Travel Diary

Memoria

VNiVERSiDAD
D SALAMANCA

Trabajo de fin de Grado
Grado en Ingeniería Informática

Tutor/es
Iván Álvarez Navia

Alumno
Jorge García Prieto

Año de presentación
Septiembre de 2023

Certificado del/los tutor/es TFG

D./Dña. Iván Álvarez Navia, profesor/a del Departamento de Informática y Automática de la Universidad de Salamanca,

HACE/N CONSTAR:

Que el trabajo titulado "Travel Diary", que se presenta, ha sido realizado por Jorge García Prieto, con DNI ****1054F y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Ingeniería Informática en esta Universidad.

Salamanca, __ de _____ de 20____

Fdo.: _____

Resumen

Durante el año 2022, más de 900 millones de personas realizaron viajes internacionales, siendo esto todavía, un 37% menor a los movimientos de personas realizados previo a la pandemia SARS-CoV-2 acontecida principalmente en el año 2020. Si lo analizamos a nivel económico, el movimiento de personas representó 7,4 billones de dólares estadounidenses, es decir, un 6,1% del PIB mundial corresponde al turismo. Estas cifras mostradas, hablan por si solas de la influencia que tiene el turismo a nivel mundial, estando todavía recuperándose a niveles normales pre-pandemia.

La torre Eiffel, en París, Francia, recibió durante el año 2022 casi 5,8 millones de visitantes. Casi 7 millones de personas visitaron el Coliseo Romano en Roma. Alrededor de 4,8 millones de visitantes deciden subir al mirador del Empire State Building cada año.

Los datos previamente expuestos, muestran que uno de los principales hobbies del ser humano consiste en viajar a diferentes lugares a lo largo del mundo, recorriendo calles, accediendo a edificios históricos o simplemente probando la comida local. Es por esto, que la aplicación Travel Diary cobra sentido, pues proporciona a los usuarios un diario de viaje desde el que se puede tener organizado todo el itinerario de un viaje, desde próximas actividades, imágenes tomadas durante el viaje, billetes de medios de transporte o de las mismas actividades e incluso destacar diferentes momentos del viaje para que los usuarios tengan la posibilidad de incluir notas. Por último, esta aplicación hará uso de su sistema de notificaciones para recordar a sus usuarios del comienzo de una de sus actividades, e incluso si estos han decidido adjuntar el billete para la actividad en la aplicación, al pulsar sobre la notificación, podrán visualizar el billete de forma automática.

Summary

During the year 2022, more than 900 millions of people traveled internationally, meaning this number is still a 37% lower than the usual number of tourists before the SARS-CoV-2 pandemic suffered during the year 2020. Economic wise, tourism generated almost 7,4 billion of United States Dollars, a 6,1 of the global GDP. Those previous numbers speak for themselves about the influence of tourism globally, even while recovering to pre-pandemic numbers.

Eiffel tower, Paris, France received almost 5,8 millions of visitors during 2022. Around 7 million tourists visited the Roman Colosseum in Rome. The Empire State building receives every year around 4,8 millions of visitors too.

Data previously exposed was just an example of how many people has traveling as one of their favorite hobbies, visiting historical buildings, just walking through new cities of even just trying new tastes and food. For those reasons, the app Travel Diary makes sense enabling users to easily manage their itinerary, next destinations, planned activities, images, tickets and even highlight some moments for the eternity. At last, the app will use its notification system to inform the users if an activity starts soon, even clicking over the notification will open the ticket if they decided to upload it as more information for the activity.

Palabras clave

Turismo, viajes, diario, planificar, gestión.

Keywords

Tourism, travel, diary, management.

Tabla de contenido

Tabla de ilustraciones

Tabla de tablas

1. Introducción

El presente documento recoge la memoria del Trabajo de Fin de Grado titulado 'Travel Diary' en la titulación del Grado en Ingeniería Informática. Ha sido realizado por el alumno Jorge García Prieto durante el curso 2022-2023 y ha sido dirigido por D. Iván Álvarez Navia.

En los tiempos en los que vivimos el uso de las tecnologías es algo fundamental y necesario para la realización de gran cantidad de nuestras actividades del día a día. Cada vez más acciones que se realizaban a mano, escritas en folios, cuadernos, pequeñas agendas o 'stickers' han pasado a realizarse de forma digital en teléfonos móviles, tabletas u ordenadores.

Gracias a todas estas tecnologías los seres humanos hemos pasado de comprar un cuaderno para cada asignatura, a llevar tu tableta u ordenador para escribir todo en las diferentes aplicaciones disponibles en las tiendas de aplicaciones. Se ha pasado de imprimir los billetes en papel para acceder a diferentes sitios, a llevar un código QR en tu dispositivo móvil, escanearlo y acceder al mismo sitio para el que antes tus padres hubiesen impreso el billete. Estos no son más que meros ejemplos que el avance de las tecnologías han permitido.

Volviendo un poco al pasado, y para hilarlo con el avance de las tecnologías, me encontraba de viaje en Edimburgo. Como todo turista suele hacer, fui a visitar el Castillo de Edimburgo. Al comprar el billete, venía con un código de barras, que habría que escanear al entrar, como muchas personas hacen, guardé este documento en una conversación conmigo mismo de 'WhatsApp'. Fue ahí cuando me decidí a realizar una aplicación que permitiese almacenar tus billetes y te notificase al comenzar una actividad permitiendo abrir el documento instantáneamente. Con el avance de las tecnologías esto se ha ido recogiendo en 'Google Wallet' o 'Apple Wallet' que almacenan los billetes y notifican previamente, como había ideado, aunque en muchos sitios web solo te dejan descargar estos billetes para almacenarlos para el sistema operativo de tu dispositivo (si tienes iPhone para iOS, si tienes otro teléfono, te permite descargar únicamente el billete para Android). Esto si vas con un grupo de amigos sigue sin ser óptimo. Precisamente por el perfeccionamiento estas aplicaciones 'cartera', se decidió a comienzos de curso, ampliar la aplicación a un diario de viaje, que permita al usuario a gestionar y organizar el viaje de una forma sencilla, donde en una sola aplicación pueda crear el viaje, añadir sus actividades, documentos, imágenes del viaje e incluso le permita tomar notas de algún momento en concreto.

El objetivo principal es el desarrollo de una aplicación móvil que permita a los usuarios toda la información sobre un viaje que ellos deseen. Desde esta, una vez que los usuarios se hayan registrado e iniciado sesión, podrán tanto gestionar sus próximos viajes, como observar las imágenes de viajes anteriores, notas que decidiesen añadir, etc.

En este documento se recogen los detalles de su desarrollo y las posibles líneas de futuro que podría seguir el proyecto.

Los puntos que va a seguir el documento serán los siguientes:

- **Introducción**
  Este punto es la sección actual donde se indica al lector la temática del proyecto y la razón de su realización.

- **Objetivos**
  En este punto se han descrito los objetivos que se van a tratar de conseguir durante la realización del proyecto.

- **Conceptos teóricos, técnicas y herramientas utilizadas**
  En este, se han descrito diferentes conceptos de importancia, como son los lenguajes de programación, herramientas, etc. utilizados durante el desarrollo del proyecto.

- **Aspectos relevantes del desarrollo**
  Se comentan los aspectos más relevantes durante el desarrollo del sistema

- **Conclusiones**
  Se comentan las conclusiones tras la realización del proyecto

- **Líneas de trabajo futuras**
  Se describen posibles ampliaciones y mejoras del proyecto

- **Referencias**
  Se enumeran las citas empleadas en el documento

Además, a parte de este documento, se adjunta también el código, y la documentación técnica que está compuesta por los siguientes anexos:

- **Anexo I. Planificación temporal del proyecto software**
  En este documento se realiza la planificación del proyecto en diferentes Sprints, comentando finalmente la consecución de los objetivos para cada uno de estos.

- **Anexo II. Especificación del Product Backlog**
  En este anexo se comentan las diferentes Historias de Usuario, junto a sus respectivas tareas para completar cada Historia. Adicionalmente, se ha descrito la jerarquía de actores, realizado el modelo de dominio de la aplicación, etc.

- **Anexo III. Diseño del sistema software**
  Apoyándonos sobre el Anexo II se ha elaborado la documentación de la estructura de proyecto.

- **Anexo IV. Manual de programador**
  Recoge la documentación referente al código desarrollado, dando una explicación y facilitando la comprensión de este.

- **Anexo V. Manual de usuario**
  Este anexo tiene como objetivo servir de guía de uso de la aplicación. Se recogen todas las acciones que el usuario podrá realizar en la aplicación.

2. Objetivos

El objetivo final del proyecto es el diseño e implementación de una aplicación Android que permita a los usuarios tener un diario de viaje que permita poder incluir sus actividades, futuros viajes, fotos, etc. De forma sencilla.

2.1 Objetivos funcionales

A lo largo del proyecto se tratarán de cumplir diferentes objetivos software:

- **Gestión de usuarios**: el sistema deberá ser capaz de permitir a los usuarios registrarse en ella sin dificultad. Permitiendo únicamente el acceso a esta únicamente a usuarios registrados. Se mostrará únicamente la información de cada usuario específico.

- **Gestión de viajes**: el sistema deberá ser capaz de añadir, modificar y eliminar diferentes viajes. Estos estarán asociados a un usuario, que podrá consultar su información en cualquier momento (habiendo iniciado sesión primero)

- **Gestión de actividades**: el sistema deberá ser capaz de añadir, modificar y eliminar diferentes actividades. Estas estarán relacionadas con el usuario y también con un viaje y día en concreto. Adicionalmente el sistema deberá controlar la inclusión de archivos a la actividad para gestionar notificaciones para esta actividad.

- **Gestión de Imágenes**: el sistema deberá ser capaz de permitir incluir imágenes a las actividades y por supuesto, de eliminar estas actividades. También se podrá marcar imágenes como favoritas. Por último, se creará una galería con las imágenes incluidas por el usuario.

- **Gestión de notificaciones**: el sistema deberá notificar a los usuarios cuando una actividad esté a punto de comenzar (el tiempo previo para notificar será configurable por el usuario). En caso de existir un archivo adjunto a la actividad, el sistema deberá reconocerlo y cambiar la notificación para permitir abrir este documento al seleccionar sobre esta.

- **Gestión de archivos:** el sistema deberá permitir añadir, modificar y eliminar archivos en formato PDF. Si el usuario selecciona sobre este archivo, el sistema deberá permitir al usuario visualizar el contenido de dicho archivo.

2.2 Objetivos no funcionales

A lo largo del proyecto se tratarán de completar distintos objetivos técnicos:

- **Tiempo de respuesta**: el sistema deberá responder dentro de un tiempo aceptable, intentando satisfacer al usuario lo más rápido posible dando, además, la sensación de fluidez.

- **Usabilidad**: el sistema será fácil e intuitivo de utilizar. Con esto se busca lograr la mayor satisfacción posible del usuario.

- **Concurrencia**: el sistema deberá ser capaz de funcionar sin errores ni problemas, sin importar el número de personas utilizándolo al mismo tiempo.

- **Bases de datos**: se tendrán que utilizar peticiones a la base de datos de Firebase para obtener la información solicitada por el usuario al acceder a las diferentes pantallas de la aplicación.

- **Mantenibilidad**: el sistema deberá cumplir con este requisito para minimizar los esfuerzos a la hora de conservar el correcto funcionamiento de este a futuro.

- **Escalabilidad**: El sistema deberá ser capaz de adaptarse a la infraestructura de cualquier tamaño. Se podrán añadir nuevas funcionalidades con el crecimiento del sistema.

