UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Aedo, Aplicación de experiencias turísticas entre usuarios: Frontend móvil

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Matarredona Coloma, Joan

Tutor/a: Bort Mir, Lorena

CURSO ACADÉMICO: 2022/2023

El proyecto “AedoApp” ha sido realizado en conjunto por cuatro alumnos de la Escuela Técnica Superior de Ingeniería Informática de la Universidad Politécnica de Valencia en el campus de Vera durante el curso 2022/2023.

Aunque el desarrollo se ha realizado de manera conjunta, cada alumno presentará una parte como trabajo de fin de grado, es por ello que los tres primeros apartados de este documento son comunes en las cuatro memorias de trabajo de fin de grado así como el anexo “Manual de uso de la aplicación”.

Los alumnos implicados en el proyecto y los trabajos que van a presentar son:

- Hernández Bonet, Álvaro: Inteligencia artificial
- Matarredona Coloma, Joan: Frontend de la aplicación móvil
- Mut Portes, Andreu: Backend de la aplicación
- Palacios Martínez, Diego: Frontend de la aplicación web

Agradecimientos

Este trabajo de fin de grado no hubiera podido ser realizado si no fuera por la participación del equipo de trabajo de Aedo Álvaro Hernández, Andreu Mut y Diego Palacios. También dar las gracias al equipo de diseño Demetrio Villena y Rafael Cruañes.

Resumen

El turismo globalizado tiende a concentrarse en destinos específicos del planeta, lo que puede generar problemas tanto para los residentes locales como para el ecosistema de las ciudades. Casos emblemáticos como Venecia o Barcelona han experimentado los efectos negativos del turismo masificado. A este problema se le suma la despoblación de los entornos rurales que cada vez tienen menos visibilidad y servicios. Para abordar esta problemática, se ha desarrollado una plataforma que ofrece una alternativa al turismo centralizado, enfocándose en lo que los habitantes de cada zona pueden ofrecer. La plataforma busca descentralizar el turismo y ofrecer visibilidad a las comunidades rurales, promoviendo destinos menos conocidos y evitando la sobrecarga turística en lugares ya saturados. Al permitir que los habitantes compartan sus perspectivas y conocimientos, se fomenta una interacción más auténtica y enriquecedora entre los visitantes y las comunidades locales.

En este Trabajo de Fin de Grado (TFG), se ha trabajado en la creación de una plataforma que permite a cualquier persona compartir su contribución a la cultura y el patrimonio de su lugar de residencia. Específicamente, se ha centrado en el desarrollo de la aplicación móvil, enfocándose en el desarrollo frontal de la misma. Para lograrlo, se ha trabajado en equipo utilizando metodologías ágiles, lo que ha permitido una colaboración coordinada y eficiente en el proceso de desarrollo.

Durante el proceso de desarrollo y en concreto en este TFG, se ha dado especial atención al diseño de la interfaz de usuario, asegurando una experiencia intuitiva y atractiva para los usuarios de la aplicación. Asimismo, se ha realizado un trabajo exhaustivo de pruebas y refinamiento para garantizar un funcionamiento fluido y eficiente. Se ha desarrollado con la ayuda de Start.inf, el espacio de emprendimiento de la ETSINF, construyendo dos MVP (producto mínimo viable) con sus correspondientes experimentos.

Palabras clave: Descentralización del turismo, contribución a la cultura, metodologías ágiles, interfaz de usuario, MVP.

Abstract

Globalized tourism tends to concentrate on specific destinations around the world, which can create problems for both local residents and the ecosystem of cities. Emblematic cases such as Venice or Barcelona have experienced the negative effects of mass tourism. This problem is compounded by the depopulation of rural areas, which are increasingly experiencing reduced visibility and services. To address this problem, a platform has been developed to offer an alternative to avoid centralised tourism, focusing on what local people have to offer. The platform seeks to decentralise tourism and give visibility to rural communities, promoting lesser-known destinations and avoiding tourist overload in already saturated places. By allowing local people to share their perspectives and knowledge, it fosters a more authentic and enriching interaction between visitors and local communities.

In this Final Degree Project (FDP), we have worked on the creation of a platform that allows anyone to share their contribution to the culture and heritage of their place of residence. Specifically, it has focused on the development of the mobile application, focusing on its front-end development. To achieve this, teamwork using agile methodologies has been used, which has allowed for a coordinated and efficient collaboration in the development process.

During the development process and specially in this FDP, special attention has been given to the design of the user interface, ensuring an intuitive and attractive experience for the users of the application. In addition, extensive testing and refinement work has been carried out to ensure smooth and efficient operation. It has been developed with the help of Start.inf, the entrepreneurship space of the ETSINF, developing two MVP (minimum viable product) with their corresponding experiments.

Keywords : Decentralization of tourism, contribution to culture, agile methodologies, user interface, MVP.

Índice general

Contenido

Índice de figuras

Índice de tablas

1 Introducción

1.1 Introducción

Cuando se piensa en una ciudad o una región, se suelen destacar una serie de lugares o experiencias de mayor calado turístico que la gran mayoría de las personas conoce, pero ¿qué ocurre con todas las tradiciones y la cultura que se esconde tras esos sitios?

Actualmente, existen en el mercado numerosas aplicaciones para ayudar al viajero/a que no sabe qué actividades realizar en la localidad de destino a la que se dirige. Estas aplicaciones ofrecen, por lo general, una serie de tours convencionales que suelen dejar una imagen sesgada de la cultura, las tradiciones o la realidad de la sociedad de una ciudad o una región. Un/a turista que visita la ciudad de Valencia suele ir a la Ciudad de las Artes y las Ciencias, a la Catedral, a la Plaza del Ayuntamiento, a probar una paella en la playa... pero ¿de dónde viene ese arroz que se ha comido en la paella, qué importancia tiene la preparación de la paella para la cultura valenciana, o qué rito gastronómico supone la típica reunión familiar de los domingos para degustar ese plato? ¿Qué supone el "esmorzaret" para un valenciano? ¿Cuál es un típico "esmorzaret"? Todas estas cuestiones son las que al final marcan el carácter y la tradición de una sociedad, y estos elementos no tienen cabida en la típica visita a la ciudad.

Aedo nace de la necesidad y de la importancia de poder ofrecer unas actividades turísticas de calidad a los viajeros/as, que permitan poner en valor las tradiciones y la cultura de cualquier lugar, por pequeño o inhóspito que sea. Además, ofrece una plataforma a todas estas localidades para poder publicitar la riqueza de sus tierras, atrayendo a nuevos visitantes con todas las ventajas que esto puede suponer.

1.2 Objetivos

La finalidad de este trabajo es realizar un proyecto de emprendimiento que ofrezca como resultado una aplicación que pueda ser usada por cualquier usuario/a y en la que pueda encontrar actividades (gastronómicas, culturales, al aire libre, etc.) que se ofrezcan en distintas poblaciones de España realizadas por gente local. Para conseguir este producto se han definido una serie de objetivos comunes a todos los miembros del equipo que habría que alcanzar durante el desarrollo de la aplicación:

- Elaborar, desarrollar y evaluar una idea de negocio teniendo en cuenta el mercado actual.
- Obtener una aplicación que cumpla con las características establecidas por el equipo de desarrollo.
- Valorar mediante experimentos la idea de negocio.
- Desarrollar funcionalidades atractivas e intuitivas para los usuarios en el ámbito móvil.
- Crear una estructura sencilla y fácil de mantener para el desarrollo de la presentación de la aplicación móvil.

1.3 Estructura de la memoria

El presente documento expone una idea de negocio y el desarrollo posterior de una aplicación que va a ser el producto software resultante. Los distintos apartados de la memoria van a ser los siguientes:

- Apartado 2: Se realiza el resumen de la aplicación junto a un estudio de la idea de negocio y de mercado para comparar la aplicación con el *software* competidor. Además, se presenta el modelo de negocio, una proyección económica, en análisis de la matriz DAFO y una serie de conclusiones.
- Apartado 3: Se desarrolla la idea de negocio con los dos productos *MVP (Minimum Viable Product)* y sus respectivos experimentos.
- Apartado 4: Explica los distintos aspectos técnicos de la aplicación de una manera general para posteriormente enfocarse en *front-end* móvil.
- Apartado 5: Expone la cronología seguida durante los meses empleados para la realización del trabajo.
- Apartado 6: Aparecen las conclusiones del trabajo realizado en base a los objetivos propuestos, así como las limitaciones del mismo y el posible trabajo futuro que se podría realizar con la aplicación.

El documento finaliza con las referencias bibliográficas consultadas durante el trabajo y un apartado de anexos sobre el funcionamiento general de la aplicación, las preguntas de las encuestas y los experimentos realizados.

2 Evaluación de la idea de negocio

2.1 Resumen de Aedo

Antes de explicar el contenido de la aplicación, es importante explicar la terminología utilizada en el ámbito del proyecto. Empezando por el propio nombre, Aedo proviene del vocablo griego que hacía referencia a los personajes conocidos en la cultura occidental como juglares, y se ha utilizado como un símil a esa tradicional vía oral de transmisión de información, de noticias, tradiciones etc. En este marco histórico, el usuario que utilice la aplicación se convertirá en un "Odiseo", el cual deberá juntarse con el Aedo para que le transmita ese conocimiento ancestral, y en el momento en el que el Odiseo haya protagonizado esa aventura del saber, podrá convertirse a su vez en un Aedo, al publicar sus propias experiencias.

Aedo es una aplicación tanto móvil como web que permite a cualquier usuario publicar y reservar experiencias turísticas. Estas experiencias pueden ser de cualquier calado, aunque la idea subyacente es que se realicen actividades que pongan en valor las tradiciones, la cultura, la gastronomía, etc. de los lugares en los que se realizan.

El principal atractivo de la aplicación es la posibilidad de realizar actividades ofertadas por personas locales de los lugares que se desean visitar, ayudando a paliar problemáticas como la despoblación, las desigualdades laborales o la perdida de trabajos artesanales.

2.2 Idea de negocio

La idea de negocio se trata del producto o el servicio que se ofrece al mercado. Dicho producto puede existir en el mercado actual o ser creado. Después se le aporta una propuesta de valor para el mercado al cual se dirige, es por ello que dicha solución puede generar beneficios para el impulsor del producto y otros beneficios para el usuario final de la aplicación.

Para poder elaborar la idea de negocio de una manera más sencilla y visual, se ha utilizado le técnica del *Lean Canvas (Maurya, 2022)* ya que el *Lean Canvas* es una herramienta de visualización de modelos de negocio rápida porque en una única hoja, de un vistazo, se dispone de toda la información importante y necesaria para llevar a cabo la idea de negocio. El tablero *Lean Canvas* se divide en nueve apartados: clientes, problema, proposición de valor, solución, canales, ingresos, costos, métricas y la ventaja competitiva.

Como resultado de la aplicación de esta técnica, se obtiene la **Figura 1**, la cual recoge todos los apartados mencionados anteriormente. Del tablero cabría destacar los dos puntos que resultan más interesantes: la proposición de valor y la ventaja competitiva, ya que tienen un contenido común, las experiencias. La ventaja competitiva de Aedo son las experiencias personales que se transmiten de persona a persona junto a la gamificación y el chat interno, mientras que la proposición de valor contempla la realización de experiencias para aprender tradiciones de unos a otros para que estas no se pierdan lo cual fomenta el turismo de zonas despobladas, que es donde suelen estar las tradiciones.

Esta versión del Lean Canvas de la **Figura 1** corresponde a la etapa inicial de la idea de negocio de la aplicación y podría sufrir variaciones a lo largo del tiempo, ya que la industria del *software* está en constante cambio y es muy posible que una idea que a priori resultaba innovadora

después sea implementada por otras aplicaciones competidoras. Así pues, la ventaja competitiva planteada inicialmente podría dejar de serlo.

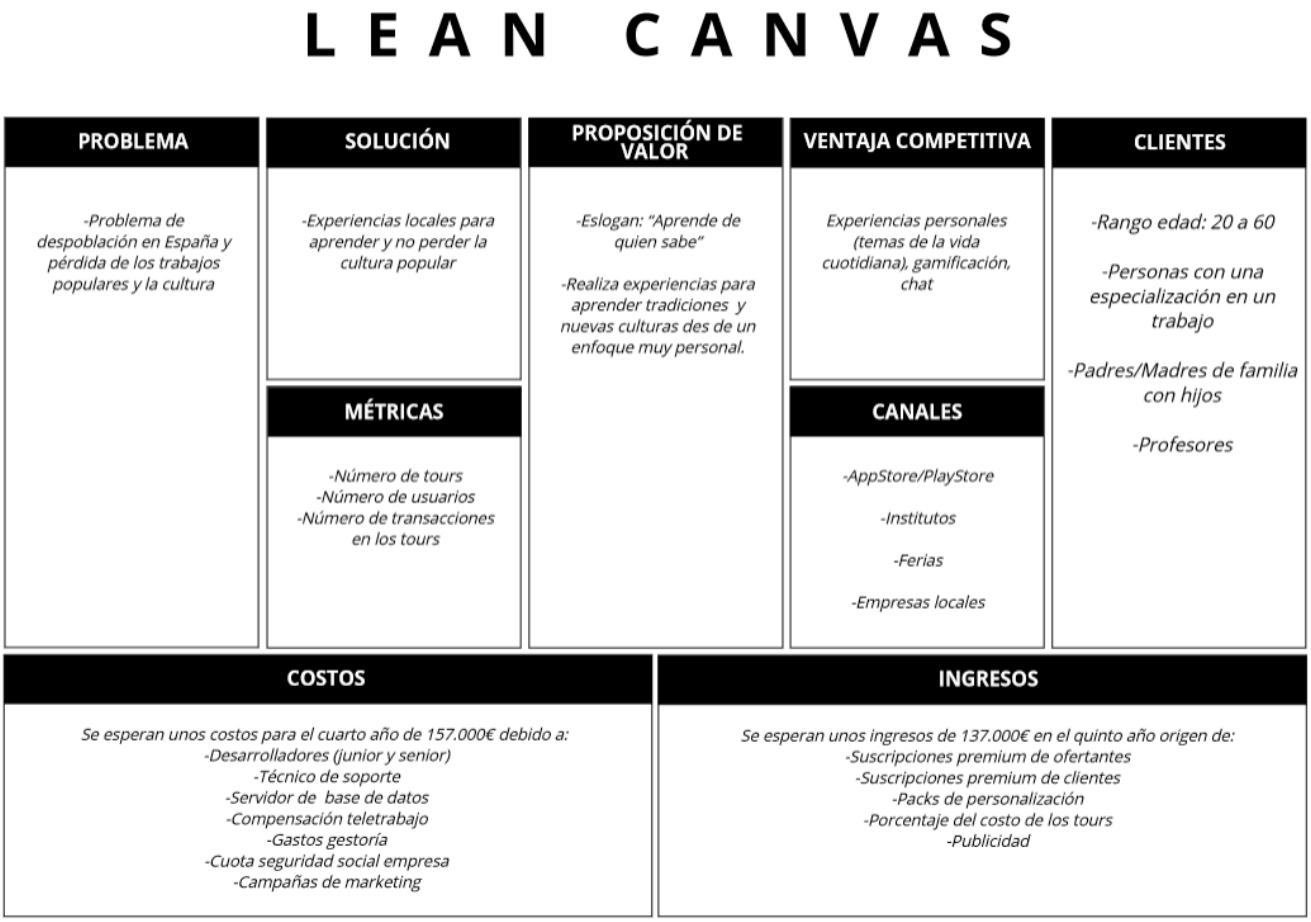

Figura 1: *Lean canvas Aedo App*

2.3 Estudio de mercado

Tras observar las características principales de la aplicación, se realizó un estudio de las posibles alternativas competidoras. Se ha hecho una elección basándose en la similitud del objetivo final de la herramienta, destacando los puntos en común y las diferencias respecto a Aedo.

2.3.1 Civitatis

Esta aplicación[1] (ver **Figura 2**) está centrada en destinos de viajes a las principales ciudades y destinos de las mismas. Es una aplicación que tiene más de 500 mil descargas en

[1] Civitatis: https://www.civitatis.com/es

Google Play; esto la convierte en uno de los competidores más importantes. Además, también incorpora la opción de buscar tours cerca de la localización actual.

***Figura 2**: Web Civitatis (Fuente: civitatis.com/es/)*

Civitaris cuenta con una aplicación específica con guías para varias ciudades europeas para centrarse más en el destino elegido por los usuarios (**Figura 3**).

***Figura 3**: Web Civitatis (Fuente: civitatis.com/es/)*

2.3.2 Guruwalk

Guruwalk[2] (ver **Figura 4**) está centrada en ***free tours*** por las principales ciudades del mundo. Según ellos, se encuentran más de **2300** *free tours* en español, inglés y otros idiomas. Un *free tour* es un tipo de visita turística en la que el viajero paga lo que quiere al final del recorrido, según su satisfacción y presupuesto.

Esta app establece un sistema de propinas: los/as turistas pueden dar una propina a su guía turístico/a al final del tour si están satisfechos/as con la experiencia. Además, fomenta el turismo responsable, sostenible y cultural apoyando a los guías locales, promoviendo el contacto directo y contribuyendo a la diversificación y la desestacionalización de la oferta turística, al ofrecer *free tours* en lugares menos masificados y durante todo el año.

***Figura 4**: Captura de GuruWalk (Fuente: Google play GuruWalk)*

2.3.3 FreeTour.com

FreeTours.com[3] (ver **Figura 5**) es una plataforma online que ofrece tours gratis o de bajo coste con guías locales en más de 120 países. Tiene un sistema de valoración y comentarios de los usuarios/as que ayuda a elegir el mejor tour y a mejorar la calidad del servicio.

Según SimilarWeb, FreeTour.com tiene un tráfico mensual estimado de 1.3 millones de visitas en los últimos 6 meses, con un promedio de 3.5 páginas vistas por visita y un tiempo de permanencia de 4:28 minutos.

[2] GuruWalk: https://www.guruwalk.com/
[3] Freetour.com: https://www.freetour.com/

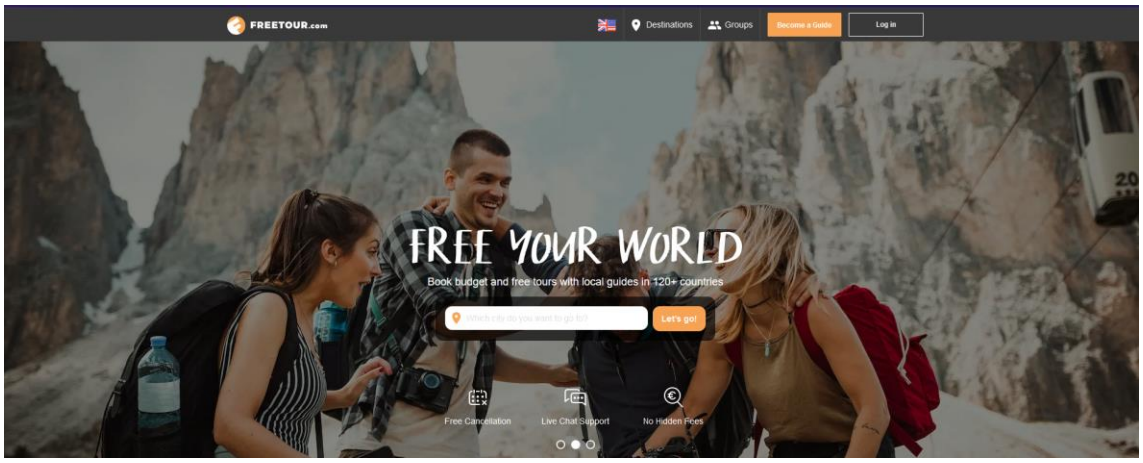

Figura 5: *Web FreeTour.com(Fuente: https://www.freetour.com/)*

Además, también dispone de una versión móvil con más de 100k descargas en Google play (ver **Figura 6**).

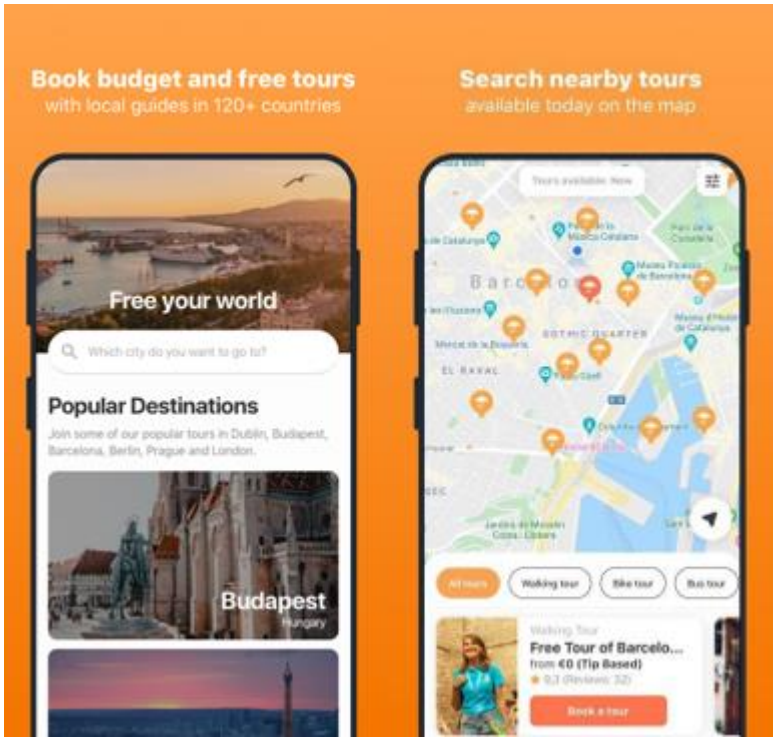

Figura 6*: Web de FreeTour.com (Fuente: freetour.com/es)*

2.3.4 Identify

La *app* de turismo cultural Identify[4] (ver **Figura 7**) es una guía digital que permite descubrir más de 300.000 puntos culturales en todo el mundo. Se puede personalizar la experiencia según los intereses y la geolocalización del usuario/a, así como participar en un juego

[4] Identify: https://www.identifytravel.app/

donde se ganan recompensas y premios por identificar elementos culturales y compartirlos con otros usuarios/as.

***Figura 7**: Web Identify (Fuente: identifytravel.app)*

2.3.5 Withlocals

Withlocals[5] (ver **Figura 8**) es una plataforma que conecta a viajeros/as con gente local que ofrece comida y experiencias únicas. Se puede disfrutar de una ciudad como un lugareño, conocer su cultura y sus secretos, y apoyar el turismo sostenible. Hay más de 100 destinos disponibles en todo el mundo. Las experiencias son guiadas por expertos locales que conocen la historia, la cultura y las normas sanitarias de cada lugar.

[5] Withlocals: withlocals.com/es

***Figura 8**: Withlocals (Fuente: withlocals.com/es/)*

2.3.6 Tabla comparativa

Tras analizar las distintas aplicaciones competidoras, se ha confeccionado una tabla comparativa (**Tabla 1**) en la que se comparan una serie de funcionalidades típicas en este tipo de aplicaciones y se tiene un registro de, por cada programa, cual implementa las características analizadas.

Tabla 1: Tabla comparativa Aedo con otras aplicaciones.

Característica:	**Civitatis**	**Guruwalk**	**FreeTour.com**	**Identify**	**Withlocals:**	**Aedo**
Buscador	✓	✓	✓	✓	✓	✓
Usuarios registrados consumidores	✓	✓	✓	✓	✓	✓
Usuarios registrados productores	✓	✓	✓		✓	✓
Filtros	✓	✓		✓	✓	✓
Compras integradas	✓				✓	✓
Soporte técnico	✓	✓	✓	✓	✓	✓

Diferentes idiomas de interfaz	✓			✓	✓	Español, Inglés
Diferentes idiomas del tour		✓	✓		✓	✓
Distintos tipos de tours	Entradas, espectáculos, escapadas gastronómicas, transportes	Alternativo, arte urbano, bici, leyendas...	Tours gratis, diarios, urbanos, bares, bicicleta, gastronómicos	Sólo muestra puntos de interés	Gastronómicas, bici, arte, música, nocturnas, ...	✓
Sin gastos de reserva	✓	✓	✓		✓	✓
Servicio de la APP gratuito	✓		✓	✓	✓	✓
Ver fotos, reseñas, detalles, descripciones y disponibilidad del tour (quien oferta el tour)	✓	✓	✓	✓	✓	✓
Calificar y comentar el tour/guía	✓	✓	✓		✓	✓
Wishlist de tours	✓		✓			✓
Encontrar tours por cercanía (a la localización actual)	✓		✓	✓		✓
Oferta de entradas a espectáculos	✓					
Recursos para estudiantes y centros educativos				✓		✓
Rutas personaliza-das				✓		✓
Verificación de cuenta	✓	✓	✓	✓	✓	✓
Valoración del ofertante sobre los clientes						✓
Chat Bot						✓

Gamifica-ción						✓

Así pues, se puede observar en la Tabla 1 que hay una serie de características comunes a todas las aplicaciones, como pueden ser buscador, registro de usuarios, soporte técnico, ver fotos y reseñas y la verificación de cuenta. Después, existen algunas características que están en prácticamente todas las aplicaciones como pueden ser los filtros, las compras integradas, uso gratuito de la app o publicar comentarios, por ejemplo. Finalmente están las características propias de la aplicación Aedo, las cuales no están contempladas en la competencia en la actualidad y son: la gamificación, un chat interno, un chat bot para soporte técnico y la valoración por parte de la persona que explica la experiencia sobre los clientes que asisten. Además, Aedo ofrece recursos para estudiantes y centros educativos y rutas personalizadas, característica solo ofrecida por Identify. Sin embargo, si se tuviera que remarcar el elemento diferenciador de Aedo sería la posibilidad de que cualquier usuario pueda publicar actividades, y que no se restrinja la creación de estas a la empresa propietaria de la aplicación. Se ha considerado, que esta es la puesta en valor del proyecto respecto a la competencia, y lo que puede favorecer más a los objetivos de desarrollo sostenible perseguidos por la aplicación.

