginas de la aplicación, CSS, para darle estilo a este esqueleto, y Typescript, para manejar la vista que se ha montado. Se podría decir que Ionic utiliza la infraestructura de Angular JS para montar las aplicaciones móviles, algo que facilita la construcción de estas debido al enorme potencial de Angular.

[6] **Framework:** Marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software.
[7] **Angular JS:** https://angularjs.org

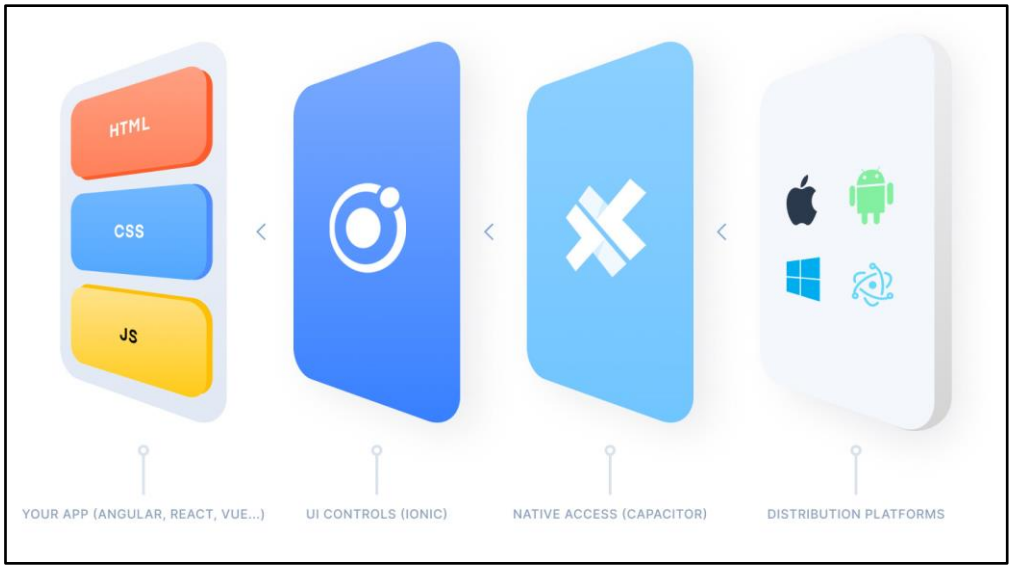


Figura 22 - Arquitectura interna frontend

Como se ve en la Figura 22, Ionic es usado con HTML, CSS y Typescript. Lo que se ve a la derecha es lo que diferencia un *frontend* de una página web con uno de una aplicación móvil. Las aplicaciones tienen que comunicarse con los sistemas operativos nativos del teléfono para obtener cierta información (como por ej. la geolocalización, las notificaciones, la cámara o la galería de fotos del teléfono) que se necesita para la App. Todo esto es gestionado a través de una herramienta como puede ser Capacitor JS[8] [11], que comunica el Framework de Ionic con las plataformas nativas como Android e IOS.

4.2.2 Tecnología usada en el *backend*

La tecnología *backend* fue la decisión más complicada de tomar debido al gran número de tecnologías potentes que existen en este contexto. Finalmente se decidió usar Spring Boot[9], una extensión de Spring Framework que usa Java y utiliza las mejores prácticas sin perder flexibilidad.

La arquitectura del *backend* que ofrece Spring Boot es una arquitectura que incluye 3 de las 4 capas de nuestra arquitectura del sistema.

- Capa de negocio: Maneja toda la "lógica de negocio" o funcionalidad. Consta servicios y utiliza los servicios proporcionados por las capas de acceso a datos.
- Capa de persistencia: La capa de persistencia contiene toda la lógica de almacenamiento de datos y traduce los objetos de negocio desde y hacia la base de datos.
- Capa de base de datos: En esta capa se realizan las operaciones CRUD (crear, recuperar, actualizar, eliminar).

[8] **Capacitor JS**: https://capacitorjs.com

[9] **Spring Boot:** https://spring.io/projects/spring-boot

Spring Boot usa Hibernate[10] y JPA [12], siendo la primera una herramienta de mapeo objeto-relacional que transforma los modelos o clases de Java, en tablas de base de datos automáticamente. La segunda se trata de una API de persistencia que combinada con Hibernate permite tener todo lo necesario para consultar, modificar y crear nuestra BD[11] con un nivel de abstracción más elevado que de normal. Todo esto lo hace siempre y cuando se haya cumplido las normas relacionales a la hora de definir atributos y relaciones en los modelos.

En la Figura 23 se muestra como es la arquitectura estándar que utiliza SpringBoot en los proyectos.

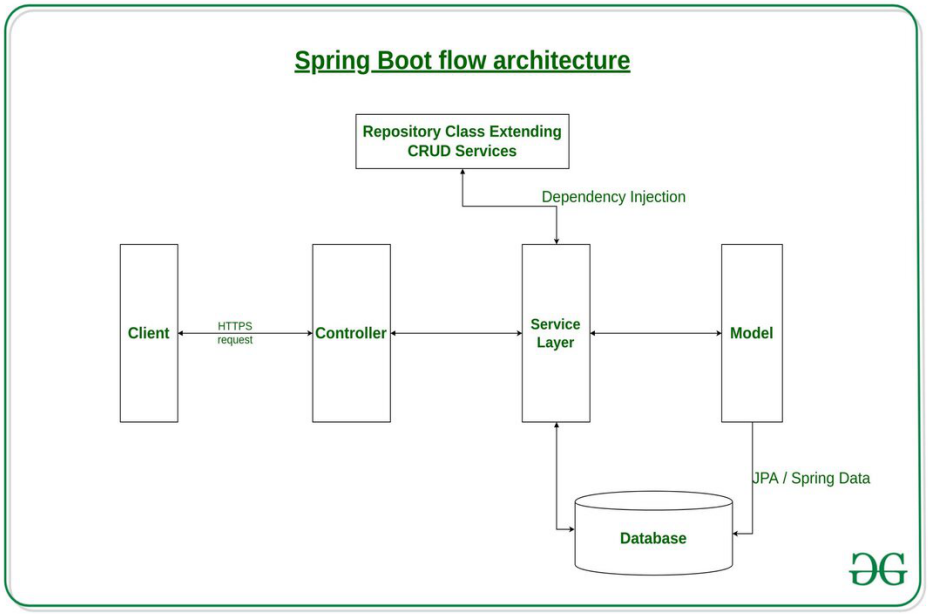


Figura 23 - Arquitectura y flujo de Spring Boot

4.2.3 Tecnología usada para la Base de datos

El dilema de las bases de datos hoy en día es si usar una BD no relacional o una Relacional. Un sistema con un diseño de base de datos complejo es más complicado de trabajar en una BD no relacional, sin embargo, una relacional puede ser más complicada de montar de primeras, pero luego facilita bastante el trabajo a la hora de usarla.

Por este motivo se ha escogido el sistema de gestión de bases de datos PostgreSQL[12], el cual usa PgAdmin como aplicación de escritorio para gestionar las BD's.

[10] **Hibernate:** https://hibernate.org
[11] **BD :** Diminutivo de Base de datos
[12] **PostgreSQL:** https://www.postgresql.org

Esta BD se ha creado usando los servicios de Amazon “AWS” (Amazon Web Services)[13] donde se puede crear una base de datos gratuita. Estará hospedada y soportada por Amazon, lo cual resta preocupaciones para el desarrollador. Inicialmente estará un poco limitada pero la ventaja es que escala automáticamente. Por lo tanto, si hay pocos usuarios, es gratis, pero en cambio si la aplicación es muy concurrida, la BD escala automáticamente para que soporte este tráfico, teniendo en cuenta que si escala se te cobrará por ello.

4.3 Diseño de la interfaz gráfica del usuario

En esta sección se van a mostrar parte de los mockups de la aplicación y explicar a qué funcionalidades corresponden Posteriormente se va a realizar una validación de los mockups con usuarios reales. Esto se hará para comprobar si son de calidad, visualmente atractivos o si hay fallos de usabilidad en los mismos. Son los usuarios los que se darán cuenta que algo no cuadra si su intención es realizar una acción y no es intuitivo hacerla

4.3.1 Mockups

Se van a incluir todas las imágenes de todas las pantallas necesarias para desarrollar la aplicación. Este diseño se ha realizado con el software Figma[14], un software de diseño gráfico profesional.

Los mockups que se presentan en este apartado se realizaron en colaboración con estudiantes de bellas artes para un proyecto que se desarrolló en la asignatura PIN del grado en Ingeniería Informática durante el cuatrimestre 4A. Uno de los objetivos de la aplicación era ofrecer una buena experiencia de usuario. Para ello, se perseguía diseñar un interfaz con una sintonía de colores adecuada, de tal manera que fuese gratificante y agradable de usar.

A continuación, se muestran los mockups correspondientes al subsistema de usuarios estándar.

Las figuras Figura 24Figura 25 Figura 26 muestran la página de buscar actividad, la página de actividades recomendadas y la página de inicio respectivamente.

[13] AWS: https://aws.amazon.com/es/

[14] Figma: https://www.figma.com

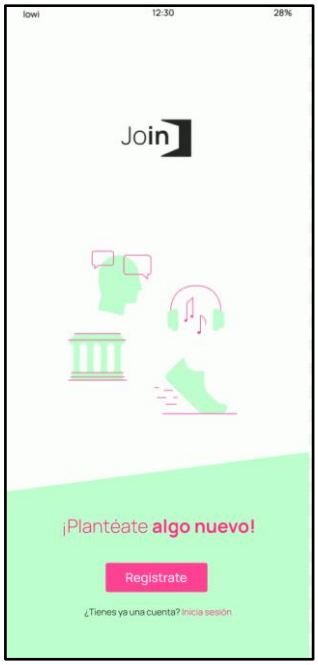


Figura 26 – Join, Mockup Página de Inicio

Figura 25 - Join, Mockup Actividades Recomendadas

Figura 24 - Join, Mockup Buscar Actividad

La funcionalidad *crear actividad* se lleva a cabo a través de una serie de pasos. Para cada uno de ellos se ha diseñado una página. La Figura 27 muestra el todo el paso en orden que el usuario debe completar para la creación de una actividad.

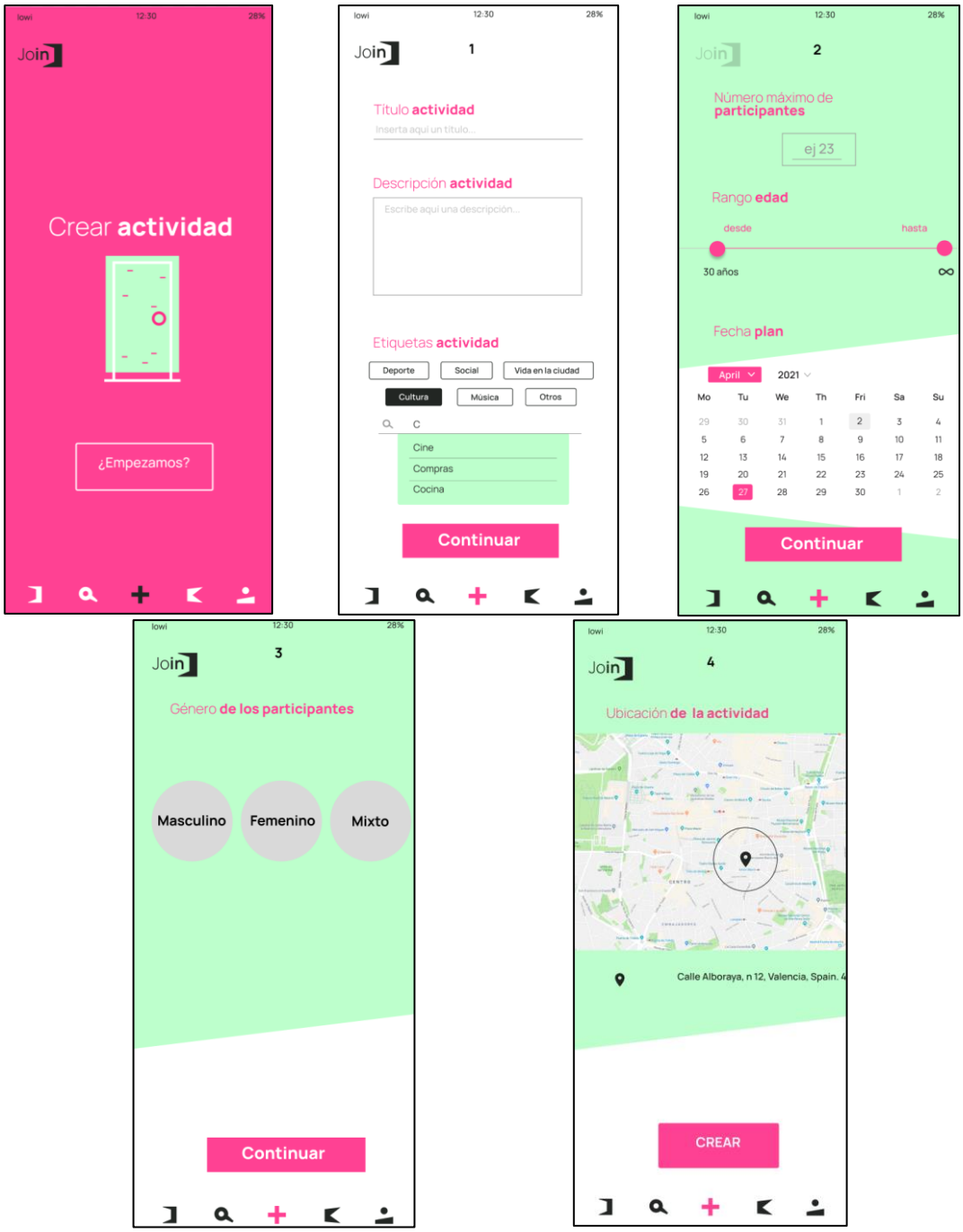


Figura 27- Join, Mockups de Proceso de Crear Actividad

Las figuras Figura 29 y Figura 28 muestran la pantalla de *Mis Actividades* y el chat, estas dos pantallas estarán disponibles desde la misma pestaña de navegación inferior.