2.3 Personales

El desarrollo del proyecto también supondrá una serie de objetivos personales:

- Introducción a la industria del desarrollo orientado a dispositivos móviles: a través del aprendizaje para desarrollar para el sistema operativo Android se va a conocer más sobre el funcionamiento de esta industria, así como de '*frameworks*' y librerías que faciliten el desarrollo.

- Conocimiento de lenguajes de programación: aprendizaje de lenguajes no utilizados durante el grado como JavaScript o TypeScript

- Utilización del entorno de Google en bases de datos y gestión de usuarios: la aplicación hace uso de la base de datos de Firebase además de realizar la gestión de usuarios de esta librería, esto permite aprender sobre otra herramienta que no se ha tratado en el Grado, lo que añade nuevas capacidades para el mercado laboral.

- Aprendizaje Scrum: la metodología ágil Scrum es utilizada en muchas empresas por lo que se tratarán de aprender sus conceptos.

3. Conceptos teóricos

A lo largo del documento se van a utilizar diferentes términos que puede ser conveniente aclarar en este punto, para que no sea necesario ir aclarándolos a lo largo del documento.

Estos conceptos son los siguientes:

3.1 Aplicación Móvil

Una aplicación móvil, es una aplicación diseñada específicamente para dispositivos móviles o tabletas. Incluso siendo estas aplicaciones de tamaño limitado, proporcionan al usuario funcionalidad aislada, por ejemplo, la aplicación calculadora, sirve para realizar cálculos matemáticos, no va a permitirte abrir una red social. La principal ventaja de los dispositivos móviles radica en su tienda de aplicaciones donde los usuarios podrán seleccionar las aplicaciones necesarias para sus necesidades en el día a día [1].

Generalmente, el desarrollo de una aplicación móvil, y en este caso concreto, una aplicación Android, se realiza en el IDE proporcionado por Google, Android Studio, del que se hablará en mayor profundidad más adelante, en los lenguajes Kotlin o Java. El 'framework' de Ionic y posteriormente la herramienta Capacitor (como ocurre con Android Studio, más adelante se hablará en mayor profundidad sobre ambos), permiten desarrollar una aplicación Android como si se tratase de un desarrollo web, utilizando Visual Studio Code para codificar en lenguajes JavaScript y TypeScript. Una vez finalizada la codificación, Capacitor realizará la portabilidad del proyecto web a un proyecto de Android Studio, consiguiendo de esta manera la obtención de una aplicación Android.

3.1.1 Componentes de una aplicación Android

Los componentes de la aplicación son bloques de creación esenciales de una aplicación para Android. Cada componente es un punto de entrada por el que el sistema o un usuario ingresan a la aplicación. Algunos componentes dependen de otros [2].

Actividades

Una actividad es el punto de entrada de interacción con el usuario. Representa una pantalla individual con una interfaz de usuario.

Por ejemplo, en el proyecto, existen varias de estas pantallas donde el usuario puede introducir datos para crear nuevos viajes, actividades, etc.

Servicios

Un servicio es un punto de entrada general que permite mantener la ejecución de una aplicación en segundo plano por diversos motivos. Es un componente que se ejecuta en segundo plano para realizar operaciones de ejecución prolongada o para realizar tareas de procesos remotos. Un servicio no proporciona una interfaz de usuario.

En el proyecto, esto se ve reflejado cuando un usuario graba un mensaje de voz en un momento destacado. En el momento en el que desea escucharlo, este se reproduce incluso saliendo de la aplicación, en un segundo plano.

Receptores de emisiones

Un receptor de emisiones es un componente que posibilita que el sistema entregue eventos a la aplicación fuera de un flujo de usuarios habitual, lo que permite que la aplicación responda a los anuncios de emisión de todo el sistema. Por ejemplo, una aplicación puede programar una alarma para publicar una notificación sobre un evento futuro destinado al usuario (el ejemplo se ajusta a lo que la aplicación realiza cuando un usuario crea una actividad e indica su hora de comienzo).

Proveedores de contenido

Un proveedor de contenido administra un conjunto compartido de datos de la aplicación que puedes almacenar en un sistema de archivos, en una base de datos SQLite, en la Web o en cualquier otra ubicación de almacenamiento persistente que tenga acceso a la aplicación.

En este caso, el proveedor de contenido sería **Firebase** de Google, qué actúa de Base de Datos, proporciona la autenticación al sistema, etc.

Ilustración 1: Aplicaciones móviles

3.2 Metodologías ágiles

Las metodologías ágiles son aquellas que adaptan la forma de trabajar a las condiciones del proyecto, de esta forma se consigue una mayor flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las especificaciones del entorno y sus circunstancias.

Ejemplos de metodología ágil son XP (Extreme Programming), Scrum, Kanban, Agile Inception o Design Sprint [3].

Estas metodologías surgen a raíz del 'Manifiesto Ágil' (ver Ilustración 2), este incluye los valores sobre los que se asientan estos métodos y fue creado en 2001 por 17 profesionales del software críticos con los modelos de producción basados en procesos. Estos valores son [4]:

- Los individuos y su interacción por encima de los procesos y las herramientas.
- El software funcionando por encima de los procesos y las herramientas
- Colaboración con el cliente por encima de la negociación contractual
- Respuesta al cambio por encima del seguimiento de un plan.

Estos valores se asientan sobre principios más detallados:

- La prioridad principal es satisfacer al cliente mediante tempranas y continuas entregas de software que le reporten un valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Se aprovecha el cambio para dar una ventaja competitiva al cliente.
- Entregas frecuentes de software funcional, desde un par de semanas a un par de meses, con el menor intervalo posible entre entregas.
- Los responsables del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y apoyo necesario y confiarles la ejecución del trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
- Las mejores arquitecturas, requisitos y diseños surgen de equipos organizados por sí mismos.

- El software que funciona es la medida principal de progreso.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- A intervalos regulares el equipo reflexiona sobre como ser más efectivo para, a continuación, ajustar y perfeccionar su comportamiento en concordancia.

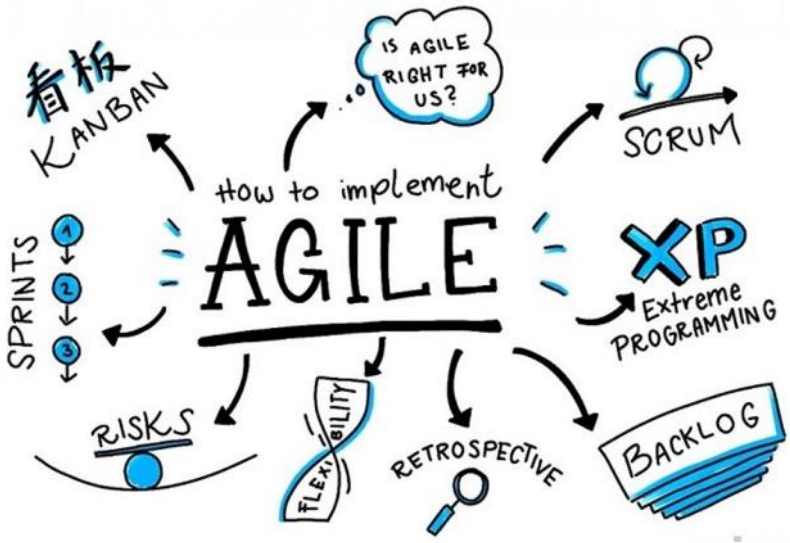

Ilustración 2: Manifiesto ágil [5]

3.3 Scrum

Scrum, según la propia guía Scrum es un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente [6].

Scrum es:

- Liviano
- Fácil de entender
- Difícil de llegar a dominar

No es un proceso o una técnica para construir productos sino un marco de trabajo dentro del cual se pueden aplicar varios procesos y técnicas.

3.3.1 Teoría de Scrum

Scrum está basado en la teoría de control de procesos empírica o empirismo. Asegura que el conocimiento procede de la experiencia y de tomar decisiones a raíz de lo que se conoce. Emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo.

Está pasado en tres pilares fundamentales.

Transparencia

Todos los aspectos significativos deben ser visibles para los responsables del resultado. Es decir, se requiere que dichos aspectos sean definidos por un estándar común para que los observadores tengan un entendimiento común.

Inspección

Los usuarios de Scrum han de inspeccionar los artefactos Scrum y el progreso para detectar variaciones indeseadas. No deben realizarse inspecciones tan frecuentemente como para que interfiera en el trabajo.

Adaptación

Si se detecta que algún aspecto se desvía de límites aceptables, el proceso deberá ajustarse. Este ajuste tendrá que realizarse cuanto antes para minimizar posibles desviaciones mayores.

3.3.2 Equipo Scrum

El Equipo Scrum o '*Scrum Team' (ver* Ilustración 3*)* es auto organizado y multifuncional, eligen la forma de realizar el trabajo y no son dirigidos por personas ajenas al equipo. Tiene todas las competencias necesarias para llevar a cabo su trabajo y está diseñado con el objetivo de optimizar la flexibilidad, la creatividad y la productividad [7].

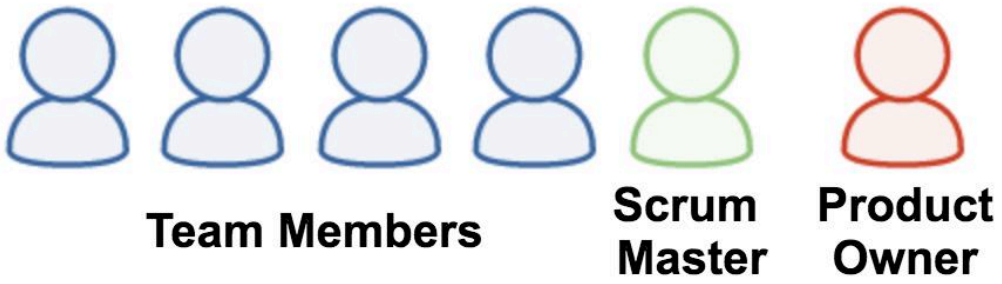

Ilustración 3: Equipo Scrum

Dueño de Producto (Product Owner)

El Dueño de Producto es el responsable de maximizar el valor del producto. Es la única persona responsable de gestionar la Lista del Producto. Dicha gestión incluye:

- Expresar claramente los elementos de la Lista del Producto
- Ordenar los elementos para alcanzar los objetivos
- Optimizar el valor del trabajo
- Asegurar que es visible, transparente y clara
- Asegurar que el Equipo de Desarrollo entiende los elementos

Toda la organización debe respetar sus decisiones que se reflejan en el contenido y la priorización de la Lista de Trabajo.

Equipo de Desarrollo (Development Team)

El Equipo de Desarrollo está formado por los profesionales que realizan el trabajo de entregar un Incremento de producto que cumpla con la definición de "Terminado".

- **Auto organizado**: nadie indica cómo convertir elementos de la Lista del Producto en Incrementos.
- **Multifuncionales**: cuentan con todas las habilidades para crear un Incremento.
- **No reconoce títulos**: dentro del equipo, todos son Desarrolladores.
- **No reconoce sub-equipos**
- **Pueden tener habilidades especializadas**

El tamaño óptimo del Equipo de Desarrollo es suficientemente pequeño para permanecer ágil y lo suficientemente grande como para completar una cantidad de trabajo significativa, generalmente suele ser de tres a nueve personas.