2.4 Modelo de negocio

Existen distintos modelos de negocio que se podrían aplicar en Aedo pero, al tratarse de una aplicación móvil, algunos de ellos no serían bien recibidos. Es por esa razón que, de antemano, el modelo de negocio tradicional, basado en la venta de licencias en la que se adquiere el producto por un precio y se dispone de total libertad para utilizarlo, no encaja con la aplicación. La razón por la que se descarta es que Aedo necesita de una amplia base de datos de usuarios para poder ofertar experiencias por parte de las personas que quieren ofrecerlas y de clientes que asistan a dichas experiencias. Si los/as usuarios/as no utilizan la aplicación, no hay tránsito y por tanto la aplicación no aumentará el número de usuarios activos. Muchos de estos usuarios rehusarían de la aplicación si tienen que pagar por ella, además que probablemente otra empresa podría vislumbrar el potencial de la aplicación y la implementaría con otro modelo de negocio.

Es por esta razón que la opción que se considera acertada para el proyecto es un modelo *freemium*, en el que los usuarios pueden descargar el *software* de manera gratuita y utilizarlo para, posteriormente, adquirir los servicios *premmium* si así lo desean. Entre las distintas posibilidades de opciones de compra se han pensado las siguientes:

- **Suscripción *premmium* para los ofertantes**: Se trata de un servicio para los/as usuario/as que ofertan experiencias. Mediante este servicio se podrían posicionar las experiencias en el buscador de otra manera. A su vez, permitiría añadir más imágenes a la hora de crear las experiencias para que los usuarios puedan tener más información sobre lo que se van a encontrar. Además, desaparecen los anuncios publicitarios.
- **Suscripción *premmium* para los clientes**: Un cliente *premmium* no tiene anuncios publicitarios en su cliente móvil y podría tener prioridad a la hora de acceder a una experiencia para un día dado si dicha experiencia tiene el cupo de usuarios completo. Un usuario *premmium* se beneficia de prioridad en la cola. También se anularía la publicidad para este tipo de usuarios.

- **Packs de personalización**: Es común en las aplicaciones dar la posibilidad a los/as usuarios/as de adquirir elementos para su perfil de usuario, que son visibles des de la aplicación y los distinguen de otros usuarios que no han adquirido la membresía *premmium* como pudiera ser un icono o alguna insignia.

Por otra parte, también se consideraría como fuentes externas de ingresos la implementación de anuncios publicitarios en la aplicación y una pequeña comisión del precio de las experiencias ofertadas en la plataforma cada vez que un cliente se apunta a las mismas.

2.5 Proyección económica a 6 años

Para realizar la estimación de ingresos y gastos durante los seis primeros años, se ha dividido este período de tiempo en doce semestres. Se parte de una pequeña cantidad de usuarios para el primer semestre estimada en 500 hasta llegar a 60.000 usuarios activos en el semestre doceavo.

La **Figura 9**, muestra la proyección económica preestablecida para el proyecto en cuestión, mostrando los diferentes gastos desglosados y los previstos ingresos según la fuente proveniente.

Por una parte, los **ingresos** provienen, como se ha explicado en el apartado de **modelo de negocio** de este documento, de distintas fuentes a ser las suscripciones *premmium* para ofertantes y clientes, los packs de personalización, las comisiones de las experiencias y la publicidad. Cabe destacar que no se prevén ingresos publicitarios hasta el semestre décimo momento en el cual ya se dispone de una base de usuarios activos suficiente como para ser rentable que los anunciantes publiciten sus productos. Es por esta razón que resulta tan complicado adquirir ese punto de equilibrio en el que se empiezan a generar beneficios.

Por otra parte, los **gastos** se deben a distintos servicios de *hosting* necesarios para el funcionamiento de la aplicación, a la adquisición de los dispositivos necesarios para realizar el desarrollo del *software* y los salarios de los trabajadores, además de campañas de *marketing*.

Analizando la **Figura 9** se puede apreciar que el primer año los gastos van a ser constantes durante los dos semestres y los ingresos van a ser prácticamente nulos debido a la carencia de usuarios, es por ello por lo que se optará por una campaña de publicidad más agresiva de cara al segundo año.

En el segundo año, el número de usuarios se incrementa, y con ello los ingresos. Los gastos también aumentarán debido a los costes de gestoría y a la campaña publicitaria. Dicha campaña publicitaria se refleja en el tercer año, en el que se multiplica por cinco la cantidad de usuarios respecto a inicios del segundo año. Esto permite aumentar aún más los ingresos, pero a su vez se requerirá de la contratación de un desarrollador senior.

A partir de este momento, cuarto año, se prevé que junto a campañas publicitarias y el boca a boca de los usuarios el número de usuarios activos seguirá aumentando y en este momento

la empresa empezará a generar beneficios semestrales, pero no será hasta el quinto año donde se equilibren los gastos y beneficios.

Usuarios activos	500	1000	2000	6000	10000	15000	20000	30000	40000	50000	55000	60000
Número de Semestre	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Tipos de suscripciones	**Año 1 / S1**	**Año 1 / S2**	**Año 2 / S1**	**Año 2 / S2**	**Año 3 / S1**	**Año 3 / S2**	**Año 4 / S1**	**Año 4 / S2**	**Año 5 / S1**	**Año 5 / S2**	**Año 6 / S1**	**Año 6 / S2**
Suscripción premium ofertantes	1500	3000	6000	18000	30000	45000	60000	90000	120000	150000	165000	180000
suscripción premium para clientes	200	400	800	2400	4000	6000	8000	12000	16000	20000	22000	24000
Packs de personalización	50	100	200	600	1000	1500	2000	3000	4000	5000	5500	6000
porcentaje odiseas	90	180	360	1080	1800	2700	3600	5400	7200	9000	9900	10800
publicidad										33600	36960	40320
Total Ingresos Semestrales	**1840**	**3680**	**7360**	**22080**	**36800**	**55200**	**73600**	**110400**	**147200**	**217600**	**239360**	**261120**
Gastos Anuales	**Año 1 / S1**	**Año 1 / S2**	**Año 2 / S1**	**Año 2 / S2**	**Año 3 / S1**	**Año 3 / S2**	**Año 4 / S1**	**Año 4 / S2**	**Año 5 / S1**	**Año 5 / S2**	**Año 6 / S1**	**Año 6 / S2**
Servidor BBDD	100 €	200 €	400 €	1.200 €	2.000 €	3.000 €	4.000 €	6.000 €	8.000 €	10.000 €	11.000 €	12.000 €
mobiles IOS / Android	800 €	0 €	0 €	0 €	800 €	0 €						
Compensación teletrabajo	3.600 €	3.600 €	3.600 €	3.600 €	7.200 €	7.200 €	7.200 €	7.200 €	7.200 €	7.200 €	7.200 €	7.200 €
Gestoría	0 €	0 €	1.000 €	1.000 €	2.000 €	2.000 €	2.000 €	2.000 €	2.000 €	2.000 €	2.000 €	2.000 €
Desarrollador Senior Android / IOS	0 €	0 €	0 €	0 €	20.000 €	20.000 €	20.000 €	20.000 €	20.000 €	20.000 €	20.000 €	20.000 €
Desarrollador junior	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €
Desarrollador junior	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €	12.500 €
Tecnico de soporte	0 €	0 €	0 €	0 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €	10.000 €
Cuota seguridad social empresa	300 €	300 €	1.200 €	1.200 €	1.200 €	1.200 €	1.200 €	1.200 €	1.200 €	1.200 €	1.200 €	1.200 €
Campaña marketing	2000	2000	4000	6000	6000	5000	5000	5000	5000	5000	5000	5000
Total Gastos	**31.800 €**	**31.100 €**	**35.200 €**	**38.000 €**	**74.200 €**	**73400**	**74400**	76400	78400	80400	81400	82400
Resultado Semestral	-29.960 €	-27.420 €	-27.840 €	-15.920 €	-37.400 €	-18.200 €	-800 €	**34.000 €**	68.800 €	137.200 €	157.960 €	178.720 €
Resultado Semestral Acumulado	-29.960 €	-57.380 €	-85.220 €	-101.140 €	-138.540 €	-156.740 €	-157.540 €	-123.540 €	-54.740 €	82.460 €	240.420 €	419.140 €

***Figura 9**: Proyección económica*

Como se puede apreciar en la **Figura 9**, la previsión establece un *bypass* semestral entre ingresos y gastos a partir del segundo semestre del cuarto año. Esta situación se generalizaría en el acumulado a partir del segundo semestre del quinto año, obteniendo unos beneficios aproximados en dicho plazo de unos 82.000€. En la tabla también aparecen desglosados los diferentes gastos e ingresos proyectados para dicho proyecto y que se han considerado estándares para un trabajo de esta complejidad.

La **Figura 10** muestra el balance de pérdidas/beneficios acumulados por semestre que se han explicado en el documento. En ella se pueden observar las pérdidas de los primeros semestres hasta llegar al punto de equilibrio en el semestre diez (segundo semestre del quinto año), donde la aplicación empezará a generar beneficios.

Figura 10: *Resultado semestral acumulado*

2.6 Análisis DAFO

La **Tabla 2**, conocida como matriz DAFO, establece las Debilidades, las Amenazas, Fortalezas y Oportunidades que se han observado para el proyecto. En este caso, se debe destacar la poca experiencia del equipo, equilibrada por el potencial del proyecto para diferentes sectores de mecenazgo de la sociedad (instituciones, organizaciones etc.) Además, se ha considerado positivo el hecho de contar con un equipo reducido para potenciar la facilidad en la toma de decisiones.

Tabla 2: *Matriz DAFO*

Debilidades	**Amenazas**
• Equipo junior con poca experiencia. • Poca financiación en el proyecto inicial. • Equipo reducido. • Dirección estratégica inicial poco establecida.	• Entrada de empresas competidoras con mayor financiación. • Cambios demográficos intensos. • Crecimiento lento del mercado. • Futuros cambios en la política económica de la empresa. • Políticas adversas en otros países.
Fortalezas	**Oportunidades**
• Nicho con poca competencia. • Funcionalidad novedosa. • Capacidad de decisión limitada a un equipo pequeño. • Políticas afines. • Basado en ODS con posible financiación europea. • Flexibilidad organizativa. • Equipo joven con ideas modernas y atractivas.	• Afinidad con diferentes sectores de la población. • Afinidades políticas con instituciones. • Aprovechamiento del potencial turístico del país. • Políticas territoriales alineadas con la visión de la empresa. • Financiación a nivel estatal y europea.

	• Crecimiento rápido de la popularidad de la aplicación.

2.7 Conclusiones de la evaluación

A grandes rasgos, el presente documento ha pretendido mostrar la factibilidad de un proyecto con un gran nicho de mercado, con gran potencial y elementos distintivos, que, a pesar de requerir de un presupuesto inicial elevado, debido a la complejidad del mismo, el riesgo que ello supone es limitado en relación con las posibles oportunidades observadas.

El hecho de aunar en una aplicación diferentes características tan atractivas para diferentes sectores de la sociedad, como actividades de ocio, tiempo libre, medio ambiente, puesta en valor de tradiciones, cultura, etc.., se ha visto como un potente aliciente por parte del equipo para confiar en la buenaventura del proyecto y el compromiso con el mismo.

3 Desarrollo de la idea de negocio

A continuación se muestra el mapa de características de la aplicación.

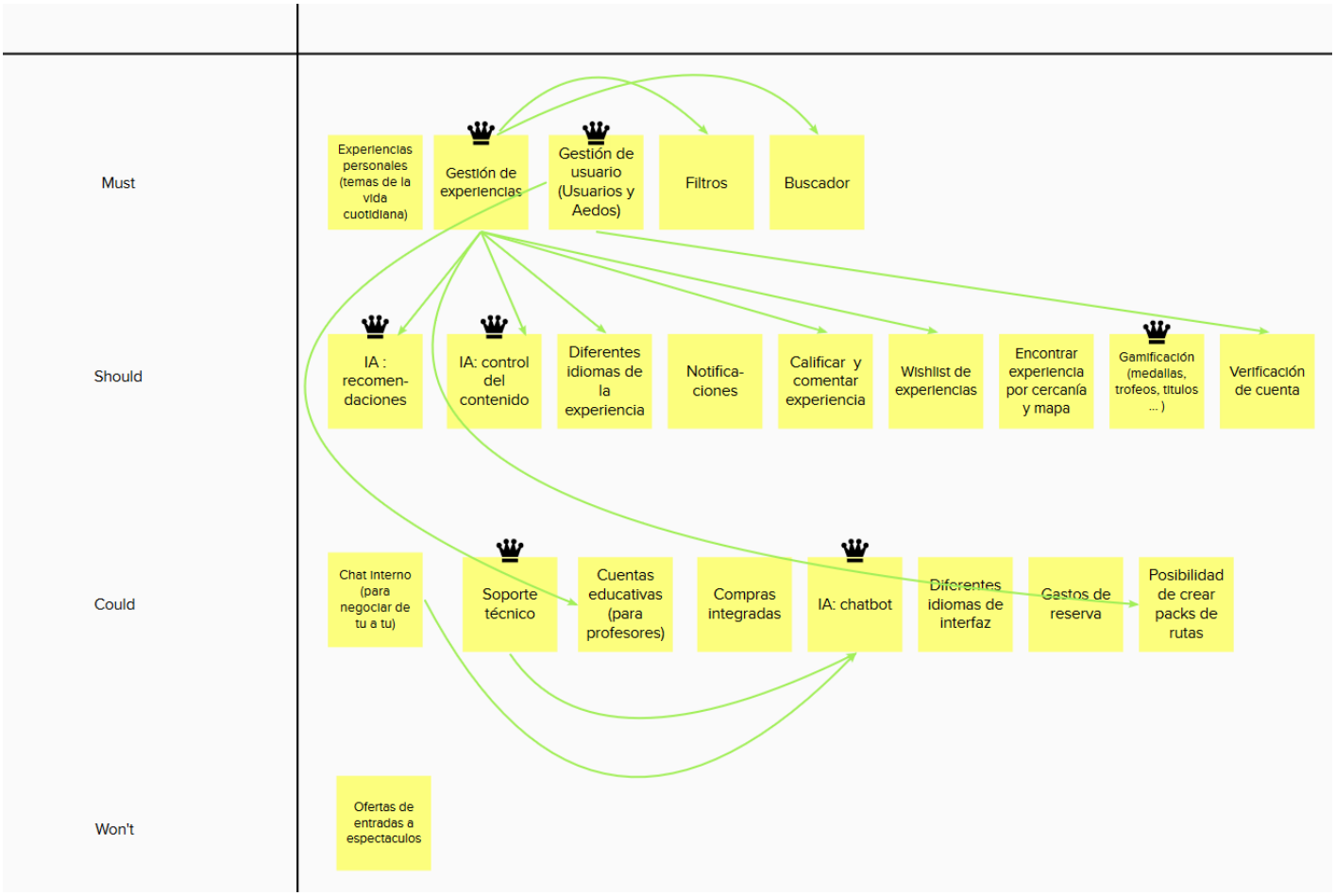

Figura 11: *Mapa de características inicial*

En esta figura podemos observar los diferentes elementos que se han considerado importantes a implementar en la aplicación durante su desarrollo hasta consolidarse como un producto maduro. También se ha realizado un análisis MOSCOW (Kuhn, 2009) para priorizar las diferentes características, basándose en criterios de funcionalidad y usabilidad.

Destacar funciones básicas como la gestión de experiencias o usuarios, que establecen la base de la aplicación, u otras más avanzadas como la IA de recomendaciones o de control de contenido, que darían un toque distintivo a la misma, mejorando la experiencia de uso de los usuarios.

3.1 Desarrollo del primer MVP

El desarrollo de la aplicación Aedo empezó en el marco de la asignatura de Proyecto de Ingeniería del Software, con una metodología preestablecida por el profesorado de esta. En ese ámbito se utilizó la herramienta *Worki*[6] para gestionar el esfuerzo dedicado al proyecto y el trabajo individual de cada miembro. Se realizaron 3 *Sprints* de un mes aproximadamente, en el que se

[6] Worki: https://cliente.tuneupprocess.com/web/

implementaron varias funcionalidades básicas del proyecto para posteriormente presentarlo en la Feria de Proyectos de la ETSINF. La elección de dichas funcionalidades se basó en un criterio MVP según las características que se consideraron básicas de la aplicación según la **Figura 11**.

El diseño de la aplicación fue desarrollado por el grupo de estudiantes de Bellas Artes integrados en el grupo de desarrolladores de software, técnica novedosa llevada a cabo en dicha asignatura para incrementar la cooperación entre estudiantes de diferentes facultades y crear sinergias de trabajo similares a una experiencia laboral real.

3.1.1 Experimento 1: Feria de proyectos

En la Feria de Proyectos se presentó la aplicación, y se estableció contacto con empresas y otros desarrolladores que la analizaron y opinaron sobre sus funcionalidades. De esta experiencia se extrajo el aprendizaje de la importancia de la viralización en una aplicación de este tipo y de las diferentes posibilidades de realizar experiencias en grupo, por personas desconocidas entre sí, y que decidieran conocerse realizando dicha actividad.

***Figura 12**: Fotografía del stand y el equipo de Aedo (Fuente: Fotografía de la Etsinf)*

3.2 Desarrollo del segundo MVP

El segundo *MVP* estuvo enfocado a aumentar las funcionalidades de la aplicación, elaborar una inteligencia artificial para las recomendaciones de experiencias, elaborar la página web de la aplicación para las gestiones administrativas y tener una versión básica de la aplicación en una versión de navegador. De entre las funcionalidades implementadas, cabría destacar el apartado educativo de la aplicación. Tras el segundo *MVP*, la aplicación permitía disponer de cuentas educativas y de generar experiencias colectivas para clases de alumnos enteras. También se incluyeron mejoras de interfaz y funcionalidades para conocer, de una manera más clara, cuánta gente estaba apuntada a una experiencia, mejoras en el perfil, categorías extra, etc.

Para el segundo y tercer experimento, tras tener la aplicación con las últimas funcionalidades implementadas, se presentó la aplicación a dos perfiles diferentes, pero a su vez enfocados en la creación de experiencias.

3.2.1 Experimento 2: Presentar la aplicación a un trabajador del campo.

Con el objetivo de conocer la opinión de un potencial creador de contenido de la aplicación, se contactó con un agricultor interesado en realizar talleres con personas aficionadas a las labores del campo. En este caso, la persona seleccionada era un conocido de uno de los miembros del equipo de desarrollo, con un perfil joven e interesado en la temática de la aplicación, con intención real de aportar contenido a la misma.

Teniendo en cuenta, que uno de los objetivos de desarrollo sostenible defendidos por la cultura de la aplicación era el de "Ciudades y Comunidades Sostenibles", se consideró un posible usuario muy apropiado para los intereses del proyecto.

De esta forma, se le presentó la aplicación y se le realizaron una serie de preguntas orientadas a conocer, de primera mano, los puntos positivos y negativos que un usuario primerizo podría observar de la aplicación. Las preguntas se realizaron verbalmente *in situ* al interesado, no obstante, también se le pidió que contestara a las preguntas de un cuestionario *online* a través de un enlace para tener el registro de las respuestas. El Anexo 2 recoge las preguntas del formulario y el Anexo 3 fotos del experimento. El resultado fue satisfactorio, destacando, por parte del encuestado, la interfaz intuitiva y la facilidad de uso. De igual forma, se recalcó la importancia de disponer de un chat de usuarios en el que poder responder a las dudas y cuestiones que le realizaran los posibles clientes. Dicha funcionalidad ya había sido valorada por el equipo de desarrollo, pero no se ha dispuesto de tiempo suficiente para presentarla en la presente versión.

3.2.2 Experimento 3: Presentar la aplicación a un profesor

Ya que uno de los objetivos perseguidos durante la segunda fase del desarrollo de la aplicación fue la implementación de un tipo de cuentas educativas, se consideró oportuno contactar con un profesor de secundaria para conocer su opinión al respecto. La elección de dicha persona vino condicionada por ser excompañero de clase de unos de los miembros del equipo y ser una persona joven con facilidad en el uso de tecnologías.

Se le explicó el proyecto y se le presentaron las funcionalidades de este, destacando el uso de las cuentas educativas y las opciones para reservar actividades para grupos escolares. Ya que el proyecto pretende poner en valor las tradiciones y la cultura, el encuestado recalcó la importancia de este tipo de actividades dentro del ámbito educativo de los centros de secundaria. También sugirió la posibilidad de exportar las descripciones de las actividades a pdf para poder compartirlas con los padres de los alumnos, y firmar los permisos para que los alumnos las realicen. El equipo de desarrollo valoró dicha opción, y se tiene pensado implementar un sistema de exportación a diferentes formatos y de compartir las actividades a través de aplicaciones de terceros en futuras versiones.

Se le facilitó al docente un cuestionario *online* (ver Anexo 2) para conocer su opinión y registrar sus posibles sugerencias. Las imágenes del experimento pueden verse en el Anexo 3.

4 Aspectos técnicos: Desarrollo front-end móvil

Se debe hacer una consideración inicial antes de empezar con la parte específica del *front-end*. Como ya se ha comentado, el proyecto de Aedo nació en la asignatura de proyecto de ingeniería del software (PIN). Fue un proyecto multidisciplinar entre la escuela de informática y la escuela de bellas artes, esto es importante tenerlo presente de cara a la consideración.

Fue decidido por el equipo de programadores utilizar una terminología basada en la antigua Grecia para la aplicación de Aedo, es por ello que los usuarios mencionados hasta este momento en el documento, en el código se les conoce como odiseos y las experiencias en el código aparecerán como odiseas. Por tanto, siempre que aparezca la palabra odiseo en este presente documento, se refiere a los usuarios de la aplicación (los que no son aedos, los aedos son los usuarios que ofrecen las experiencias al resto de usuarios) y cuando aparezca odisea, realmente se refiere a las experiencias a las que los usuarios se inscriben.

Esta discrepancia de nombres tiene un motivo: A la hora de presentar el proyecto en la Feria de Proyectos se mostró la aplicación a los alumnos de bellas artes con unas semanas de antelación y se propuso cambiar el nombre de odisea por algo más amigable ya que la palabra odisea se relaciona con algo muy complicado o imposible de hacer y puede dar una mala impresión al usuario sobre a qué actividad se están inscribiendo. Como se tuvo que cambiar la palabra odisea, no tenía sentido mantener la palabra odiseo para los usuarios, sin embargo, toda la terminología interna de la aplicación (la que no es visible por el usuario), como puedan ser los nombres de ventanas y componentes, las APIs, los modelos y los objetos, siguen manteniendo esta terminología antigua de odiseos y odiseas.

4.1 Modelo de dominio

En la **Figura 13** se puede ver representado el modelo de dominio del proyecto tras los cambios realizados con el equipo de diseño, protagonizado por las Experiencias como pilar fundamental de la aplicación, con una serie de relaciones con los demás elementos de este. Este apartado es importante remarcarlo en este TFG de *front-end* para entender los conceptos y entidades clave del dominio. Como se puede observar, todos los elementos propios de una aplicación de este tipo estarían representados, destacando las reservas, los comentarios, los usuarios, las fechas disponibles, etc.