Figura 29 - Join, Mockup de Mis Actividades

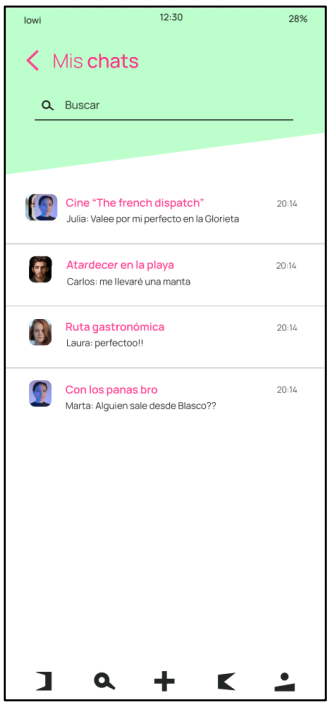


Figura 28 - Join, Mockup de Mis Chats

La funcionalidad *Registrar Usuario* se lleva a cabo también a través de una serie de pasos. Para cada uno de ellos se ha diseñado una página. La Figura 30 muestra el todo el paso en orden que el usuario debe completar para la creación de una actividad.

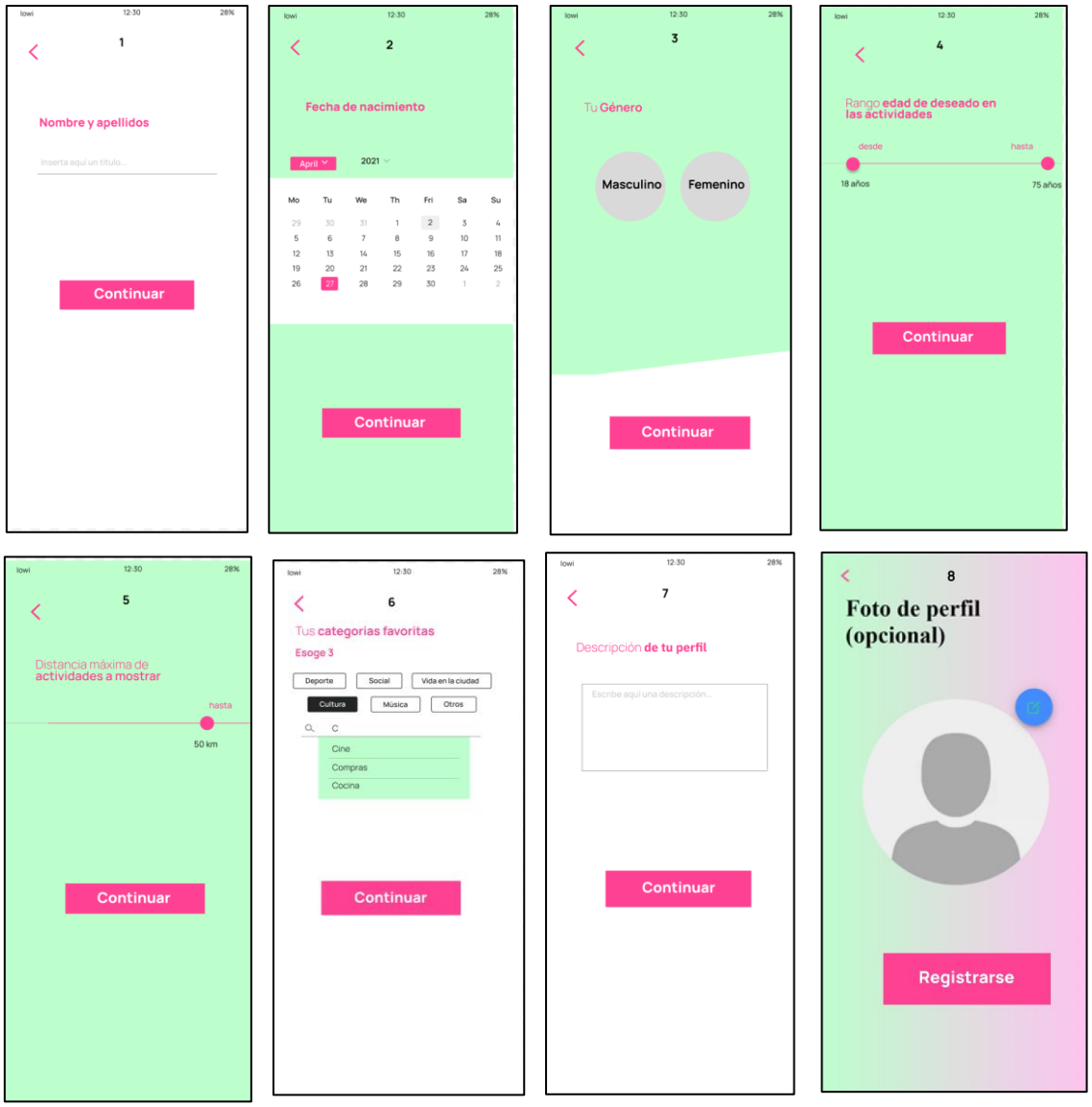


Figura 30 - Join, Moclups de proceso de registro de usuario estándar

A continuación, se muestran las pantallas del subsistema de entidades organizadoras

Al igual que el proceso de *Crear actividad* normal, la Figura 31 muestra los pasos que debe seguir la organización para crear una Actividad de pago.

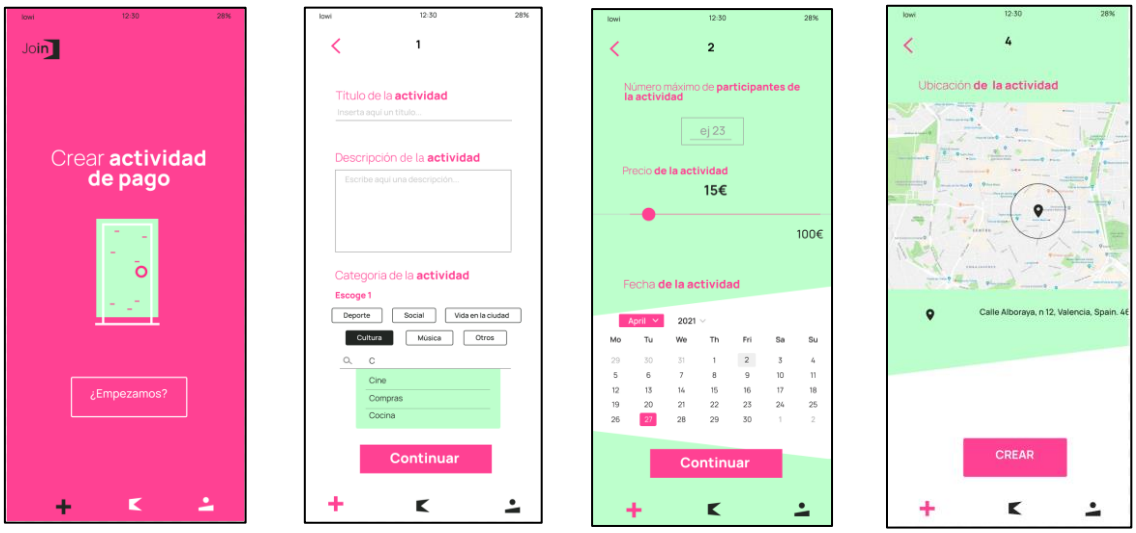


Figura 31 - Join, Mockups de Crear Actividad de pago

La Figura 32 muestra la pestaña desde donde las organizaciones gestionarán sus actividades.

Figura 32 - Join, Mockup de actividades creadas por la entidad

4.3.2 Validación de los mockups con el usuario

La validación se ha llevado a cabo con la realización de un experimento en el que han participado **4 usuarios.** Se han reclutado 4 estudiantes jóvenes que están acostumbrados a usar aplicaciones diariamente, pero que sin embargo no tienen conocimientos informáticos. Este perfil se corresponde con un usuario medio de la aplicación. A estos usuarios se les ha explicado brevemente el funcionamiento de la aplicación, y a continuación se les ha mostrado el diseño de las interfaces correspondientes a las funcionalidades principales: Crear Actividad, Buscar Actividad, Actividades Recomendadas, Mis Actividades y Registro de usuario

Los usuarios han visto las interfaces y se les ha pedido que se imaginen usando la funcionalidad que es visible en esa interfaz y que observen detalladamente a todos los elementos durante 15 minutos.

Una vez pasado el tiempo se les ha preguntado a los usuarios una serie de preguntas definidas en un formulario que se ha diseñado para la prueba (Ver Anexo 2: Formulario de validación de mockups).

Los resultados del experimento han permitido recoger información sobre aspectos de la aplicación que los usuarios consideraban que son mejorables. A continuación, se destacan algunos de los puntos a mejorar mencionados por el usuario.

Usuario 1:

- En algunas pantallas no hay flecha hacia atrás para volver a donde uno estaba
- Los símbolos de las pestañas de la barra de navegación de la aplicación son un poco abstractos, con formas extrañas.
- A la hora de elegir temática cuando un usuario se registra y crea una actividad, puede no llegar a ser intuitivo el funcionamiento de elegirla.

Usuario 2:

- Sería recomendable poner un género más aparte de masculino y femenino; “otro”, para aquellas personas que quizá no se identifiquen.
- Podría ser recomendable que la persona que crease la actividad pudiese añadir a gente con la que ya ha realizado actividades y con la cual tenga una amistad. Algo parecido a cuando creas un grupo de WhatsApp que el creador puede añadir a la gente antes de crearlo.
- Temática visual y colores demasiado llamativos para una App.

Usuario 3:

- Cuando creas una actividad habría que tener en cuenta que pudieses introducir una hora a la que se va a realizar la actividad, y que no solo se escriba en la descripción de esta.
- Cuando añades la ubicación de una actividad, estaría bien incluir una barra de búsqueda al estilo Google Maps.

Usuario 4:

- Algunas funcionalidades, como por ejemplo buscar grupo o crear grupo, están puestas con letras, quizá sería más idóneo usar iconos.

4.3.3 Análisis de los resultados de la validación

Tras analizar los resultados del experimento se ha identificado los siguientes puntos a mejorar en la aplicación:

- La elección del género de los participantes de la actividad que estás creando. únicamente se puede elegir entre género masculino y femenino. 2 usuarios han mencionado que sería interesante añadir otra opción para cualquier persona que no se identifique con uno de estos dos géneros.

- La hora de la actividad, en el mockup no estaba la hora que se llevaría a cabo la actividad, y esto es un requisito indispensable.

- Las flechas de navegación hacia atrás, algunas de ellas se han pasado por alto, es importante se coloquen todas para una correcta navegación de la aplicación.

- La elección de las temáticas de la actividad. Se ha mencionado que la elección no es muy intuitiva, esto se cambiará para una mejor experiencia de usuario.

- La barra de búsqueda de Google Maps, necesaria si quieres buscar una ubicación por calle o distrito.

- Por último, sustituir el texto de los botones de crear grupo o buscar grupo por iconos que se adapten mejor a la aplicación.

Estos resultados del experimento han sido claves para realizar cambios en las interfaces con el objetivo de obtener una aplicación fácil y agradable de usar como se tenía ideado.

Hay que mencionar también que tres de los cuatro usuarios han destacado lo atractiva y diferente que se veía la aplicación en cuanto a colores y diseño. Algo que sin duda ha reafirmado lo atrevido que es el diseño, pero a la vez el potencial que tiene, sin duda será algo que se mantendrá en el desarrollo.

5 Desarrollo de la solución propuesta

Tras realizar todo el trabajo previo de análisis, diseño y evaluación, finalmente ha llegado la sección donde se va a explicar cómo se ha desarrollado la aplicación.

Este apartado muestra el desarrollo del primer MVP de la aplicación. Este MVP ha sido el subsistema de usuario estándar, donde de momento no se ha creado el subsistema de entidades organizadoras debido a las horas estipuladas al TFG. Este MVP abarca todas las funcionalidades de la aplicación relacionadas con la comunicación Usuario–Usuario.

La sección se estructura en cuatro subsecciones dedicadas a: presentar las herramientas de soporte utilizadas en el desarrollo (ver sección 5.1), al desarrollo del *frontend* y *backend* (ver secciones 5.2 y 5.3) las cuales se ilustran con la funcionalidad de "*Crear Actividad*" especificada como requisito en la sección 3.1.1 y por último a presentar los retos o dificultades que se han encontrado y abordado en el desarrollo (ver 5.4).

5.1 Herramientas de soporte

Durante el desarrollo de un sistema software siempre es interesante utilizar herramientas que facilitan la organización y el propio desarrollo de este. A continuación, se introducen las herramientas que han sido utilizadas para el desarrollo de ***Join***.

Software de control de versiones.

Se ha utilizado el software de control de versiones GitHub[15], una de las herramientas de control de versiones más utilizada. Esto ha permitido llevar un control de todos los cambios, y dividir estas versiones por funcionalidades, donde cada funcionalidad era un *Commit*[16] dentro de la plataforma. La Figura 33 muestra el repositorio de archivos sobre el que se ha trabajado.

Repositorio de ***Join***:

[15] **GitHub:** https://github.com

[16] **Commit:** Captura una instantánea de los cambios preparados en ese momento del proyecto.

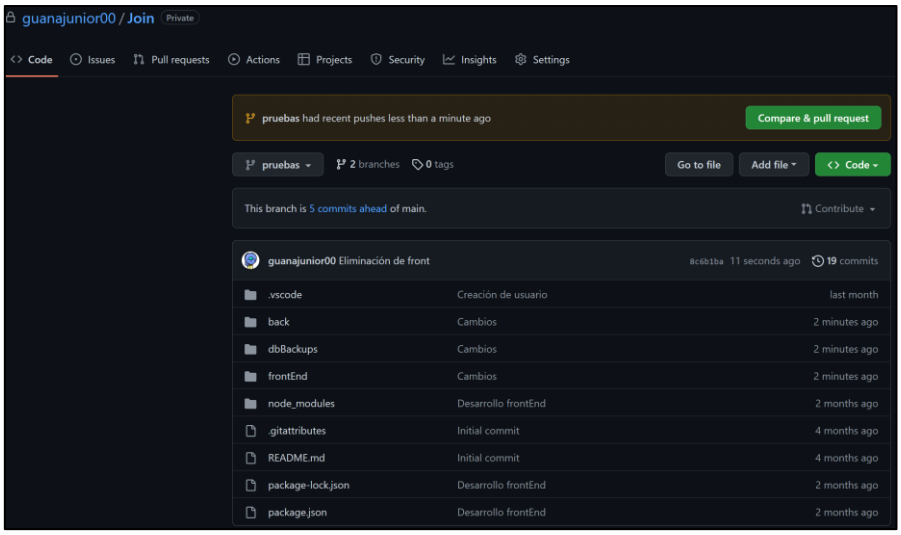


Figura 33 - Repositorio GitHub de Join

Un gestor de proyectos software.