Scrum Master

Es el responsable de asegurar que Scrum se entienda y se adapte. También es el encargado de dar servicio al Dueño del Producto. Sus funciones serán las siguientes:

- Encontrar técnicas para gestionar la Lista de Producto.
- Ayudar al Equipo de Desarrollo a entender la necesidad de contar con elementos de la Lista de Producto claros y concisos
- Entender la planificación
- Asegurarse de que el Dueño del Producto sepa cómo ordenar la Lista de Producto
- Entender y practicar la agilidad
- Facilitar los eventos Scrum según se necesite
- Guía del Equipo de Desarrollo
- Motivar cambios que incrementen la productividad

3.3.3 Eventos Scrum

Existen eventos predefinidos con el objetivo de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Una vez comienza un Sprint tendrá una duración máxima definida y no podrá modificarse [8].

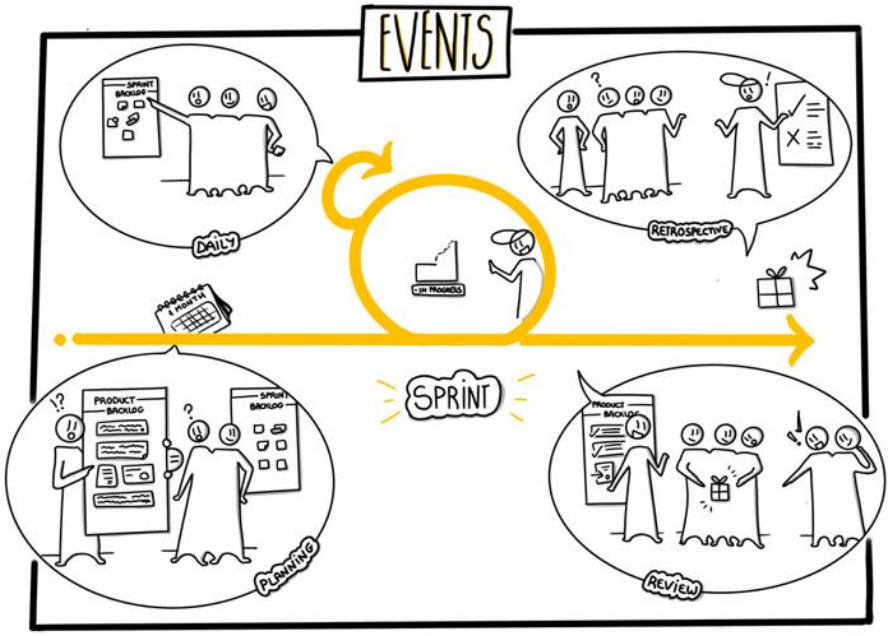

Ilustración 4: Eventos Scrum

Sprint

Es un bloque de tiempo aproximadamente de un mes o menos en el que se crea un Incremento de producto "Terminado". Cada Sprint comienza inmediatamente después de la finalización del anterior.

Contienen y consisten en la Planificación del Sprint, los Scrums Diarios, el trabajo de desarrollo, la Revisión del Sprint y la Retrospectiva del Sprint.

Durante el Sprint:

- No se realizan cambios que puedan afectar al Objetivo del Sprint.
- Los objetivos de calidad no disminuyen.
- El alcance puede clarificarse y renegociarse entre el Dueño del Producto y el Equipo de Desarrollo a medida se va aprendiendo.

Los Sprints habilitan la predictibilidad al asegurar inspección la inspección y adaptación del progreso al menos en cada mes calendario, limitando también el riesgo.

Planificación del Sprint (Sprint Planning)

La Planificación del Sprint tiene un máximo de duración de ocho horas. El Scrum Master enseña al Equipo Scrum a mantenerse dentro del bloque de tiempo.
Este evento responde a las preguntas:

- ¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?
- ¿Cómo se conseguirá el trabajo necesario para la entrega del Incremento?

Al finalizar la Planificación del Sprint, el Equipo de Desarrollo tiene que ser capaz de explicar al Dueño de Producto y al Scrum Máster cómo va a trabajar para lograr el Objetivo del Sprint y crear el Incremento esperado.

Objetivo del Sprint (Sprint Goal)

El Objetivo del Sprint es una meta establecida que puede lograrse mediante la implementación de la Lista de Producto.

A medida que el Equipo de Desarrollo trabaja, mantiene el Objetivo del Sprint en mente con el fin de satisfacerlo. Si el trabajo acaba siendo diferente, ellos colaboran con el Dueño de Producto para negociar el alcance de la Lista de Pendientes del Sprint.

Scrum Diario (Daily Scrum)

El Scrum Diario es una reunión de quince minutos para que el Equipo de Desarrollo sincronice sus actividades y cree un plan para las siguientes veinticuatro horas.
En estas reuniones, cada miembro explica:

- ¿Qué hice ayer que ayudara al Equipo de desarrollo a lograr el Objetivo del Sprint?
- ¿Qué haré hoy para ayudar al Equipo de Desarrollo a lograr el Objetivo del Sprint?
- ¿Hay algún impedimento que no permita al Equipo de Desarrollo alcanzar el Objetivo del Sprint?

Los Scrum Diarios mejoran la comunicación, eliminan la necesidad de realizar otras reuniones, identificar impedimentos que haya que eliminar posteriormente, resaltan y promueven la toma de decisiones rápida y mejoran el nivel de conocimiento del Equipo de Desarrollo.

Revisión del Sprint (Sprint Review)

Al final del Sprint se lleva a cabo una revisión para inspeccionar el Incremento y adaptar la Lista de Producto si se diera el caso.

- Está restringida a cuatro horas.
- Los asistentes son el Equipo Scrum y los interesados clave invitados por el Dueño de Producto.
- Se explican qué elementos de la Lista de Producto se han "Terminado" y cuáles no.
- El Equipo de Desarrollo habla acerca de qué estuvo bien durante el Sprint, qué problemas surgieron y cómo se resolvieron.
- El Equipo de Desarrollo hace una demostración del trabajo "Terminado" y responde preguntas.
- El Dueño de Producto habla sobre la Lista de Producto.
- El grupo completo trata sobre qué hacer a continuación como puntos de entrada para Reuniones de Planificación de Sprint siguientes.
- Revisión tanto de cómo el mercado o uso potencial podría haber cambiado y de la línea de tiempo, presupuesto, etc.

Retrospectiva del Sprint (Sprint Retrospective)

La Retrospectiva del Sprint es una oportunidad para que el Equipo Scrum se inspeccione a sí mismo y elabore un plan de mejoras de cara al siguiente Sprint.

Tiene una duración máxima de tres horas y se trata lo siguiente:

- Análisis del último Sprint (personas, relaciones, procesos y herramientas).
- Identificar y ordenar los elementos más importantes que salieron bien y posibles mejoras a futuro.
- Crear un plan para implementar las mejoras a la forma que el Equipo Scrum desarrolla su trabajo.

3.3.4 Artefactos Scrum

Los artefactos Scrum representan trabajo o valor de una manera útil para proporcionar transparencia y oportunidades para la inspección y la adaptación del proyecto [9].

Ilustración 5: Artefactos Scrum

Lista de Producto (Product Backlog)

La Lista de Producto es una lista ordenada de todo lo que podría ser necesario en el producto y es la única fuente de requisitos para cualquier cambio a realizarse en el producto.

Algunas de sus características más importantes son:

- Nunca está completa, es decir, evoluciona.
- Enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para futuras entregas.
- Los elementos de orden más alto son generalmente más claros y detallados que los de menor orden.

Lista de Pendientes del Sprint (Sprint Backlog)

La Lista de Pendientes del Sprint es el conjunto de elementos de la Lista de Producto seleccionados para el Sprint más un plan para entregar el Incremento del producto y conseguir el Objetivo del Sprint.

Hace visible todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el Objetivo del Sprint.

A medida que el Sprint avanza, el Equipo de Desarrollo va modificando la Lista de Pendientes del Sprint actualizando la estimación de trabajo restante.

Incremento

El Incremento es la suma de todos los elementos de la Lista de Producto completados durante un Sprint. Debe cumplir con la Definición de "Terminado" del Equipo Scrum y debe estar en condiciones de utilizarse sin importar si el Dueño de Producto decide o no liberarlo.

3.3.5 Definición de "Terminado" (*Definition of "Done" – DOD*)

Todo el mundo debe entender lo que significa "Terminado", pudiendo esto variar significativamente para cada Equipo Scrum. Los miembros del Equipo deben tener un entendimiento compartido de lo que significa que el trabajo esté completado para asegurar la transparencia.

Se utiliza para evaluar cuándo se ha completado el trabajo sobre el Incremento del producto.

A medida que los Equipos Scrum maduran, se espera que su definición de "Terminado" se amplíe para incluir criterios más rigurosos y una mayor calidad.

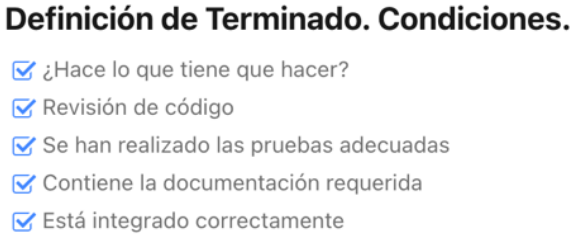

Ilustración 6: Definición de 'terminado'

3.4 Turismo

'El turismo es el desplazamiento de las personas de manera temporal y voluntaria. Dentro de este concepto deben ser incluidos las relaciones humanas que conllevan la prestación de servicios. Si bien los motivos del turismo son variados, suelen estar relacionados con el ocio

El turismo resulta muy importante desde el punto de vista **social, cultural y económico**' [10].

Este proyecto está enfocado en la creación de una aplicación cuya finalidad es ser de utilidad para el turista en su itinerario. Es por esto que vamos a desarrollar algunos aspectos socioeconómicos del turismo en este nuestro País.

3.4.1 Aspectos socioeconómicos en España

La importancia del sector del turismo en España se pone de manifiesto a través de su peso en relación con las distintas macromagnitudes de la economía española. Así, el PIB (Producto Interior Bruto) asociado al turismo alcanzó los ciento cincuenta y tres mil ciento setenta y cinco millones de euros, en el año dos mil diecinueve suponiendo el doce con tres por ciento del PIB. Por su parte, en el sector del turismo, el empleo alcanzó, en dos mil diecinueve los dos millones y medio de afiliados representando el trece por ciento del total de afiliados a la seguridad social en España

Comunidades autónomas como Canarias o las Islas Baleares están fuertemente sometidas a nivel económico al turismo, debido a que su actividad productiva, directa o indirectamente se centra en el turismo. [11]

Según el INE (Instituto Nacional de Empleo), por cada cien empleos en ramas de actividad en contacto directo con el turista, se generan sesenta y siete empleos adicionales en otros sectores, y por cada cien euros de valor añadido, se aportan sesenta y dos en otros sectores.

Además, el sector turístico también es clave para la re-dinamización de entornos rurales y la España 'vaciada', debido entre otros, a la extensa red de casas rurales presente en más de cuatro mil setecientos municipios, más del cincuenta por ciento de los que componen la estructura geográfica española. [12]

Agenda del sector: Visión 2030. [12]

La agenda propone trabajar en tres áreas clave:

- Compromiso y acción: Maximizar el impacto de las acciones que se acometen en el campo de la Responsabilidad Social, reforzando la vocación de contribuir y tomar como referencia iniciativas internacionales y fomentando el desarrollo y bienestar de las comunidades locales con el máximo respeto ante el patrimonio, la cultura, los habitantes locales y sus tradiciones.