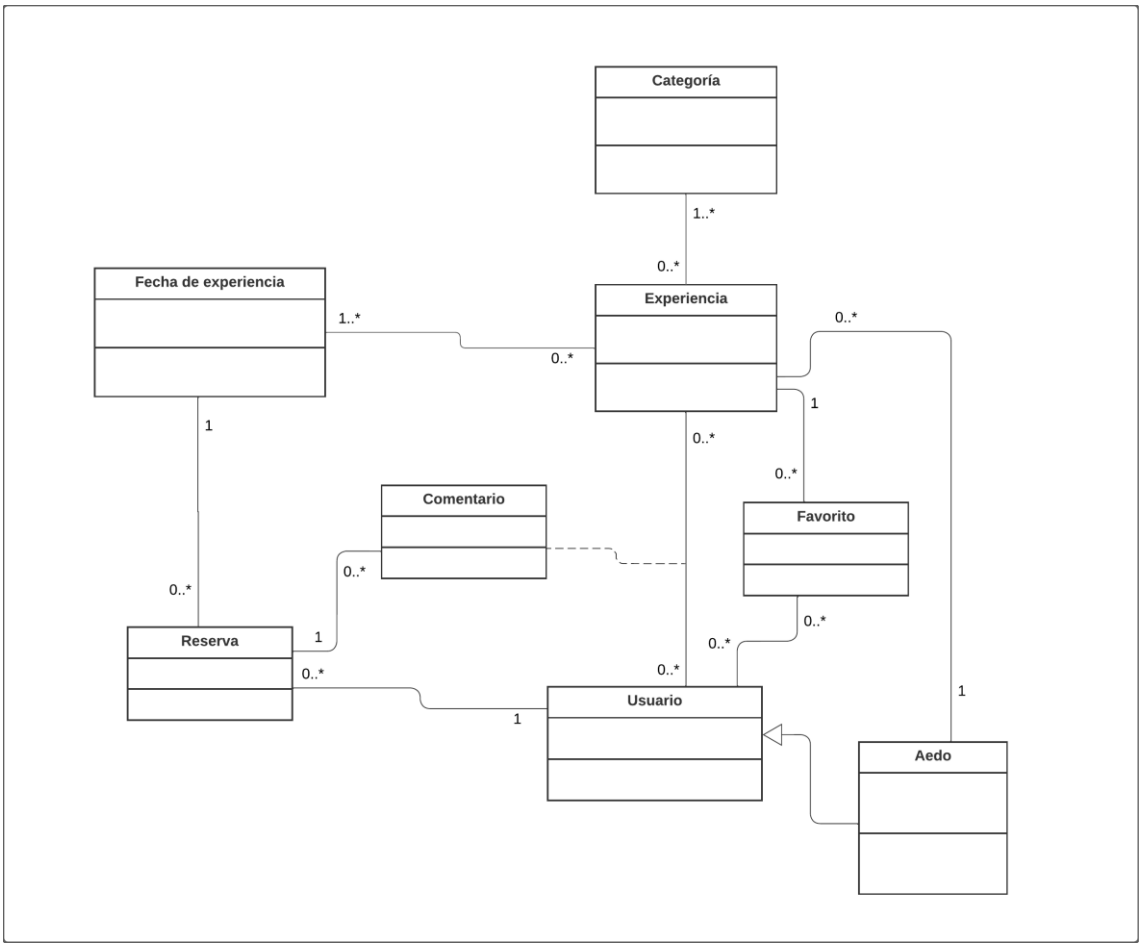

***Figura 13**: Modelo de dominio Aedo*

4.2 Herramientas utilizadas

En este proyecto móvil se ha elegido la tecnología de React Native para la realización del *front-end* móvil. React Native (*¿Qué es React Native?*, s. f.) es un *framework* de programación de aplicaciones móviles multiplataforma que está basado en JavaScript y React.js. Permite construir aplicaciones móviles utilizando solamente JavaScript y React. Utiliza el mismo diseño que React.js, permitiendo usar elementos de interfaz de usuario móvil. Se ha trabajado con Expo, una plataforma de desarrollo de aplicaciones móviles que permite a los desarrolladores crear aplicaciones nativas y multiplataforma con la tecnología de React Native.

En cuanto al control de versiones se ha utilizado GIT, en el cual existe una rama *main* donde se encuentra la última versión de la aplicación y existen ramas para cada requisito que se realice.

En este proyecto se ha utilizado un *linter, una* herramienta encargada de la mejora del código identificando fallos o problemas en el sistema, en concreto SonarLint[7], para mejorar la calidad y mantenibilidad del código. Además, se ha trabajado juntamente con SonarQube[8] para observar la deuda técnica y la categoría de fallos obtenidos.

[7] SonarLint: https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarlint-vscode

[8] SonarQube: https://docs.sonarqube.org/latest/

Para la gestión del proyecto con el equipo se han utilizado dos herramientas distintas. Para los cuatro primeros *sprints* se utilizó Worki[9], y para los siguientes se ha usado Jira[10]. De esta forma se han experimentado diferentes formas de trabajar y organizar los proyectos.

4.3 Entornos de desarrollo

Todo el proyecto *front-end* móvil se ha trabajado con el entorno de desarrollo VS code de Microsoft, con acceso a múltiples librerías que facilitan la mantenibilidad del código y facilitan el trabajo en equipo. Algunas de ellas son:

SonarLint: Proporciona análisis estático del código en tiempo real. Ayuda a identificar problemas de calidad del código, detectar errores y aplicar las mejores prácticas de codificación. Además, ofrece sugerencias y recomendaciones para mejorar la calidad y la legibilidad del código.

Live Share: Permite la colaboración en tiempo real entre desarrolladores. Con Live Share, varios desarrolladores pueden trabajar en el mismo proyecto al mismo tiempo, incluso si están utilizando diferentes sistemas operativos o lenguajes de programación. Permite compartir código, depurar de forma colaborativa, editar en conjunto y realizar reuniones en línea.

Prettier: Es una herramienta de formateo de código que ayuda a mantener una consistencia en el estilo de código en un proyecto. La extensión de Prettier para Visual Studio Code automatiza la tarea de formatear el código según las reglas de estilo establecidas.

Simple React Snippets: Simple React Snippets es una librería de código abreviado (snippets) para el desarrollo de aplicaciones React en Visual Studio Code. Proporciona fragmentos de código predefinidos que permiten generar rápidamente estructuras comunes de React, como componentes, propiedades y métodos. Esto acelera el proceso de desarrollo al reducir la necesidad de escribir código repetitivo.

Todo Tree: Ayuda a gestionar las tareas pendientes dentro del código fuente. Permite buscar y resaltar comentarios etiquetados con palabras clave como "TODO" o "FIXME" en el código, lo que facilita la identificación y seguimiento de tareas o mejoras pendientes.

Para trabajar con la emulación de un dispositivo Android se ha trabajado con Android Studio. Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones móviles Android. Proporciona un conjunto completo de herramientas y recursos para ayudar a los desarrolladores a crear, depurar y emular aplicaciones Android.

4.4 Estilo artístico de la aplicación

Para explicar el estilo artístico elegido por el equipo de Aedo es importante entender el contexto y a quien va dirigida la aplicación. Como se ha mencionado anteriormente, Aedo es una aplicación que tiene como uno de sus principales objetivos dar visibilidad a los pueblos y culturas de España.

[9] Worki: https://cliente.tuneupprocess.com/web
[10] Jira: https://www.atlassian.com/es/software/jira

Un claro ejemplo son las ilustraciones creadas por el equipo de diseño referentes a la cultura de diferentes puntos de España. Otro ejemplo es el icono de la aplicación (ver **Figura 14**), que consiste en la visión que tiene el equipo del Aedo, término proveniente del castellano antiguo que representa los poetas y cuenta cuentos de la Grecia antigua, el cual ha sido diseñado con inspiración del personaje literario Don Quijote.

***Figura 14**: Icono Aedo*

La tonalidad de la aplicación se basa en tres paletas de colores con tonos verdosos, blancos y rojos. Jugando con estas únicas tonalidades se consigue crear una interfaz atractiva para el usuario y darle personalidad.

Diferentes tipos de estilos de texto han sido escogidos y serán representados en los estilos globales de la aplicación.

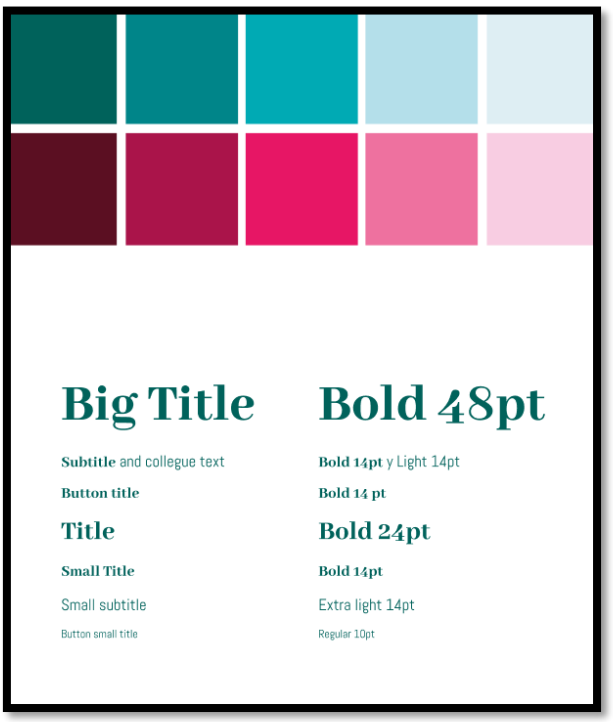

***Figura 15**: Gama de colores y tipografías utilizadas*

Se ha optado por disponer de la barra inferior (*bottomBar*) con únicamente las opciones más usadas por los usuarios (ver **Figura 16**). A continuación, se detallan las principales ventanas:

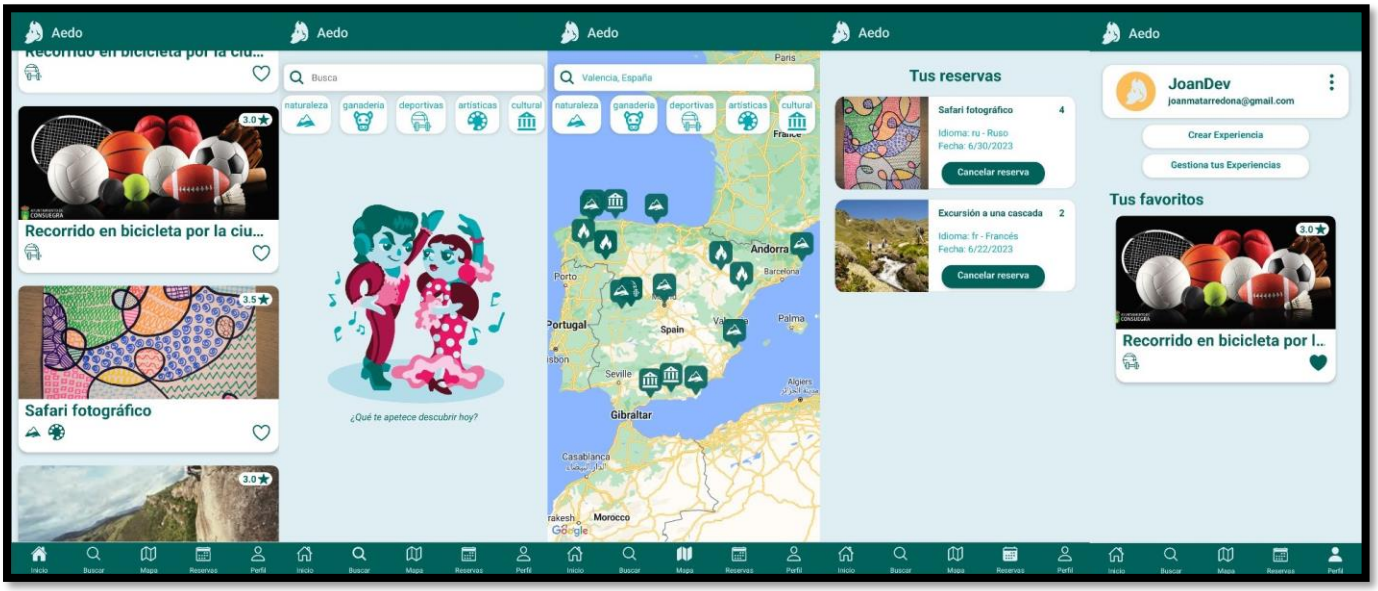

***Figura 16**: Ventanas de la barra de navegación*

Ventana Inicio: Esta es la ventana principal, es el primer contacto del usuario con la aplicación. En esta se encuentra un variado de experiencias de las que el usuario puede estar interesado y un apartado de recomendaciones que realiza la aplicación en función de los gustos previos del usuario.

Ventana Buscar: En esta ventana el usuario puede hacer uso de la búsqueda de experiencias por ciudad o utilizar los filtros de categorías. Por defecto esta ventana mostrará las experiencias cercanas al usuario.

Ventana Mapa: Esta ventana tiene el mismo fin que la del Buscar , es decir, encontrar experiencias, pero aquí se agrega un mapa donde se pueden visualizar distintos marcadores, cada uno indicando una de las categorías que puede tener una experiencia, que podrían ser por ejemplo: agricultura, deportes, música, etc, cada una con un icono distinto. Además, tiene los filtros de buscar por ciudad o por tipo de categoría.

Ventana Reservas: Esta ventana permite gestionar las reservas realizadas de las experiencias por los usuarios.

Ventana Perfil: Aquí el usuario puede iniciar sesión o registrarse. Una vez el usuario tenga la sesión iniciada se pueden visualizar los datos del perfil, modificarlos, cerrar la sesión, crear una experiencia o gestionar las ya creadas. Además, se pueden observar qué experiencias están marcadas como favoritas por el usuario.

4.5 ¿Qué es React Native?

React Native es un *framework* de JavaScript para crear aplicaciones móviles nativas para iOS y Android. Está basado en la biblioteca de JavaScript React para la creación de componentes

visuales, desarrollada por Meta, la anterior Facebook. React Native permite a los desarrolladores crear aplicaciones móviles nativas utilizando JavaScript y React.

Al utilizar React Native, los desarrolladores pueden crear aplicaciones móviles nativas utilizando un solo código base. Esto significa que los desarrolladores pueden crear aplicaciones móviles nativas tanto para IOS como para Android utilizando el mismo código base y de esta forma facilitando el desarrollo de las aplicaciones multiplataforma.

React Native es una herramienta muy popular entre los desarrolladores y que tiene mucho apoyo por parte de ellos para crear librerías útiles para el *front-end*. Las librerías son una herramienta importante para el desarrollo de aplicaciones móviles con React Native, ya que permiten a los desarrolladores reutilizar código y evitar escribir código desde cero. Al instalar una librería, los desarrolladores pueden agregar nuevas funcionalidades a su aplicación de manera eficiente, lo que les permite ahorrar tiempo y recursos.

La plataforma de React Native dispone de una amplia cantidad de documentación que proporciona información acerca de los componentes básicos y su funcionamiento. Además, la comunidad de desarrolladores ha generado una cantidad considerable de documentación en torno a las librerías que ofrece recursos y conocimientos adicionales para la utilización de estas herramientas. En conjunto, la documentación de la plataforma y de la comunidad conforman un conjunto de recursos valiosos que contribuyen al conocimiento y uso efectivo de las librerías en el contexto de React Native.

4.6 Alternativas a React Native

Existen varias alternativas a React Native que también ofrecen características similares y un desarrollo multiplataforma. En esta sección se comparan tres alternativas a React Native: Flutter, MAUI y Ionic.

Flutter (*Flutter documentation*, s. f.) es un *software* de desarrollo de aplicaciones móviles desarrollado por Google. Flutter utiliza un lenguaje de programación multiplataforma llamado Dart. Uno de los puntos fuertes de Flutter es su capacidad para crear interfaces de usuario personalizadas y animadas, gracias a su motor de renderizado de gráficos en 2D y 3D. Su principal ventaja radica en que genera código 100% nativo para cada plataforma, con lo que el rendimiento y la UX es totalmente idéntico a las aplicaciones nativas tradicionales.

.NET Maui (Davidbritch, 2023) es una plataforma de desarrollo móvil propiedad de Microsoft. A diferencia de Flutter y React Native, Maui usa el lenguaje de programación C#. Maui permite a los desarrolladores escribir una sola base de código para aplicaciones móviles para iOS, Android y Windows, y también puede utilizar las API nativas de cada plataforma para acceder a características específicas. Otra ventaja de Maui es su integración con Visual Studio, el entorno de desarrollo integrado de Microsoft, que puede ser útil para los desarrolladores que ya están familiarizados con ese entorno. Sin embargo, Maui puede tener una curva de aprendizaje más empinada que React Native y a veces las aplicaciones desarrolladas en Maui pueden tener un proceso de carga más lento.

Ionic es un framework de desarrollo (*Introduction to Ionic | Ionic Documentation*, s. f.) de aplicaciones móviles basado en Angular, un *framework* popular de desarrollo web. Ionic usa

HTML, CSS y JavaScript para crear aplicaciones móviles multiplataforma para iOS y Android. Una de las ventajas de Ionic es que los desarrolladores pueden utilizar su experiencia en Angular para crear aplicaciones móviles, lo que puede acelerar el proceso de desarrollo. Sin embargo, Ionic se basa en tecnologías web, lo que puede afectar el rendimiento de la aplicación en comparación con React Native o Flutter.

En conclusión, cada alternativa a React Native tiene sus pros y sus contras. Flutter es una buena opción para los desarrolladores que quieren crear interfaces de usuario personalizadas y animadas, pero puede tener una curva de aprendizaje más empinada. Xamarin es una buena opción para los desarrolladores que ya conocen C# y Visual Studio, pero puede ser más difícil de aprender. Ionic es una buena opción para los desarrolladores que ya están familiarizados con Angular, pero puede tener un rendimiento inferior en comparación con otras opciones.

React Native (*React Native · Learn once, write anywhere*, s. f.) sigue siendo una opción popular para el desarrollo de aplicaciones móviles multiplataforma. Con una gran comunidad de desarrolladores y una amplia gama de recursos disponibles, React Native ofrece una curva de aprendizaje relativamente suave y una gran flexibilidad para los desarrolladores. También cuenta con una amplia variedad de bibliotecas y herramientas que pueden simplificar el proceso de desarrollo y aumentar la eficiencia del equipo de desarrollo.

4.7 Atomic design

Antes de empezar a explicar la estructura utilizada para organizar todos los elementos del *front-end* es importante explicar el patrón básico del cual se centra toda la aplicación.

Como se ha mencionado anteriormente, los componentes en React son elementos básicos de la interfaz de usuario (UI) que se utilizan para construir aplicaciones. Funcionan como bloques de construcción reutilizables que permiten crear una experiencia de usuario coherente en toda la aplicación. Los componentes de React están diseñados para ser flexibles y se pueden personalizar para adaptarse a las necesidades específicas de cada aplicación. La ventaja de utilizar componentes en React Native es que permiten la reutilización de código ya que permiten ser compartidos entre diferentes partes de la aplicación y con otras aplicaciones.

Atomic Design (*Atomic Design Methodology | Atomic Design by Brad Frost*, s. f.) es una metodología para construir sistemas de diseño efectivos y orientados al medio web o digital. Esta metodología se basa en la idea de que los sistemas vivos son sistemas abiertos en constante interacción con otros sistemas.

En React, Atomic Design se utiliza para construir componentes reutilizables y escalables. Los componentes se dividen en cinco niveles: átomos, moléculas, organismos, plantillas y páginas (ver **Figura 17**).

Los átomos son los bloques más pequeños y simples que componen una interfaz de usuario. Las moléculas son combinaciones de átomos que forman elementos más complejos. Los organismos son combinaciones de moléculas que forman secciones más grandes de una interfaz de usuario. Las plantillas son diseños estructurales que contienen organismos y moléculas. Las páginas son instancias específicas de plantillas que contienen contenido real.

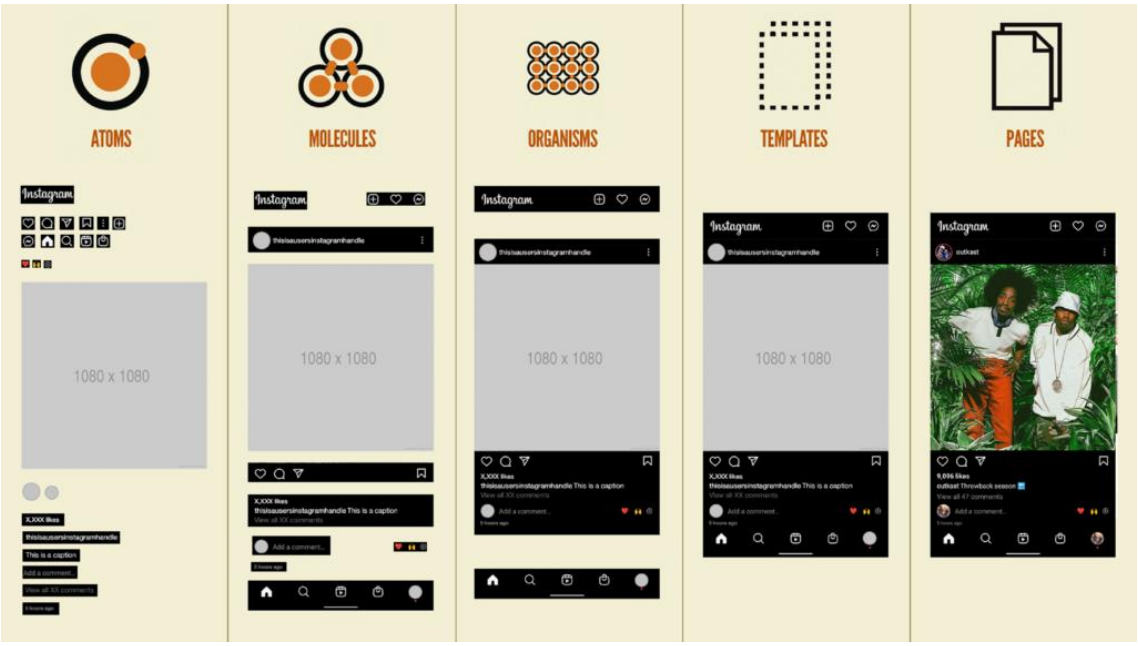

***Figura 17**: Ejemplo de Atomic Design en el desarrollo web (Fuente: https://leruggeri.com/atomic-design-que-es/)*

En la estrategia seguida por el equipo de Aedo para el desarrollo móvil se ha realizado una adaptación de la metodología de diseño atómico. Se ha seguido el enfoque basado en átomos, moléculas y organismos, de la misma forma que en la filosofía del diseño web. Sin embargo, las plantillas y páginas han sido sustituidas por las ventanas. Estas últimas representan la capa más alta en la cual se emplearán los componentes de la estructura atómica del sistema.

Para entender más esta metodología se puede explicar el siguiente ejemplo. Se quiere crear un buscador de ciudades y pueblos de España (Ver **Figura 18**) que contendrá, además, filtros del tipo de experiencia que se quiere buscar en la ciudad elegida.

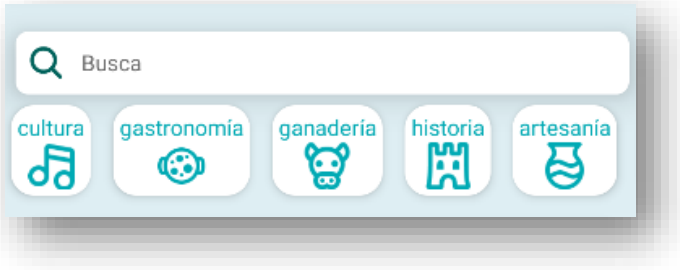

***Figura 18**: Barra de búsqueda con filtros de categorías*

Este componente se va a utilizar en varias ventanas, en concreto en la que contiene al mapa (ventana Mapa) y la que contiene un listado de experiencias (ventana Buscar). Este

componente formaría parte del grupo más alto, el de los organismos. Este organismo estará formado por moléculas: una será el buscador sin filtros, y la otra los filtros (Ver **Figura 19** y **Figura 20**).

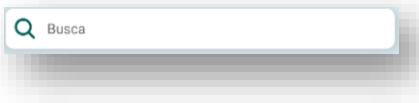

***Figura 19**: Buscador*

***Figura 20**: Filtros*

La implementación de la metodología de diseño atómico en el desarrollo móvil permite reutilizar componentes de manera efectiva y reducir la duplicación de código. Por ejemplo, si se quisiera crear un organismo distinto al que se muestra, pero que contenga el mismo buscador con filtros distintos, se podría reutilizar la molécula del buscador, lo que reduciría la cantidad de código duplicado.

Además, es posible fraccionar aún más el código de manera efectiva. Por ejemplo, el filtro podría estar compuesto por átomos de distintas categorías. De hecho, el átomo utilizado para el filtro es el mismo, pero dependiendo de los parámetros que se le pasen, especialmente el título y la imagen, puede formar una categoría distinta. Esta modularidad en la implementación de componentes atómicos permite una mayor flexibilidad y una mejor gestión del código.

4.8 Estructura de las carpetas

La estructura de carpetas (Jain, 2022) de un proyecto hecho en React Native que se ha utilizado incluye diferentes carpetas que contienen diferentes tipos de archivos y recursos (ver **Figura 21**), cada una de las cuales cumple un propósito específico en el proyecto.

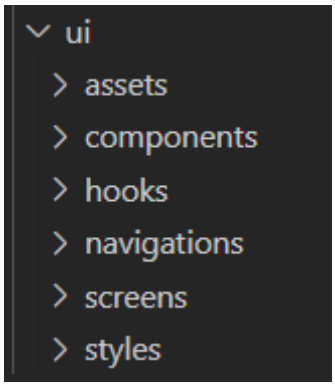

***Figura 21**: Estructura de carpetas del front-end móvil*

La carpeta de *assets* contiene recursos como imágenes, fuentes y otros archivos multimedia que se utilizan en el proyecto.

La carpeta de *components* incluye los componentes reutilizables que se utilizan en diferentes partes del proyecto. Estos componentes pueden ser simples o complejos y se pueden utilizar en diferentes partes de la aplicación para mejorar la modularidad y la reutilización de código. Esta carpeta contiene tres subcarpetas relacionadas con el *Atomic Design*, *atoms*, *molecules* y *organisms*.

La carpeta de *hooks* se utiliza para definir diferentes *hooks* personalizados que se utilizan en el proyecto. Los *hooks* son funciones que permiten a los desarrolladores utilizar el estado y otros métodos de ciclo de vida de los componentes de React en componentes funcionales. Proporcionan una forma de reutilizar la lógica de un componente sin necesidad de utilizar clases o hacer complejas abstracciones.

La carpeta de *navigations* se utiliza para definir la estructura de navegación de la aplicación. En esta carpeta se pueden encontrar diferentes tipos de navegadores, como *Stack*, *Drawer*, *Tab* y otros tipos de navegación que se utilizan para proporcionar una experiencia de navegación fluida y fácil de usar en la aplicación.