Para la gestión del proyecto se ha utilizado la herramienta ClickUp[17]. Esta herramienta ha sido clave a la hora de desarrollar el proyecto. Dentro de ella son muchas las funciones que se pueden usar, pero la más útil para ***Join*** ha sido la de gestionar tareas y funcionalidades (UT's). Se pueden definir todas las tareas, con fechas límite, prioridades, módulos, tiempos registrados y muchas más opciones. Esto hace que se pueda tener un control sobre el trabajo realizado y el que queda por realizar, algo indispensable en ingeniería del software. En la Figura 34 parte del panel de creación de la funcionalidad *Crear Actividad.*

[17] **ClickUp:** https://clickup.com

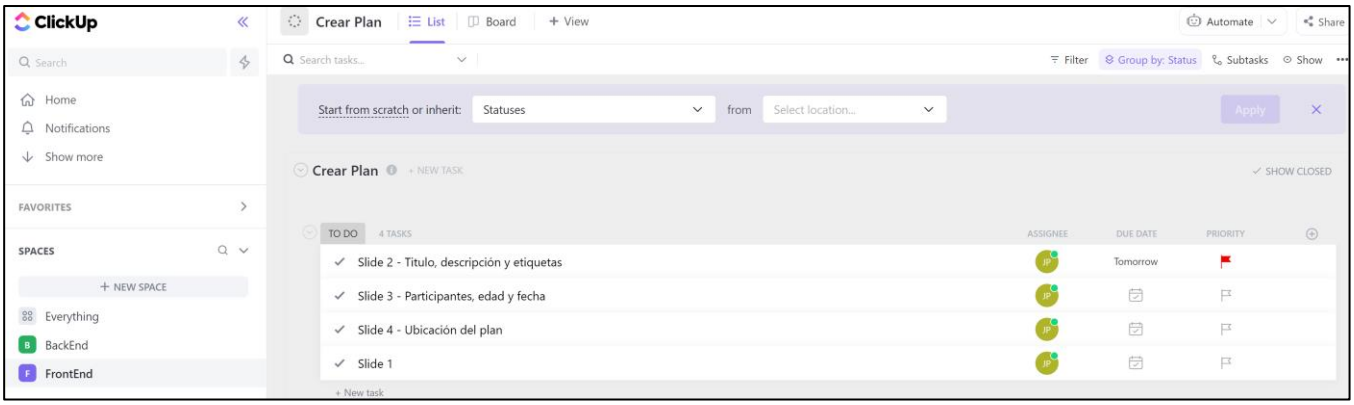


Figura 34 - DashBoard de Join en ClickUp

5.2 Desarrollo *FrontEnd*

5.2.1 Organización interna

La estructura del proyecto se ha realizado siguiendo las buenas prácticas descritas por Angular. El uso de estas buenas prácticas permite mantener una buena organización durante todo el ciclo de vida del proyecto, evitando estructuras complejas que dificulten entender la distribución de este. La Figura 35 muestra esta estructura.

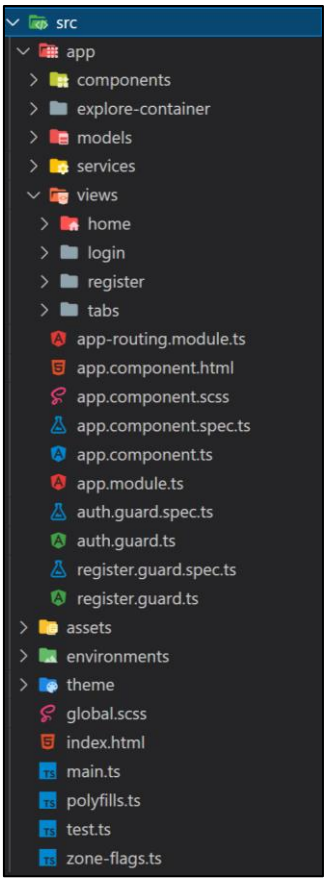


Figura 35 - Estructura de carpetas del FrontEnd

Como se puede ver en la figura, se tiene una carpeta *componentes*. Los componentes son una de las bases de Ionic y angular, se tratan de elementos reutilizables que pueden ser usados en cualquier parte de la aplicación. Como por ejemplo una barra de búsqueda, o una flecha de ir hacia atrás, en resumen, es todo aquel elemento que requiere de un trozo de código y que es reusable

La carpeta *models* agrupa los modelos de los objetos que son necesarios definir en el *frontend* para poder crear instancias de estos y enviarlas al *backend* como un DTO. Estos modelos pueden ser Actividad y Usuario, con sus respectivos atributos.

En *services* están los servicios que se crean para utilizar en distintas situaciones, en la mayoría de los casos se usarán para realizar las peticiones HTTP al *backend* para comunicarse con la API REST que se crea ahí.

En *views* se encuentra lo más importante, las pantallas principales de la aplicación, donde se tiene el login, registro y los tabs. Siendo estos últimos las pantallas de los botones del menú inferior de navegación de la aplicación.

Por último, están las carpetas *assets* y *environments*, donde en la primera se guardarán todos aquellos archivos e imágenes necesarios que no vayan ligadas a nada en la BD, y en la segunda se definirán las variables que se quiere que sean accesibles desde cualquier archivo Typescript del proyecto.

5.2.2 Desarrollo de las vistas

En esta sección se muestra un ejemplo de la creación de una de las vistas y funcionalidades, *Crear actividad*. Esta funcionalidad estaba formada por 4 vistas, donde en cada una se introduce información de la actividad, hasta llegar a la última donde se le da al botón de crear. Por esto mismo, se han de crear las subcarpetas *step1*, *step2*, *step3* y *step4* como se observa en la Figura 36.

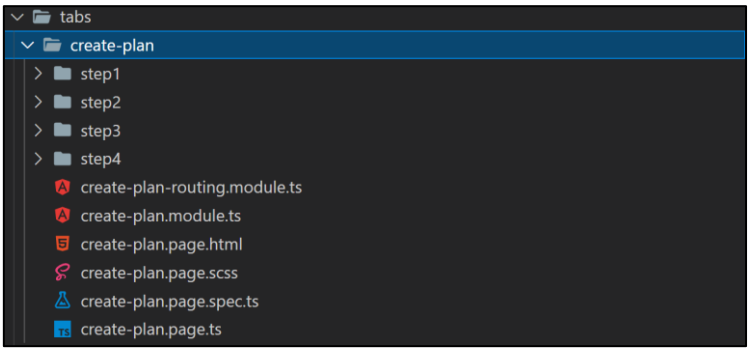


Figura 36 - Estructura de carpetas del proceso de Crear Actividad

Los archivos que hay dentro de cada sub vista o vista son los necesarios para poder crearla:

- Un archivo Typescript para definir rutas internas, de manera que, si esa vista tiene sub vistas, puedas navegar con los hijos.
- Un archivo Typescript que es un módulo, para importar componentes y elementos de otras partes de tu proyecto y de las librerías de Ionic y Angular.
- Un archivo HTML, para definir el esqueleto de la vista
- Un archivo SCSS, versión mejorada de CSS que permite dar estilo a nuestra vista
- Un archivo Typescript que permite controlar la vista definida en el HTML, dándole funcionalidad a dicha.
- Un archivo spec Typescript, que es para hacer y definir tests y pruebas sobre esa vista.

La Figura 37 muestra el archivo HTML5 correspondiente al segundo paso de “crear actividad”, donde el usuario tiene que introducir el número de participantes deseados, su rango de edad y la fecha de la actividad.

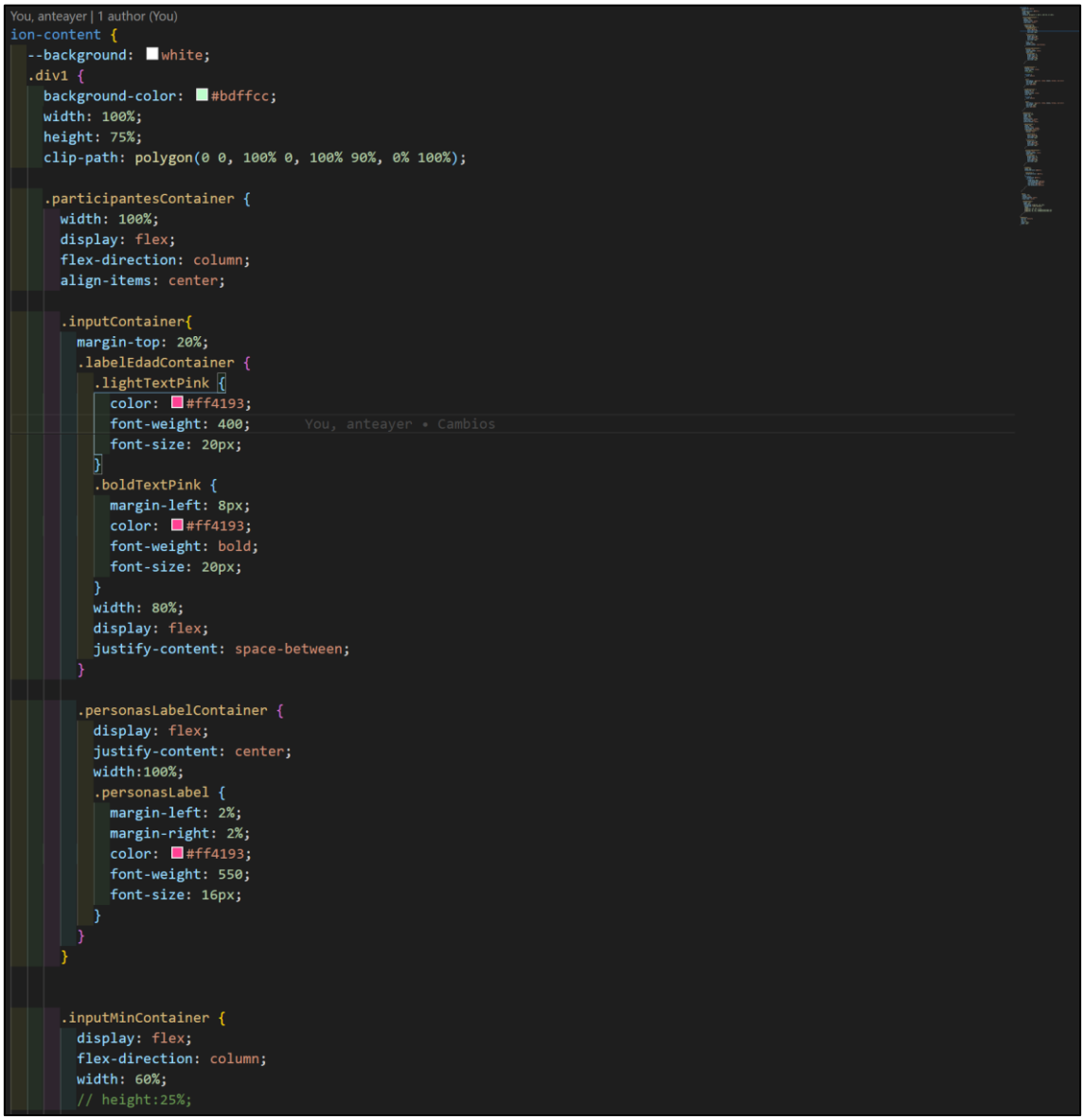
```
<ion-content>
  <div class="div1">
    <img
      (click)="goBackInNavegation()"
      class="backArrow"
      src="../../../../../assets/icon/leftArrow.png"
    />

    <div class="participantesContainer">

      <div class="inputContainer">
        <div class="labelEdadContainer">
          <p class="lightTextPink">Rango de</p>
          <p class="boldTextPink">Edad</p>
        </div>
        <div class="personasLabelContainer">
          <p class="personasLabel">{{ageRangeParticipants.lower}}</p>
          <p class="personasLabel">-</p>
          <p class="personasLabel">{{ageRangeParticipants.upper}}</p>
        </div>
        <ion-range
        [(ngModel)]="ageRangeParticipants"
          ticks="true"
          pin="false"
          mode="ios"
          dualKnobs="true"
          min="18"
          max="75"
          step="1"
          color="danger"
          ></ion-range>
      </div>
    </div>

    <div class="edadContainer">
      <div class="labelContainer">
        <div class="labelEdadContainer">
          <p class="lightTextPink">Número de</p>
          <p class="boldTextPink">Participantes</p>
        </div>
        <div class="personasLabelContainer">
          <p class="personasLabel">{{numberRangeParticipants.lower}}</p>
          <p class="personasLabel">-</p>
          <p class="personasLabel">{{numberRangeParticipants.upper}}</p>
        </div>
      </div>

      <div class="rangeEdad">
        <ion-item>
          <ion-range
          [(ngModel)]="numberRangeParticipants"
          ticks="true"
          pin="false"
          mode="ios"
          dualKnobs="true"
          min="1"
          max="25"
          step="1"
          color="danger"
          ></ion-range>
        </ion-item>
      </div>
    </div>
  </div>

  <div class="div3">      You, anteayer • Cambios …
    <ion-button
    [disabled]=""
    (click)="navigateForward()"
    expand="block"
    fill="solid"
    shape="round"
    ngClass="buttonContinue"
    >Continuar</ion-button
    >

    <ion-datetime
    [(ngModel)]="date"
    min="{{minYear}}"
    max="{{maxYear}}"
    minuteValues="0,5,10,15,20,25,30,35,40,45,50,55"
    hour-cycle="h23"
    first-day-of-week="1"
    size="cover"
    mode="md"
    presentation="date-time"
    (ionChange)="dataTimeChanged($event)"
  ></ion-datetime>
  </div>
</ion-content>
```

Figura 37 - Ejemplo de archivo HTML creado para el Step 2 de CrearActividad

En este archivo HTML se utilizan etiquetas nativas de Ionic y nomenclaturas personalizadas de este *framework*. Se introducen muchos atributos en las etiquetas HTML que en la versión oficial no existen, pero que sin embargo en Ionic se pueden usar, estos facilitan y flexibilizan el desarrollo. Esto se ha utilizado a lo largo del proyecto para realizar todas las pantallas que había que crear.