- Alianzas sectoriales e intersectoriales: que ayuden a aunar objetivos y gestionar las acciones de todos los actores que integran la amplia cadena de valor implicados en la actividad turística, tomando como base la cohesión, articulación y un diálogo constante y consistente

- Identificación de las claves del sector sobre las que incidir. De ello se derivan cinco oportunidades para maximizar el impacto positivo en la sociedad

    - Desarrollo de políticas de empleo que den respuesta a estos retos sociales relacionados con la integración laboral, generando empleo de calidad e inclusivo.
    - Fomento de actuaciones de promoción de los valores de la sociedad e impulso de la relación con sus entidades sociales.
    - Trabajo con proveedores involucrados en el compromiso social de la empresa.
    - Impulso de la mejora de calidad de vida y el bienestar de la sociedad, incorporando temas medioambientales
    - Favorecer la accesibilidad física y tecnológica para no dejar a nadie atrás.

3.5 Diario de Viaje

'Un diario de viaje es un cuaderno que incluye todos los recuerdos de tu viaje. Siempre puedes llevarlo encima y pegar e incluir en él todos los detalles que te van a acompañas en las vacaciones. Desde entradas a museos hasta crónicas detalladas de las anécdotas que has vivido, cualquier momento tiene espacio en el diario de viaje'. [13]

Los diarios de viaje digitales han llegado para quedarse tras los avances tecnológicos, y es evidente que estos no convencerán a todo el público objetivo, puesto que muchas personas siguen prefiriendo crear sus propios diarios desde cero, pudiendo añadir todos los colores, diseños y formas de la forma que mejor les represente, pudiendo tener la sensación de que ese diario es verdaderamente tuyo. De todas formas, la aplicación desarrollada durante este proyecto cumple con los puntos desarrollados en el párrafo anterior, convirtiéndose en una opción legítima como diario de viaje, que podría competir con otras opciones ya existentes en el mercado.

Ilustración 7: Diario de viaje

4. Técnicas y herramientas

Se comentan las diferentes técnicas y herramientas utilizadas para el desarrollo de la aplicación.

4.1 Entornos de desarrollo

4.1.1 Visual Studio Code

"Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y MacOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código". [14]

Se ha escogido este editor ya que, debido a las múltiples extensiones instalables, permite realizar un desarrollo cómodo y simple. También va a permitirnos en tiempo real ver los cambios que se van efectuando a la aplicación directamente en el navegador web de preferencia o en el emulador de Android Studio.

4.1.2 Android Studio

IDE oficial para el desarrollo de aplicaciones para la plataforma Android. Este va a compilar la aplicación una vez exportada desde nuestro framework de trabajo. También va a permitir emular diferentes dispositivos Android, con diferentes versiones del sistema operativo, para probar en estos el funcionamiento de la aplicación creada.

4.2 Frameworks

4.2.1 Ionic

'Ionic Framework es un SDK de frontend de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JavaScript). Es decir, un framework que nos permite desarrollar aplicaciones para iOS nativo, Android y la web, desde una única base de código. Su compatibilidad y, gracias a la implementación de Cordova e Ionic Native, hacen posible trabajar con componentes híbridos. Se integra con los principales frameworks de frontend como Angular, React y Vue'. [15]

Se ha elegido este framework por la facilidad que supone para el desarrollo de una aplicación Android utilizando las tecnologías HTML y CSS mencionadas anteriormente. También nos permite integrarse con React para realizar diferentes tareas de una forma más sencilla. Por último, nos permite utilizar diferentes APIs que dotarán a nuestra aplicación de más funcionalidad [16].

4.2.2 Capacitor

'CapacitorJS es un Framework de código abierto que permite construir aplicaciones móviles nativas para varias plataformas utilizando tecnologías web como HTML, CSS y JavaScript'. Se integra con herramientas populares como Ionic y permite acceder a las características nativas de los dispositivos móviles a través de una API unificada.' [17]

En nuestro se ha utilizado para realizar la portabilidad del proyecto hacia Android Studio. También se ha hecho uso de alguno de los Plugin ofrecidos, en este caso se ha hecho uso de los *Plugin* **Local Notifications** [18] y **Camera** [19]**.** Estos plugin han permitido el uso

de notificaciones y tomar fotos o elegir estas de la galería para incluirlas en los viajes, respectivamente.