La carpeta de *screens* contiene las diferentes pantallas que se utilizan en la aplicación. Cada pantalla se define en un archivo separado y se utiliza para mostrar diferentes tipos de información y funcionalidades de la aplicación.

La carpeta de *styles* se emplea para establecer los estilos generales de la aplicación. En esta carpeta se pueden encontrar diversos tipos de archivos de hojas de estilo, incluyendo hojas de estilo globales que se aplican de manera uniforme a toda la aplicación.

4.9 El interior de un componente

En React, un componente (*Tu primer componente – React*, s. f.) es una unidad básica y reutilizable de una interfaz de usuario. Un componente de React es esencialmente una función o clase que devuelve un elemento React. Están diseñados para ser autónomos y modularizados, lo que permite que sean reutilizados en diferentes partes de una aplicación. Actualmente, las últimas versiones de React recomiendan el uso de componentes funcionales. A dichos componentes funcionales se les pueden pasar datos por medio del objeto props (proveniente de propiedades) y usarlos para renderizar contenido dinámico. También pueden actualizar su estado interno a través de acciones del usuario o de llamadas a funciones, lo que permite que la interfaz de usuario responda a los cambios en tiempo real. Para construir los elementos gráficos React y React Native utilizan JSX.

JSX (*Escribir marcado con JSX – React*, s. f.) es una extensión de sintaxis para JavaScript que permite escribir código que mezcla HTML y JavaScript en un solo archivo. En React la sintaxis es muy similar a la de HTML, sin embargo, en React Native la sintaxis cambia un poco para asemejarse a los entornos de desarrollo móvil.

Un ejemplo de componente en Recta Native básico podría verse en la **Figura 22**.

```
import React from 'react';
import { Text, View } from 'react-native';


export default function App() {
  return (
    <View>
      <Text>¡Hola mundo!</Text>
    </View>
  );
};
```

***Figura 22*: Ejemplo de componente básico en React Native**

En este ejemplo (ver **Figura 22**), la etiqueta <View> se utiliza para definir un contenedor de elementos, mientras que la etiqueta <Text> se utiliza para definir un elemento de texto.

Por ejemplo, un componente simple de React podría verse así (ver **Figura 23**):

```
import React from 'react';


export default function App() {
  return (
    <div>
      <h1>¡Hola mundo!</h1>
  </div>
  );
};
```

***Figura 23*: Ejemplo de componente básico en React**

En ambos casos, el código JSX se compila a código JavaScript puro antes de ser ejecutado en el navegador o en un dispositivo móvil. Esto significa que se pueden utilizar todas las características de JavaScript, como variables, funciones y bucles, dentro de los elementos JSX.

4.10 Buenas prácticas

A continuación, se explican algunas de las buenas prácticas que se pueden aplicar al escribir componentes en React y React Native:

1. Organizar el componente: Es recomendable organizar el componente de manera clara y legible, colocando las diferentes secciones en un orden lógico. Por ejemplo, es común

empezar con los atributos, seguido de los *hooks*, las funciones de lógica y, finalmente, la función de renderizado.

2. Definir variables y funciones antes de la función de renderizado: Es una buena práctica definir las variables y funciones utilizadas en el componente antes de la función de renderizado. De esta manera, se puede separar la lógica del componente de la presentación visual.
3. Utilización de los props: Las props son los datos que se pasan a un componente como argumentos. Es importante obtener las props en el componente y utilizarlas para definir el comportamiento del componente.

A la hora de escribir las funciones variables, etc., se han tenido en cuenta las características del estándar de escritura de JavaScript ES6 (*JavaScript ES6*, s. f.).

4.10.1 ¿Como organizar un componente?

La estructura de un componente en React puede variar dependiendo de las preferencias y necesidades del programador, pero la estructura que se muestra en la **Figura 24** tiene algunas ventajas que pueden ayudar a organizar mejor el código y hacerlo más legible.

A continuación, se explican las secciones por orden de la estructura y su importancia:

1. Atributos: En esta sección se pueden definir los atributos del componente, como los valores iniciales de los estados, las propiedades que recibe, los eventos que maneja, etc. Definir los atributos al principio del componente ayuda a tener una idea clara de las funciones y propiedades que tendrá el componente antes de entrar en detalles con el código.
2. *Hooks*: Los *hooks* son una forma de utilizar el estado y otras características de React en componentes funcionales. En esta sección se pueden declarar los *hooks* que necesite el componente, como *useState*, *useEffect*, *useContext*, etc.
3. Funciones de Lógica: En esta sección se pueden definir las funciones que manejan la lógica del componente, como funciones que actualizan el estado, funciones que hacen peticiones a una API, funciones que procesan los datos, etc. Esta sección ayuda a separar la lógica del componente de la estructura del mismo y a hacer el código más modular y fácil de mantener.
4. Funciones de Renderizado: Esta sección incluye la función de retorno del componente, que devuelve el JSX que se renderizará en la pantalla.

```
export default function component() {

    // Atributes
    const {value1, value2} = props;
    // Hooks
    const [value, setValue] = useState("");
    // Logic Functions
    const create = () => {

    }

    return (
        <div>
        </div>
    );
}
```

***Figura 24**: Estructura buenas prácticas de un componente*

4.10.2 ¿Como obtener las propiedades?

En React y React Native, los componentes a menudo reciben datos o propiedades (props) de un componente superior o del componente que los está utilizando. Estas propiedades son pasadas como un objeto y se accede a ellas a través del parámetro props en la definición del componente.

La sintaxis *const { value1, value2 } = props* es una forma de desestructurar el objeto props y extraer valores específicos de una forma clara y sin ocupar un exceso de espacio en el código.

4.11 Patrones de diseño de los componentes

4.11.1 Especialización de componentes

En ocasiones, se establece una relación de subordinación entre componentes, considerando algunos como casos particulares de otros. Por ejemplo, en Aedo hay diferentes usos para el mapa; en concreto, en el más usado aparecen los marcadores de las experiencias ya existentes y solo se ofrece la posibilidad de movernos por él sin poder modificar ningún marcador. En cambio, cuando se crea una experiencia, es necesario poder marcar en un mapa donde se encuentra esta. Por ello, para facilitar la reducción de código, se puede utilizar la Especialización de componentes.

En React, esto también se consigue por composición (CloudAPPi, 2022), en la que un componente más específico renderiza uno más genérico y lo configura con props. En este caso el componente *MapComponent* es el genérico y el componente *PointMapComponent* es el específico.

En las siguientes figuras (**Figura 25** y **Figura 26**) se puede observar como el componente *MapComponent* puede recibir una función a través de la propiedad *handleOnPress* y el

componente *PointMapComponent* lo utiliza para cambiar la funcionalidad del componente general y de esta forma poder modificar el mapa añadiendo un marcador.

```
export default function MapComponent(props) {
  const {
    navigation,
    _map,
    markers,
    handleOnPress,
    region,
    setRegion,
    selectedTags,
  } = props;

  const calloutHandler = (odisea) => {···
  };

  return (
    <MapView···
    </MapView>
  );
}
```

Figura 25: *Componente Genérico*

```
export default function PointMapComponent(props) {
  const { selectedCoords, setSelectedCoords, region, setRegion } = props;

  const [markPointer, setMarkPointer] = useState({ loadLastMarkPointer });

  // Carga el marcador anterior si hay
  const loadLastMarkPointer = () => {···
  };

  // Cambia el punto seleccionado
  const handleSelectedMark = (coordinate) => {···
  };

  return (
    <MapComponent
      handleOnPress={(mark) => {
        {
          handleSelectedMark(mark.nativeEvent.coordinate);
        }
      }}
      markers={[markPointer]}
      region={region}
      setRegion={setRegion}
    ></MapComponent>
  );
}
```

***Figura 26**: Componente Específico*

4.12 Documentación de los componentes

Para facilitar la reutilización y la comprensión de los componentes se ha realizado una documentación de los diferentes Átomos, Moléculas y Organismos con JSDoc (*Use JSDoc: Index*, s. f.). JSDoc es una sintaxis para agregar documentación de la API al código fuente de JavaScript. La sintaxis JSDoc es similar a la sintaxis de Javadoc, usado para documentar el código de Java, pero se ha especializado para trabajar con la sintaxis de JavaScript. Se han utilizado estas etiquetas:

@module Etiqueta marca el archivo actual como si fuera su propio módulo

@param Proporciona el nombre, el tipo y la descripción de un parámetro de función.

@example Muestra de un ejemplo de uso del componente.

Un ejemplo de documentación realizada es la **Figura 27** y **Figura 28**: Documentación del componente ProfileImage, consiste en el componente *SearchBar* que hace referencia a la barra de búsqueda.

```
/**
 * Componente que muestra una barra de búsqueda con un icono de lupa y un campo para introducir una ciudad
 * con el componente GooglePlacesCities.
 * @module Molecules/SearchBar
 * @param {Object} props -Las propiedades del componente.
 * @param {Object} props.region - El objeto que contiene la región inicial del mapa.
 * @param {function} props.setRegion - La función que actualiza la región del mapa.
 * @example
 * const region = {
    latitude: 39.479424,
    longitude: -0.343158,
    latitudeDelta: 0.09,
    longitudeDelta: 0.04,
  };
  const setRegion = (region) => {
    //Lógica necesaria
  }
  return (
    <SearchBar region={region} setRegion={setRegion} />
  );
 */
export default function SearchBar(props) {
```

***Figura 27**: Documentación del componente SearchBar*

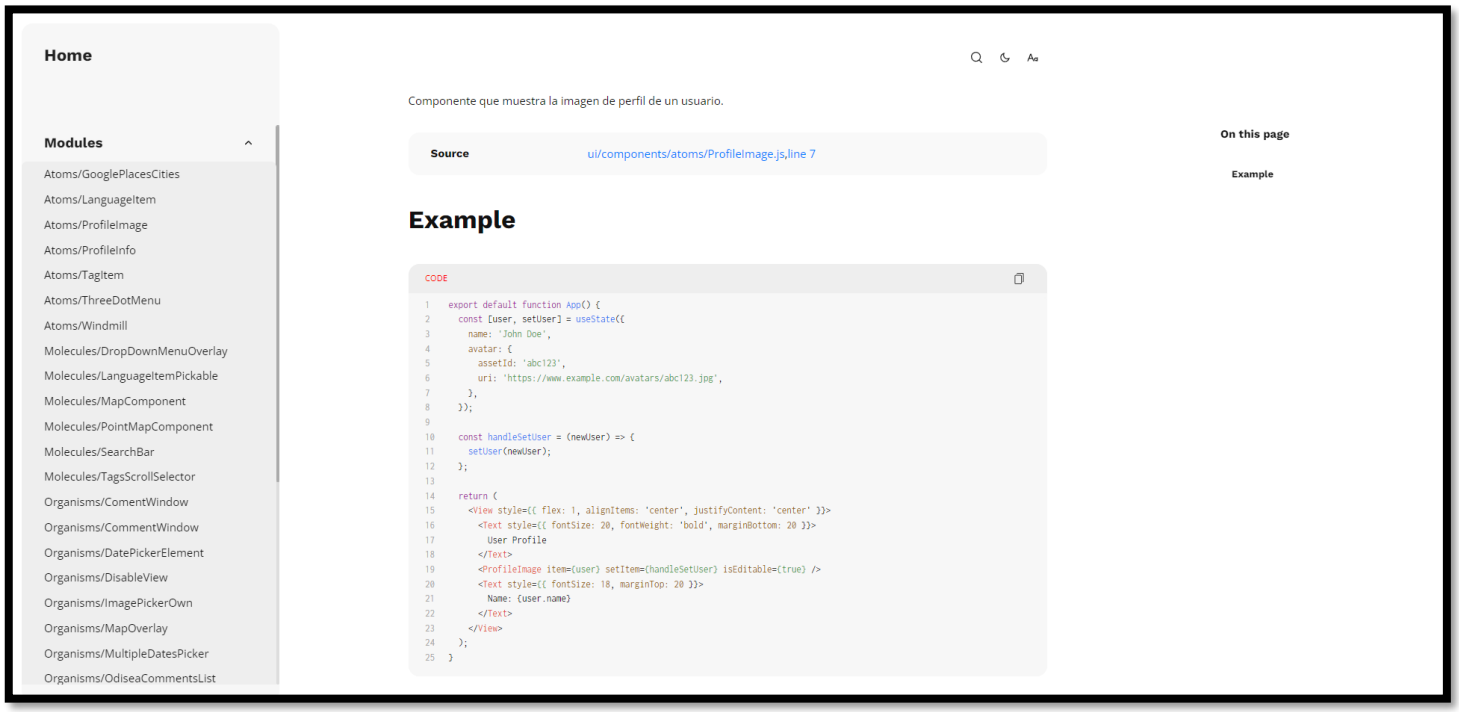

***Figura 28**: Documentación del componente ProfileImage*

4.13 Hooks

4.13.1 ¿Qué es un hook en React?

En React, un *hook* (*Hooks integrados de React – React*, s. f.) es simplemente una función que permite agregar características a un componente de una manera más fácil y organizada.

Un *hook* como *useState* permite añadir estado a un componente de React sin tener que crear una clase de componente. De esta manera, se puede actualizar el estado del componente y React se encargará de actualizar la vista automáticamente.

Supongamos que queremos crear un componente de React que muestre un contador que se puede incrementar haciendo clic en un botón. Podemos usar *useState* para mantener el estado interno del contador. La función *setContador* se utiliza para actualizar el estado del contador cada vez que se hace clic en el botón. La vista del componente se actualizará automáticamente cada vez que se actualice el estado del contador.

Otro ejemplo de *hook* como *useEffect* te permite ejecutar código en respuesta a cambios en el componente, por ejemplo, para realizar operaciones de efecto secundario como hacer una petición HTTP o actualizar el título de la página.

4.13.2 Hooks personalizados

Este *hook* se llama *useLocation* y se encarga de obtener la ubicación del dispositivo y devolverla en un estado de React. Utiliza los *hooks useState* y *useEffect* de React para manejar el estado de la ubicación y para solicitar permisos de ubicación al usuario. El usuario puede proporcionar una ubicación por defecto a través de los parámetros de la función (ver **Figura 29),** de lo contrario se utilizará una ubicación del dispositivo predefinida (ver **Figura 30**).

```
export default function MapOverlay(props) {
  const { item, setItem, visibleMapOverlay, setVisibleMapOverlay } = props;

  const [region, setRegion] = useLocation(item.location);
  const [selectedCoords, setSelectedCoords] = useState(item.location);
```

***Figura 29**: Utilización del hook para obtener introduciendo una ubicación por defecto*

```
export default function GeoLocationScreen({ navigation, route }) {
  const isFocused = useIsFocused([]);
  const [region, setRegion] = useLocation();
  const [markers, setMarkers] = useState([]);
  const [auxMarkers, setAuxMarkers] = useState([]);
  const [tags, setTags] = useState([]);
  const [tagsChanged, setTagsChanged] = useState(false);
```

***Figura 30**: Utilización del hook para obtener ubicación del dispositivo*

El *useEffect* se utiliza para solicitar los permisos de ubicación del usuario y obtener la ubicación actual del dispositivo utilizando la función *getCurrentPositionAsync* de Expo *Location*

(ver **Figura 31**). Si no se proporciona una ubicación por defecto a través de los parámetros, la ubicación actual se utiliza para actualizar la ubicación en el estado de *Location*.

```
useEffect(() => {
  (async () => {
    //Obtiene los permisos de ubicación del usuario
    let { status } = await Location.requestForegroundPermissionsAsync(
      "Always"
    );
    if (status !== "granted") {
      return;
    }

    // Obtiene la localización actual del dispositivo
    let currentPosition = await Location.getCurrentPositionAsync({});

    // Comprueba si hay una localización por defecto introducida
    if (givenLocation == undefined) {
      // Cambia la variable de Estado de la localización del hook
      handleLocationChange(currentPosition.coords);
    }
  })();
}, []);
```

***Figura 31**: UseEffect utilizado para pedir los permisos de ubicación del usuario*

En resumen, el *hook useLocation* proporciona una forma sencilla de obtener y actualizar la ubicación del dispositivo utilizando la biblioteca Expo Location, explicada en el apartado **4.15.3 expo-location**, en una aplicación de React Native.

4.14 UseContext

Durante el desarrollo de la aplicación surgieron problemas con algunos datos que se utilizaban en múltiples ventanas y había que realizar las respectivas llamadas cada vez que fuera necesario, como puede ser todos los datos relacionados con el usuario como el nombre, el correo, etc. Para solucionar este problema, React ofrece una solución llamada *useContext*.

En el contexto de React y React Native, useContext (*useContext – React*, s. f.) es un *hook* que permite acceder y utilizar datos o funcionalidades compartidas desde un componente hijo sin pasar explícitamente esas propiedades a través de la jerarquía de componentes. El *useContext* se utiliza en conjunto con el objeto *Context* de React para proporcionar una forma sencilla de compartir datos entre componentes sin la necesidad de pasar props manualmente en cada nivel.

Antes de utilizar el *Context* es importante mencionar que se debe utilizar de forma moderada porque hace que la reutilización de componentes sea más difícil.

Para la creación de la solución al problema de compartir los datos del usuario para que sean accesibles a través de cualquier componente se ha creado un proveedor de contexto *UserProvider* utilizando *React.createContext() (Ver* **Figura 32***)*. A continuación, se define una función llamada *getUser(),* que utiliza *useContext(UserContext)* para acceder al valor del contexto *UserContext*. La idea es que esta función se puede utilizar en cualquier componente de la aplicación para obtener el usuario autenticado.

Después, se define un componente llamado *UserProvider*. Este componente se encargará de gestionar el estado del usuario autenticado y proporcionar el valor actualizado del usuario a los componentes hijos a través del contexto.

Dentro del componente *UserProvider*, se utiliza *useEffect* para suscribirse al evento *onAuthStateChanged* de *auth*. *Auth* se refiere a una instancia de autenticación proporcionada por la biblioteca de autentificación de Firebase [11]. Esta instancia de autenticación contiene funcionalidades relacionadas con el inicio de sesión y la autenticación de usuarios en la aplicación y en este caso permite obtener el usuario autentificado.

El evento *onAuthStateChanged* se dispara cuando el estado de autenticación del usuario cambia. Dentro del *callback* [12] del evento, se verifica si el usuario está autenticado (*user2*) y se actualiza el estado del usuario utilizando la función *setUser*. Si el usuario no está autenticado, se establece el estado del usuario como *null*.

A continuación, se utiliza *useState* para inicializar el estado del usuario con el valor actual de *auth.currentUser*. Esto proporcionará una referencia inicial al usuario autenticado cuando el componente se monte por primera vez.

Por último, se renderiza el componente *<UserContext.Provider>* y se pasa el valor del usuario como *value*. Esto permitirá que los componentes hijos accedan y consuman el valor del usuario a través del contexto.

[11] Firebase Authentication: Servicio de autenticación proporcionado por Firebase https://firebase.google.com/docs/auth?hl=es-419

[12] Callback: Es una función que se pasa como argumento a otra función y se ejecuta en algún momento posterior o en respuesta a un evento específico.

```jsx
import React, { useContext, useEffect, useState } from "react";
import { auth } from "../../database/firebase";

// Contexto para almacenar la información del usuario
export const UserContext = React.createContext();

// Función para obtener el usuario desde el contexto
export function getUser() {
  return useContext(UserContext);
}

const UserProvider = ({ children }) => {
  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged((user2) => {
      if (user2) {
        // Actualizar el estado del usuario si hay un usuario autenticado
        setUser(user2);
      } else {
        // Establecer el estado del usuario como null si no hay un usuario autenticado
        setUser(null);
      }
    });
    return unsubscribe;
  });

  // Estado para almacenar el usuario actual
  const [user, setUser] = useState(auth.currentUser);

  // Proporcionar el valor del usuario a los componentes hijos envueltos por este proveedor
  return <UserContext.Provider value={user}>{children}</UserContext.Provider>;
};

export default UserProvider;
```

***Figura 32**: Componente UserProvider*

Una vez explicado el funcionamiento interno del componente, para utilizar la funcionalidad explicada se debe utilizar el componente *UserProvider* embolviendo a los componentes hijos que se quiera utilizar la función *getUser()*. En el caso de la **Figura 33** el componente envuelve al hijo *NavigationContainerScreen*, que es un componente el cual contiene todas las ventanas existentes de la aplicación. De esta forma, desde cualquier ventana se puede obtener el usuario autenticado en cualquier momento haciendo uso de la función *getUser()*.

```jsx
export default function App() {
  return (
    <UserProvider>
      <NavigationContainerScreen />
    </UserProvider>
  );
}
```

***Figura 33**: Ejemplo de uso del componente UserProvider*

En resumen, este ejemplo utiliza el contexto de React para proporcionar y acceder al estado del usuario autenticado en diferentes componentes de la aplicación. El componente

UserProvider gestiona el estado del usuario y proporciona el valor actualizado a través del contexto, mientras que la función *getUser()* permite a los componentes hijos acceder al valor del usuario de manera sencilla.

4.15 Librerías importantes front-end

En este apartado se explicarán algunas de las librerías de react-native utilizadas para realizar el *front-end* móvil de Aedo.

Un aspecto importante para destacar es que todas las librerías que se van a mencionar tienen un enfoque multiplataforma, lo que significa que funciona de manera consistente en dispositivos iOS y Android. Lo que facilita considerablemente el desarrollo.

4.15.1 react-navigation

React Navigation (*React Navigation*, s. f.) es una librería ampliamente utilizada en el desarrollo de aplicaciones móviles con React Native. Proporciona una solución eficiente para gestionar la navegación y la transición entre pantallas en una aplicación. Esta librería implementa un enfoque declarativo para la navegación, lo que significa que se define la estructura y el comportamiento de la navegación utilizando componentes de React.

React Navigation ofrece una variedad de navegadores (*navigators*) predefinidos, como Stack Navigator, Tab Navigator y Drawer Navigator, que permiten organizar las pantallas de la aplicación de diferentes maneras. Estos navegadores se encargan de gestionar el historial de navegación, el apilamiento de pantallas y las transiciones entre ellas.

En Aedo se ha utilizado el Tab Navigator para generar una barra inferior de navegación donde se encuentran las funcionalidades más usadas por los usuarios (Ver **Figura 34**).

***Figura 34**: Barra inferior de navegación de Aedo*

Además, se ha utilizado esta librería para navegar entre las diferentes ventanas de la aplicación conectando cada componente con un nombre de ruta.

Existe un componente *NavigationContainer* (Ver **Figura 35**) que es el contenedor principal de la navegación y envuelve todas las pantallas y navegadores de la aplicación. Dentro de él, se encuentra el componente *Stack.Navigator* que define un conjunto de rutas o pantallas disponibles en la aplicación.

A continuación, se definen las diferentes pantallas dentro del *Stack.Navigator* utilizando el componente Stack.Screen (Ver **Figura 36**). Cada *Stack.Screen* representa una pantalla de la aplicación y tiene un nombre y un componente asociado. También se pueden proporcionar opciones de navegación específicas para cada pantalla y establecer la pantalla inicial que en el caso de Aedo es *Home* (Ventana inicio).

```
const Stack = createNativeStackNavigator();

export default function NavigationContainerScreen() {
  return (
    <NavigationContainer>
      <Stack.Navigator ···
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

***Figura 35**: Componente que contiene las rutas a las ventanas*

```
export default function NavigationContainerScreen() {
  return (
    <NavigationContainer>
      <Stack.Navigator
        screenOptions={{ ···
        }}
      >
        <Stack.Screen
          name="Root" ···
          }}
        />
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen
          name="ModifyProfile"
          component={ModifyProfileScreen}
          options={{ title: " Modifica tu perfil " }}
        />
        <Stack.Screen
          name="ModifyPassword"
          component={ModifyPasswordScreen}
          options={{ title: " Modifica tu contraseña " }}
```

***Figura 36**: Ejemplos del componente Stack.Screen*

Una vez que has definido el componente *NavigationContainer* en tu aplicación, puedes utilizarlo en el archivo principal de la aplicación, normalmente llamado App.js (Ver **Figura 37**). De esta forma se establecerá el contexto de navegación para toda la aplicación con la página de inicio definida en la propiedad *initialRouteName* definida en el componente *Stack.Navigator*.