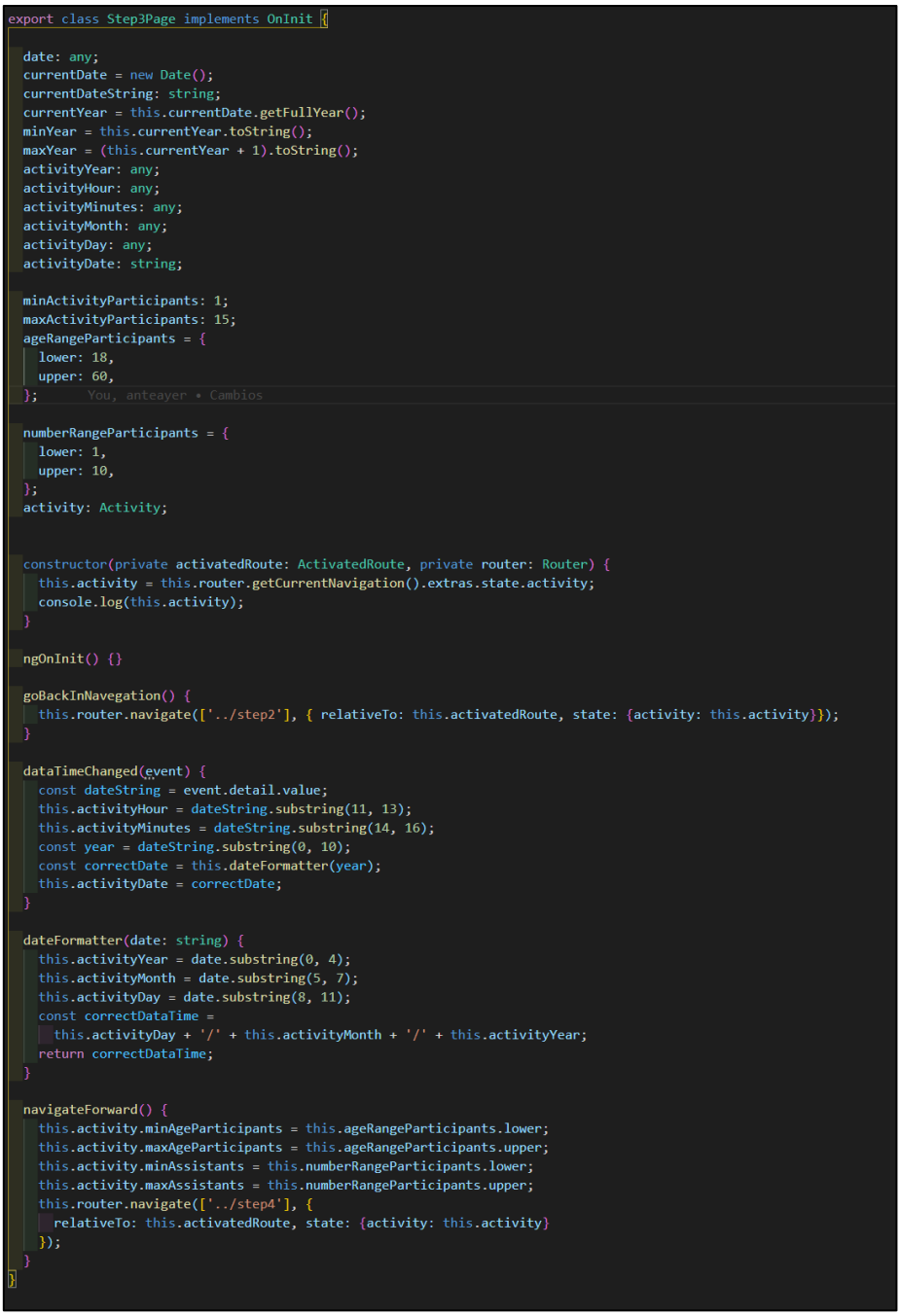
Es importante mencionar que en Ionic se puede introducir lógica en el HTML, de hecho, es recomendable, ya que esto hace que nuestro archivo Typescritp, que se verá más adelante, se quede mucho más limpio y la refactorización sea nula o mínima.

La Figura 38 muestra una captura del archivo SCSS que da estilo a los elementos del archivo HMTL mostrado anteriormente. La captura introducida es tan solo un fragmento del archivo completo debido a la longitud extensa del archivo.



```
You, anteayer | 1 author (You)
ion-content {
  --background: white;
  .div1 {
    background-color: #bdffcc;
    width: 100%;
    height: 75%;
    clip-path: polygon(0 0, 100% 0, 100% 90%, 0% 100%);

    .participantesContainer {
      width: 100%;
      display: flex;
      flex-direction: column;
      align-items: center;

      .inputContainer{
        margin-top: 20%;
        .labelEdadContainer {
          .lightTextPink {
            color: #ff4193;
            font-weight: 400;       You, anteayer • Cambios
            font-size: 20px;
          }
          .boldTextPink {
            margin-left: 8px;
            color: #ff4193;
            font-weight: bold;
            font-size: 20px;
          }
          width: 80%;
          display: flex;
          justify-content: space-between;
        }

        .personasLabelContainer {
          display: flex;
          justify-content: center;
          width:100%;
          .personasLabel {
            margin-left: 2%;
            margin-right: 2%;
            color: #ff4193;
            font-weight: 550;
            font-size: 16px;
          }
        }
      }

      .inputMinContainer {
        display: flex;
        flex-direction: column;
        width: 60%;
        // height:25%;
```

Figura 38 - Fragmento del archivo SCSS del Step 2 de Crear Actividad



```
export class Step3Page implements OnInit {

  date: any;
  currentDate = new Date();
  currentDateString: string;
  currentYear = this.currentDate.getFullYear();
  minYear = this.currentYear.toString();
  maxYear = (this.currentYear + 1).toString();
  activityYear: any;
  activityHour: any;
  activityMinutes: any;
  activityMonth: any;
  activityDay: any;
  activityDate: string;

  minActivityParticipants: 1;
  maxActivityParticipants: 15;
  ageRangeParticipants = {
    lower: 18,
    upper: 60,
  };       You, anteayer • Cambios

  numberRangeParticipants = {
    lower: 1,
    upper: 10,
  };
  activity: Activity;


  constructor(private activatedRoute: ActivatedRoute, private router: Router) {
    this.activity = this.router.getCurrentNavigation().extras.state.activity;
    console.log(this.activity);
  }

  ngOnInit() {}

  goBackInNavegation() {
    this.router.navigate(['../step2'], { relativeTo: this.activatedRoute, state: {activity: this.activity}});
  }

  dataTimeChanged(event) {
    const dateString = event.detail.value;
    this.activityHour = dateString.substring(11, 13);
    this.activityMinutes = dateString.substring(14, 16);
    const year = dateString.substring(0, 10);
    const correctDate = this.dateFormatter(year);
    this.activityDate = correctDate;
  }

  dateFormatter(date: string) {
    this.activityYear = date.substring(0, 4);
    this.activityMonth = date.substring(5, 7);
    this.activityDay = date.substring(8, 11);
    const correctDataTime =
      this.activityDay + '/' + this.activityMonth + '/' + this.activityYear;
    return correctDataTime;
  }

  navigateForward() {
    this.activity.minAgeParticipants = this.ageRangeParticipants.lower;
    this.activity.maxAgeParticipants = this.ageRangeParticipants.upper;
    this.activity.minAssistants = this.numberRangeParticipants.lower;
    this.activity.maxAssistants = this.numberRangeParticipants.upper;
    this.router.navigate(['../step4'], {
      relativeTo: this.activatedRoute, state: {activity: this.activity}
    });
  }
}
```

Figura 39 - Archivo .Ts del Step 2 de Crear Actividad

La Figura 39 muestra el archivo Typescript que define todas las funciones necesarias para que la pantalla tenga el comportamiento que se desea. Como en esta vista hay que elegir la fecha de la Actividad, hay ciertas funciones que trabajan con dicha fecha y la formatean de manera correcta.

5.2.2.1 Secuencia de desarrollo

Han sido muchas las vistas y pantallas que se han tenido que desarrollar en ***Join***. La creación de estas se ha llevado a cabo en un orden lógico, de manera que las más prioritarias o aquellas en las que se definiese información que fuese necesaria se desarrollaron antes que las demás.

El orden fue el siguiente:

Registro de usuario - Necesario para poder crear una actividad

Inicio de sesión

Crear actividad - La base de la aplicación gira en torno a esta funcionalidad

Buscar actividad - Necesario para poder unirte a cualquier actividad

Mis actividades - Necesario para poder gestionar las actividades que has creado y a las que te has unido. Incluyendo el apartado de crear y buscar grupo , el cual esta englobado en esta pestaña de navegación.

Mi perfil

Planes Recomendados

				
Crear actividad	Buscar actividad	Mis actividades	Mi perfil	Planes recomendados

5.3 Desarrollo del *backend*

En este apartado se va a presentar la estructura del *backend* y el proceso de desarrollo de este.

El *backend* se ha estructurado utilizando las buenas prácticas de SpringBoot según su documentación oficial [13].

La estructura definida para todo el proyecto en la parte del servidor se puede ver en la Figura 40. Se han definido las siguientes carpetas:

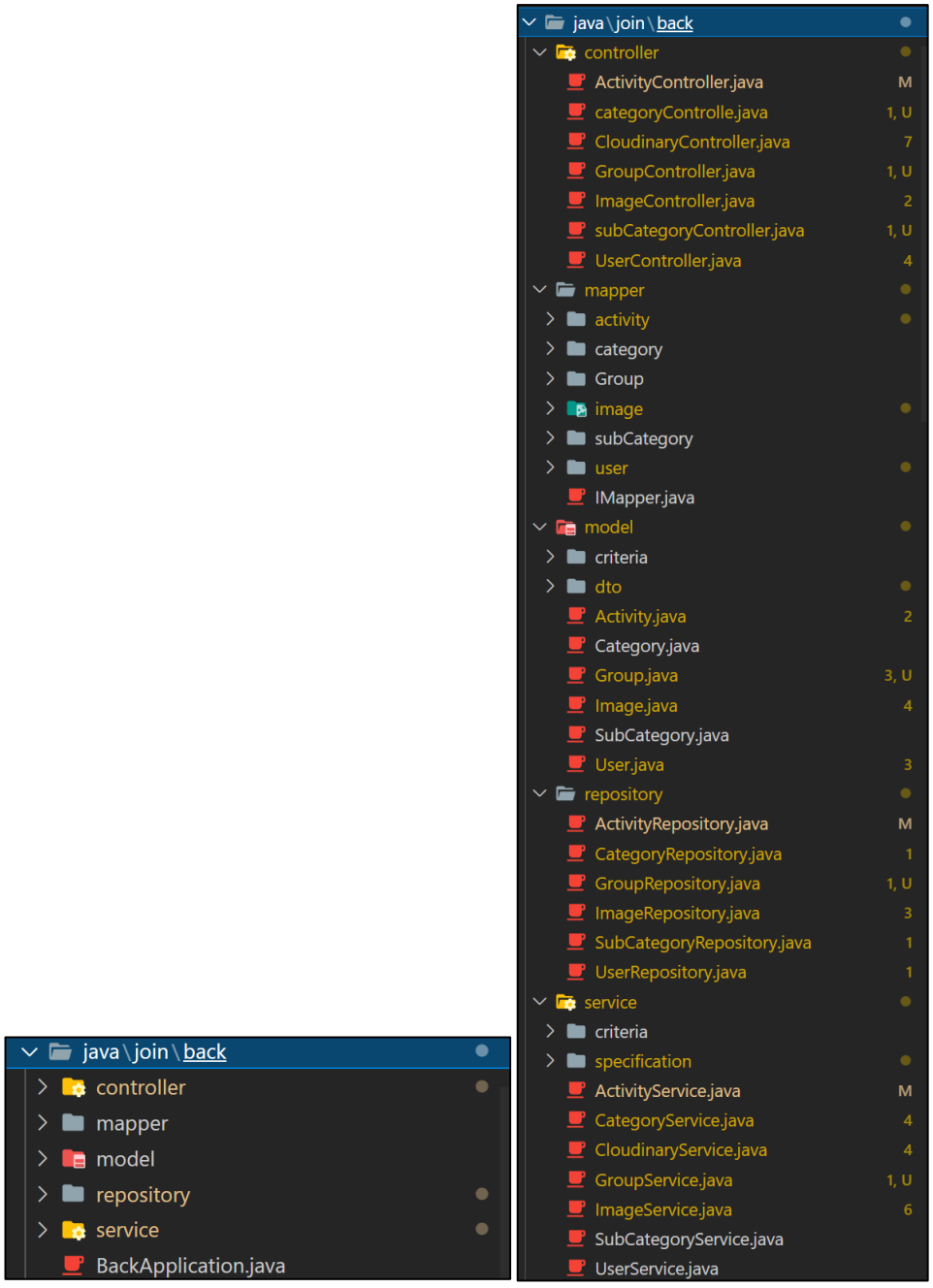


Figura 40 - Estructura de carpetas del Backend de Join

- Controladores, donde se define las API REST para recibir las peticiones

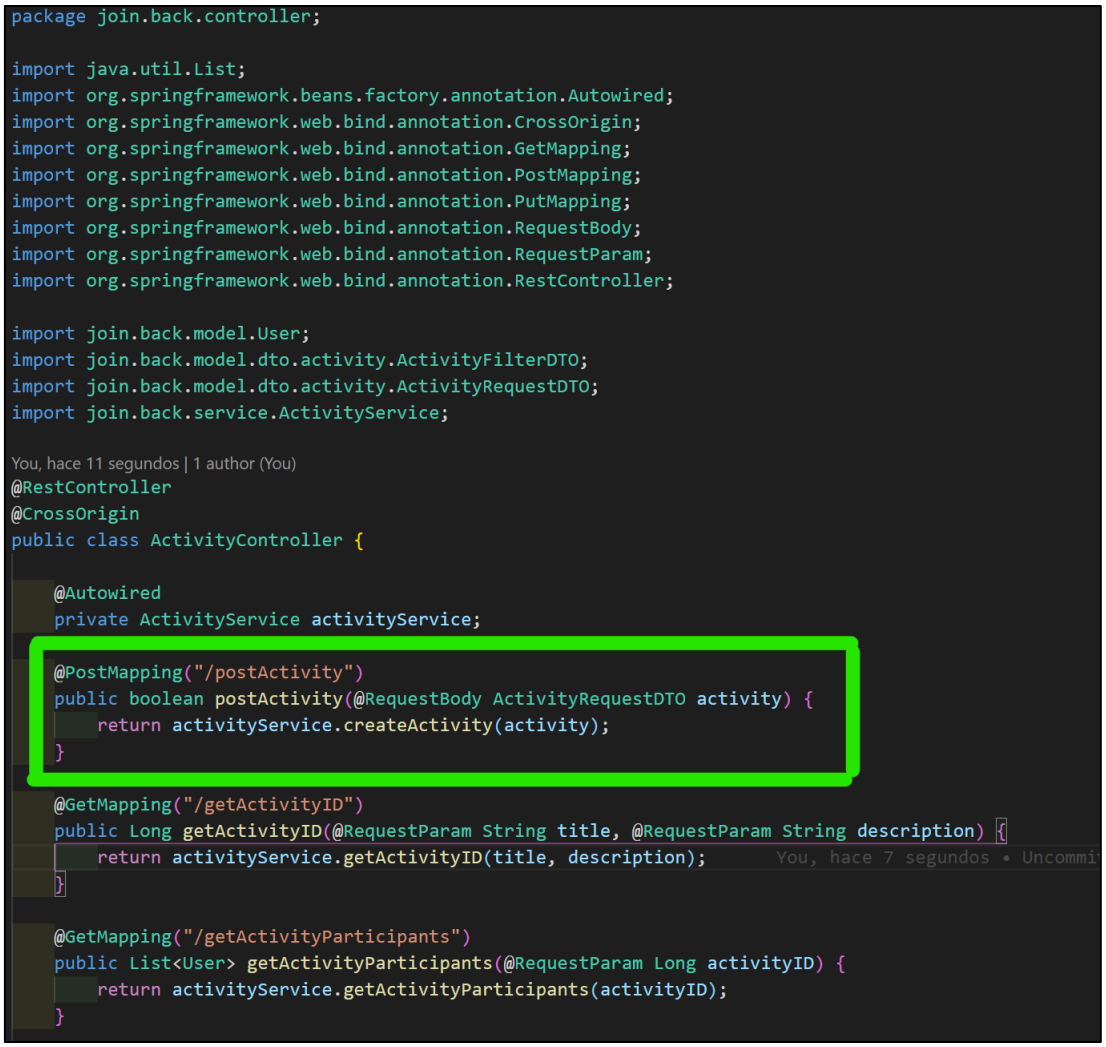
- Servicios, utilizados para realizar ciertas funciones, como por ejemplo utilizar los "Repositories" para llamar a la BD y usar los Mappers.
- Repositorios, son las interfaces que extienden de JPA (Java Persistence API) y tiene una serie de funciones que realizan funciones CRUD sobre la BD
- Mapeadores, transforma los DTO`s que se reciben del *frontend*, en modelos/entidades de la BD y viceversa.
- Modelos, aquí se definen los DTO`s y los Modelos y entidades de la BD.