Estos *Plugin* necesitan unos permisos específicos que deben ser habilitados en el proyecto creado en Android Studio por Capacitor. En la Ilustración 8 se pueden observar los diferentes permisos utilizados por esta aplicación.

```
<!-- Permissions -->

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

Ilustración 8: Permisos de la aplicación

4.3 Gestor de paquetes y entorno de ejecución

4.3.1 NodeJS

'Node.js es un entorno de tiempo de ejecución de JavaScript (de ahí su terminación en .js'). Este entorno de tiempo de ejecución en tiempo real incluye todo lo necesario para ejecutar un programa escrito en JavaScript. Va a utilizar un modelo de entrada y salida sin bloqueo, controlado por eventos que lo hace ligero y eficiente (con entrada nos referimos a solicitudes y con salida a las respuestas a estas solicitudes). Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer solicitudes HTTP'. [20]

Se ha optado por este entorno debido a las recomendaciones realizadas por el framework Ionic.

4.3.2 NPM

NPM (Node Package Module) es el gestor de paquetes para JavaScript, por lo tanto, servirá también para TypeScript. Es el gestor por defecto en NodeJS. Con él podrá descargar e instalar diferentes librerías.

La elección fue bastante evidente al utilizar NodeJS.

4.4 Bibliotecas

Para el desarrollo de la aplicación, se han utilizado diferentes bibliotecas que nos aportan funcionalidad externa al sistema.

4.4.1 Firebase

Esta biblioteca nos ha permitido conectar nuestro proyecto con la suite en la nube de Google Firebase. Esta suite nos permite realizar diferentes acciones relacionadas con la base de datos desde la aplicación [21]:

- Alta, baja, modificación y consulta de la base de datos de Cloud Firestore.
- Subir imágenes y archivos en formato PDF a Cloud Storage.
- Gestionar el acceso y salida de usuarios del sistema con Firebase Authentication.

Se ha optado por Firebase debido a la sencillez para realizar las acciones de backend requeridas por el sistema usando únicamente un elemento. Además, nos garantiza el control de los usuarios de forma centralizada y garantizando que estos se registran en el sistema para acceder.

4.4.2 Leaflet [22]

'Leaflet es una librería JavaScript que permite crear mapas interactivos de una forma muy sencilla. Es de código abierto y es mobile-friendly'. [23]

Nuestra aplicación va a necesitar de un mapa donde se pueda añadir la ubicación donde tendrán lugar las diferentes actividades del viaje, además de mostrar nuestra localización como guía al usuario de la distancia a destino.

Se ha optado por esta opción debido a la sencillez de uso, su documentación está bien redactada y es muy completa y otras opciones en consideración (Google Maps) son de pago. La funcionalidad aportada por esta biblioteca ha sido perfecta para lo buscado en el proyecto.

4.4.3 React [24]

'ReactJS es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Creada por Facebook, React contiene una colección de fragmentos de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI) llamadas componentes.' [25]

Se ha seleccionado esta biblioteca debido a su gran compatibilidad con el framework Ionic. Además, dispone de una documentación bien redactada que facilita en gran medida el desarrollo de la aplicación.

4.5 Lenguajes de programación

Para el desarrollo del proyecto se han utilizado los siguientes lenguajes de programación:

4.5.1 JavaScript

'JavaScript es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico'. [26].

En nuestro proyecto se ha utilizado para la generación del mapa y la iteración con el mismo para dotarle de la funcionalidad requerida.

4.5.2 TypeScript

TypeScript es un lenguaje de programación basado en JavaScript añadiendo tipos estáticos a las variables y objetos basados en clases.

En el desarrollo de la aplicación se ha utilizado para la realización de las diferentes pantallas de la aplicación.

4.5.3 CSS

'CSS es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuarios escritas en HTML '. [27].

Se ha utilizado para dar un diseño concreto a una serie de componentes específicos.

4.6 Herramientas utilizadas para gestión del proyecto y documentación

4.6.1 Wrike

Wrike es una empresa desarrolladora de software de gestión de proyectos [28]. El producto principal de Wrike, llamado de la misma forma, es una herramienta en línea de gestión de proyectos y colaboración. Permite que los usuarios ajusten sus planes de proyectos, prioricen tareas, estén al tanto de la planificación y colaboren en línea con los compañeros de equipo [29].

La herramienta ha sido utilizada para toda la planificación temporal del proyecto.

4.6.2 Visual Paradigm

Visual Paradigm es una herramienta UML Case. Se ha utilizado para la creación de todos los distintos diagramas de los diferentes documentos para así facilitad la comprensión de la estructura del proyecto

4.6.3 Microsoft Word

Herramienta utilizada como editor de texto durante todo el proyecto tanto para la realización de los anexos como para la memoria.

5. Aspectos relevantes del desarrollo

5.1 Planificación del proyecto

Este apartado está dedicado a los aspectos más relevantes de la planificación temporal del proyecto. Se mostrarán algunas partes del desarrollo del proyecto, se puede ver más detalladamente en el Anexo I [30].

Para el proyecto se ha seguido el marco de trabajo de Scrum con algunas variaciones. Este marco consiste en el desarrollo incremental e iterativo, es decir, se parte de un prototipo funcional a partir del cual se van incorporando nuevas funcionalidades y mejoras hasta su finalización.

Dado que se ha podido trabajar desde muy temprano, el tiempo disponible será de cinco meses (abril – agosto), es decir, cinco Sprint ya que hemos definido la duración de cada uno de ellos a un mes calendario.

El Equipo Scrum en este caso estará formado por:

- **Dueño de Producto**: este rol será desempeñado por el tutor, que será el encargado de la verificación de los Incrementos, así como del Producto Final.

- **Scrum Master y Equipo de Desarrollo**: ambos roles serán desempeñados por el estudiante que se encargará de desarrollar el proyecto completo.

Este proyecto parte de una idea recogida en la Epic: "Como usuario quiero un diario de viaje". A partir de esta Epic obtendremos Historias de Usuario más pequeñas a las que poder asociarle Tareas para realizar el desarrollo del producto.

A continuación, se mostrarán algunas partes del desarrollo del proyecto, se mostrará la planificación de cada uno de los Sprints así como el Sprint Review de cada uno de ellos. Con esto se pretende mostrar cómo se ha planificado el proyecto y cómo se ha trabajado en cada uno de los Sprints.

Para mayor profundidad del proceso de desarrollo se puede consultar el Anexo I.

5.1.1 Sprints

En la metodología Scrum, como se ha explicado previamente en el apartado 3.3.3 Eventos Scrum, se trabaja en base a unos Sprints con una duración determinada. Por lo tanto, en este apartado se va a hablar más en detalle acerca de los Sprint realizados. En el Anexo I se puede encontrar la información acerca de la planificación temporal del proyecto, así como toda la información necesaria acerca de las Historias de Usuario y las Tareas.

Primer Sprint (SP-01)

Sprint Planning

En el primer Sprint se comenzará a trabajar sobre los primeros aspectos de la aplicación, antes que nada, el estudiante deberá aprender el lenguaje Javascript + Ionic + React, por lo que hará un curso en la plataforma Udemy, y posteriormente comenzará con la creación del proyecto, configuración de la base de datos, diseño de la interfaz, etc. Por tanto, se

comienza a trabajar sobre la gran mayoría de Historias de Usuario sin ahondar en exceso en ninguna de ellas, se empieza la casa por la estructura y se trata de hacer un buen trabajo en este apartado, para que no ocurran errores que puedan frenar el avance del proyecto en el futuro.

Segundo Sprint (SP-02)

Sprint Planning

En este segundo Sprint, se va a tratar de empezar a desarrollar la gestión de usuarios de la aplicación, es decir, crear todo lo necesario para que la autenticación de usuarios se haga de forma correcta, incluyendo la posibilidad de cambiar las credenciales y eliminar la cuenta.

Realizamos también ahora la especificación de casos de uso para tener claro más adelante las acciones a implementar dentro de la aplicación.

Tercer Sprint (SP-03)

Sprint Planning

Durante este Tercer Sprint, el objetivo será comenzar a desarrollar el grueso de la funcionalidad, empezar a crear los scripts que permitirán crear viajes, incluyendo el propio calendario de viaje, toda la edición o eliminado del mismo, gestionar la antelación con la que la aplicación notificará a los usuarios e incluso comenzar a trabajar en las actividades a crear.

Cuarto Sprint (SP-04)

Sprint Planning

Durante el cuarto Sprint, el objetivo será dejar el proyecto prácticamente finalizado, es decir, terminar el desarrollo de la aplicación, o al menos dejar unos pequeños flecos para el siguiente Sprint donde se realizará gran parte de la documentación. Todas las tareas estarán dirigidas hacia el desarrollo de la funcionalidad de la aplicación y no habrá distracciones, a no ser que surjan durante las semanas de duración del Sprint nuevas necesidades.

Sprint Review

No se han podido completar todas las tareas previstas antes de empezar el Sprint por dos motivos:

- El primero han sido las dificultades surgidas que inicialmente no fueron contempladas, es decir, el desarrollo de alguna de las funcionalidades ha sido más complejo de lo que se pensaba inicialmente. En este caso las Tareas T-026 y T-027 Creación de Galería y Navegación en la Galería respectivamente, han supuesto un mayor esfuerzo del inicialmente planeado.

- El segundo motivo han sido las dos tareas que han surgido a lo largo de este Sprint, qué al añadir más carga de trabajo, han limitado el tiempo que se le ha podido dedicar a otras tareas. Las nuevas tareas agregadas no contempladas inicialmente son T-030 Geolocalización en Actividad y T-031 Grabar mensaje de voz en Momentos Destacados.

Tarea (ID)	Historia (ID)	Estado
T-001	*	Completada
T-002	HU-001, HU-002, HU-005	Completada
T-003	HU-002, HU-006, HU-007	Completada
T-004	HU-002, HU-005, HU-006, HU-007, HU-008	Completada
T-005	HU-002, HU-003, HU-004, HU-005, HU-006, HU-008	Completada