```
export default function App () {
  return (
    <NavigationContainer>
      <AppNavigator />
    </NavigationContainer>
  );
}
```

***Figura 37**: Componente inicial de la aplicación App.js*

4.15.2 expo-image-picker

La librería *expo-image-picker* es una parte integral de la plataforma Expo que proporciona una interfaz fácil de usar para acceder a la galería de imágenes y la cámara del dispositivo móvil, permitiendo a los desarrolladores incorporar funcionalidades relacionadas con la selección y captura de imágenes en sus aplicaciones.

Además, proporciona opciones adicionales como la posibilidad de ajustar la calidad y el tamaño de las imágenes seleccionadas o capturadas, así como la opción de recortar y redimensionar las imágenes.

4.15.3 expo-location

Esta librería de expo permite acceder a la ubicación del dispositivo móvil, brindando a los desarrolladores la capacidad de incorporar funcionalidades basadas en la ubicación en sus aplicaciones.

Ofrece características adicionales, como la capacidad de solicitar permisos de ubicación al usuario, administrar el modo de ahorro de energía y detectar cambios en la configuración de la ubicación del dispositivo.

4.15.4 react-native-maps

La librería *react-native-maps* es una herramienta muy útil en el desarrollo de aplicaciones móviles con React Native, que permite incorporar mapas interactivos en las aplicaciones. Esta librería utiliza los servicios de mapas nativos de cada plataforma (Google Maps para Android y Apple Maps para iOS) y proporciona una interfaz para interactuar con ellos de manera sencilla y eficiente.

Una de las principales ventajas es su capacidad de personalización. Los desarrolladores pueden configurar y ajustar diversos aspectos visuales de los mapas, como el estilo de mapa, el zoom inicial, los controles de navegación, los tipos de marcadores y las animaciones de transición (Ver **Figura 38**). Además, la librería permite superponer capas adicionales en el mapa, como

imágenes personalizadas, polígonos o polilíneas, para adaptarlos a las necesidades específicas del proyecto.

***Figura 38**: Mapa con marcadores personalizados*

4.15.5 react-native-google-places-autocomplete

Esta librería facilita la búsqueda y selección de ubicaciones en tiempo real dentro de la aplicación, mejorando la experiencia del usuario al interactuar con la funcionalidad de búsqueda de lugares.

Esta librería proporciona una interfaz sencilla para implementar un campo de entrada de texto que ofrece sugerencias de ubicaciones a medida que el usuario escribe. Al utilizar la API de Google Places, la librería ofrece resultados de autocompletado precisos y relevantes, lo que permite a los usuarios encontrar rápidamente lugares específicos.

Además, proporciona opciones para personalizar la apariencia y el comportamiento del campo de entrada de texto, como el estilo del componente, los tipos de lugares permitidos y los límites geográficos de búsqueda. Esta personalización se hace a través del atributo *query*.

Es importante destacar que la librería requiere configuraciones adicionales, como la clave de API de Google Places, que se obtiene a través del servicio de desarrolladores de Google.

En Aedo se ha utilizado para poder situar al usuario en una ubicación y así recomendarles experiencias de esa ubicación concreta (Ver **Figura 39**).

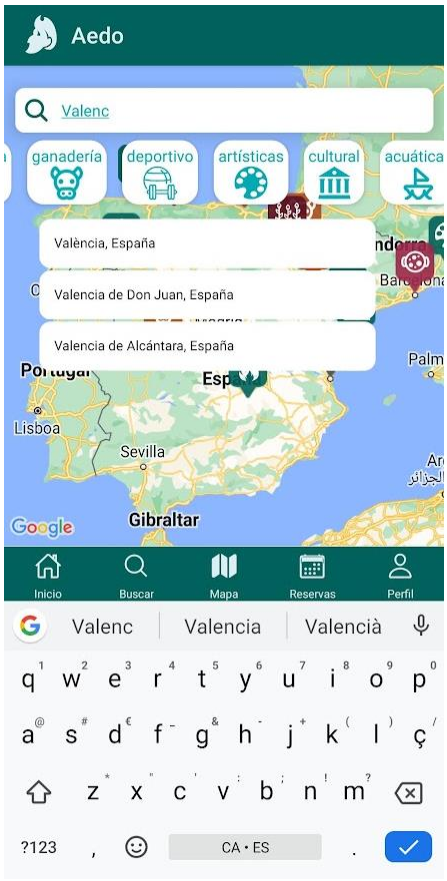

***Figura 39**: Ejemplo busqueda con google-places-autocomplete*

4.15.6 react-native-paper-dates

La librería *react-native-paper* es una popular librería de componentes de interfaz de usuario (UI) para React Native, que sigue los principios de Material Design[13], la guía de diseño de Google. Esta librería ofrece una amplia gama de componentes personalizables.

En concreto en la aplicación de Aedo se ha utilizado la parte de la librería que ofrece selectores de fechas *react-native-paper-dates*. Se ofrecen gran variedad de tipos de selectores de fechas[14] como el elegir un solo día, elegir múltiples días o elegir un rango de días.

4.15.7 Rreact-native-elements:

React Native Elements (*Overview | React Native Elements*, s. f.) es una biblioteca de componentes de interfaz de usuario para React Native, un marco de desarrollo de aplicaciones móviles multiplataforma. Diseñada para simplificar y acelerar el proceso de desarrollo de

[13] Material design: https://m3.material.io/

[14] Tipos de selectores de fechas que ofrece react-native-paper-dates: https://www.reactnativepaperdates.com/

interfaces de usuario en aplicaciones móviles, React Native Elements proporciona una variedad de componentes predefinidos y personalizables que pueden ser utilizados para construir interfaces de usuario atractivas y coherentes.

La biblioteca React Native Elements incluye una amplia gama de componentes, como botones, encabezados, iconos, tarjetas, formularios, listas y mucho más. Estos componentes están diseñados con una apariencia y comportamiento coherentes con estándares de diseño móvil como Material Design. Además, permite la fácil modificación de estos estilos a gusto del desarrollador para adaptarse a la apariencia de los componentes de la marca.

En resumen, React Native Elements es una amplia biblioteca de componentes de interfaz de usuario que simplifica y agiliza el desarrollo de aplicaciones móviles en React Native.

4.16 El uso de los estilos

En un proyecto de React native hay bastantes formas de definir los estilos de los diferentes componentes. Se pueden distinguir dos tipos de estilos: el estilo local, que es específico de un componente, y el estilo global, que se utiliza en varios componentes de la aplicación.

4.16.1 Estilos Locales

Para definir los estilos locales se ha utilizado el componente ofrecido por React Native llamado Stylesheet.

StyleSheet (ver ejemplo en la **Figura 40**) proporciona un método *create* que permite definir estilos mediante objetos de JavaScript. Estos objetos contienen propiedades que representan los estilos para los componentes, como el tamaño, color y posicionamiento.

Este contenedor de estilos se define debajo de la función del componente. De esta forma, moviendo los estilos lejos de la función de renderizado se facilita la lectura del código.

```
/**...
export default function MapOverlay(props) {···
}

const styles = StyleSheet.create({
  mainView: {
    width: 350,
    maxHeight: 650,

    alignItems: "center",
  },
  buttonsView: {
    position: "absolute",
    flex: 1,
    color: "#18910E",
    bottom: 6,
    alignSelf: "flex-end",
    flexDirection: "row",
    marginTop: 10,
    marginBottom: 10
```

***Figura 40**: Ejemplo del StyleSheet del componente MapOverlay*

4.16.2 Estilos Globales

En el caso de necesitar estilos globales para elementos como botones que se pueden utilizar en distintos componentes y ventanas de la aplicación, se han definido una serie de módulos, dentro de la carpeta *styles*, de los cuales cada uno contiene su propio contenedor de estilos (Stylesheet). Estos son para botones, iconos, *inputs*, textos, *views* y sombras (ver **Figura 41**). Cada stylesheet se puede reutilizar tantas veces como sea necesario.

Estos contenedores de estilos globales necesitan constantes que estén definidas en diferentes módulos como los tipos de colores, las tipografías y las escalas. Para comunicarse con estos módulos se ha creado un módulo *Index* que hace de intermediador para reducir el acoplamiento.

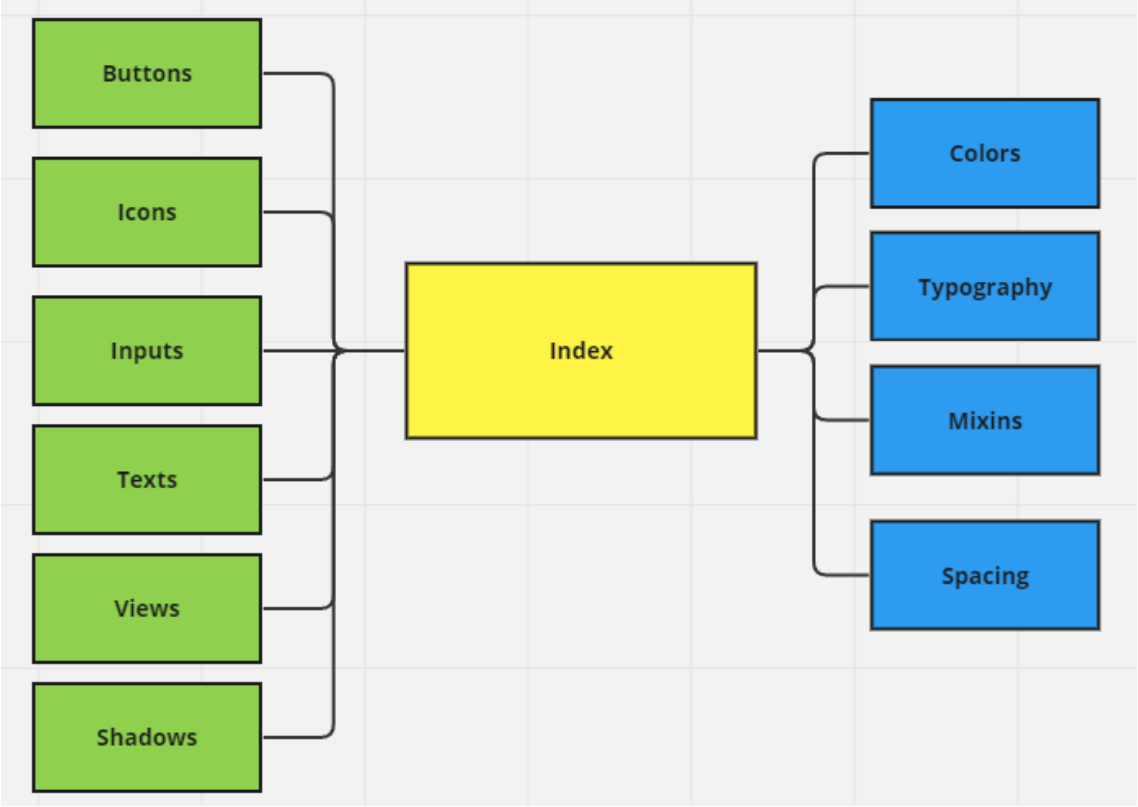

***Figura 41**: Distribución de los estilos globales*

4.17 Principios SOLID de React Native en Aedo

Los principios SOLID (Moniz, 2022b) son un conjunto de prácticas que se pueden utilizar a la hora de organizar de escribir código. Estas prácticas son muy recomendadas para la programación orientada a objetos pero se han expandido a otros tipos de estilos de programación. Se explicarán estas 5 prácticas en referencia a los entornos de desarrollo *front-end* de React y React Native enfocados en el desarrollo de componentes:

- El principio de única responsabilidad (*Single-responsibility principl*e). Una clase, un componente, una función solo deberían resolver un problema específico. En el contexto de React, una mala práctica del principio podría ser un componente que tenga demasiadas responsabilidades o que realice operaciones que no corresponden a su función principal. Esto hace que cuando vayas a modificar la funcionalidad del código se complique el mantenimiento.
  Esta regla se puede aplicar a muchas partes del código y depende mucho de la profundidad de la responsabilidad que se le quiera dar.

  Un ejemplo claro ejemplo del principio es el caso explicado en el apartado **4.7 Atomic design**. Inicialmente todo estaba metido en un mismo componente, es decir el mismo componente se encargaba de buscar una localización y de filtrar los resultados según un listado de categorías. Para dividir las responsabilidades se formaron dos componentes distintos cada uno con su función.

- El principio de abierto cerrado (*open-closed principle*). Las entidades deben de estar abiertas a extensión, pero cerradas a modificación. El principio de abierto/cerrado en React se trata de diseñar componentes que sean fáciles de extender y que puedan ser reutilizados sin tener que modificar su código fuente original.

  Un ejemplo de este principo es el explicado en el apartado **4.11.1 Especialización de componentes.** Al utilizar la composición de componentes, se evita tener que modificar el código fuente original del componente *MapComponent* para añadir la funcionalidad de renderizar un marcador en el mapa. En cambio, se utiliza la propiedad *handleOnPress* para pasar una función al componente genérico que puede ser utilizada por el componente específico para extender su comportamiento.

  De esta manera, se cumple con el principio de abierto/cerrado ya que se permite extender la funcionalidad de un componente sin tener que modificar su código fuente original, lo que promueve la modularidad y la reutilización de código.

- El principio de sustitución de Liskov (*Liskov substitution principle*). Cada clase que hereda de otra puede usarse en substitución de esta sin que haya problemas.
  Aplicado a React se puede ver este prencipio con los componentes. Se puede explicar con una estructura seguida con una de las librerías utilizadas por Aedo es *GooglePlacesAutocomplete*[15], una librería o componente de React que proporciona una interfaz para interactuar con el servicio de Autocompletado de lugares de Google Maps. Esta librería es utilizada por el componente padre *GooglePlaces* el cual tiene la configuración básica de la librería y tiene como propiedades: *region*, *setRegion* y *query*. Este componente será utilizado por componentes hijos especializados: *GooglePlacesCities*, *GooglePlacesRegions*, etc (ver **Figura 42**).

  En el ejemplo mencionado, este principio implica que cualquier componente que espere recibir una instancia de los componentes *GooglePlacesCities* y *GooglePlacesRegions* debe ser capaz de recibir, sin ocasionar problemas, instancias del componente *GooglePlaces*, que

[15] GooglePlacesAutocomplete: https://developers.google.com/maps/documentation/places/web-service/autocomplete?hl=es-419

es el tipo de los subtipos *GooglePlacesCities* y *GooglePlacesRegions*. Esto se consigue utilizando los mismos props comunes, es decir, que uno tenga *region* y otro *city* para referirse a la propiedad que contiene la ubicación provocaría un error en caso de que en un futuro se quisiera intercambiar un componente por otro.

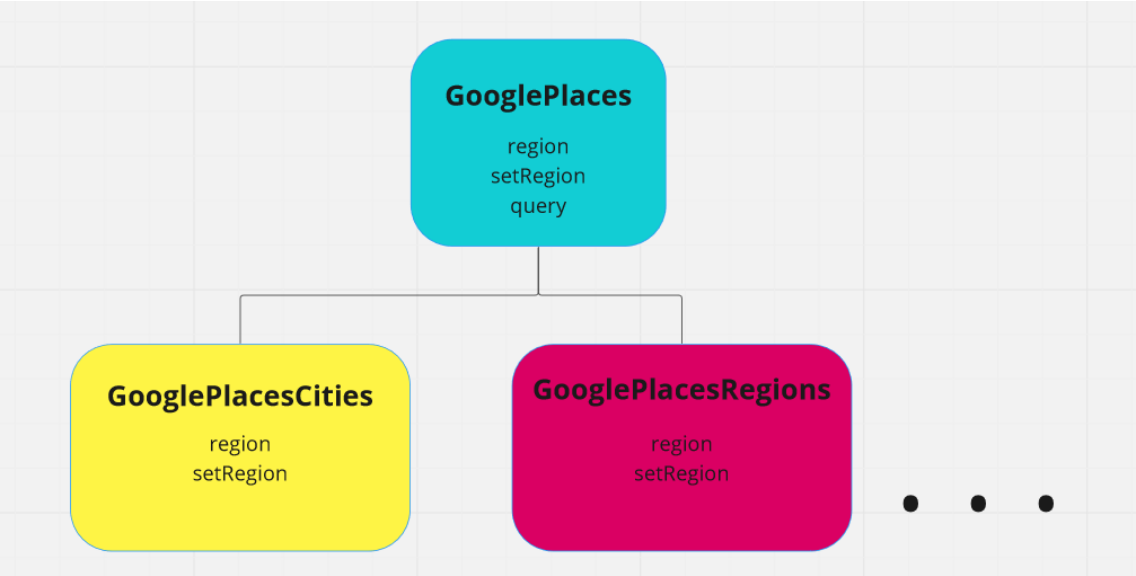

***Figura 42**:Esquema gerarquía de componentes GooglePlaces*

- El principio de segregación de interfaz (*Interface segregation principle*). Es recomendable tener una interfaz pequeña y cohesionada; por ejemplo, en un compnente de React solo se debería de tener las propiedades relevantes. Por ejemplo, en el componente del botón de favoritos de las experiencias (Ver **Figura 43**), solo se necesita el ID de la experiencia generada por la base de datos para añadirla al listado de favoritos del usuario. Por lo tanto, no hace falta pasar por las propiedades del componente todo el objeto que almaceno los datos de la experiencia.

  Un caso similar es el componente que genera una imagen del avatar del usuario (Ver **Figura 44**). Solo se necesita el parámetro relacionado con la imagen y no hace falta pasar todo el objeto usuario.

```
> /** ...
export function FavouriteButton(props) {
    const { odiseaID } = props;
```

***Figura 43**: Componente del botón de favoritos*

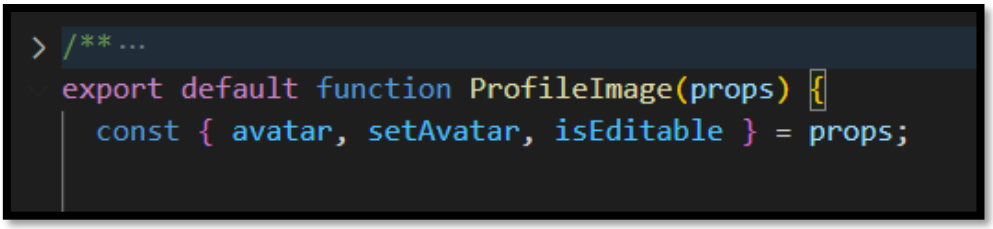

***Figura 44**: Componente del avatar de un usuario*

Cabe destacar que no todo es blanco o negro, habrá casos en los que sí que sea recomendable pasar todo el objeto, aunque sea información innecesaria, por ejemplo, un componente que genere un apartado donde se muestre información del usuario como el nombre o el correo (Ver **Figura 45**). Si en el momento actual no se desea mostrar el teléfono puede ser que en un futuro sí que se necesite por cualquier razón. Dependerá de la situación y el contexto el decidir cuál es la mejor opción.

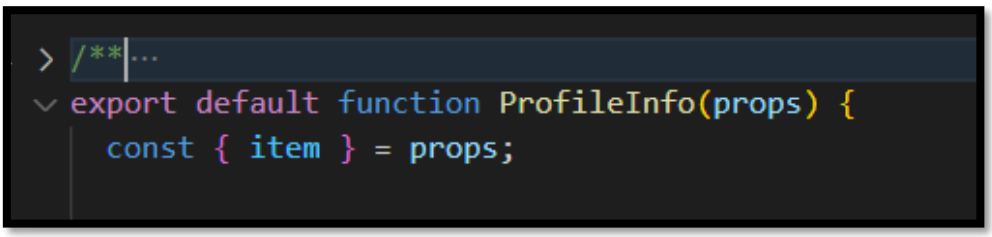

***Figura 45**: Componente de la información del perfil*

- El principio de inversión de dependencias (*Dependency inversion principle*). En este principio se recomienda depender de abstracciones y no de componentes concretos. En el caso de React Native se puede aplicar este principio para el uso de conexiones a APIs [16] externas. Por ejemplo, en el caso de que se quisiera obtener una imagen de una experiencia seleccionada por el usuario, se podría llamar directamente a la función de Firebase[17], el servicio de bases de datos actual de la aplicación Aedo, desde el componente que se necesite, pero esto incumpliría rotundamente el principio de inversión de dependencias.

  Esto puede generar un problema si en un futuro se quisiera cambiar completamente el servicio utilizado para el almacén de los datos, ya que habría que rehacer todas las llamadas realizadas a esta función.

  Para evitar este problema se ha creado un servicio intermedio llamado *ImageApi* (Ver **Figura 46**) el cual define funciones como *uploadImage*, *downloadImage*, *downLoadCategoryIcon* y *getCategoriesUris* que encapsulan la lógica para interactuar con el almacenamiento de

[16] API: Significa *Application Programming Interface* (Interfaz de Programación de Aplicaciones) y se refiere a un conjunto de reglas y protocolos que permiten a distintas aplicaciones comunicarse y compartir datos entre sí.

[17] Firebase: https://firebase.google.com/?hl=es-419

imágenes a través del módulo *server* que hace referencia a un módulo específico de un tipo de bases de datos. Estas funciones ofrecen una forma de utilizar el servicio de almacenamiento de imágenes sin exponer directamente los detalles de implementación, como la comunicación con Firebase.

Al tener esta capa de abstracción, los componentes de la interfaz de usuario pueden depender de la API *ImageApi* en lugar de depender directamente de la implementación concreta del módulo *server*. Esto permite una mayor flexibilidad, ya que puedes cambiar la implementación subyacente del módulo *server* sin afectar directamente a los componentes que utilizan la API *ImageApi*.

En resumen, el código proporcionado muestra un ejemplo de cómo se puede aplicar el principio de inversión de dependencias en React Native al utilizar una capa de abstracción, en este caso, la API *ImageApi*, para separar los componentes de la interfaz de usuario de los detalles de implementación específicos del almacenamiento de imágenes.

```
/**
 * Api para tratar los imagenes que se suben/descargan al storage
 * @external ImageApi
 */
export const ImageApi = {
  /**...
  uploadImage: async function (imageName, uri) {
    return await server.uploadFile("images/" + imageName, uri);
  },