El *backend* se ha desarrollado al mismo tiempo que el *frontend*, esto significa que cuando una funcionalidad era necesaria en la parte del cliente, como por ejemplo obtener todas las actividades, se desarrollaba en el *backend* esa funcionalidad y luego se continuaba con la parte visual. Esta misma metodología de trabajo se ha aplicado para todas las funcionalidades que han sido necesarias.

5.3.1 Proceso de desarrollo de un caso de uso / funcionalidad:

En esta sección se muestra cómo se ha implementado el caso de uso "Crear Actividad". Se van a explicar todos los pasos necesarios para completar este caso de uso. El proceso de implementación del resto de casos de uso es muy similar.

Paso 1: Creación del método para recibir la petición en las API / Controlador correspondiente



```java
package join.back.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import join.back.model.User;
import join.back.model.dto.activity.ActivityFilterDTO;
import join.back.model.dto.activity.ActivityRequestDTO;
import join.back.service.ActivityService;

You, hace 11 segundos | 1 author (You)
@RestController
@CrossOrigin
public class ActivityController {

    @Autowired
    private ActivityService activityService;

    @PostMapping("/postActivity")
    public boolean postActivity(@RequestBody ActivityRequestDTO activity) {
        return activityService.createActivity(activity);
    }

    @GetMapping("/getActivityID")
    public Long getActivityID(@RequestParam String title, @RequestParam String description) {
        return activityService.getActivityID(title, description);       You, hace 7 segundos • Uncommi
    }

    @GetMapping("/getActivityParticipants")
    public List<User> getActivityParticipants(@RequestParam Long activityID) {
        return activityService.getActivityParticipants(activityID);
    }
```

Figura 41- API Rest y metodo creado para recibir peticiónes de Crear Actividad

Spring utiliza anotaciones con @ para definir ciertas propiedades. En el método *postActivity* subrayado de la Figura 41 se está definiendo que se va a recibir una petición POST y de parámetro se recibe un objeto Actividad DTO, que debe tener las mismas propiedades y atributos que las definidas en la clase *ActivityRequestDTO*. Spring se encargará de adaptar ese objeto del *frontend* y transformarlo a nuestro DTO, siempre y cuando se cumpla lo mencionado anteriormente sobre los atributos.

En un controlador no puede haber nunca funcionalidades puras, de esto se encarga el servicio, es por eso por lo que se usará un método del servicio que se encargue de todo.

Lo que hace dentro del método *postActivity* con la línea ***activityService.createActivity(activity)*** es llamar a un método que está en el servicio específico de Actividad. Este se creará de tal manera que devuelve un booleano si se ha realizado correctamente.

Paso 2: Creación del método del servicio:

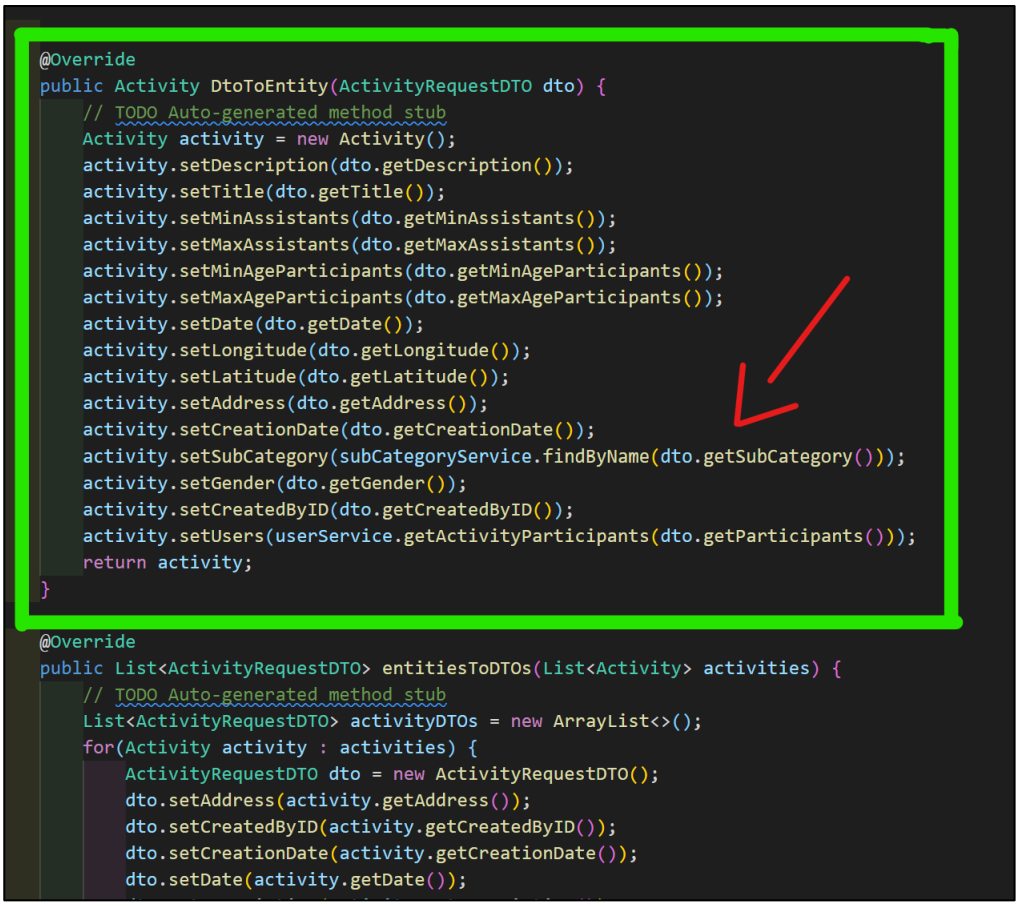
```java
@Service
public class ActivityService {
    @Autowired
    private ActivityRepository activityRepository;

    @Autowired
    private ActivityRequestMapper activityRequestMapper;

    @Autowired
    private ActivityCriteriaService activityCriteriaService;

    @Autowired
    private ActivitySpecificationService activitySpecificationService;

    public boolean createActivity(ActivityRequestDTO activityRequestDTO){
        Activity activity = activityRequestMapper.DtoToEntity(activityRequestDTO);
        try{
            activityRepository.save(activity);
        }
        catch(IllegalArgumentException e){
            return false;
        }

        return true;
    }

    public Long getActivityID(String title, String description){
        Activity activity = new Activity();
        activity = activityRepository.findByTitleAndDescription(title, description);
        return activity.getId();
    }

    public List<User> getActivityParticipants(Long activityID){
        List<User> participants = new ArrayList<>();
        participants = activityRepository.getById(activityID).getUsers();
        return participants;
    }
```

Figura 42 - Método del servicio de Crear Actividad

En el método subrayado *createActivity* de la Figura 42 se necesita recibir la actividad para poder publicarla, por lo tanto, lo que se recibirá será lo que llegue del controlador; una ActivityRequestDTO. Aquí surge el primer problema, JPA y nuestra BD solo entienden de los modelos/entidades que se han definido para ellos, situados en la carpeta Models. Por lo tanto, de alguna manera se tienen que transformar ese DTO en una Entidad, y para ello se usan los "Mappers" o mapeadores que se definen. Dentro de estos mappers, habrá uno por modelo/entidad que se tenga definida en la BD. Dentro de ellos se ha de definir los siguientes métodos:

- EntityToDTO
- DTOToEntity
- EntitiesToDTOS
- DTOStoEntieties

Paso 2.1: Uso los métodos de los mappers



```java
@Override
public Activity DtoToEntity(ActivityRequestDTO dto) {
    // TODO Auto-generated method stub
    Activity activity = new Activity();
    activity.setDescription(dto.getDescription());
    activity.setTitle(dto.getTitle());
    activity.setMinAssistants(dto.getMinAssistants());
    activity.setMaxAssistants(dto.getMaxAssistants());
    activity.setMinAgeParticipants(dto.getMinAgeParticipants());
    activity.setMaxAgeParticipants(dto.getMaxAgeParticipants());
    activity.setDate(dto.getDate());
    activity.setLongitude(dto.getLongitude());
    activity.setLatitude(dto.getLatitude());
    activity.setAddress(dto.getAddress());
    activity.setCreationDate(dto.getCreationDate());
    activity.setSubCategory(subCategoryService.findByName(dto.getSubCategory()));
    activity.setGender(dto.getGender());
    activity.setCreatedByID(dto.getCreatedByID());
    activity.setUsers(userService.getActivityParticipants(dto.getParticipants()));
    return activity;
}

@Override
public List<ActivityRequestDTO> entitiesToDTOs(List<Activity> activities) {
    // TODO Auto-generated method stub
    List<ActivityRequestDTO> activityDTOs = new ArrayList<>();
    for(Activity activity : activities) {
        ActivityRequestDTO dto = new ActivityRequestDTO();
        dto.setAddress(activity.getAddress());
        dto.setCreatedByID(activity.getCreatedByID());
        dto.setCreationDate(activity.getCreationDate());
        dto.setDate(activity.getDate());
```

Figura 43- Método del Mapper de Actividad DtoToEntity

En este caso se necesita pasar de un DTO a una Entity, por lo tanto, se debe mapear todos sus atributos a una actividad y devolvérsela al servicio para que la utilice (ver Figura 43). Es posible que para mappear algún atributo que sea una propia Entity el mismo, sea necesario usar algún servicio para mappearlo y obtenerlo, ya que en las DTOS solo se pueden definir atributos de tipos primitivos. Como se puede observar en la flecha roja marcada en la Figura 43. Entonces se puede dar el caso que se obtenga un String del nombre de por ejemplo la categoría, pero se necesita el objeto categoría correspondiente a ese nombre, por lo tanto, se usa un servicio para buscar y obtener esa categoría de la BD que tiene el nombre del String que recibimos en el DTO.

Una vez mapeado, se devuelve la actividad (Entity) creada, con los atributos correctos.

Paso 3: Dentro del servicio, interactuar con los "repositories" y la BD

Una vez el servicio obtiene de vuelta la actividad que se va a publicar ya en forma de Entity, se debe guardarla en la BD. Para ello se utilizan los "repositories" que se han definido en la estructura del *backend*. Estos repositorios se tratan de interfaces que extienden de JPA y dan acceso a muchos métodos que permiten hacer operaciones CRUD sobre la BD, incluyendo una nomenclatura para definir nuestros propios métodos para consultar y acceder a ella.

Si se observa de nuevo la Figura 43, usando una instancia de ActivityRepository se llama al método "save", ya definido por JPA, para guardarlo en la base de datos. Si la Actividad a guardar

es por algún motivo errónea o nula, saltará una excepción, de tal manera que dependiendo de si se guarda correctamente o no se devolverá un true o un false respectivamente. Este booleano será el que se reciba en el *frontend* para confirmar si se ha realizado correctamente o ha saltado algún tipo de excepción.

5.4 Problemas y dificultades

Durante el desarrollo del proyecto han surgido varios problemas que han dificultado y ralentizado el mismo. Han surgido problemas técnicos tanto en el *frontend* como en el *backend*.

5.4.1 Problemas en el *frontend*

Los problemas principales encontrados en esta sección han sido los siguientes:

- Implementar exactamente el diseño que se estipuló. Construir una interfaz gráfica de usuario con HTML y CSS para dejarla exactamente igual que un diseño hecho con figma con tantos elementos ha sido bastante complejo de realizar. Con el tiempo y experiencia poco a poco se han ido refinando las vistas.
- Asincronías en algunas funcionalidades usando TypeScript, acabando en muchas ocasiones en el comúnmente conocido "callback Hell". Muchas de ellas han sido solucionadas con el uso del patrón Aync /await de Typescript.
- Integración de los mapas de Google. Aunque en principio puede parecer sencillo, se presentaron bastantes problemas a la hora de hacerlos funcionar correctamente. Al ser un componente reusable, una vez se consiguió configurar, era posible utilizarlo en toda la aplicación.
- Ciertos problemas integrando la aplicación con capacitor para las plataformas IOS e Android. Específicamente en IOS, donde para añadir ciertas propiedades que conectaban con este sistema operativo era necesario hacer uso de un Mac que tuviese el programa XCode. Algo que al principio no fue posible.

5.4.2 Problemas en el *backend*

Problemas principales encontrados durante el desarrollo *backend*;

- Curva de aprendizaje de SpringBoot bastante lenta. No ha sido sencillo utilizar el framework de manera fluida y rápida, sobre todo al comienzo. Es complicado hacerse a una tecnología si no se ha utilizado previamente ya que requiere un gran tiempo de adaptación.
- Restricciones en SpringBoot. Ha sido complejo seguir algunas restricciones que impone SpringBoot y que se han de cumplir, y más si se está acostumbrado a una metodología de programación más ágil y rápida. Esto ha ocasionado la mayoría de los errores del *backend*.