T-006	HU-001, HU-002, HU-003, HU-004, HU-006, HU-008	Completada
T-007	HU-003, HU-004, HU-006, HU-008	Completada
T-008	HU-002, HU-003, HU-004, HU-006, HU-008	Completada
T-009	HU-003, HU-004, HU-006, HU-008	Completada
T-010	HU-002, HU-005	Completada
T-011	HU-002, HU-005	Completada
T-012	HU-002, HU-005	Completada
T-013	HU-002, HU-005	Completada
T-014	HU-001, HU-003, HU-005	Completada
T-015	HU-003, HU-005	Completada
T-016	HU-001, HU-003	Completada
T-017	HU-001, HU-003, HU-005	Completada
T-018	HU-003, HU-005	Completada
T-019	HU-004, HU-007	Completada
T-020	HU-001, HU-004, HU-006, HU-008	Completada
T-021	HU-001, HU-003, HU-006, HU-008	Completada
T-022	HU-001, HU-003, HU-004	Completada
T-023	HU-001, HU-004, HU-005	Completada
T-024	HU-001, HU-004, HU-005	Completada
T-025	HU-004, HU-005, HU-007	Completada
T-026	HU-001, HU-003, HU-004, HU-005, HU-006	Completada
T-027	HU-001, HU-003, HU-004, HU-005, HU-006	Completada
T-028	HU-004, HU-007	Completada
T-029	HU-001, HU-003, HU-005, HU-008	En proceso
T-030	HU-001, HU-004, HU-005	Completada
T-031	HU-001, HU-005, HU-008	En proceso

Tabla 1: Sprint 4

Quinto Sprint (SP-05)

Sprint Planning

Quinto y último Sprint del proyecto. Se terminará de desarrollar la aplicación dejando resueltos los flecos sueltos que puedan haber quedado del Sprint anterior. Se realizarán toda clase de pruebas y corrección de errores, también se terminará con la documentación requerida para la entrega del Trabajo de Fin de Grado: memoria, manual de usuario, manual de programador, diagramas de despliegue, secuencia, etc.

5.2 Requisitos

Dado que estamos utilizando la metodología ágil Scrum, las funcionalidades son recogidas en las Historias de Usuario y posteriormente descompuestas en Tareas de carácter más específico que son las que se desarrollarán con el objetivo de completar el proyecto.

5.2.1 Historias de Usuario y Tareas Scrum

Se mostrarán, de ejemplo, algunas tabla relacionadas con las Historias de Usuario y sus Tareas, para más detalle en los anexos I y II se dispone de la información completa.

5.2.1.1 HU-001 Interfaz Gráfica

En la Tabla 2, correspondiente a la primera Historia de Usuario, encontramos la correspondiente a la interfaz gráfica. Estas tablas están relacionadas con las necesidades que tendrá un usuario dentro de la funcionalidad planeada para la aplicación. En este caso, al ser una aplicación Android, la interfaz gráfica es algo clave, puesto que el usuario deberá visualizar toda la información en su pantalla.

[Historia] Interfaz Gráfica
ID HU-001
Caso de Uso **Como** usuario **Necesito** tener una navegación sencilla **Para que** no necesite ayuda a la hora de utilizar la aplicación
Descripción El usuario final debe disponer de una disposición de los elementos en su pantalla organizada y fácil de entender. Las diferentes pantallas de las que esté compuesta la aplicación deben conectadas fácilmente.
Criterios de aceptación **Dado que soy** usuario **Cuando yo** necesite realizar cualquier acción **Entonces** debo saber como realizarla y hacia donde dirigirme

Tabla 2: HU-001 Interfaz Gráfica

5.2.1.2 T-001 Curso Ionic + React

Al contrario de lo que ocurre con las Historias de Usuario, las Tareas para completar estas historias, están centradas en el desarrollo qué debe hacer el equipo desarrollador para satisfacer las necesidades del usuario, esto se ve reflejado en la Tabla 3, donde la Tarea

T-001, al ser tan general, pues se trata de realizar un curso para aprender a trabajar con el '*framework*' elegido para el desarrollo, va a afectar directamente a todas las necesidades del usuario.

[Tarea] Curso Ionic + React		
Estado Completed	Asignado Jorge García Pri...	Fecha 3 – 14 abr. (10d)
Añadir subelemento	Añadir archivos	2 más 0:00
ID T-001		
Historia de Usuario •		
Caso de Uso **Como** desarrollador **Necesito** aprender Ionic + React **Para que** pueda llevar a cabo el desarrollo del proyecto		
Descripción Completar un curso en la plataforma Udemy que permita aprender a crear aplicaciones Android con Ionic + React apoyándose en Capacitor en lenguaje Javascript		

Tabla 3: T-001 Curso Ionic + React

Como en Scrum no se produce una fase de análisis y diseño dada la mutabilidad de los requisitos, estos se definen durante el desarrollo de este. De todas maneras, se ha optado por realizar una serie de diagramas para facilitar la comprensión del proyecto.

5.2.2 Casos de uso

Actores

Jerarquía de actores

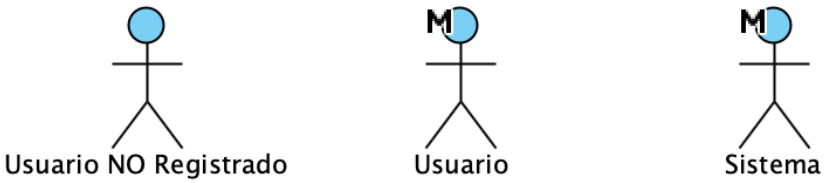

Ilustración 9: Jerarquía de actores

Definición de actores

- **Usuario NO Registrado**: persona que accede por primera vez a la aplicación y todavía no ha creado su cuenta de usuario. Este actor no cuenta con funcionalidades dentro de la aplicación.
- **Usuario**: usuario con acceso a todas y cada una de las funcionalidades de la aplicación. Es imprescindible estar registrado en la aplicación para acceder a la misma.
- **Sistema**: usuario que representa la parte automática del propio sistema.

Diagramas de casos de uso

Gestión inicio

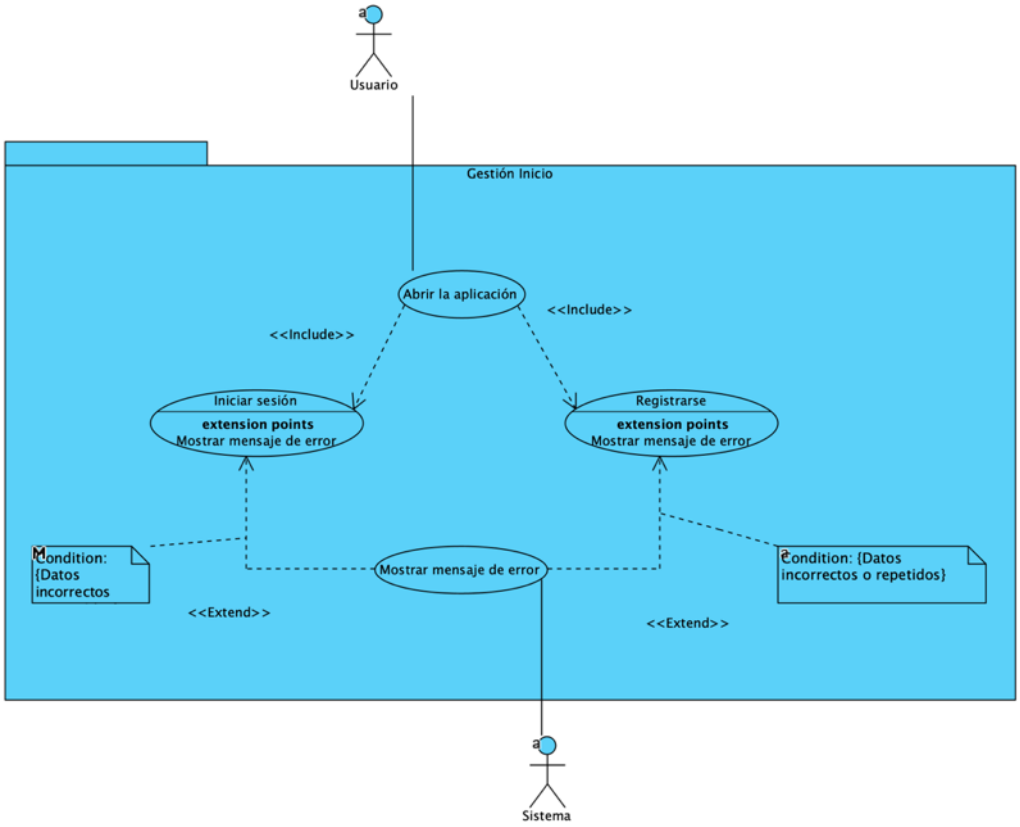

Ilustración 10: Gestión inicio

5.2.3 Modelo de dominio

Diagrama de clases

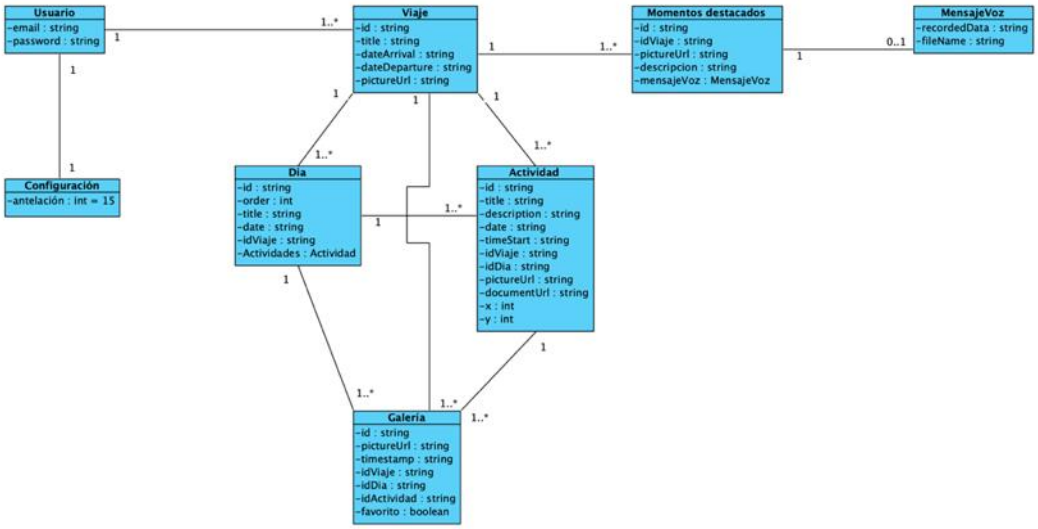

Ilustración 11: Diagrama de clases

Glosario de clases

- **Usuario**: accede a la aplicación a través de su correo electrónico y contraseña almacenados en la base de datos con su identificador asociado.
- **Configuración**: ajustes personales del usuario que podrá modificar a conveniencia.
- **Viaje**: Cada uno de los destinos pasados/futuros del usuario
- **Dia**: forman parte del calendario de un viaje
- **Actividad**: evento que tiene lugar para un viaje en un día en concreto
- **Galería**: fotos pertenecientes a una Actividad/Dia/Viaje
- **Momentos destacados**: listado de mejores momentos del viaje.
- **MensajeVoz**: información almacenada en un mensaje de voz grabado para un Momento Destacado

5.2.4 Diagramas de secuencia

Inicio de sesión

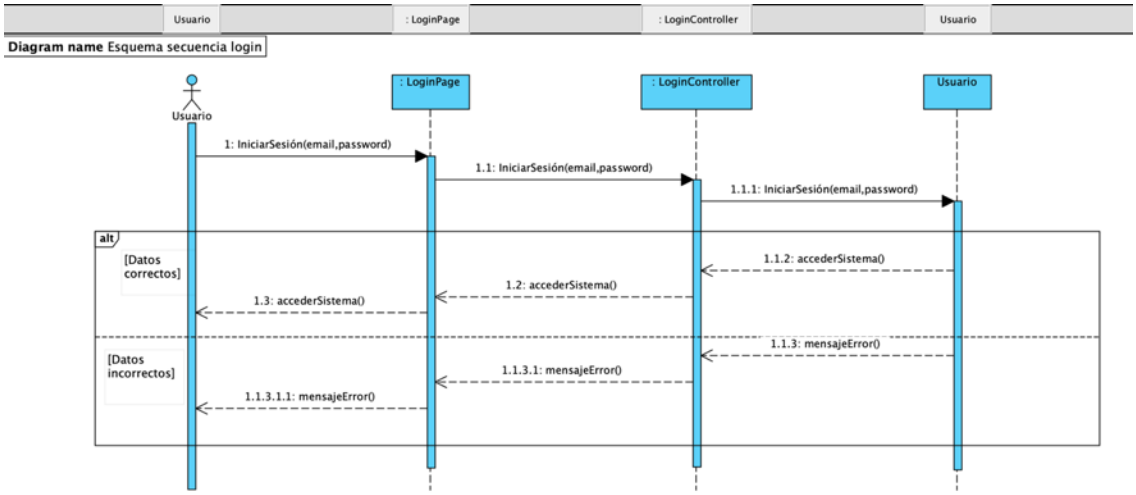

Ilustración 12: Diagrama de secuencia - Inicio de sesión

5.3 Diseño del sistema

Este apartado está dedicado al diseño del sistema que, dado que hemos utilizado la metodología Scrum, no se especificó al principio del proyecto.

De igual forma, se explicará de forma resumida la arquitectura del sistema, las relaciones entre sus elementos, etc. Se puede encontrar más detalle en el Anexo III. Diseño del sistema.

5.3.1 Patrón arquitectónico

Para realizar el proyecto se ha optado por un patrón que agilizase el proceso de desarrollo. En este caso, se ha elegido el patrón Modelo-Vista-Controlador (MVC).
Este patrón separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Estos componentes dan nombre al patrón [31]. La relación entre los diferentes componentes es como se muestra en la Ilustración 13.

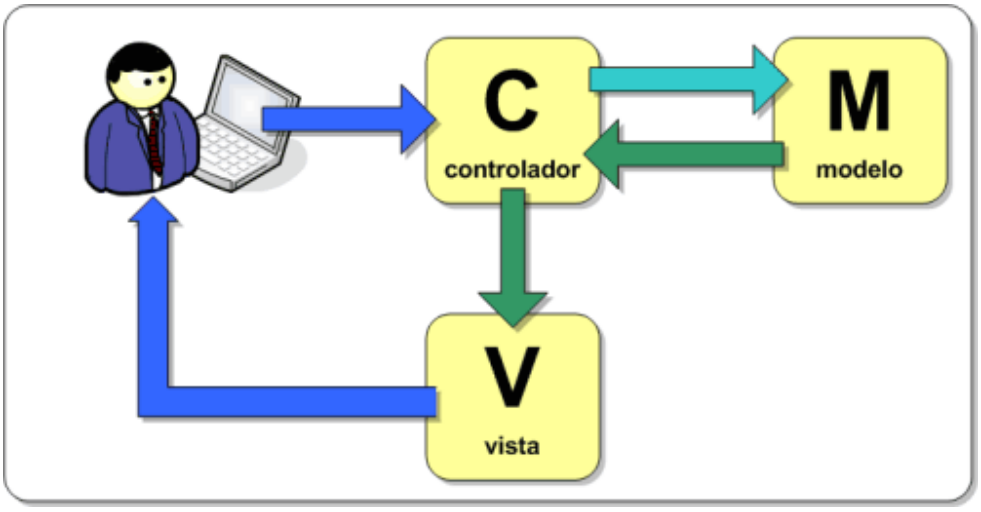

Ilustración 13: Patrón arquitectónico MVC

Este modelo separa en tres partes el sistema:

- **Modelo**: contiene los datos de la aplicación, estos a su vez, estarán almacenados en una base de datos.

- **Vista**: contiene los elementos con los que va a interactuar el usuario, mostrando los datos y obteniendo los datos nuevos.

- **Controlador**: contiene la funcionalidad principal del sistema. Va a ser el responsable de actualizar el modelo de datos obtenido en las vistas y de obtener los datos del modelo para proporcionarlos a las vistas que van a mostrarlos.

5.3.2 Clases de diseño

Siguiendo las mismas intenciones que en el apartado anterior, se busca determinar un sistema con bajo acoplamiento y alta cohesión. Para ello, se ha dividido el sistema en diferentes subsistemas individuales. Todos se van a relacionar entre sí para poder trabajar e intercambiar datos de forma satisfactoria.

En la Ilustración 14 encontramos el subsistema del 'Controlador'. Encontramos aquí todos los controladores necesarios para el correcto funcionamiento del sistema, así como las funciones incluidas en cada uno de estos. En el Anexo III. Diseño del Sistema se puede encontrar información extra acerca del resto de subsistemas.

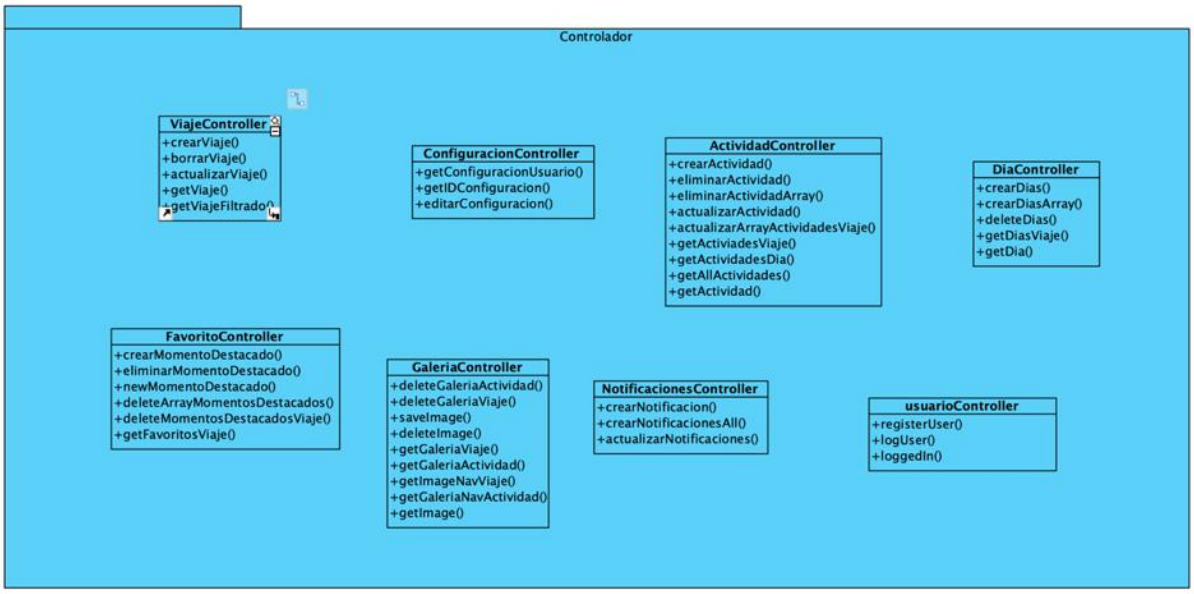

Ilustración 14: Controlador de la aplicación

5.3.3 Diagramas de actividad

Editar Viaje

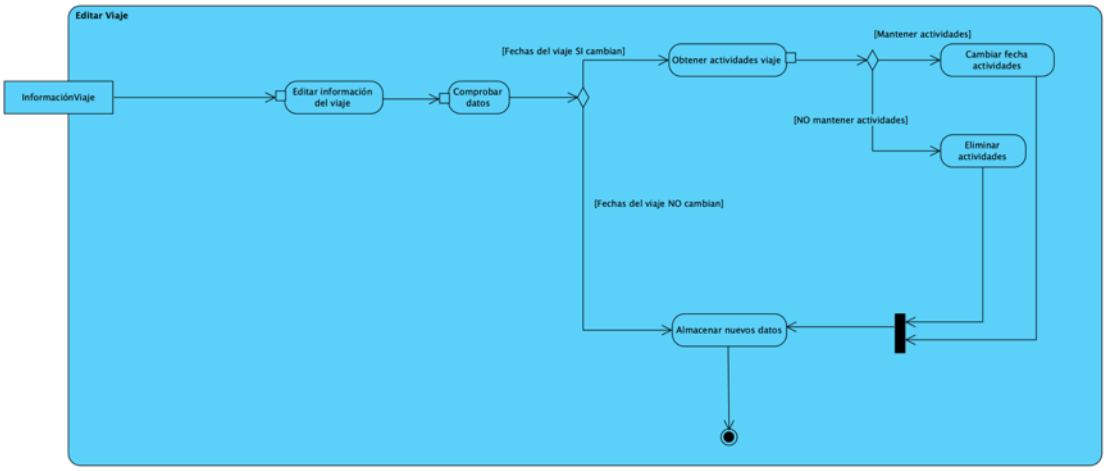

Ilustración 15: Diagrama de actividad – Editar Viaje

Grabar mensaje de voz

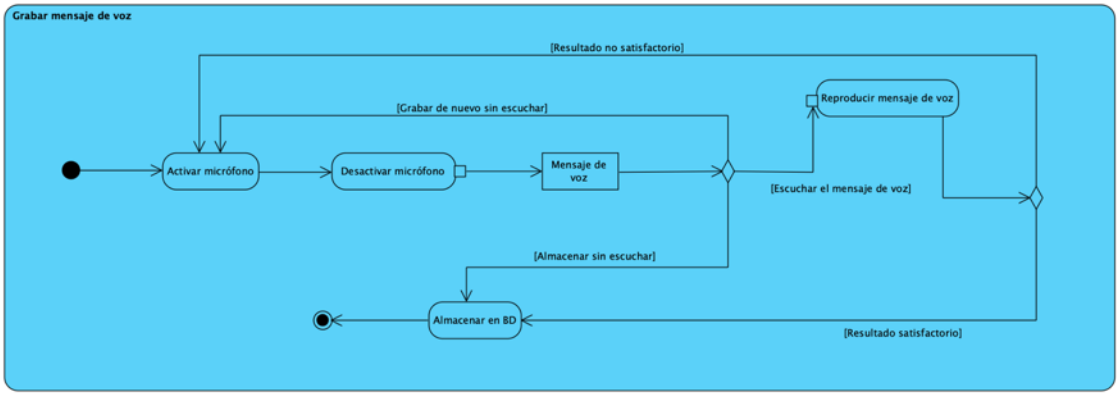

Ilustración 16: Diagrama de actividad – Grabar mensaje de voz

5.3.4 Modelo de despliegue

El objetivo del diagrama de despliegue es mostrar la disposición de las partes físicas de nuestro sistema y sus relaciones.

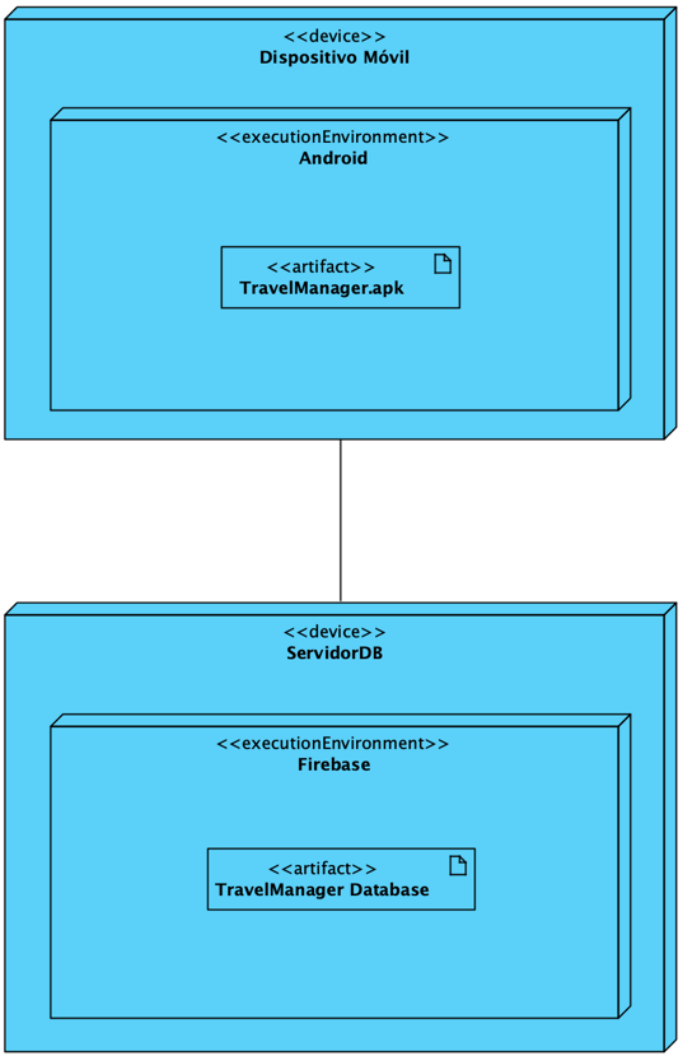

Ilustración 17: Diagrama de despliegue - Sistema general

En el sistema vamos a encontrar 2 nodos:

- **Dispositivo Móvil**: dispositivo donde se va a instalar la aplicación. El sistema operativo que se va a usar es Android. A través de la aplicación, los usuarios van a poder acceder a las diferentes funcionalidades comentadas en los diferentes anexos.

- **Servidor DB**: servidor donde se va a almacenar la información referente a los usuarios y diferentes viajes que va a crear el usuario dentro de la aplicación. La base de datos utilizada va a ser **Firebase Firestore.**

5.4 Implementación

En este apartado se tratarán los aspectos más relevantes de la programación, para un análisis más en detalle se puede encontrar todo tanto en el anexo IV [32].

5.4.1 Visual Studio Code

La aplicación ha sido desarrollada en Visual Studio Code, en lenguaje JavaScript. Se van a tratar los aspectos más relevantes de la aplicación explicando así los asuntos más complejos de la programación de este.

Como se explica en el Anexo IV, se ha trabajado con Visual Studio Code pero se han utilizado las diferentes tecnologías para el desarrollo:

- Framework **Ionic** con **React**.
- Base de datos **Firebase Firestore**
- Almacenamiento de archivos en **Firebase Storage**
- Autenticación en el sistema con **Firebase Authentication**
- Conversión de desarrollo web a proyecto en Android Studio con **Capacitor**.

Editar datos de un viaje

Para editar un viaje habrá que tener en cuenta toda la estructura de componentes, puesto que modificar las fechas del viaje ya implica modificar también el calendario de viaje, y modificar el calendario de viaje va a modificar la fecha de las actividades, por lo tanto, con las siguientes imágenes se irá explicando el proceso de edición de viaje.