  /**...
  downloadImage: async function (imageName) {
    return server.downloadFile("images/" + imageName);
  },
  downLoadCategoryIcon: async function (iconNameID) {
    return server.downloadFile("icons/" + iconNameID);
  },
  getCategoriesUris: async function (category) {
    return server.downloadFile("icons/" + category.name + "/" + category.id);
  },
};
```

***Figura 46**: Servicio/Api creada para manejar las diferentes imagenes desde la base de datos a la parte de presentación*

4.18 Intercambio de información con la web

La parte frontal de la aplicación se comunica con la parte de Datos y lógica a través de una capa que funciona como una API (Zanini, 2021). Al colocar los archivos en carpetas que representen una arquitectura multicapa, puedes convertir un proyecto en una estructura mucho más organizada. Con este enfoque, cada archivo tendrá su propio lugar. Esto permitirá que el equipo establezca estándares de arquitectura que hagan que todo el código sea más robusto y mantenible y simplifiquen el proceso de desarrollo.

La estructura escogida por el equipo ha sido crear una API por cada modelo de datos y algunas APIs generales como la de manejo de ficheros (Ver **Figura 47**). Por ejemplo, se ha creado una api para una reserva (*ReservationAPI*) que contiene funcionalidades como crear, actualizar, viasualizar una reserva con ID o eliminarla.

La creación de APIs específicas para cada modelo de datos, como la mencionada *ReservationAPI*, es una práctica común en el diseño de sistemas. Esto permite encapsular la funcionalidad relacionada con ese modelo en particular y proporcionar una interfaz clara y coherente para manipular los datos asociados.

En resumen, es importante destacar que el uso de una capa API para comunicar la parte frontal de la aplicación con la lógica y los datos proporciona una separación clara de responsabilidades y facilita la interoperabilidad entre los distintos componentes del sistema.

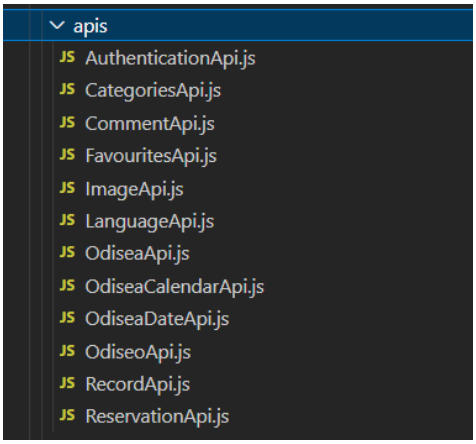

***Figura 47**: Listado de APIs de la aplicación Aedo*

4.19 Mantenibilidad del Software

La mejora la calidad del software es una de las prioridades para el desarrollo de la aplicación Aedo. Para ello se ha utilizada la herramienta SonaQube que realiza un análisis exhaustivo del código fuente y proporciona información detallada sobre la calidad de este en términos de mantenibilidad, confiabilidad, seguridad y eficiencia.

Durante el tercer sprint de desarrollo que realizó el primer análisis y los resultados obtenidos fueron los que se visualizan en la siguiente captura:

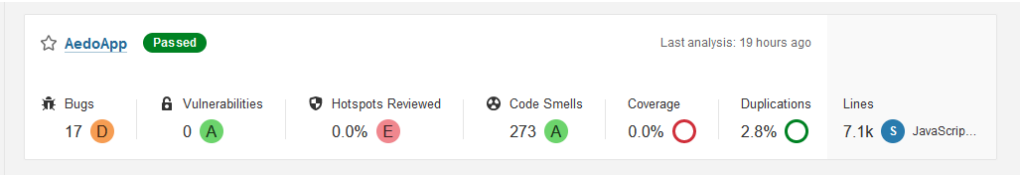

***Figura 48**: Primer análisis obtenido de SonarQube*

SonarQube detectó que la aplicación no tenía vulnerabilidades (característica no funcional de seguridad) en cambio sí tenía problemas de bugs (característica de fiabilidad de la aplicación) y *code smells* (característica de mantenibilidad).

En cuanto a los bugs, se califican con distintas severidades no siendo ninguno de ellos bloqueantes de la aplicación (Ver **Figura 49**). El informe que genera ayuda a encontrar donde están los problemas para poder solucionarlos.

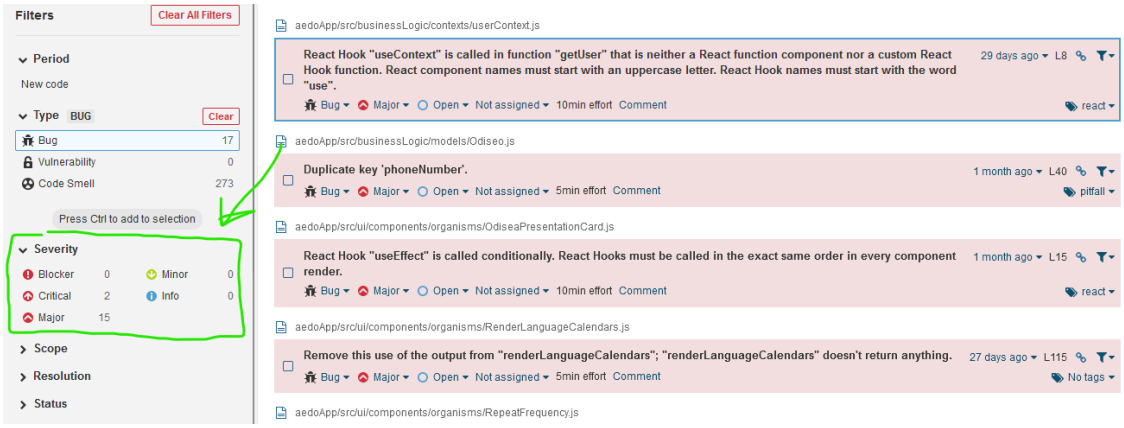

***Figura 49**: Tiraje de bugs realizado por SonarQube*

Además Esta herramienta calcula la deuda técnica del software analizado, es decir el tiempo que se tardaría en corregir los *bugs* y *code smells* detectados, en el caso de Aedo, en el momento que se realizó el análisis, se obtuvo una deuda técnica de 2 días y 6 horas.

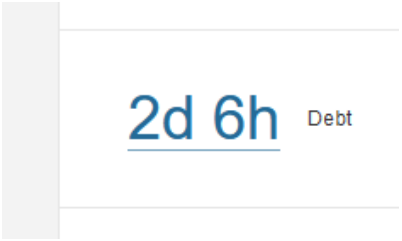

***Figura 50**: Deuda técnica ofrecida por SonarQube*

Después de llevar a cabo la depuración del código, se logró reducir los *code smells* a 94 y los errores a 14. Esto ha resultado en una disminución de la deuda técnica de aproximadamente 1 día, dejándola en un total de 1 día y 5 horas como se puede ver en la **Figura 51**.

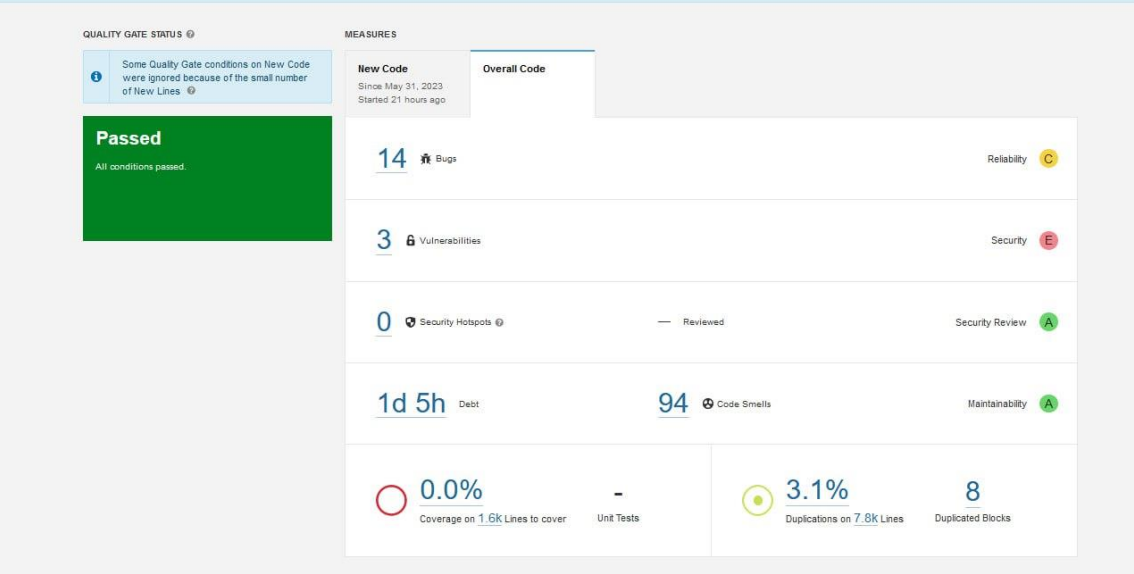

Figura 51: *Análisis SonarQube tras una limpieza de código*

4.20 Pruebas realizadas

4.20.1 Jest

Para realizar las pruebas de automatización se ha consultado las recomendaciones de React Native (*Testing · React Native*, 2023), y en concreto la parte de *user interactions*, para de esta forma probar que los requisitos realizados funcionen correctamente en todo momento del desarrollo.

Se ha empleado Jest (*Jest*, s. f.), un marco de pruebas (*testing framework*) desarrollado por Facebook. Está diseñado principalmente para probar aplicaciones escritas en JavaScript, especialmente aplicaciones de React., en conjunto con la librería testing-library[18] del propio react-native. Ambas trabajan en conjunto para poder ofrecer las funcionalidades de testing buscadas. Se ha conseguido probar la inserción de datos de muchos formularios en la aplicación lo cual garantiza que los usuarios y las odiseas, tanto las operaciones de creación como de modificación de estos, se realizarán de una manera correcta.

El funcionamiento de la herramienta se va a explicar a partir de la siguiente captura, mencionando las líneas de código, comentar que dicha captura de pantalla corresponde a la prueba de aceptación de crear experiencias/odiseas (Ver **Figura 52**).

[18] Testing-library: https://testing-library.com/docs/react-native-testing-library/intro/