6 Pruebas

En esta sección se van a explicar cómo se ha probado la aplicación de ***Join***. Se mostrarán 3 distintos tipos de pruebas. Primero se explicarán las pruebas unitarias realizadas (6.1), después las pruebas de integración (6.2), y por último las pruebas con usuarios (6.3).

6.1 Pruebas unitarias

En las pruebas unitarias se ha decidido usar una tecnología de *testing* de Java. Se ha usado JUnit5[18], con la cual es posible escribir pruebas unitarias utilizando una cierta nomenclatura que comprueba la salida de la prueba.

6.1.1 Diseño de las pruebas

Como ejemplo para mostrar en la memoria se han cogido las pruebas realizadas para 2 funcionalidades un cierto servicio. Si se recuerda lo explicado en el apartado 5.3, los servicios eran las clases que englobaban toda la funcionalidad del back. Los métodos pertenecen al servicio de Actividad, y los dos métodos elegidos son los siguientes:

[18] JUnit5: https://junit.org/junit5/

```java
@Test
void testCreateActivity() {

    //creamos una actividad de prueba que vamos a subir a la BD
    ActivityRequestDTO activityDTO = new ActivityRequestDTO();
    activityDTO.setCreationDate("10/06/2022");
    activityDTO.setAddress("Blasco Ibañez 73");
    activityDTO.setDate("15/07/2022");
    activityDTO.setDescription("Decripción de prueba");
    activityDTO.setGender("male");
    activityDTO.setLatitude(-0.70107);
    activityDTO.setLongitude(38.26218);
    activityDTO.setTitle("Titulo de prueba");
    activityDTO.setMaxAgeParticipants(25);
    activityDTO.setMinAgeParticipants(18);
    activityDTO.setMinAssistants(1);
    activityDTO.setMaxAssistants(5);
    activityDTO.setCreatedByID(1L);
    activityDTO.setSubCategory("Voleyball");

    //aqui comprobamos si se ha creado correctamente la actividad
    // el método create Activity devuelve true si se ha creado correctamente.
    assertTrue(activityService.createActivity(activityDTO));
}
```

Figura 44 - Prueba unitaria del método Crear Actividad

En la Figura 44 se muestra el diseño del código elaborado para probar la funcionalidad. Lo que se ha ideado para comprobar si se crea una actividad correctamente, es crear una ActividadDTO, simulando el objeto que se envía del *frontend*, y luego se ha llamado al método correspondiente, que como vimos en la sección 5.3.1, devuelve true, si se ha creado correctamente el método.

```java
@Test
void testGetActivityParticipants() {

    //creamos una lista de participantes y añadimos un usuario a parte del creador
    // esta lista de participantes se la asginaremos ala actividadDTO, que luego mapeará todos los atributos
    List<Long> participantsDTO = new ArrayList<>();
    participantsDTO.add(1L);
    participantsDTO.add(2L);

    //creamos la actividadDTO y le damos unos valores ya que tendremos que subir la actividad
    ActivityRequestDTO activityDTO = new ActivityRequestDTO();
    activityDTO.setCreationDate("10/06/2022");
    activityDTO.setAddress("Blasco Ibañez 73");
    activityDTO.setDate("15/07/2022");
    activityDTO.setDescription("Esta es la descripción escrita para la segunda prueba");
    activityDTO.setGender("male");
    activityDTO.setLatitude(-0.70107);
    activityDTO.setLongitude(38.26218);
    activityDTO.setTitle("Esta es el titulo escrito para la segunda prueba");
    activityDTO.setMaxAgeParticipants(25);
    activityDTO.setMinAgeParticipants(18);
    activityDTO.setMinAssistants(1);
    activityDTO.setMaxAssistants(5);
    activityDTO.setCreatedByID(1L);
    activityDTO.setSubCategory("Voleyball");
    activityDTO.setImageID(null);
    activityDTO.setParticipants(participantsDTO);

    //creamos la actividad, para asegurarnos de que se sube a la BD
    assertTrue(activityService.createActivity(activityDTO));

    //cogemos esa actividad creada de la BD
    Activity activity = activityRepository.findByTitleAndDescription(activityDTO.getTitle(), activityDTO.getDescription());

    //Una tenemos la actividad de la BD, usamos el método getActivityParticipants, que nos devolvera una lista de Users
    // si este devuelve una lista con los usuarios cuyos ID's son 1 y 2 , significa que funciona correctamente el método a probar
    // ya que estos son los ID's de los participantes que hemos añadido manualmente de prueba
    List<User> participants = new ArrayList<>();
    participants = activityService.getActivityParticipants(activity.getId());

    //Una vez tenemos la lista de Users, tenemos que obtener uan lista con sus ID's,
    // esto se hace para poder comprobarlos con los que hemos introducido manualmente
    List<Long> participantsIDS = new ArrayList<>();
    for(User participant: participants){...}

    // hacemos la comprobación de que todos los participantes que hemos añadido, se han retornado correctamente y coinciden.
    assertTrue(participants.containsAll(participantsDTO));
}
```

Figura 45 - Prueba unitaria del método getActivityParticipants

El código de la Figura 45 es un tanto más complejo. Primero de todo se crea una ActivityDTO como en el anterior método, a esta se le asignan unos participantes. Si la actividad se sube a la BD, y se usa el método getActivityParticipants, donde se le pasa el ID de la actividad deseada, en nuestro caso la que se acaba de crear. Este método debería entonces retornar los mismo ID’s que los introducidos en el objeto DTO, esto es lo que se prueba con la última línea del método. Se usa la cláusula *assertTrue* para comprobar si los participantes de la actividad retornada son los mismo que los que se han añadido a esa misma actividad, donde consecuentemente; funciona de manera correcta.

6.1.2 Resultados de las pruebas

En JUnit, cada prueba es ejecutada como un método. Si se cumplen las pruebas, al ejecutar ese método aparece un *check ok* confirmando que las restricciones escritas usando los *asserts* se han cumplido, y, por ende, se ha pasado.

Los resultados de los dos ejemplos mostrados anteriormente son los siguientes que se muestran en la Figura 46.

Figura 46 - Resultado de ejemplos de pruebas unitarias

6.2 Pruebas de integración

En este apartado se va a mostrar un ejemplo de una prueba de integración realizada, esta se realiza mediante el uso de un software de pruebas automático llamado Cypress[19]. Con este software, diseñas un script que probará automáticamente desde el *frontend*, un proceso completo. Posteriormente se mostrarán los resultados del script automático ejecutado sobre la aplicación.

En este caso se va a usar de nuevo el caso de uso más importante, *Crear Actividad.* En la prueba se va a ejecutar automáticamente una serie de pasos, de manera que se cree una actividad automáticamente, y compruebe si en el resto de las funcionalidades, como puede ser buscar Actividad, aparece está nueva actividad creada. Esto comprobará si estás dos funcionalidades de la aplicación están bien integradas juntas.

6.2.1 Diseño de las pruebas

[19] Cypress: https://www.cypress.io

```
describe('CREAR ACTIVIDAD', () => {
    let date = new Date()
    let title= `test ${date.toUTCString()}`
    let descripciçon= `descripción de prueba`
    before(()=>{
    cy.get('#tab-button-create-plan').click();
    })
    after(()=>{
        cy.visit('/views/tabs/search-plan')
        cy.get('ion-searchbar').type(title)
        cy.get('ion-searchbar').trigger('search')
        cy.get('ion-card').then((card)=>{
            expect(card.length).to.equal(1)
        })
    })
    it('should create a plan',()=>{
        let place = 'blasco ibañez valencia'
        let description = 'plan de testeo en marcha'
        cy.get('[formControlName=title]').type(title)
        cy.get('[formControlName=description]').type(description)
        cy.contains('Voleyball').click()
        cy.get('.buttonContinue').click()
        cy.get('[formControlName=when]').click()
        cy.contains('Done').click()
        cy.get('.buttonContinue').click()
        cy.get('.maleButton').click()
        cy.get('.buttonContinue').click()
        cy.get('ion-searchbar').type(place)
        cy.wait(5000)
        cy.get('.buttonCreateActivity').click()
        cy.wait(5000)
    })
})
```

Figura 47 - Test integración Crear Actividad - Buscar Actividad

En la Figura 47 se ve el script automático que se va a ejecutar. En él, se pasa por todos los pasos necesarios hasta finalmente crear la actividad. Al principio del código con las cláusulas *before* y *after* se define lo que se hace antes y después de ejecutar el cuerpo del script.

En el *after*, se comprueba en *Buscar Actividad* si buscando el título de la actividad que se acaba de crear, aparece o no. De esta manera se sabe si estos dos módulos están integrados correctamente.

6.2.2 Resultados de las pruebas

Los resultados esperados serán encontrar un elemento *ion-card* en la pestaña *Buscar Actividad* ya que esto es lo que se espera en el script de testing. Al crear una actividad, se sube a la BD y cuando buscas la actividad creada te debe aparecer un elemento *ion-card* que corresponda con la actividad creada.

A continuación, se ve el trozo final de ejecución del script:

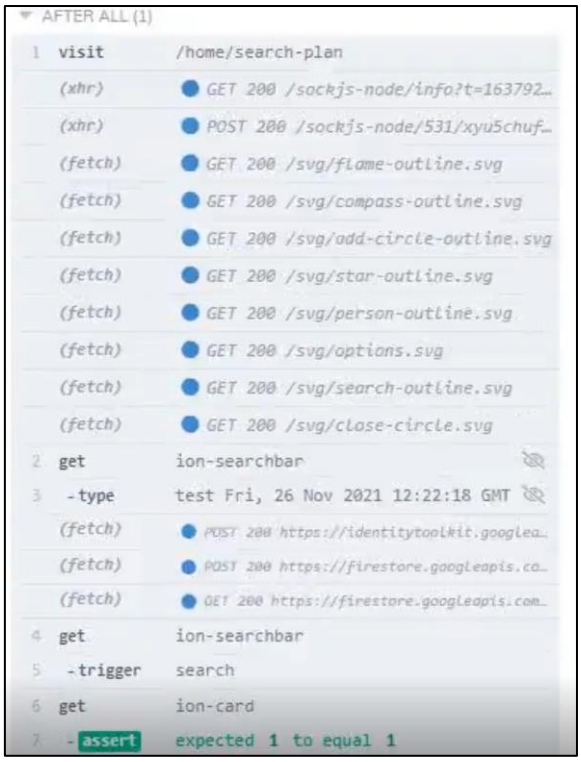


Figura 48 - Resultado de pruebas de integración

En la Figura 48 se observa parte de la ejecución que hace el navegador cuando recibe las instrucciones del script. La prueba se ha pasado ya que se esperaba un elemento *ion-card* y se cómo se puede observar el Assert es positivo y se ha cumplido ya que se ha recibido una instancia de este elemento. Por lo tanto, las funcionalidades *Crear Actividad* y *Buscar Actividad* están integradas correctamente, esto es lo que se ha hecho con todos los componentes que se relacionan entre sí. Des está manera se puede probar todos los componentes integrados con el uso de testing automático

6.3 Pruebas con usuario

Por último, tras realizar las pruebas unitarias y de integración, se van a diseñar unas pruebas dirigidas al usuario. Estas irán dirigidas a los mismos 4 sujetos a los que se realizó las pruebas de validación de los mockups.

6.3.1 Diseño de las pruebas

A lo usuarios que van a ser testeados se les va a entregar el MVP realizado de la aplicación. Estos 4 usuarios van a ser los mismos usuarios que a los que se les realizó las pruebas de validación de mockups. Por lo tanto, estarán algo familiarizados, pero solo visualmente, ya que no han podido usar la aplicación aún.

De nuevo, se les va a pedir que hagan uso de la aplicación, se registren y usen las funcionalidades principales: Crear Actividad, Buscar Actividad, Actividades Recomendadas, Mis Actividades y Registro de usuario.

Se les exigirá que prueben todo lo que encuentren dentro de esas pantallas a lo largo de 20 minutos. Mas tarde se les entregará un formulario con ciertas preguntas que tendrán que responder (Ver Anexo 3: Formulario de pruebas de usuario).

El objetivo será obtener *feedback* real de la aplicación con las preguntas que hagamos. Toda mejora que nos propongan será tomada en cuenta para futuras actualizaciones y mejoras de ***Join***.

6.3.2 Resultados de las pruebas

Los resultados obtenidos por parte de los cuatro usuarios han sido bastante positivos. Cada uno de ellos destaco aspectos buenos sobre la aplicación y algunos aspectos mejorables.

Aspectos destacados:

Usuario 1:

- La aplicación es sencilla y agradable de usar, muy parecida a lo que el usuario tenía en mente cuando hizo las pruebas de los mockups.
- Al crear una actividad, sería ideal que se pudiese poner la fecha y hora de la actividad a menos de una hora de la hora actual, ya que puede ocasionar que nadie se una a esa actividad. Es idóneo por lo menos crear una actividad con hora y media de antelación.
- El usuario ha sabido que hacer en cada pantalla sin estar perdido en ningún momento.

Usuario 2:

- Ha destacado que al igual que en los mockups la aplicación es demasiado llamativa, distrayendo quizá demasiado.
- El usuario no ha encontrado ningún bug en la aplicación
- El usuario ha sabido que hacer en cada pantalla sin estar perdido en ningún momento.

Usuario 3:

- La aplicación ha resultado ser muy parecida a los mockups, es decir visualmente muy atractiva, cautivando de esta manera la atención del usuario.
- El usuario ha detectado que no había ningún apartado de historial de actividades a las que te has apuntado una vez haya pasado la fecha de esa actividad. Ha mencionado que con esto podría encontrar gente con la que hizo esas mimas actividades o incluso la entidad organizadora que la organizo, si este fuera el caso.

- El usuario no ha encontrado ningún bug en la aplicación
- El usuario ha sabido que hacer en cada pantalla sin estar perdido en ningún momento.

Usuario 4:

- La aplicación ha sido agradable de usar para el usuario y sorprendentemente rápida.
- El usuario no ha encontrado ningún bug en la aplicación
- El usuario ha encontrado un tanto enrevesado el que las funciones de crear y buscar grupo estén dentro de una subpestaña de navegación, *Mis Actividades* en este caso, en lugar de tener una propia.
- El usuario ha comentado que no ha encontrado una manera de compartir actividades entre con otros usuarios directamente debido a que no hay sistema de amigos en la aplicación.