```
// Guardamos la actualización del viaje en la BBDD
const handleUpdate = async () => {

    setStatus({loading: true, error: false});

    const viajeData = {id, title, dateArrival, dateDeparture, pictureUrl};

    var cambioFechas = false

    // Comprobamos que las fechas tienen sentido
    if (checkdate(dateArrival, dateDeparture) && title != "") {

        // Actualizamos viaje en BBDD y obtenemos los nuevos días del mismo
        var updateViaje = await actualizarViaje(userId, viajeData, oldImg, oldDateArrival, oldDateArrival);

        if (dateArrival != oldDateArrival || dateDeparture != oldDateDeparture) {

            setDias(updateViaje);
            //setDias(await crearDiasArray(userId, viajeData.dateArrival, viajeData.dateDeparture, id))

            // Permitimos elegir al usuario donde colocar cada actividad
            cambioFechas = true;
        }

        showToast('Viaje modificado con éxito!');

        if (cambioFechas) {

            setStatus({loading: false, error: false});

            setActividades(await getActividadesViaje(userId, id));

            if (actividades.length > 0) {
                // presentamos por pantalla el formulario para reordenar las actividades
                setOrdenarActividades(true);
            } else {
                history.goBack();
            }
```

Ilustración 18: Almacenamiento en base de datos de un viaje actualizado

La Ilustración 18 muestra la función que es lanzada (en la vista) una vez el usuario rellena el formulario para editar los datos de un viaje y pulsa el botón 'Guardar viaje'.
Una vez creado el objeto con toda la información del viaje y comprobadas fechas y título, se llama a la función '**actualizarViaje'** (Ilustración 19) en el controlador, que guarda en base de datos la nueva información, borra el calendario existente y genera un nuevo calendario, que es devuelto al terminar la función, por eso se hace uso de la variable '**updateViaje'**.

```
export async function actualizarViaje(userId: string, viajeData: any, oldImg: string, oldDateArrival: string, oldDateDeparture: string) {

    return new Promise<Dia[]>(async (resolve, reject) => {

        var arrayDias: Dia[] = [];

        if (viajeData.pictureUrl != oldImg) { // Cambio de imagen de perfil

            if(!oldImg.startsWith('/assets/')) {
                // Borramos la anterior imagen
                const deleteRef = ref(storage, oldImg)
                await deleteObject(deleteRef);
            }

            // Guardamos la nueva
            const pictureRef = await ref(storage, `users/${userId}/pictures/${uuid()}`);
            const response = await fetch(viajeData.pictureUrl);
            const blob = await response.blob();
            const uploadTask = await uploadBytesResumable(pictureRef, blob);
            viajeData.pictureUrl = await getDownloadURL(uploadTask.ref);
        } else if (viajeData.pictureUrl.startsWith('/assets')) {
            viajeData.pictureUrl = '/assets/placeholder.png';
        }

        // Gestión de nuevas fechas
        if (viajeData.dateArrival != oldDateArrival || viajeData.dateDeparture != oldDateDeparture) {

            // borramos los días anteriores
            await deleteDias(userId, viajeData.id);

            // creamos los nuevos días
            arrayDias =  await crearDiasArray(userId, viajeData.dateArrival, viajeData.dateDeparture, viajeData.id);
        }

        // Actualizamos el viaje en BBDD
        const viajeRef = doc(firestore, `users/${userId}/viajes/${viajeData.id}`);
        await updateDoc(viajeRef, viajeData);

        resolve(arrayDias);
    })
}
```

Ilustración 19: Comprobación nuevos datos del viaje

Volviendo a la Ilustración 18. Se comprueba si las fechas del viaje han cambiado, en caso afirmativo es cuando asignamos la variable '**updateViaje'** al nuevo calendario de viaje. También se recogen las actividades del viaje (Ilustración 20) para mostrarlas por pantalla y de esta forma que el usuario pueda elegir si eliminarlas o cambiar su fecha.

```
/**
 * Obtiene todas las actividades planificadas para un viaje
 * @param userId Identificador de usuario
 * @param idViaje Identificador de viaje
 * @returns Array con las actividades del viaje
 */
export async function getActividadesViaje(userId: string, idViaje: string) {

    return new Promise<Actividad[]>((resolve, reject) => {

        const q = query(collection(firestore, `users/${userId}/actividades`), where("idViaje", "==", idViaje), orderBy("date", "asc"));

        const unsubscribe = onSnapshot(q, (observer) => {

          var arrayActividades: Actividad[] = [];

          observer.forEach((actividad) => {
              arrayActividades.push(toActividadData(actividad));
          });

          resolve(arrayActividades);
        });
    })
}
```

Ilustración 20: Obtención de actividades del viaje

Esta función, tras pasar por '**actividadModel'** (Ilustración 21) para formatear los datos de cada actividad, devuelve el listado de actividades al usuario para presentarlo por pantalla.