```
1  import React from "react";
2  import { fireEvent, render, screen } from "@testing-library/react-native";
3  import ComentWindow from "../../../src/ui/components/organisms/ComentWindow";
4
5  import CreateOdiseaScreen from "../../../src/ui/screens/odisea/CreateOdiseaScreen";
6
7  describe("Create odisea screen", () => {
8    test("Crear odisea", () => {
9      render(<CreateOdiseaScreen />);
10     const nombre = screen.getByPlaceholderText("Llegar a Itaca ...");
11     fireEvent.changeText(nombre, "Elaboracion de fartons");
12     expect(nombre.props.value).toBe("Elaboracion de fartons");
13
14     const descripcion = screen.getByPlaceholderText(
15       "Vuelve a tu hogar y descansa ..."
16     );
17     fireEvent.changeText(
18       descripcion,
19       "En 2 horas te enseño como se elaboran los fartons de manera tradicional"
20     );
21     expect(descripcion.props.value).toBe(
22       "En 2 horas te enseño como se elaboran los fartons de manera tradicional"
23     );
24
25     const aforo = screen.getByPlaceholderText(
26       "Nº máximo de compañeros... (10 por defecto)"
27     );
28     fireEvent.changeText(aforo, 20);
29     expect(aforo.props.value).toBe(20);
30
```

***Figura 52**: Test en Jest para la ventana de crear una experiencia*

En este apartado es necesaria la instalación de Jest y testing-library, así como la importación de *fireEvent*, *render* y *screen* (línea 2), posteriormente se explicará la utilidad. Después se importa el componente a probar, para este ejemplo de la imagen, se importa *CreateOdiseaScreen* (línea 5), el cual se encarga de crear las experiencias/odiseas en la aplicación.

Es recomendable proporcionar una descripción clara y concisa en el bloque *describe* (línea 7) que indique qué aspecto específico del componente se va a testear. Se pueden realizar distintos tests en un componente en concreto, todos se crean con la estructura *test* de la línea 8 y se le suele asignar un nombre representativo. Acto seguido, línea 9, aparece la función *render* importada anteriormente. Esta función se utiliza para renderizar el componente que queramos probar y que Jest más *testing-library* puedan realizar las pruebas. Para el ejemplo se prueba *CreateOdiseaScreen*.

A continuación, mediante *screen* (línea 10), se obtiene el *TextInput* del componente utilizando su *placeholder*, el cual está presente en el código del componente. El *fireEvent* (línea 11) simula la introducción de un texto (por ejemplo, "Elaboración de fartons") en el formulario.

Finalmente, se utiliza el *expect* (línea 12) se marca el resultado esperado. Si los resultados no coinciden el test falla. Con esto se garantiza que el valor que introduce el usuario en el

formulario es el que realmente recibe el *front-end* de la aplicación el cual será posteriormente guardado en la base de datos.

El test se ejecuta con el comando *npm test* y tras pasar los tests devuelve si han pasado (Ver **Figura 53**) o en caso de tener algún error devuelve el error y dónde se encuentra el mismo, así como una descripción.

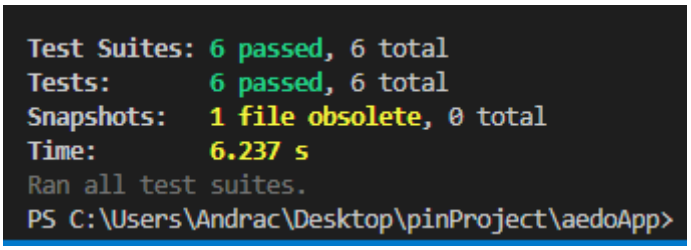

Figura 53: *Ejemplo de resultados correctos de los Tests realizados*

4.20.2 Test de usabilidad

Durante el desarrollo de este proyecto, se han realizado tests de usabilidad en los experimentos realizados al usuario relacionado con la educación y a un usuario relacionado con la oferta de experiencias de agricultura y naturaleza. Posteriormente, las mismas preguntas del cuestionario fueron presentadas a distintos conocidos del equipo de desarrollo con un rango de edad comprendido entre 20 y 30 años para conocer su opinión.

Para ello se ha utilizado como referencia el cuestionario de usabilidad en sistemas informáticos (CSUQ), el cual fue desarrollado para recopilar el mayor número de cuestionarios aplicados a las pruebas de usabilidad. Las preguntas que se formularon durante los experimentos quedan recogidas en el Anexo 2. Estas preguntas están enfocadas a la usabilidad y calidad del producto software ofrecido.

Las puntuaciones para las distintas preguntas obtenidas fueron superiores a ocho puntos sobre diez de media, siendo, en su mayoría, puntuaciones superiores a nueve puntos, la pregunta con menor puntuación fue de 8,36 en la que algunos encuestados opinan que la aplicación no brinda la información suficiente sobre cómo proceder cuando existe algún problema. Sin embargo, estos usuarios no han mencionado nada al respecto en el apartado de comentarios del cuestionario. Por otra parte, los apartados de interfaz de usuario y creación de experiencias han recibido una puntuación media de diez sobre diez, seguido por 9,82 puntos en la pregunta de la creación de experiencias.. En el anexo 2, se comparten los datos de la encuesta y los resultados.

Cuatro de los encuestados compartieron sugerencias para mejorar la aplicación. Entre ellas sugirieron implementar un chat interno en la aplicación, exportar la información de las experiencias a distintos formatos, la posibilidad de compartirlas en redes sociales y la posibilidad de añadir una hora a la experiencia y no solo el día. Todas estas sugerencias han sido tomadas en cuenta y serán implementadas en versiones futuras de la aplicación.

La realización de estas pruebas de usabilidad fue fundamental para comprender las necesidades y preferencias de los usuarios, permitiendo así realizar ajustes y mejoras en la interfaz y funcionalidades de la aplicación para próximas iteraciones del producto. Por ejemplo, respondiendo a la puntuación obtenida en la pregunta de la información ofrecida sobre cómo proceder cuando existe algún problema, se ha investigado el comportamiento de algunos

formularios y se han añadido algunos mensajes de error que no se mostraban al introducir datos erróneos al intentar iniciar sesión.

4.21 Manual de usuario de la aplicación móvil

Ventana de inicio:

En esta ventana (ver **Figura 54**) aparecen todas las experiencias propuestas por los distintos usuarios en la aplicación. En la parte inferior aparecen recomendaciones personalizadas por una inteligencia artificial.

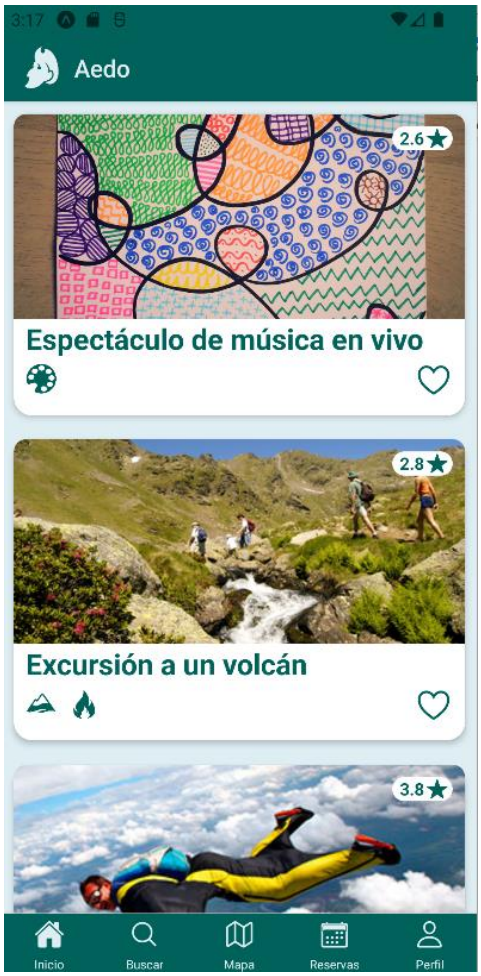

***Figura 54**: Ventana de inicio*

Ventana de información de una experiencia:

Aparece la información de la experiencia, así como los comentarios y valoraciones dados por otros usuarios de la aplicación. Además, se puede reservar la experiencia (ver **Figura 55** y **Figura 56**).

***Figura 55**: Ventana de información de una experiencia con comentarios*

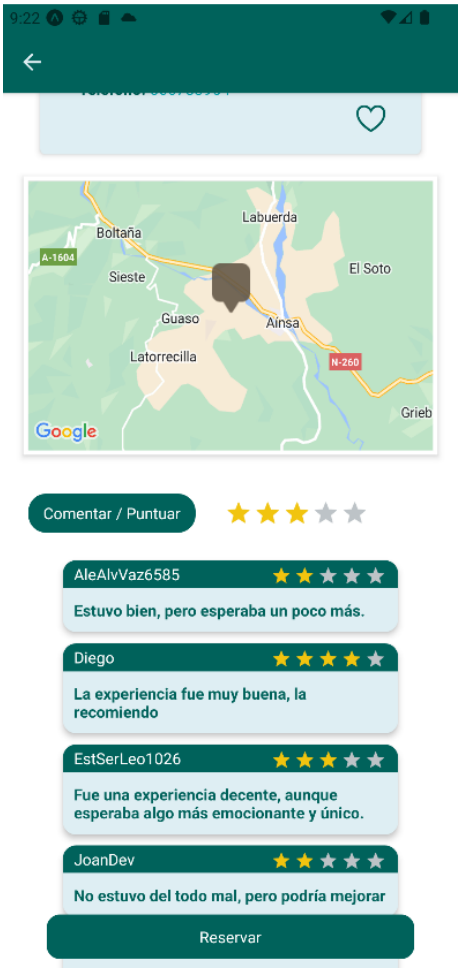

***Figura 56**:Ventana de información de una experiencia con comentarios*

Ventana de selección de fecha de reserva:

Al intentar reservar una experiencia, primero se pueden seleccionar cuantas reservas se quiere (ver **Figura 58**) y posteriormente aparecen los distintos idiomas disponibles con un calendario para poder escoger la fecha de la reserva (ver **Figura 57**).

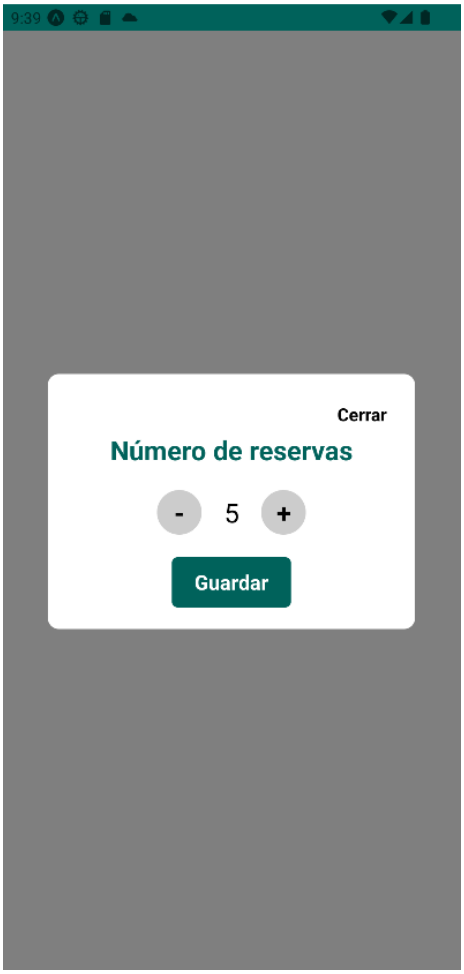

***Figura 58**: Ventana de selección de número de reservas*

***Figura 57**: Ventana de selección de día de reserva*

Ventana de buscar:

Esta pantalla muestra experiencias próximas a la ubicación del dispositivo. Además, ofrece un buscador para buscar por localidad y un sistema de filtros por categorías de experiencias (ver **Figura 59**).

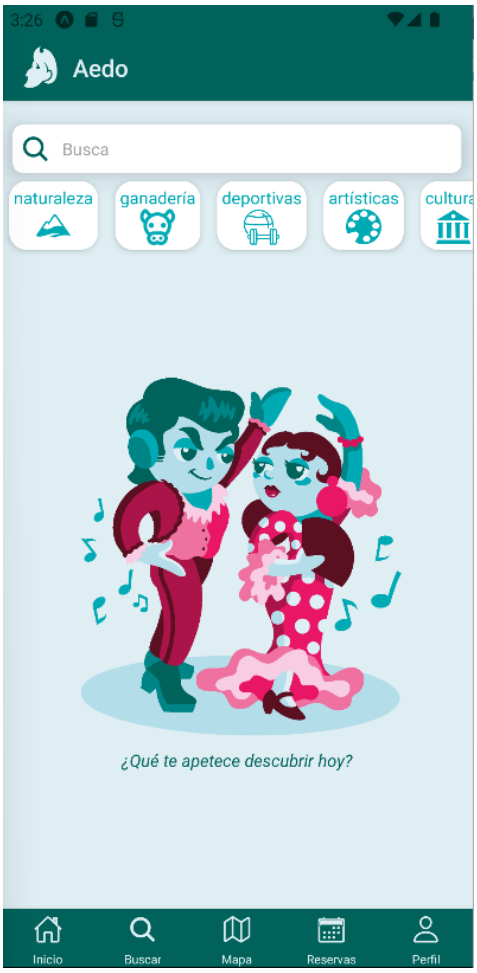

***Figura 59**: Ventana de buscar*

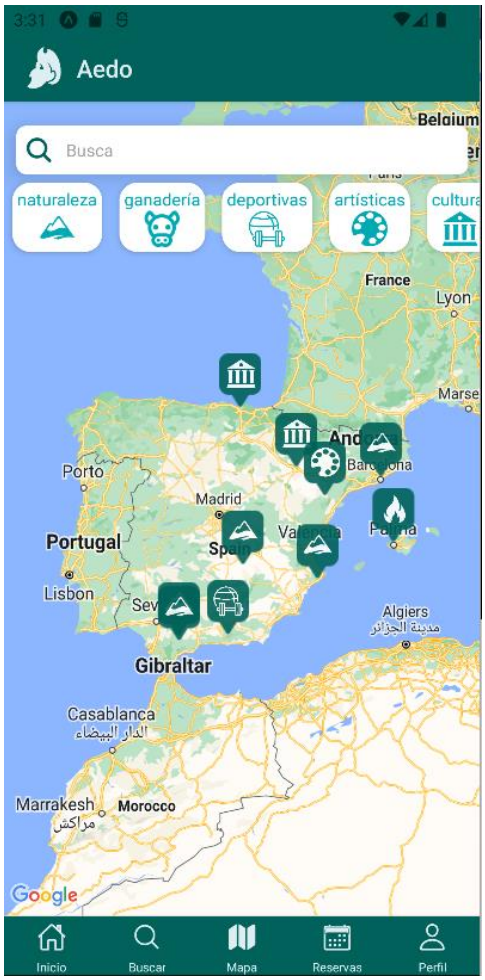

***Figura 60**: Ventana el mapa*

Ventana del mapa:

En este elemento aparece el mapa y todas las experiencias clasificadas con sus categorías y la ubicación. Se puede acceder a la "ventana de información de una experiencia" tocando el icono y el nombre en el mapa (ver **Figura 60**).

Ventana de tus reservas:

En esta interfaz aparecen las reservas realizadas por el usuario que ha iniciado sesión y la información de estas como la fecha, el idioma y el número de reservas realizadas. Permite cancelar la reserva también (ver **Figura 61**).

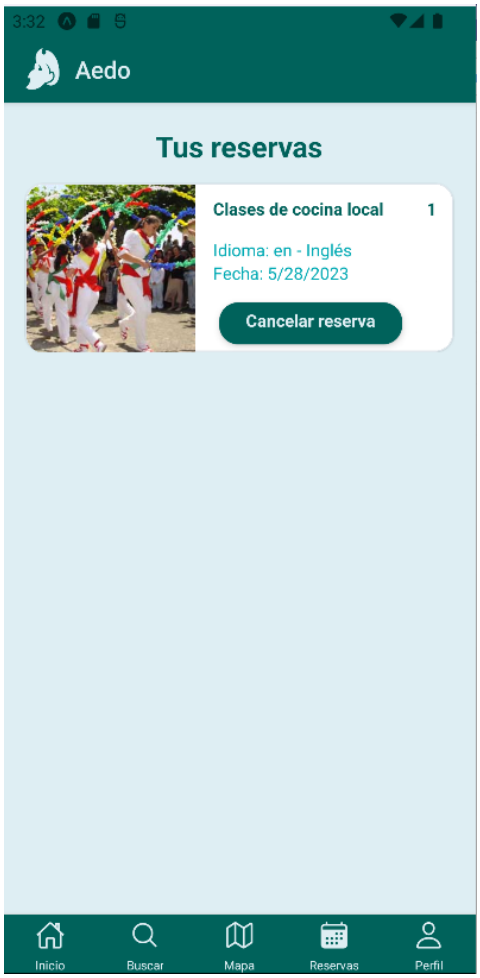

***Figura 61**: Ventana de tus reservas*

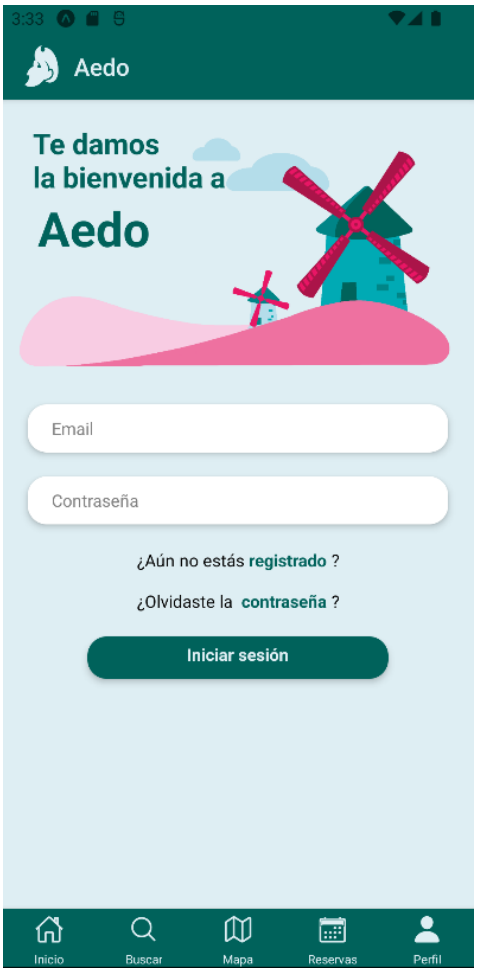

Figura 62: Ventana de inicio de sesión

Ventana de inicio de sesión:

Interfaz para iniciar sesión con el correo electrónico y la contraseña. Además, permite registrarse o restablecer la contraseña (ver **Figura 62**).

Figura 63: Ventana de registro

Ventana de registro:

Incluye el formulario con los datos necesarios para realizar el registro en la aplicación. Los campos marcados con un asterisco son obligatorios (ver **Figura 63**).

Al registrarse se debe confirmar el correo electrónico.

***Figura 64**: Ventana de perfil de usuario*

Ventana de perfil de usuario:

La pestaña del perfil (ver **Figura 64**) con el usuario autentificado se convierte en la pantalla de perfil, aquí se pueden crear experiencias o gestionarlas, además aparecen las experiencias favoritas del usuario y permite acceder a la modificación de datos personales, correo y contraseña.

Ventana modificar perfil:

Esta interfaz se utiliza para modificar los datos del usuario facilitados durante el registro en la aplicación. Permite además solicitar a través de un correo al equipo de Aedo una cuenta educativa. Esta permite la reserva exclusiva de una experiencia para un grupo escolar (ver **Figura 65**).

***Figura 65**: Ventana de perfil de usuario*

Figura 66: Ventana crear experiencia

Ventana crear experiencias:

Interfaz para crear experiencias en la aplicación. Permite introducir un título, una descripción, el aforo máximo, las imágenes, la categoría, el/los idioma/s y la localización (ver **Figura 66**).

Ventana periodicidad:

La ventana anterior lleva a la selección de fechas o periodicidad de la experiencia por cada idioma (ver **Figura 67**).

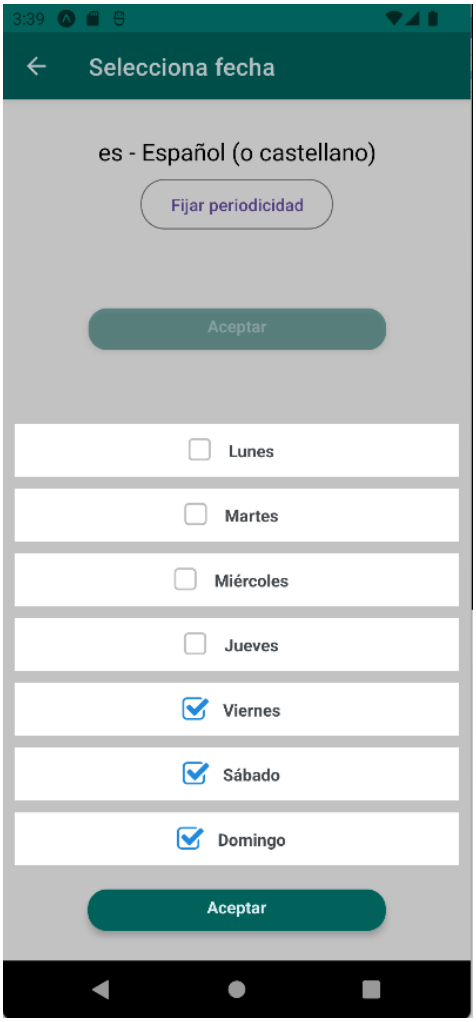

Figura 67: Ventana periocidad

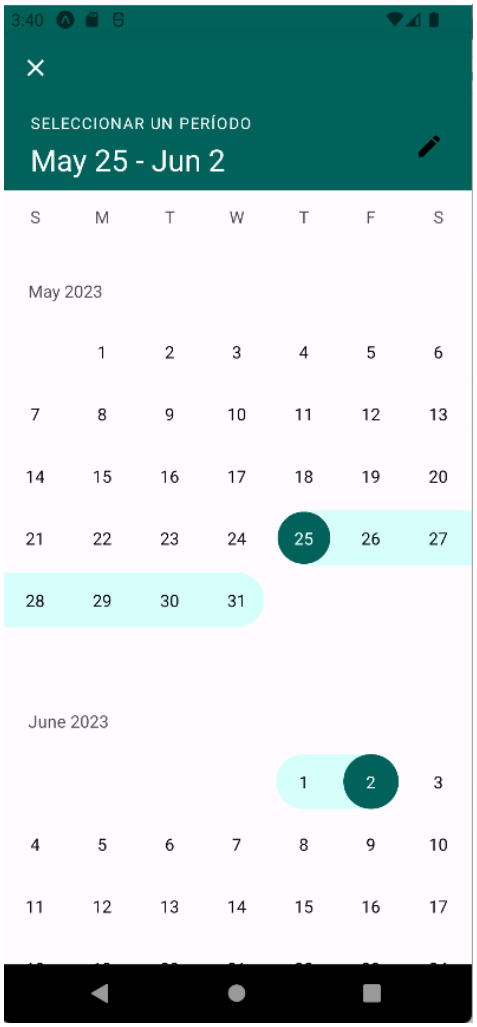

***Figura 68**: Ventana selección de rango de fechas*

Ventana selección de rango de fechas:

La ventana de creación de experiencias lleva como paso final a la selección de fechas en las que se ofrece la experiencia (ver **Figura 68**).

Ventana gestión de experiencias:

Aquí se presentan las distintas experiencias ofrecidas por el usuario registrado. Permite modificar la información, borrarlas o ver otros usuarios que se han apuntado y para qué fecha lo han hecho (ver **Figura 69**).

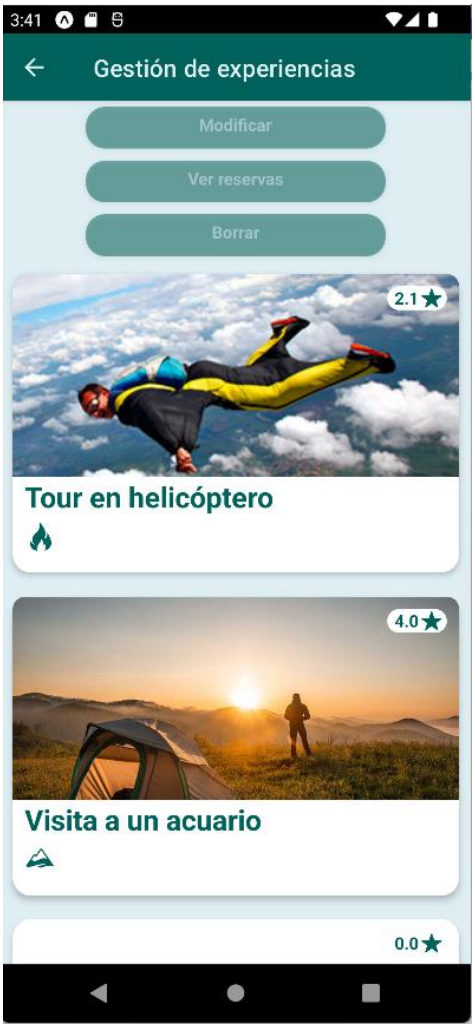

***Figura 69**: Ventana gestión de experiencias*

5 Cronología del TFG

La cronología de la aplicación se divide en seis *sprints* de programación y un séptimo inicial que se empleó para definir la idea de la aplicación, evaluar la idea de negocio, realizar el estudio de mercado, mapa de características, modelo de negocio y proyección económica… Todo el procedimiento relacionado con los proyectos de emprendimiento. También, se tuvo que elaborar un *pitch* de la idea de negocio, preparar el entorno de desarrollo y formarse los miembros del equipo para no afrontar el proyecto sin tener ningún conocimiento.

A continuación, se detallan los seis sprints principales y las áreas en las que el equipo de desarrollo se enfocó durante cada uno de ellos. Es importante destacar que todos los miembros del equipo desempeñaron tareas de desarrollo *full stack*, por lo que las actividades mencionadas abarcaron diferentes aspectos del proyecto.

5.1 Sprint 1 (PIN: Proyecto de Ingeniería del *software*)

Durante el primer sprint, se realizaron las implementaciones fundamentales para que la aplicación pudiera admitir usuarios, permitir el registro de experiencias y la capacidad de realizar reservas. Para lograr esto, se desarrollaron diversas pantallas y componentes, como una barra de navegación inferior, una pantalla de inicio que mostraba una lista de todas las experiencias disponibles, y la posibilidad de acceder a cada una de ellas para ver su información y realizar reservas. Además, se incorporaron las funcionalidades de edición y eliminación tanto para las experiencias como para los usuarios.

Estas implementaciones iniciales establecieron las bases funcionales de la aplicación, permitiendo a los usuarios interactuar con las experiencias, realizar cambios en ellas y administrar sus propios perfiles de usuario.

5.2 Sprint 2 (PIN: Proyecto de Ingeniería del *software*)

Durante el segundo sprint, se realizó una adaptación del proyecto al modelo de tres capas (Presentación, Lógica y capa de Datos). También se llevó a cabo la implementación del contexto de usuario, que resultaba esencial para acceder al usuario autenticado siempre que fuera necesario.

En lo que respecta a los nuevos requisitos, se introdujo la capacidad de registrar experiencias en varios idiomas y utilizar diferentes calendarios, lo cual implicó una reestructuración del modelo de datos existente. Este cambio permitió brindar soporte para experiencias multilingües y adaptarse a diferentes sistemas de calendario.

Además, se agregaron nuevas características clave, como la integración de un mapa para visualizar la ubicación de las experiencias, la incorporación de un buscador para facilitar la búsqueda y filtrado de experiencias específicas, la implementación de un control para gestionar el aforo máximo de las experiencias y la funcionalidad de comentar en las mismas.

Estas adiciones en el segundo sprint enriquecieron significativamente la experiencia de los usuarios al ampliar las opciones de personalización y brindarles herramientas para interactuar y compartir sus opiniones sobre las experiencias registradas.

5.3 Sprint 3 (PIN: Proyecto de Ingeniería del *software*)

Durante el tercer sprint, el enfoque principal del equipo fue refinar y mejorar los aspectos ya implementados de la aplicación, corrigiendo errores y excepciones para asegurar su funcionamiento óptimo de cara a la Feria de Proyectos. Se llevaron a cabo reuniones con los alumnos de bellas artes para colaborar en el diseño de las interfaces gráficas de las distintas pantallas, utilizando el material proporcionado por ellos.

Dado que no se contaba con maquetas para todas las ventanas de la aplicación, se implementaron las restantes siguiendo una estética similar a la proporcionada por los compañeros de bellas artes, con el objetivo de mantener la coherencia visual en toda la aplicación.

En términos de funcionalidades, se incorporó una versión básica de inteligencia artificial, se trató de un algoritmo de recomendaciones basado en las experiencias consultadas hasta el momento. Este agregado permitió ofrecer sugerencias personalizadas a los usuarios en función de sus intereses previos.

Adicionalmente, se trabajó en la mejora de los filtros de búsqueda de experiencias en la pantalla del buscador, así como en la ventana del mapa. Estas mejoras facilitaron la navegación y la localización de experiencias relevantes para los usuarios.

5.4 Sprint 4

Durante la cuarta iteración, el enfoque principal fue iniciar el desarrollo de la página web, la cual tendría la función de realizar tareas administrativas y otras acciones que podrían resultar más tediosas de llevar a cabo desde un dispositivo móvil. Se configuró el servicio de hosting de Firebase para alojar y desplegar la página web. Se comenzó implementando una versión básica de la funcionalidad de inicio de sesión y el perfil de usuario en la página web, permitiendo a los usuarios acceder a su cuenta y gestionar su información personal desde esta plataforma. Además, se agregó la capacidad de agregar y eliminar idiomas y categorías desde la página web.

En paralelo, se inició una investigación más profunda en algoritmos y técnicas para desarrollar la inteligencia artificial que se había mencionado anteriormente. Este enfoque permitiría mejorar y expandir las funcionalidades de recomendación de experiencias basadas en los algoritmos implementados hasta el momento.

5.5 Sprint 5

Durante la quinta iteración, se continuó agregando funcionalidades básicas similares a las del dispositivo móvil a la página web, con el objetivo de ofrecer una experiencia completa e integrada para los usuarios. Esto implicó desarrollar y adaptar características como la visualización de experiencias, la reserva de actividades y la gestión de perfiles desde la plataforma web.

En cuanto a la inteligencia artificial, se implementaron algoritmos y técnicas adecuadas para mejorar las recomendaciones de experiencias y personalizar aún más la interacción de los usuarios con la aplicación.

Además, se introdujo la posibilidad de tener cuentas educativas para los profesores, brindándoles un acceso especializado y permitiéndoles gestionar y supervisar las actividades de sus estudiantes dentro del sistema.

Un aspecto importante abordado durante esta iteración fue el desarrollo y mejora del script de generación de datos en Firebase. Este script era necesario para proporcionar datos de muestra y entrenar la inteligencia artificial de manera efectiva, lo que contribuyó a mejorar la calidad de las recomendaciones proporcionadas.

5.6 Sprint 6

Durante el sexto y último sprint, el equipo se enfocó en agregar nuevas funcionalidades a la página web y finalizar la programación y pulido de la inteligencia artificial. Se implementó la capacidad de realizar múltiples reservas desde una misma cuenta de usuario, lo que brindó mayor flexibilidad y comodidad a los usuarios al interactuar con la aplicación.

Además, se generó la APK de la aplicación para dispositivos móviles y se procedió a publicarla en la tienda de Google, para así realizar los experimentos con los usuarios.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En el presente Trabajo de Fin de Grado se ha realizado un esfuerzo considerable para desarrollar un producto mínimo viable que cumpla con la mayoría de los requisitos establecidos por el equipo, con el objetivo principal de lograr su utilidad y aplicabilidad.

Asimismo, se ha llevado a cabo una proyección económica y se ha establecido una hoja de ruta de requisitos en el *backlog* para futuras mejoras. El resultado obtenido es un producto con un estilo artístico profesional, en el que se han implementado funcionalidades muy útiles para el usuario, gracias al uso de las librerías apropiadas.

Durante los *sprints* 4, 5 y 6, se ha enfocado el desarrollo en mejorar la calidad y legibilidad del código de la aplicación. Se ha realizado una exhaustiva documentación de todos los componentes de la aplicación y de las APIs creadas. Además, se ha iniciado el desarrollo de pruebas de automatización y se han llevado a cabo pruebas de usabilidad durante los experimentos para identificar posibles mejoras en el producto actual.

Todo esto ha sido posible gracias al conocimiento adquirido en diferentes asignaturas. En primer lugar, el Proyecto de Ingeniería de Software (PIN) nos ha brindado las bases para comenzar este proyecto y nos ha guiado durante los primeros *sprints*, asegurándonos de seguir una metodología adecuada. Además, la asignatura de Calidad del Software (CSO) nos ha permitido trabajar con una usabilidad óptima para nuestro producto de software. Por otro lado, la asignatura de Mantenimiento y Evolución del Software (MES) nos ha proporcionado herramientas y métodos de trabajo útiles para mantener un código fácilmente mantenible a lo largo del tiempo. Por último, el Diseño del Software (DSO) ha sido fundamental para fomentar una cultura de estructuración del código y nos ha introducido el concepto de refactorización, un proceso en el desarrollo de software que implica modificar el diseño interno de un código sin cambiar su comportamiento externo.

A nivel personal, este proyecto ha brindado a cada miembro del equipo la invaluable oportunidad de embarcarse en el desarrollo de un software desde cero, con una perspectiva seria y ambiciosa. Esta experiencia ha requerido que cada uno se esforzara por encontrar soluciones y dar vida al producto, al mismo tiempo que se compartía el conocimiento adquirido con los compañeros. En el camino, el equipo se ha enfrentado a numerosos desafíos, pero gracias al trabajo en equipo, se ha sido capaz de superarlos y encontrar soluciones efectivas.