6.3.3 Análisis de los resultados

Los resultados obtenidos han sido bastante interesantes. De ellos se puede obtener varias conclusiones y pequeñas modificaciones a implementar en el futuro:

- ***Join*** es una aplicación atractiva y atrevida visualmente para la mayoría de las personas.
- Tiene una funcionalidad bastante intuitiva de usar y con pocos bugs encontrados. Todos aquellos que se han encontrado se arreglarán.
- La usabilidad ha sido bastante positiva ya prácticamente todos los usuarios han sabido que hacer o como usar las funcionalidades que han probado. Hay que destacar que uno de los ha propuesto una mejora en navegación de pantallas respecto a los grupos de interés que se tomará en cuenta. En este apartado es importante destacar que ya son 5 pestañas de navegación las que hay en la barra inferior, y añadir una quizá sería cargar de más esta barra.
- La funcionalidad de consultar el historial de actividades a las que te has apuntado y acudido puede ser crucial implementarla en ***Join***. Puede ser una herramienta muy importante para futuras actividades a las que quieras apuntarte.
- La aplicación carece de la posibilidad de compartir actividades con otros usuarios ya que no hay un sistema definido de tener “amigos” en la aplicación. Esto es algo que se no se ha buscado en ***Join*** con el objetivo de no romper esa pureza de conocer a gente nueva en actividades espontaneas y no crear ese “superficialísimo”, que se evitaba a toda costa. Sin embargo, no existe una manera de mantener contacto con esta gente que has conocido y que has establecido una amistad en la vida real. Puede ser interesante añadir una funcionalidad de añadir como “amigos” a personas con las que has compartido actividades en persona.

7 Conclusión

En esta sección se analiza si se han cumplido los objetivos propuestos al inicio del trabajo, así como también se va a mencionar los errores y problemas que se han encontrado y cómo se han solucionado, qué se ha aprendido de este trabajo tanto profesional como personalmente, y, por último, la relación del trabajo realizado con los estudios cursados.

7.1 Hipótesis y objetivos cumplidos

Los objetivos que se plantearon (ver apartado 1.2), fueron ideados para cumplirlos todos a lo largo de la memoria y en un futuro cercano de la aplicación. Se han cumplido varios de ellos realizando la memoria.

En primer lugar, se tenía como objetivo obtener un primer MVP de la aplicación, algo que se ha cumplido y nos ha servido como base en parte de la memoria. Este era sin duda uno de los más importantes ya que iba a ser la base de la aplicación.

En segundo lugar, se planteó que la aplicación tuviese una buena usabilidad, fuese agradable a la vista y poco intrusiva. Gran parte de ***Join*** se ha desarrollado entorno a ese concepto, pues ya en los mockups se tuvo muy en cuenta, y aún más en el desarrollo. Esta usabilidad fue probada con usuarios, para asegurarse de que, a pesar de intentarlo, se había conseguido ese objetivo (ver apartado 6.3).

En tercer lugar, se deseaba obtener un buen modelo de negocio que se pudiese sostener en el tiempo y estuviese dirigido a largo plazo. Se optó por el modelo descrito en el apartado 2.2.1. Un modelo que tiene dos fuentes de ingreso principales e ideas para para posibles fuentes de ingreso futuras. Por lo tanto, este objetivo se ha cumplido correctamente a lo largo del desarrollo del trabajo.

En cuarto y último lugar, se deseaba atraer a empresas y entidades que deseasen publicar sus actividades en la aplicación. Esto era un objetivo de ***Join*** en general, ya que este TFG no iba dedicado a nada relacionado con el marketing. Este objetivo es el que se va a cumplir próximamente, donde ya con un MVP que poder mostrar y planes de futuro, se puede buscar atraer a las entidades que deseábamos tener.

En conclusión, podemos decir que la primera versión obtenida ha cumplido con lo que estaba ideado. Tanto con los objetivos como con la idea en general, donde se deseaba que los usuarios pudiesen interactuar compartiendo sus actividades favoritas entre ellos.

7.2 Aprendizaje e imprevistos

En el desarrollo del trabajo de la aplicación han surgido ciertos problemas que han ralentizado el mismo. Se han encontrado problemas como:

- Uso de algunas nuevas tecnologías con curvas de aprendizaje más lentas. Este tipo de problemas se solucionaban dedicando tiempo exclusivo para aprenderlas, con libros y

documentación y también se acudió a compañeros de gremio que ya hubiesen utilizado las mismas, con el objetivo de recibir clases y consejos.
- Errores de estructuración de código que ralentizaban la aplicación al comienzo del desarrollo. Estos hacían el código difícil de leer, principalmente debido a el problema anterior de falta de conocimiento sobre el potencial y capacidad de esa tecnología. Estos fueron solucionados cuando este problema estaba ralentizando bastante el desarrollo y se decidió preguntar consejos a ciertos profesionales del ámbito sobre como optimizar el código.

Principalmente estos fueron los dos problemas que dificultaron el desarrollo del trabajo, pero fueron solucionados a tiempo. El resto de los problemas que surgieron fueron más técnicos y específicos del dominio de la aplicación.

El aprendizaje ha sido gran parte de este recorrido, y son varias las cosas que se han aprendido:

- Se ha aprendido a usar el *framework* Angular JS y estructurar un proyecto *frontend* de manera óptima.
- Se ha aprendido a usar SpringBoot de manera funcional obteniendo cualquier funcionalidad usando un código muy limpio, sencillo y de poca longitud.
- Se ha aumentado el conocimiento de diseño gráfico y usabilidad de interfaces, debido al énfasis que se ha dado a este punto.
- Se ha aprendido a gestionar una BD y hospedarla con uso de tecnologías como AWS.

En conclusión, se han obtenido conocimientos clave para el desarrollo profesional de cualquier producto software, tanto a nivel técnico con ciertas tecnologías como a nivel general. Personalmente se ha aprendido que cualquier idea u objetivo que se desee hacer realidad o de manera física conlleva más esfuerzo y trabajo del que puede parecer de primeras.

7.3 Relación del trabajo desarrollado con los estudios cursados

Los estudios que se han cursado a lo largo de los 4 años de Ingeniería informática han sido primordiales para la realización de este trabajo. Específicamente el año cursado en la rama de *Ingeniería del Software.*

En primero lugar, algo básico como las bases de programación proporcionadas en el primer y segundo año del grado han sido indispensables, sin ellas el programar una aplicación sería impensable.

En cuanto a nivel técnico se han utilizado y consultado mucha teoría de las asignaturas cursadas. Recorriendo la memoria por apartados se puede obtener en claro qué asignaturas se han aplicado en cada caso:

- Sección 2, donde se realiza una evaluación de la idea de negocio. Pasando por un análisis de competidores, de explicación del modelo de negocio, realización de una proyección

económica, análisis DAFO y Lean Canvas. Este tipo de estudios se han realizado en las asignaturas de Fundamentos de organización de empresas del cuatrimestre 1B y Gestión de proyectos del cuatrimestre 3B. En ellas se estudia el cómo analizar tus competidores y el mercado para encontrar un hueco y aportar valor, así como gestionar una proyección económica de un proyecto realista.

- Sección 3, donde se realiza un análisis del problema, obteniendo requisitos, modelo de dominio, características y casos de uso. Este proceso indispensable del desarrollo del software fue abarcado en *Análisis y Especificación de Requisitos* de la rama de IS en el cuatrimestre 4A.
- Sección 4, aquí se ha hecho uso de los conocimientos sobre arquitecturas, visto en la asignatura *Ingeniería del Software* de 3A. Para los mockups se ha hecho uso de conocimientos de usabilidad y navegabilidad aprendidos en las asignaturas *Interfaces Persona Computador* de 2B y la optativa de 4B *Diseño de sitios web.*
- Sección 5, se ha hecho uso de conocimientos obtenidos en asignatura de 3B y 4A de la rama IS como pueden ser *Proceso de Software* y *Proyecto de Ingeniería de Software, y*a que en ambas se tuvo que desarrollar un producto software al igual que en este trabajo.
- Sección 6, se ha hecho uso de los conocimientos obtenidos sobre las pruebas en las asignaturas de *Ingeniería de Software* de 3A y *Mantenimiento y Evolución de Software* de 4A.

8 Trabajos futuros

Al finalizar el trabajo realizado sobre ***Join***, quedan tareas pendientes para el futuro. El objetivo es irlas desarrollando y cumpliendo poco a poco hasta llegar a tener ***Join*** completamente terminada y en funcionamiento. Para ello se propondrá realizar las siguientes tareas:

- Se planteará, desarrollará y mejorará todo aquello que, tras las pruebas de usuario, se ha analizado y obtenido como resultado en el apartado 6.3.3.
- Se empezará a desarrollar el subsistema de entidades organizadoras con el objetivo de tenerlo en marcha en poco tiempo
- Se lanzará ***Join*** a mercado con campañas de marketing siguiendo la planificación y los presupuestos planteados en la proyección económica.
- Se integrarán poco a poco más entidades y usuarios activos en la aplicación, de esta manera se sustentará de manera más automática y podrá funcionar.
- Se propondrá ofrecer incentivos tanto a los usuarios como a las empresas con detalles como realización de eventos en vivo hechos por ***Join*** o descuentos en las propias licencias. En conclusión, se llevará a cabo los modelos de negocio futuros planteados al final del apartado 2.2.1 donde se explican estos.
- Se realizará un mantenimiento profesional de ***Join***, donde se irán corrigiendo y cambiando pequeños detalles para pulir el producto final. Mantener *Join* de manera correcta será prioridad en el futuro, ya que es en esta fase cuando se puede consolidar una buena aplicación.

9 Referencias

[1] Maria Lourdes Tapia y Marqueza Cornejo. «Redes sociales y relaciones interpersonales en internet» Accedido el 15 de abril de 2022. https://www.redalyc.org/pdf/184/18426920010.pdf.

[2] Maida, Esteban Gabriel, y Julián Pacienzia. «Metodologías de desarrollo de software», 2015. Accedido 9 de abril de 2022. https://repositorio.uca.edu.ar/handle/123456789/522.

[3] Eva María Sánchez Cano. «Freemium como modelo de negocio en Internet». Accedido 15 de abril de 2022. https://tauja.ujaen.es/jspui/handle/10953.1/4411.

[4] «Herramienta DAFO». Accedido 28 de abril de 2022. https://dafo.ipyme.org/Home.

[5] «Curso: PROYECTO DE INNOVACIÓN "EMPRENDIMIENTO E INNOVACIÓN SOCIAL DIGITAL" (PRIMER SEMESTRE 2018-19)». Accedido 3 de mayo de 2022. https://cv4.ucm.es/moodle/course/view.php?id=99042.

[6] Departamento de Ciencias de la Computación e I.A, DECSAI, «Especificación de requerimientos». Accedido 15 de mayo de 2022. https://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf.

[7] IONOS Digitalguide. «El diagrama de casos de uso en UML». Accedido 17 de mayo de 2022. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/.

[8] Vázquez Cendron, Agustín. «Arquitectura en capas: análisis y estudio de caso del modelo arquitectónico N-capas y sus variantes». Tesis, Universidad Nacional de La Plata, 2018. Accedido 17 de mayo de 2022. http://sedici.unlp.edu.ar/handle/10915/119458.

[9] hmong.wiki. «Separación de intereses ImplementaciónyOrigen». Accedido 17 de mayo de 2022. https://hmong.es/wiki/Separation_of_concerns.

[10] «Introduction to Ionic | Ionic Documentation». Accedido 2 de mayo de 2022. https://ionicframework.com/docs.

[11] Capacitor: Cross-platform native runtime for web apps. «Capacitor: Cross-Platform Native Runtime for Web Apps». Accedido 15 de mayo de 2022. https://capacitorjs.com.

[12] IBM. «IBM Docs», Accedido 13 de mayo de 2022. https://www.ibm.com/docs/es/was-liberty/nd?topic=overview-java-persistence-api-jpa.

[13] «Buenas practicas en la configuración de Spring | Marco de Desarrollo de la Junta de Andalucía». Accedido 23 de mayo de 2022. http://www.juntadeandalucia.es/servicios/madeja/sites/default/files/historico/1.3.0/contenido-libro-pautas-40.html.

10 Anexos

En este apartado se va a exponer toda aquella información que no se ha introducido en las distintas secciones de la memoria principal, esto es por motivos de relevancia y de optimización de la memoria.

10.1 Anexo 1: Mockups de la aplicación.

En el apartado 4.3.1 se muestran los mockups principales de la aplicación. No es posible mostrar todos en esa sección debido al espacio que pueden ocupar. Por lo tanto, los demás mockups se van a presentar a continuación:

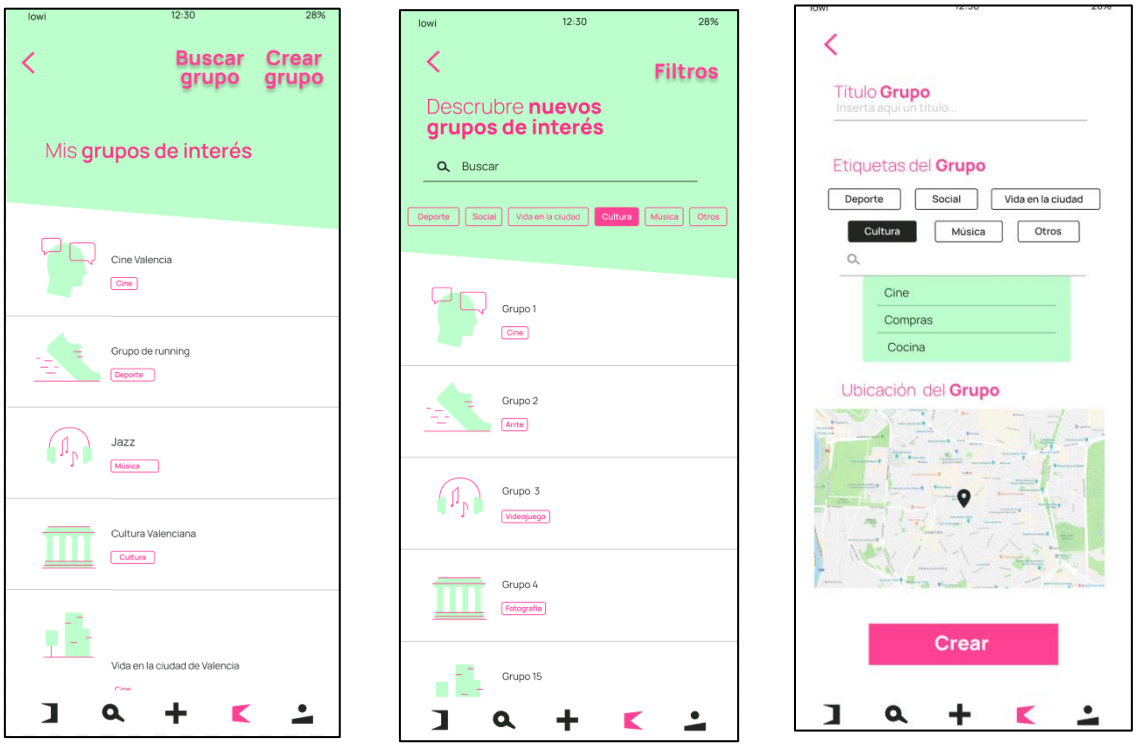


Figura 49 - Join, mockups de Mis grupos de interés, Buscar grupo de interés y Crear grupo de interés

La Figura 49 muestra todas las interfaces diseñadas entorno a los grupos de interés.

Figura 50- Join, Mockup de Mi Perfil de Entidad Organizadora

Figura 51- Join, Mockup de Mi Perfil de usuario estándar

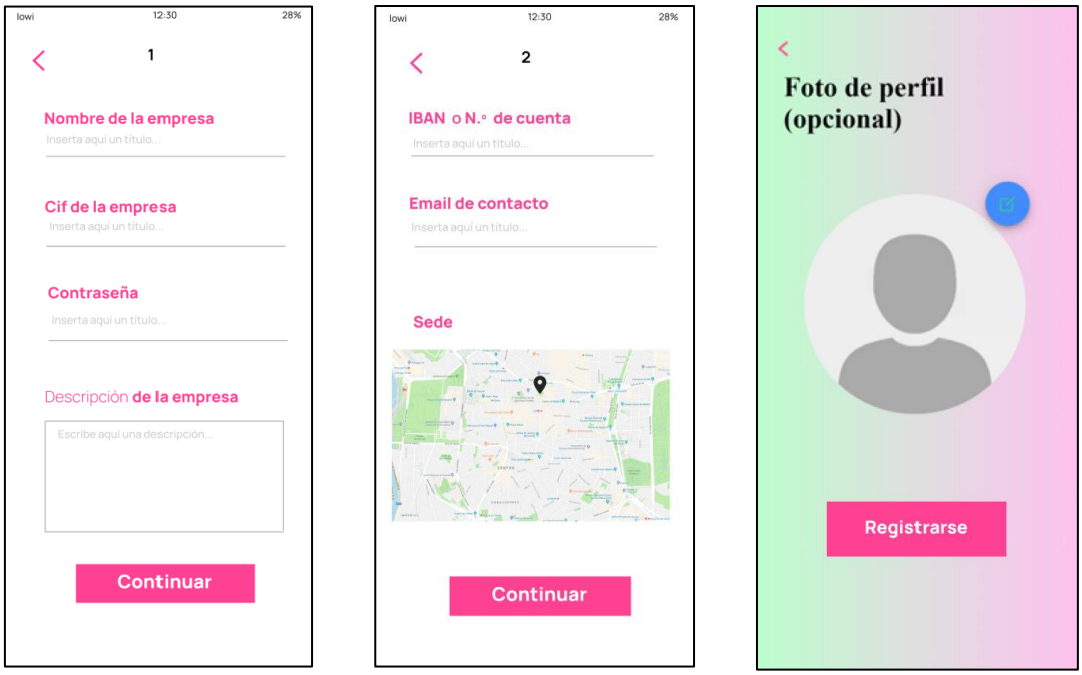


Figura 52- Join, Mockups de registro de Entidad Organizadora

Como se observa, la Figura 52 representan lo mockups del proceso de registro a la aplicación de una entidad organizadora.

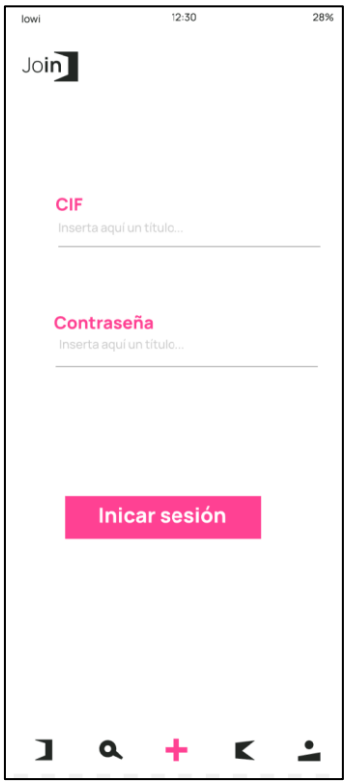


Figura 53- Join, Mockup de Iniciar Sesión de Entidad Organizadora

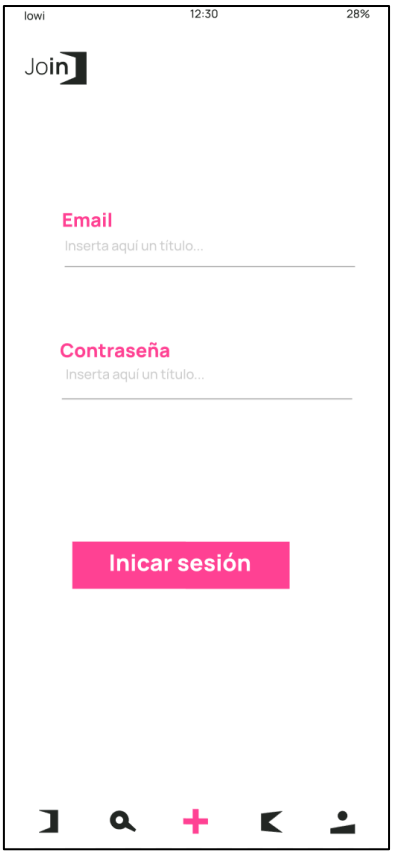


Figura 54- Join, Mokcup de Iniciar Sesión de usuario estándar.

Por último, en las figuras Figura 52 y Figura 53 se observan los mockups de las interfaces de Iniciar Sesión de un usuario estándar y de una entidad organizadora.

10.2 Anexo 2: Formulario de validación de mockups

En este anexo se va a adjuntar y comentar el formulario que se le entrego a los 4 usuarios que realizaron la prueba de validación de los mockups del apartado 4.3.2. Este formulario constaba de 4 preguntas sencillas que los usuarios tenían que responder, ante los resultados obtenidos con este formulario se realizó un análisis de estos en el apartado 4.3.3.

Figura 55 - Formulario entregado a los usuarios de la prueba de validación de mockups

En la Figura 55 se muestra el formulario entregado. Todas las preguntas exceptuando la última iban orientadas a obtener un *feedback* de aspectos a mejorar de las interfaces. Esto era nuestra prioridad antes de iniciar con el desarrollo.

10.3 Anexo 3: Formulario de pruebas de usuario

En este anexo se va a adjuntar y comentar el formulario que se le entrego a los mismos 4 usuarios que realizaron en la prueba de validación de los mockups del apartado 4.3.2, pero esta vez para realizarles unas pruebas de usuario sobre el primer MVP de la aplicación (apartado 6.3). Este formulario constaba de 4 preguntas sencillas que los usuarios tenían que responder, ante los resultados obtenidos con este formulario se realizó un análisis de estos en el apartado 6.3.3.

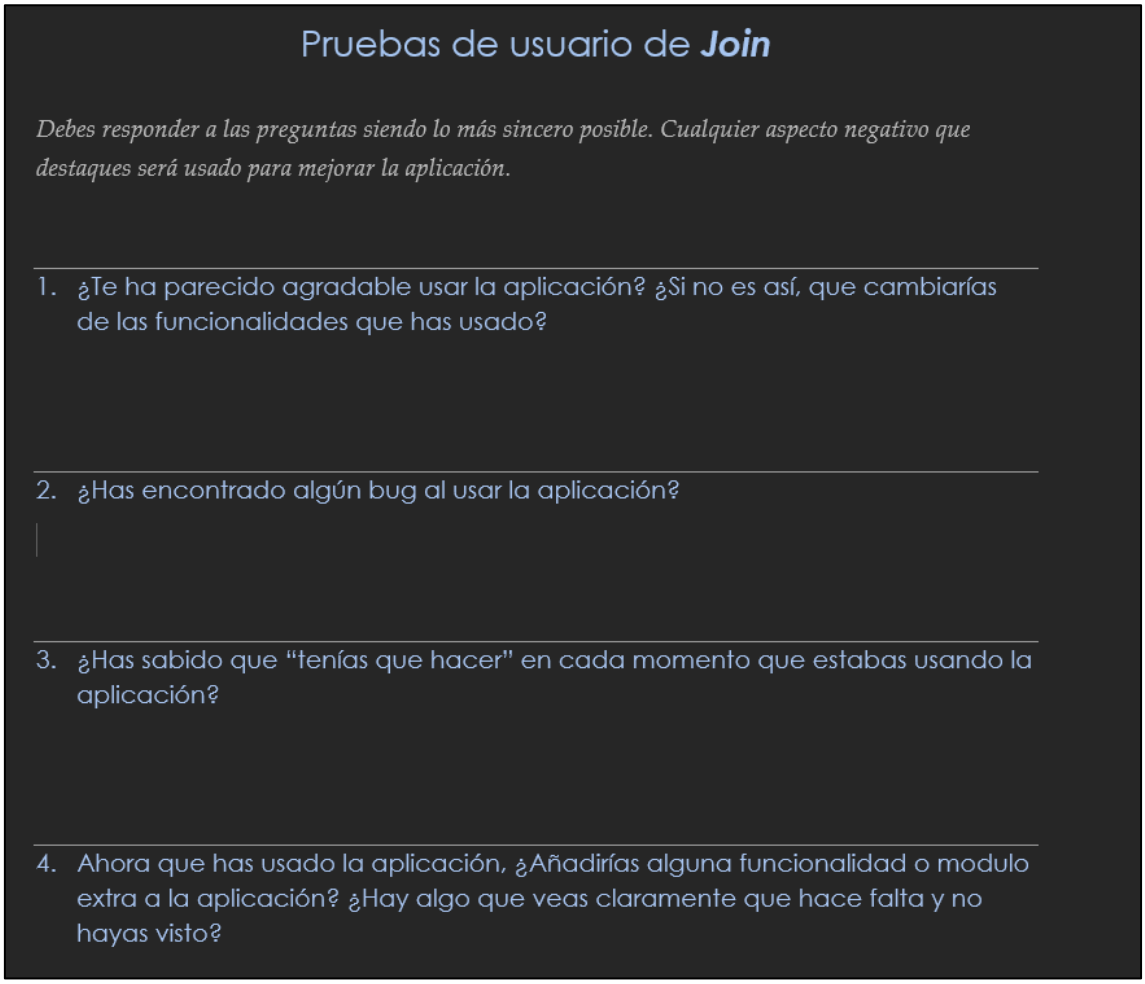


Figura 56- Formulario entregado a los usuarios para realizar las pruebas de usuario

Las preguntas de la Figura 56 estaban orientadas a encontrar de nuevo aspectos mejorables de la primera versión de ***Join***. El objetivo era recaudar toda la información y errores posibles para corregirlos cuantos antes y obtener un MVP sin errores.

10.4 Anexo 4: ODS (OBJETIVOS DE DESARROLLO SOSTENIBLE)

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
Fin de la pobreza.				**X**
Hambre cero.				**X**
Salud y bienestar.	**X**			
Educación de calidad.				**X**
Igualdad de género.		**X**		
Agua limpia y saneamiento.				**X**
Energía asequible y no contaminante.				**X**
Trabajo decente y crecimiento económico.	**X**			
Industria, innovación e infraestructuras.				**X**
Reducción de las desigualdades.				**X**
Ciudades y comunidades sostenibles.				**X**
Producción y consumo responsables.				**X**
Acción por el clima.				**X**
Vida submarina.				**X**
Vida de ecosistemas terrestres.				**X**
Paz, justicia e instituciones sólidas.				**X**
Alianzas para lograr objetivos.				**X**

Tabla 3 - Tabla de los objetivos de desarrollo sostenibles usados en Join

Reflexión sobre la relación del TFG con los ODS y con los ODS más relacionados.

Todos los objetivos anteriores fueron definidos por lideres mundiales para fomentar el desarrollo sostenible en todos los ámbitos. Estos objetivos están orientados a ser cumplidos en los próximos 15 años. Todo trabajo o proyecto que se realice hoy en día puede tener una cierta relación con alguno de estos aspectos. De hecho, si la tiene, ya está aportando un valor añadido a parte del objetivo principal del proyecto. Esto es lo que ocurre con mi aplicación ***Join***.

Es cierto que no está relacionada con la mayoría de los objetivos de la lista, esto se debe principalmente a que es una aplicación orientada a fomentar las relaciones sociales y la diversión de las personas y no tanto a otro tipo de objetivos. Sin embargo, si hay 3 objetivos de la lista que están orientados a esta misma rama de la aplicación, y sí que están muy presentes en ***Join***. Principalmente se encuentra que la salud y bienestar, la igualdad de género, y por último el trabajo decente y crecimiento económico.

A continuación, se va a comentar porque estos 3 objetivos están presentes en la aplicación, a que nivel, pero, sobre todo, de qué manera:

Salud y bienestar

En ***Join*** se fomenta la interacción social e interpersonal a la vez que realizas aquello que te gusta. Uno de los aspectos más importantes de la salud y bienestar de las personas es el salir de casa a realizar los hobbies que más te gustan, así como también salir a socializar y conocer a gente nueva en tu vida. Por lo tanto, se puede decir que este objetivo es algo que se cumple en ***Join*** en todos sus aspectos. ***Join*** trata de mejorar la vida social de distintos perfiles de personas; Personas con dificultades para conocer a gente nueva, personas que no encuentran gente con la que compartir sus gustos, personas que solo quieren pasárselo bien con otras personas sin importar quienes sean, personas que buscan pareja etc… En todos estos aspectos la salud y bienestar de la persona se beneficia de ***Join***.

Trabajo decente y crecimiento económico

Este objetivo es el segundo que más fomenta ***Join***, específicamente con el subsistema que permite a empresas o entidades organizar sus propias actividades de pago que fomenten su negocio y atraigan nuevas personas a él. De esta manera estas entidades se están beneficiando del alcance que ofrece ***Join*** a usuarios dispuestos a realizar actividades divertidas con personas nuevas.

Igualdad de género

Por último, este objetivo está menos presente en ***Join*** si se compara con los otros dos analizados. Sin embargo, en la aplicación ambos géneros tienen las mismas oportunidades y capacidades de hacer lo que ellos desean. Esto significa que internamente en ***Join*** un género nunca llegará a tener más poder de elección o algo que le favorezca en comparación con el género contrario. De hecho, en ***Join*** si alguna persona no se siente identificado con el género masculino o femenino, es posible escoger otro género que te permita emparejarte con aquellos que se siente igual que tú.