En resumen, este esfuerzo ha permitido alcanzar los objetivos propuestos, proporcionando un producto funcional y estéticamente atractivo, mientras se han sentado las bases para futuras iteraciones y mejoras, tanto en términos de calidad del código como de experiencia de usuario.

6.2 Trabajo futuro

Aunque se han abordado numerosas funcionalidades en este proyecto, como se menciona en las conclusiones, todavía quedan algunas por implementar. Estas son las funcionalidades destacadas que podrían ser consideradas para el desarrollo futuro de la aplicación:

- Mejorar el sistema de recomendación de experiencias, de modo que sugiera actividades cercanas a la ubicación actual del usuario, pero no demasiado lejos. Esto promovería la movilidad y el descubrimiento de zonas menos pobladas.

- Implementar un sistema de pago a través de dispositivos móviles, tanto para las experiencias como para posibles servicios premium en el futuro. Esto brindaría una mayor comodidad y flexibilidad a los usuarios a la hora de realizar transacciones.
- Integrar un chat interno en la aplicación, permitiendo la comunicación directa entre los usuarios y los anfitriones de las experiencias, sin necesidad de intercambiar números de teléfono u otros datos personales.
- Incorporar un sistema de notificaciones que informe a los usuarios sobre reservas próximas o cambios en las reservas existentes. La funcionalidad exacta de las notificaciones se puede definir en futuras discusiones.
- Continuar agregando funcionalidades móviles faltantes en la página web para mejorar la experiencia del usuario en diferentes dispositivos.
- Implementar la autenticación con distintas cuentas, como Google y Facebook, para ofrecer opciones de inicio de sesión más amplias y convenientes.
- Agregar la posibilidad de compartir experiencias en redes sociales, lo que permitiría a los usuarios difundir sus vivencias y atraer a más personas a la aplicación.
- Realizar tareas de mejora continua del código, con el objetivo de reducir la deuda técnica de la aplicación y generar más pruebas automatizadas. Esto garantizaría un funcionamiento más robusto y confiable.
- Incorporar la opción de seleccionar diferentes idiomas para la interfaz de usuario. Aunque la aplicación esté enfocada en España y se utilice principalmente en español, ofrecer la posibilidad de utilizar otros idiomas brindaría una mayor accesibilidad y versatilidad.

Estas funcionalidades pendientes representan oportunidades de crecimiento y de mejora de la aplicación en el futuro, permitiendo satisfacer las necesidades y expectativas de los usuarios de manera más completa y efectiva.

Glosario de términos

MVP: Minimum Viable Product. Producto viable mínimo (*Gasbarrino s.f.)*. En términos de desarrollo de aplicaciones, se trata de un producto con un número de características suficientes como para satisfacer las necesidades iniciales de los clientes.

Linter: En una definición básica y rápida (Team, 2023), un linter es un instrumento que va analizar tu código fuente para determinar si existe alguna *inconsistencia*. Esta inconsistencia puede ir desde una simple evaluación de estándares de código hasta un debugueo del mismo, donde se puede encontrar un code smell.

Code smell: Es una característica del código que permite a los programadores intuir que el código puede estar causando o causará problemas en el software, debido a su mala construcción.

Apk: *Andoid Application Package*. (Aguilar, 2020) Packete de instalación que contiene los datos de una aplicación y permite su instalación en dispositivos Android.

Framework: Un *framework* (*Framework, 2022*), o marco de trabajo, tiene como objetivo facilitar las tareas de desarrollo que surgen a la hora de programar, estos aceleran el proceso de programar gracias a que facilitan las tareas propias por tanto su objetivo al final es facilitar las tareas a los programadores.

Front-end: Complementario al *back-end*. El *front-end* (*Chapaval, 2018*) está formado por la parte visible de la aplicación, sea una interfaz web, una aplicación móvil o de escritorio. Es la parte de la aplicación con la que los usuarios interactúan. Las letras, los colores, elementos… Que los usuarios pueden visualizar.

Back-end: Complementario al *front-end*. El *back-end* (*Machuca, 2022*) está formado por todos aquellos elementos de la página web o de la aplicación los cuales no son accesibles para los usuarios finales de la aplicación.

Referencias

- Aguilar, R. (17 de Julio, 2020). Qué es un APK de Andoid, cómo se instala y diferencias con las apps normales. Xatakandroid. https://www.xatakandroid.com/aplicaciones-android/que-apk-android-como-se-instala-diferencias-apps-normales Consultado en Marzo 2023.
- Atomic Design Methodology | Atomic Design by Brad Frost. (s. f.). https://atomicdesign.bradfrost.com/chapter-2/ Consultado en Octubre 2022.
- Chapaval, N. (2018). Qué es Frontend y Backend: diferencias y características. Platzi. https://platzi.com/blog/que-es-frontend-y-backend/ Consultado en Abril 2023.
- CloudAPPi. (2022). Patrones de diseño: Composición en React. CloudAPPi. https://cloudappi.net/patrones-de-diseno-composicion-en-react/ Consultado en Octubre 2022.
- Davidbritch. (2023, 5 mayo). ¿Qué es .NET MAUI? - .NET MAUI. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/maui/what-is-maui Consultado en Mayo 2023.
- Escribir marcado con JSX – React. (s. f.). https://es.react.dev/learn/writing-markup-with-jsx Consultado en Marzo 2023.
- Flutter documentation. (s. f.). Flutter. https://docs.flutter.dev/ Consultado en Abril 2023.
- Framework. (26 de Julio, 2022). Framework. Edix. https://www.edix.com/es/instituto/framework/ Consultado en Mayo 2023.
- Gasbarrino, S. (s.f.). MVP: qué es el producto mínimo viable, cómo hacerlo y ejemplos. Hubspot. https://blog.hubspot.es/sales/producto-minimo-viable Consultado en Marzo 2023.
- Hooks integrados de React – React. (s. f.). https://es.react.dev/reference/react Consultado en Octubre 2023.
- Introduction to Ionic | Ionic Documentation. (s. f.). https://ionicframework.com/docs Consultado en Marzo 2022.
- Jain, M. (2022, 7 febrero). Best Folder Structure for React Native Project - Muskan Jain - Medium. Medium. https://medium.com/@techwithmuskan/best-folder-structure-for-react-native-project-c6d7dd6dd494 Consultado en Octubre 2022.
- JavaScript ES6. (s. f.). https://www.w3schools.com/js/js_es6.asp Consultado en Octubre 2022.
- Jest. (s. f.). https://jestjs.io/es-ES/ Consultado en Noviembre 2022.
- Kuhn, J. (2009). Decrypting the MoSCoW analysis. The workable, practical guide to Do IT Yourself, 5.
- Machuca, F. (22 de Mayo, 2022). Backend: ¿qué es y para qué sirve este tipo de programación?. Crehana. https://www.crehana.com/blog/transformacion-digital/que-es-el-backend-y-como-usarlo/ Consultado en Abril 2023.
- Maurya, A. (2022). Running lean. O'Reilly Media, Inc..
- Moniz, E. (2022b, enero 7). Applying SOLID To React - Byborg Engineering - Medium. Medium. https://medium.com/docler-engineering/applying-solid-to-react-ca6d1ff926a4 Consultado en Abril 2023.
- Objetivos de desarrollo sostenible. (25 de Setiembre, 2015). Objetivos de desarrollo sostenible. ONU. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ Consultado en Octubre 2022.

- Overview | React Native Elements. (s. f.). https://reactnativeelements.com/docs Consultado en Abril 2023.
- ¿Qué es React Native? (s. f.). Deloitte Spain. https://www2.deloitte.com/es/es/pages/technology/articles/que-es-react-native.html Consultado en Abril 2023.
- React Native · Learn once, write anywhere. (s. f.). https://reactnative.dev/ Consultado en Marzo 2023.
- React Navigation. (s. f.). https://reactnavigation.org/docs/getting-started/ Consultado en Octubre 2022.
- Testing · React Native. (2023, 12 enero). https://reactnative.dev/docs/testing-overview Consultado en Abril 2023.
- Team, K. (2023, 10 mayo). ¿Qué es un linter en programación? | KeepCoding Bootcamps. KeepCoding Bootcamps. https://keepcoding.io/blog/que-es-un-linter-en-programacion/ Consultado en Mayo 2023.
- Tu primer componente – React. (s. f.). https://es.react.dev/learn/your-first-component Consultado en Octubre 2022.
- use JSDoc: Index. (s. f.). https://jsdoc.app/ Consultado en Marzo 2023.
- useContext – React. (s. f.). https://react.dev/reference/react/useContext Consultado en Noviembre 2022.
- Zanini, A. (2021, 17 mayo). Optimize React apps using a multi-layered structure - LogRocket Blog. LogRocket Blog. https://blog.logrocket.com/optimize-react-apps-using-a-multi-layered-structure/ Consultado en Octubre 2022.

Anexo I: Manual de usuario de la página web

Pantalla principal:

Primera pantalla (ver **Figura anexo 1**) que ve el usuario al entrar a la web. En ella se muestran las distintas actividades que están disponibles y permite seleccionarlas para ver más información sobre ellas e ir a la página de reserva.

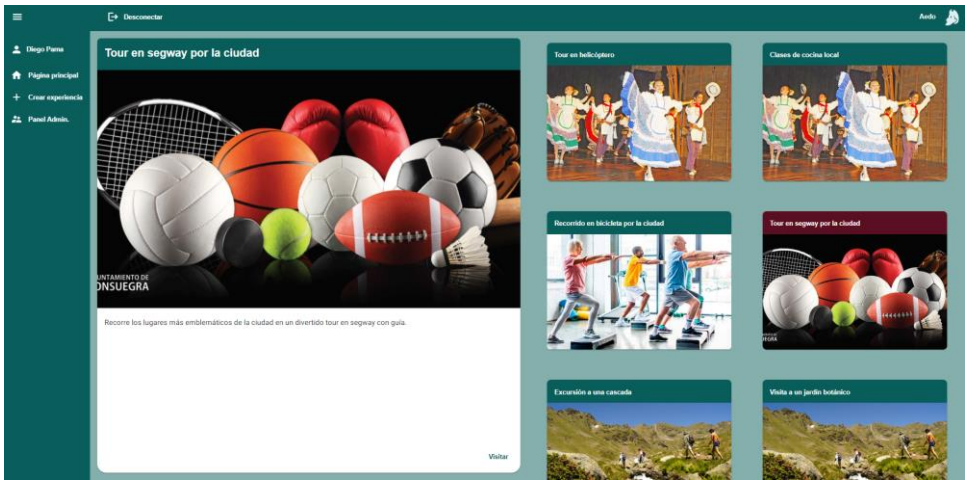

***Figura anexo 1**: Pantalla principal web*

Pantalla de administración:

Pantalla disponible para los usuarios con el rol de administrador, en ella se muestran en distintas funcionalidades de gestión (ver **Figura anexo 2**: Pantalla de administración). No es accesible para otros tipos de usuarios.

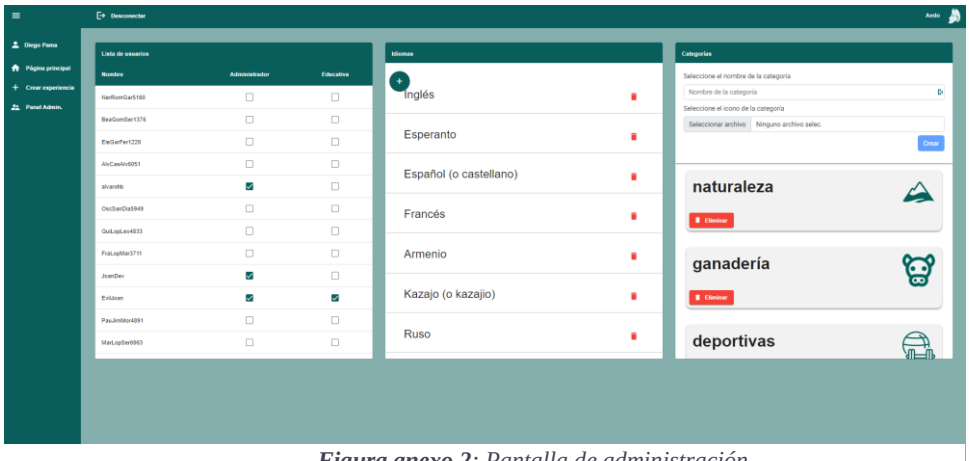

***Figura anexo 2**: Pantalla de administración*

Pantalla de registro de actividades:

Pantalla disponible para los usuarios registrados. Permite introducir un título, una descripción, el aforo máximo, las imágenes, la categoría, el/los idioma/s y la localización (ver **Figura anexo 3** , **Figura anexo 4**, **Figura anexo 5** y **Figura anexo 6**).

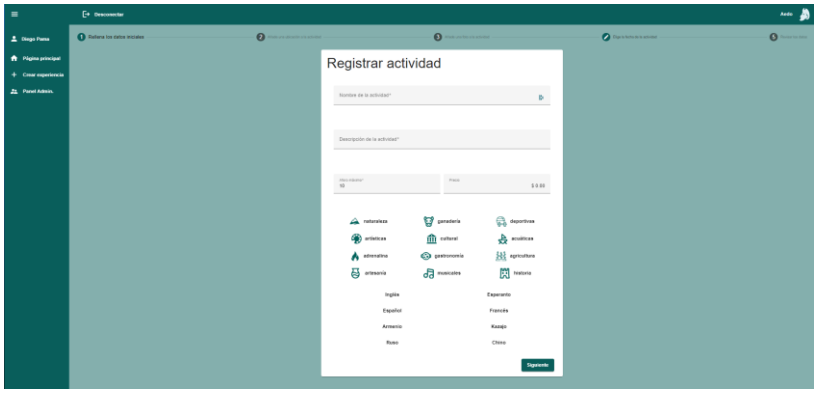

***Figura anexo 3**: Pantalla registro de actividades*

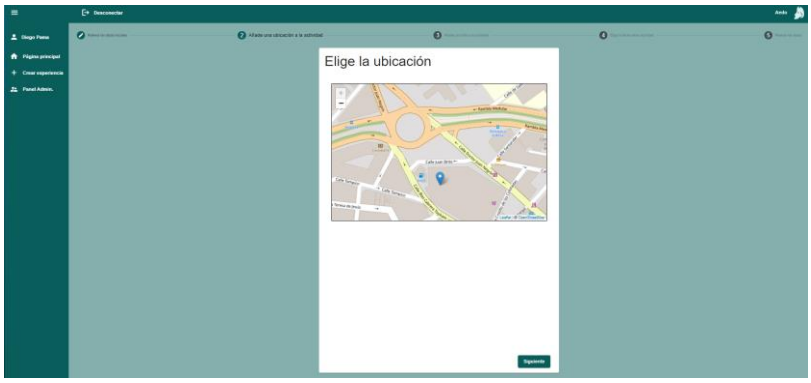

***Figura anexo 4**: Pantalla registro de actividades con mapa*

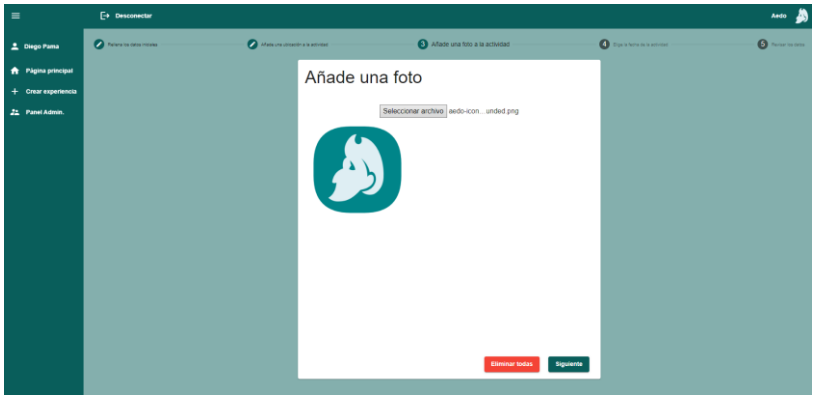

***Figura anexo 5**: Seleccionador de imágenes*

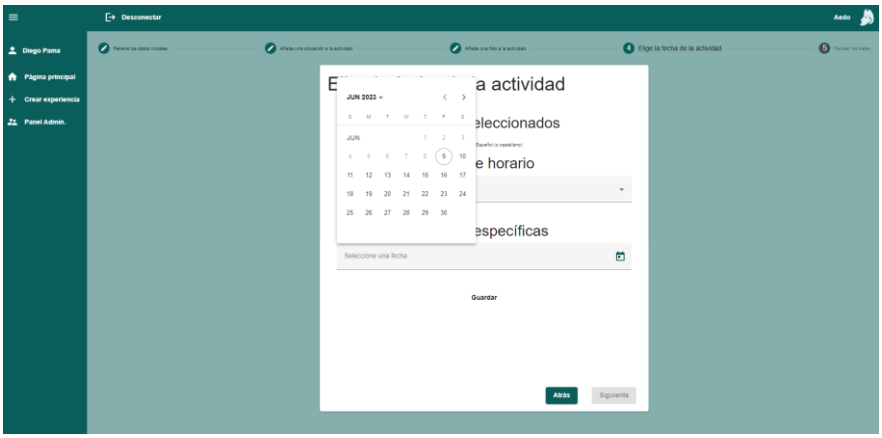

***Figura anexo 6**: Calendario de días*

Pantalla de perfil del usuario:

Muestra la información del usuario que está conectado y a su vez permite la modificación de nombre del usuario, su número de teléfono, su fecha de nacimiento, contraseña y correo (ver **Figura anexo 7**).

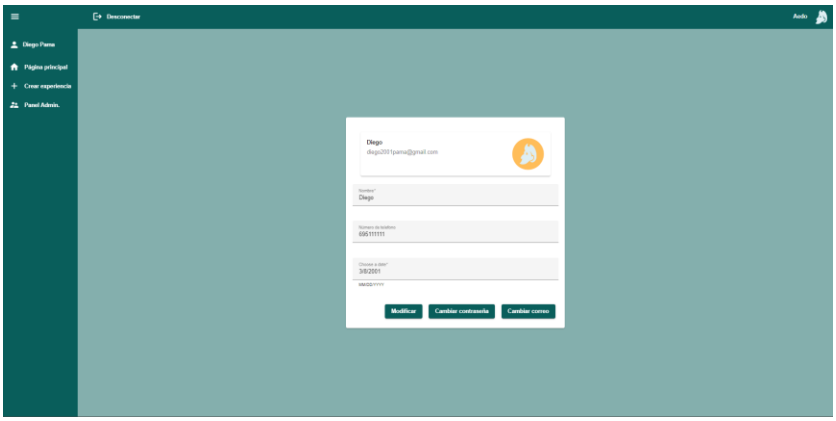

***Figura anexo 7**: Pantalla perfil de usuario*

Pantalla de información de una experiencia:

Aparece la información de la experiencia, así como los comentarios dados por otros usuarios de la aplicación. Además, permite al usuario crear una reserva para dicha experiencia (ver **Figura anexo 8**).

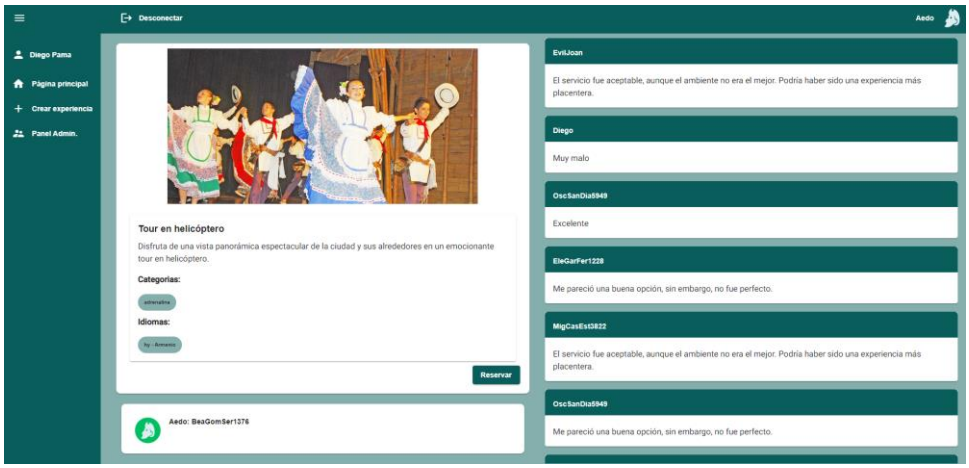

***Figura anexo 8**: Pantalla de información de una experiencia*

Pantalla de inicio de sesión:

Permite el inicio de sesión de usando la información de un usuario registrado y también da acceso a la pantalla de registro de un usuario (ver **Figura anexo 9**).

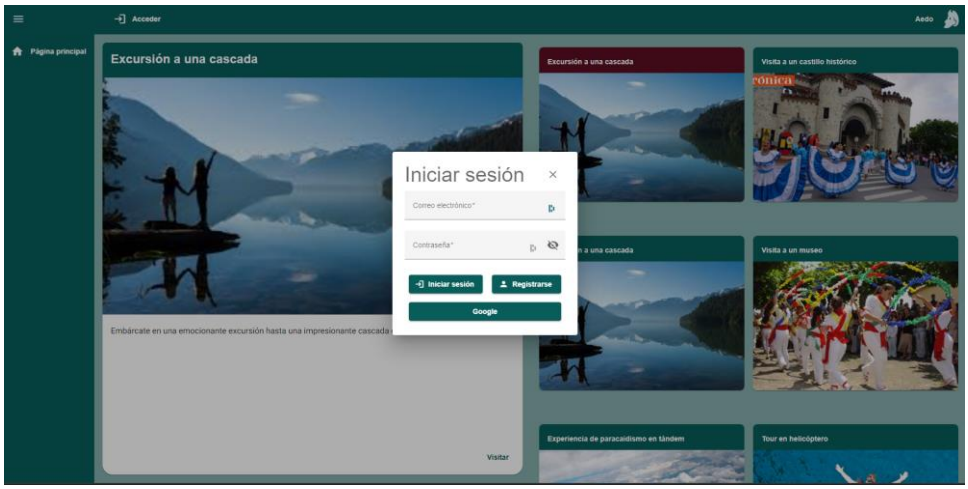

***Figura anexo 9**: Pantalla de inicio de sesión*

Pantalla de registro:

Pide al usuario un nombre de usuario, su nombre real, su número de teléfono, correo electrónico, fecha de nacimiento y contraseña con el fin de registrar el usuario en la aplicación (ver **Figura anexo 10**).

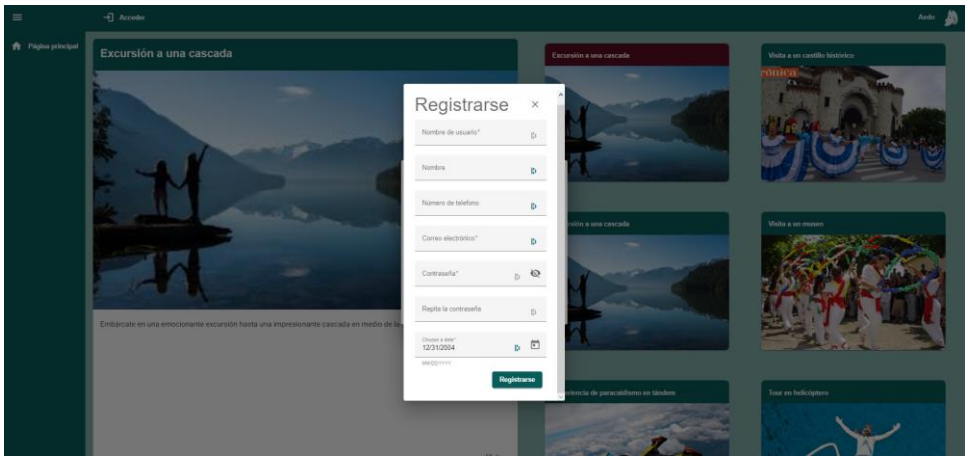

***Figura anexo 10**: Pantalla de registro*

Anexo II: Preguntas de la encuesta a los usuarios

Se realizaron dos encuestas distintas. Una encuesta era para un usuario que fuera a utilizar la aplicación y a realizar una experiencia simple, la otra encuesta iba enfocada al ámbito educativo, permitiendo reservar una experiencia para una clase entera de alumnos. A continuación, se exponen las preguntas generales para un usuario estándar (ver **Tabla anexo 1**). La encuesta "educativa" incluye las preguntas generales y un par de preguntas específicas (ver **Tabla anexo 2**).

***Tabla anexo 1**: Preguntas generales*

Pregunta	Tipo de pregunta
Grado de satisfacción con la aplicación	Escala lineal de 1 a 5
¿Qué posibilidades hay de que recomiendo Aedo a un conocido?	Escala lineal de 1 a 5
¿Considera que fue sencillo aprender a utilizar la aplicación?	Escala lineal de 1 a 5
Considero que la aplicación me informa en cómo proceder cuando existe algún problema.	Escala lineal de 1 a 5
¿Considera que siempre ha sabido como proceder en lo que deseaba hacer?	Escala lineal de 1 a 5
¿Considera que la interfaz de la aplicación es placentera?	Escala lineal de 1 a 5
¿La navegación de la aplicación le pareció intuitiva?	Escala lineal de 1 a 5
¿Considera que la aplicación cumple con las características esperadas de una aplicación para tal fin?	Escala lineal de 1 a 5
¿La velocidad de respuesta de la aplicación le parece adecuada?	Escala lineal de 1 a 5
¿Qué le ha parecido el proceso de creación de experiencias?	Escala lineal de 1 a 5
¿Qué le ha parecido la administración de experiencias y de usuarios apuntados a las mismas?	Escala lineal de 1 a 5

¿Considera que la información es suficiente?	
¿Sugerencias de la aplicación?	Abierta

***Tabla anexo 2**: Preguntas específicas de la encuesta educativa*

Pregunta	Tipo de pregunta
¿Qué le ha parecido el proceso de inscripción a experiencias Educativas?	Escala lineal de 1 a 5
¿Qué le ha parecido la administración de experiencias educativas y de usuarios apuntados a las mismas? ¿Considera que la información es suficiente?	Escala lineal de 1 a 5

Marca temporal	Pregunta 1	Pregunta 2	Pregunta 3	Pregunta 4	Pregunta 5	Pregunta 6	Pregunta 7	Pregunta 8	Pregunta 9	Pregunta 10	Pregunta 11
5/27/2023 18:12:18	5	5	5	4	4	5	4	4	4	5	4
6/10/2023 14:58:26	5	5	5	5	5	5	5	5	5	5	5
6/12/2023 8:42:34	4	4	5	5	5	5	5	4	5	5	5
6/12/2023 8:43:20	5	5	5	3	5	5	5	5	5	5	5
6/12/2023 8:44:06	5	5	4	5	5	5	5	5	5	5	5
6/12/2023 16:32:29	4	4	5	3	4	5	5	4	4	5	4
6/12/2023 16:43:49	5	5	5	4	4	5	5	5	5	5	5
14/06/2023 12:38	4	4	5	4	4	5	5	4	3	5	5
14/06/2023 14:02	4	4	5	4	3	5	4	4	4	4	5
14/06/2023 15:02	5	5	4	5	5	5	4	5	5	5	4
14/06/2023 19:00	4	4	4	4	4	5	4	5	5	5	4
Enunciados	satisfacció	posibilidad	a que fue	que la	a que	a que la	navegación	a que la	velocidad	parecido el	parecido la
sobre 10	9,09	9,09	9,45	8,36	8,73	10,00	9,27	9,09	9,09	9,82	9,27
sobre 5	4,55	4,55	4,73	4,18	4,36	5,00	4,64	4,55	4,55	4,91	4,64
numEncuestados	11										

***Figura anexo 11**: Resultados de la encuesta*

La **Figura anexo 11** muestra las respuestas respondidas por los once encuestados y las puntuaciones medias.

Anexo III: Experimentos realizados

Se realizaron tres experimentos con la aplicación. El primer experimento se realizó durante la Feria de Proyectos de la ETS Inf. Se expuso el primer *MVP* de la aplicación. Durante la experiencia en la feria se extrajo mucha información útil y feedback por parte de los asistentes; ya fueran alumnos, profesores o empresas (ver **Figura anexo 12** y **Figura anexo 13**).

***Figura anexo 12**: Foto de equipo aedo*

***Figura anexo 13**: Foto del stand*

El segundo experimento se realizó fuera del recinto universitario, directamente en Játiva con un agricultor que mostró interés en la aplicación. Se visitó el campo de trabajo y nos explicó que la aplicación le parecería útil para enseñar a los usuarios/as de qué manera se trabajan las hortalizas (ver **Figura anexo 14**, **Figura anexo 15**, **Figura anexo 16**).

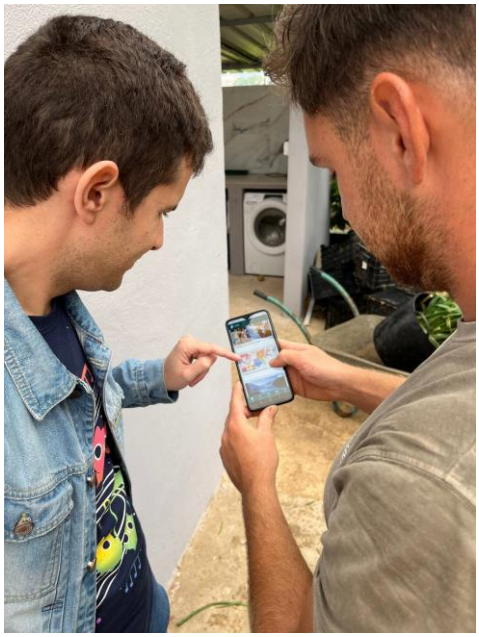

***Figura anexo 16**: Uso de la aplicación con agricultor 1*

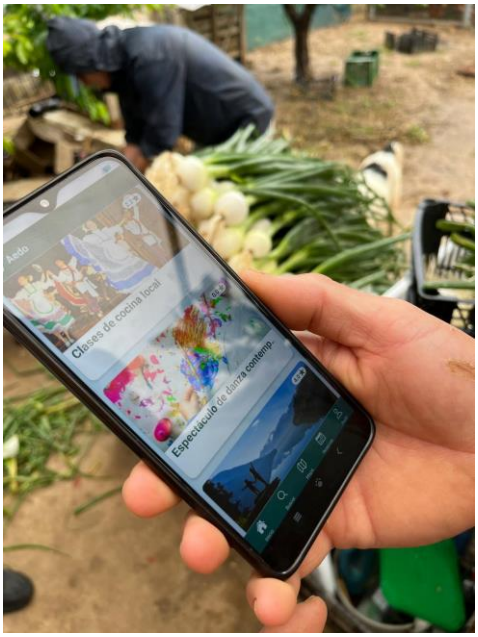

***Figura anexo 15**: Uso de la aplicación con agricultor 2*

***Figura anexo 14**: Explicación del contenido de la experiencia*

Tras la visita del campo, se nos acompañó a la tienda del barrio donde el agricultor comercializa con el producto kilómetro cero directamente desde el campo (**Figura anexo 17** y **Figura anexo 18**).

***Figura anexo 17**: Tienda del agricultor 1*

***Figura anexo 18**: Tienda del agricultor 2*

Para el tercer, y último, experimento se contactó con un profesor de historia y se le planteó la posibilidad de utilizar la aplicación para poder inscribir a su clase de alumnos a las experiencias ofrecidas en la plataforma (ver **Figura anexo 19**).

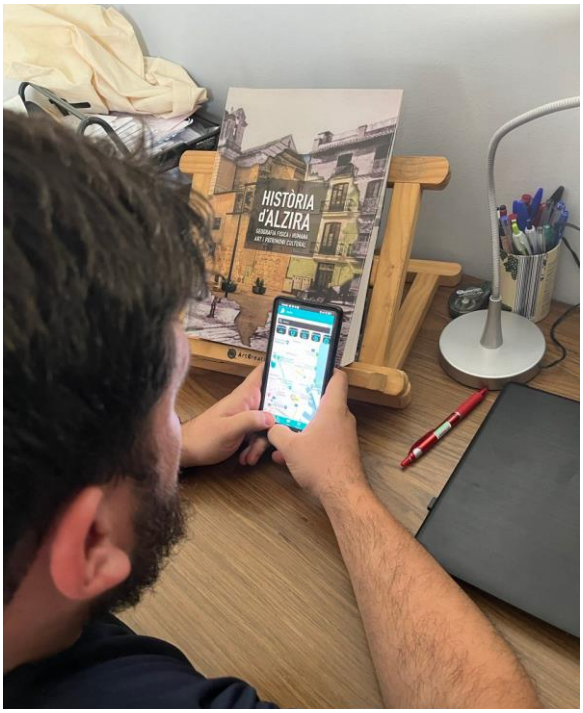

***Figura anexo 19**: Uso de la aplicación con un profesor*

Anexo IV: Objetivos de desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS) (ver **Tabla anexo 3**).

***Tabla anexo 3**: Grado de relación con ODS*

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

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

La aplicación Aedo está ampliamente alineada con tres importantes Objetivos de Desarrollo Sostenible de las Naciones Unidas: Salud y bienestar (Objetivo 3), Igualdad de género (Objetivo 5) y Ciudades y comunidades sostenibles (Objetivo 11).

- Objetivo 3: Salud y bienestar: Aedo fomenta el compartir experiencias entre usuarios/as, muchas de las experiencias son al aire libre y repercuten en un incremento de la salud, debido a la motivación por parte del equipo de Aedo en incentivar el movimiento de personas a sitios completamente distintos a los que uno puede estar acostumbrado.

- Objetivo 5: Igualdad de género: Aedo no solo permite que cualquier usuario/as, sin importar su género, se convierta en mentor de experiencias, sino que también promueve activamente la igualdad de género en todas las áreas. La aplicación destaca y promociona experiencias y logros realizados tanto por hombres como por mujeres, brindando visibilidad a personas de todos los géneros. Es por lo que Aedo favorece la igualdad de género. Gracias a la aplicación se pueden compartir, por ejemplo, trabajos realizados tanto por hombres como por mujeres. Además, asociaciones de mujeres o relacionados con la igualdad de género de los diferentes pueblos pueden hacer uso de Aedo para potenciar y publicitar sus actividades culturales.

- Objetivo 11: Ciudades y comunidades sostenibles: La aplicación Aedo tiene como objetivo primordial fomentar la descentralización del turismo en las grandes ciudades y trasladarlo a zonas menos conocidas, pero con la misma capacidad de generar interés en los turistas. Teniendo en cuenta esa premisa, Aedo "obliga" a los usuarios/as consumidores de experiencias a desplazarse. Muchas veces la ubicación de las experiencias serán localidades de la España vacía, ayudando a la economía y las personas locales de dichas poblaciones. Aedo se esfuerza por no solo descentralizar el turismo, sino también por promover la sostenibilidad y el desarrollo equilibrado de las comunidades locales. Además, Aedo fomenta la colaboración con organizaciones locales y pequeñas empresas para ofrecer experiencias auténticas y sostenibles a los usuarios/as. De esta manera, Aedo ayuda a evitar la sobrecarga turística en áreas urbanas y fomenta un turismo más responsable y consciente, que a su vez contribuye a la preservación del patrimonio cultural y natural de las comunidades visitadas.

La aplicación puede estar en menor medida relacionada con los objetivos 4 (Educación de calidad), 10 (reducción de las desigualdades), 13 (Acción por el clima), 14 (Vida submarina) y 15 (Vida de ecosistemas terrestres). La razón de este resquicio reside en que se cumplan o no los objetivos recae en el uso que dan los usuarios/as a la aplicación; por ejemplo: si un usuario/a promueve acciones por el clima, entonces la aplicación se relaciona con el ODS número 13.

En conclusión, la aplicación Aedo está alineada con los Objetivos de Desarrollo Sostenible de las Naciones Unidas al promover la salud y el bienestar a través del intercambio de experiencias al aire libre, fomentar la igualdad de género al permitir que todos los usuarios/as compartan sus trabajos y logros, y contribuir a ciudades y comunidades sostenibles al descentralizar el turismo y apoyar el desarrollo equilibrado y sostenible en áreas menos conocidas. Además, aunque en menor medida, la aplicación Aedo puede tener relación con otros Objetivos de Desarrollo Sostenible, como la educación de calidad, la reducción de las desigualdades, la acción por el clima, la protección de la vida submarina y la conservación de los ecosistemas terrestres, dependiendo del uso que los usuarios/as hagan de la plataforma y de las acciones que promuevan a través de ella.