```
/**
 * Modelo de datos para una Actividad
 * @module ActividadModel
 */
/**
 * Modelo de datos de la Actividad para la lectura desde la base de datos
 * @interface
 */
export interface Actividad {
    id: string;
    title: string;
    description: string;
    date: string
    timeStart: string;
    idViaje: string;
    idDia: string;
    pictureUrl?: string;
    documentUrl?: string;
    x?: number;
    y?: number;
}

/**
 * Función que permite pasar el dato leído de la base de datos y retornarlo cada
 * uno de los datos por separado. En este caso el tipo de dato es una Actividad.
 * @param doc Dato leído de la base de datos
 * @returns Devuelve los datos de la base de datos de manera individual
 */
export function toActividadData(doc : any) {
    return {id: doc.id, ...doc.data()};
}
```

Ilustración 21: Modelo de datos de una actividad

Una vez el usuario seleccione que actividades desea mantener, e indique la nueva fecha de las actividades, se actualizarán todas y cada una de las actividades (ver Ilustración 22).

```
// Guardamos la actualización de las actividades
const handleUpdateActividades = async () => {

    setStatus({loading: true, error: false});

    var auxStatus = false;

    actividades.forEach(async (actividad) => {

        if (actividad.date < dateArrival || actividad.date > dateDeparture) {
            auxStatus = true;
            setStatus({loading: false, error: true});
        }
    });

    if (!(auxStatus)) {

        // Actualizamos las actividades
        await updateActividades();

        // Actualizamos las notificaciones
        await actualizarNotificaciones(userId);

        showToast('Actividades modificadas con éxito!')

        history.goBack();
    }
}
```

Ilustración 22: Actualización de las actividades de un viaje

Se comprueba al pulsar el botón 'Guardar actividades' que todas las actividades restantes tengan fecha correcta y se procede a actualizar tanto las actividades como las notificaciones referentes a ellas (ver Ilustración 23).

```
/**
 * Actualiza la fecha de las actividades cuya fecha haya sido modificada al modificar las fechas de un viaje
 * @param userId Identificador de usuario
 * @param arrayActividades Listado de actividades a actualizar
 * @param idViaje Identificador de viaje
 */
export async function actualizarArrayActividadesViaje (userId: string, arrayActividades: any, idViaje: string) {

    const arrayDias = await getDiasViaje(userId, idViaje);

    arrayActividades.forEach(async (actividad) => {
        arrayDias.forEach(async (dia) => {
            if (actividad.date == dia.date) {
                actividad.idDia = dia.id
            }
            const actividadRef = doc(firestore, `users/${userId}/actividades/${actividad.id}`);
            await updateDoc(actividadRef, actividad);
        })
    })

}
```

Ilustración 23: Controlador de actividad para actualizar el array de actividades

Galería de imágenes

Para crear la galería de imágenes, existen varios aspectos clave:

- Diseño de la pantalla: Ionic proporciona una serie de componentes que permiten incluir imágenes en un recuadro (ver Ilustración 24), como en cualquier galería de un teléfono móvil. Estos recuadros serán rellenados por cada imagen y, además, serán clicables para que el usuario pueda acceder a estas en grande.

- Obtención de las imágenes de la base de datos de Firebase (ver Ilustración 25).

```
{ showGaleria==2 &&
    <IonContent>
        { galeria.map((photo) =>
            <Link to={`/my/viajes/${id}/galeria/${photo.id}`}>
                <img className="photoGallery" src={photo?.pictureUrl} />
            </Link>
        )}
        { noPhotos &&
            <IonButton className="buttonNoFotos" disabled
                expand="block" fill="clear" color="medium">
                <IonLabel className="ion-text-wrap">Todavía no has añadido imágenes a este viaje, añada nuevas imágenes en las actividades para poder ver su galería de fotos</IonLabel>
            </IonButton>
        }
    </IonContent>
}
```

Ilustración 24: Código para la galería de imágenes

```
/**
 * Se crea una lista de imágenes del viaje
 * @param userId Identificador de usuario
 * @param idViaje Identificador de viaje
 * @returns Listado de imágenes pertenecientes al viaje
 */
export async function getGaleriaViaje(userId: string, idViaje: string) {

    return new Promise<Galeria[]>((resolve, reject) => {

        const q = query(collection(firestore, `users/${userId}/galeria`), where("idViaje", "==", idViaje), orderBy('timestamp', 'asc'));

        const unsubscribe = onSnapshot(q, (observer) => {

          var arrayGaleria: Galeria[] = [];

          observer.forEach((image) => {
              arrayGaleria.push(toGaleriaData(image));
          });

          resolve(arrayGaleria);
        });
    });
}
```

Ilustración 25: Obtención de imágenes para la Galería

Observamos en la Ilustración 24, que las imágenes se introducen mediante el componente ‘img’, cuyo ‘src’ será la URL proporcionada por Firebase que descargará la imagen para su visualización. Estos componentes se encuentran dentro del componente ‘Link’, el funcionamiento de este es muy simple, una vez el usuario seleccione una imagen, el sistema abrirá la ruta en cuestión donde se visualizará la imagen en tamaño completo. Estos componentes se encuentran en la vista **‘ViajePage.tsx’.**

Como se observa en la Ilustración 41, se muestran 3 imágenes por fila, esto se consigue gracias al archivo ‘ActividadCSS.css’ como se muestra en la Ilustración 26, donde se le proporciona a cada imagen una altura de ciento cincuenta píxeles, y una anchura de un tercio de la pantalla.

```
.photoGallery {
  width: calc(100% / 3);
  height: 150px;
}
```

Ilustración 26: Colocación de imágenes en la Galería

En la Ilustración 25 encontramos el código del archivo ‘galeríaController.ts’ que obtiene todas las imágenes para un viaje. Se ordenan para que se muestren al final las imágenes más nuevas, como en cualquier otra galería.

Se comentaba previamente que el usuario puede seleccionar una imagen, y que el usuario podría entonces, visualizar esta en tamaño completo, como se observa en la Ilustración 43. Para obtener toda la información y de esta forma, poder navegar correctamente en la galería de imágenes, se necesita obtener el identificador de la imagen seleccionada, y a la vez, recorrer la galería consiguiendo los identificadores de las imágenes inmediatamente anterior y posterior, de esta forma se podrá mostrar por pantalla la información correcta, y se navegará entre imágenes de forma lógica.

En la Ilustración 27 se muestra el código correspondiente a la función **‘getImageNavViaje’** correspondiente al archivo **‘galeríaController.ts’**. Observamos que se realiza una lectura completa de las imágenes que pertenecen a un viaje, hasta encontrar la imagen deseada, para obtener el índicede la imagen deseada, y las inmediatamente anterior y posterior.

```
/**
 * Se obtiene la información de una imagen, además de la información necesaria para navegar correctamente
 * en la propia galería como los Identificadores de las fotos siguiente/anterior.
 * @param userId Identificador de usuario
 * @param idViaje Identificador de viaje
 * @param idImage Identificador de la imagen de la Galería
 * @returns Objeto con toda la información necesaria para la navegación en la Galeria
 */
export async function getImageNavViaje(userId: string, idViaje: string, idImage: string) {

    return new Promise<any>((resolve, reject) => {

        const q = query(collection(firestore, `users/${userId}/galeria`), where("idViaje", "==", idViaje), orderBy('timestamp', 'asc'));

        const unsubscribe = onSnapshot(q, (observer) => {

          var image: Galeria = {id: "", pictureUrl: "", timestamp: "", idViaje: "", idDia: "",
                                idActividad: "", favorito: false};

          var index = 0, prevIndex = 0, postIndex = 0;

          var aux: string[] = [];

          observer.forEach((foto) => {

            // Guardamos listado de Ids
            aux.push(foto.id);

            if (idImage == foto.id) {
                image = toGaleriaData(foto);

                prevIndex = index - 1
                postIndex = index + 1;
            }

            index++;
          });

          resolve({image, aux, prevIndex, postIndex, index})
        });
    })
}
```

Ilustración 27: Navegación en la Galería

Esta información es recibida por ‘GaleríaPage.tsx’ (ver Ilustración 28) donde procede a realizar la gestión de la pantalla, mostrando los componentes necesarios en la función ‘handleGallery’ (ver Ilustración 29). Los botones con flechas hacia adelante y hacia atrás (ver Ilustración 43) ya tienen asignados de nuevo el identificador de la siguiente imagen, por lo tanto, al presionar sobre estos, se repetirá este proceso, el cual, permite al usuario navegar por la galería de una forma sencilla e intuitiva.

```
// Obtenemos los datos del viaje para presentarlos por pantalla
useEffect ( () => {

    getImageNavViaje(userId, idViaje, id).then((datos) => {
        setFoto(datos.image);
        setFavorite(datos.image.favorito)

        if (datos.aux.length > 1) {
            handleGallery(datos.aux, datos.prevIndex, datos.postIndex, datos.index);
        }
        else {
            setShowRightArrow(false);
            setShowLeftArrow(false);
        }
    });

}, [id, randomNumber]);
```

Ilustración 28: Vista de Galería

```
const handleGallery = async (fotosAux: string[], prevIndex: number, postIndex: number, index: number) => {

    // Hemos construido un array con toda la galería de la actividad, también hemos obtenido el identificador de la foto anterior y la siguiente
    if (prevIndex < 0) {
        setShowRightArrow(true);
        setShowLeftArrow(false);
        setPostid(fotosAux[postIndex]);
    } else if (postIndex == index) {
        setShowLeftArrow(true);
        setShowRightArrow(false);
        setPrevid(fotosAux[prevIndex]);
    } else {
        setShowLeftArrow(true);
        setShowRightArrow(true);
        setPostid(fotosAux[postIndex]);
        setPrevid(fotosAux[prevIndex]);
    }
}
```

Ilustración 29: Gestión de la navegación

Por último, en la Ilustración 30, vemos de forma visual lo comentado previamente, al seleccionar cualquiera de las flechas de navegación, al tener asignado ya un identificador de la imagen que mostrar, vuelven a acceder a la misma pantalla, y el proceso previamente expuesto se repetirá.

```
<IonCol>
    {showLeftArrow &&
        <IonButton routerLink={`/my/viajes/${idViaje}/galeria/${previd}`}>
          <IonIcon icon={leftArrowIcon} slot='icon-only' />
        </IonButton>
    }
</IonCol>
<IonCol>
    {showRightArrow &&
        <IonButton routerLink={`/my/viajes/${idViaje}/galeria/${postid}`}>
            <IonIcon icon={rightArrowIcon} slot='icon-only' />
        </IonButton>
    }
</IonCol>
```

Ilustración 30: Flechas de navegación

Mapa en Actividad

Como se ha mencionado previamente en el apartado 4.4.2 Leaflet [22]se ha utilizado la herramienta mencionada para permitir a los usuarios indicar la ubicación de una actividad.

```
{ showMap &&
    <IonItem>
        <MapContainer center={[actividad?.y, actividad?.x]} zoom={17} style={{height:'400px', width: '100%'}}>
            <ChangeView center={[actividad?.y, actividad?.x]} zoom={17} />
            <ResizeMap />
            <TileLayer attribution="Map data &copy; <a href=&quot;https://www.openstreetmap.org/&quot;>OpenStreetMap</a> contributors"
                       url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
            <Marker position={[actividad?.y, actividad?.x]} icon={new Icon({iconUrl: "../../assets/alfiler.png", iconSize: [25,25]})}/>
            <LocationMarker />
        </MapContainer>
    </IonItem>
}
```

Ilustración 31: Componentes para el mapa

En la Ilustración 31 podemos observar los diferentes componentes utilizados para la creación y correcta gestión de este:

- '**MapContainer'**: Establece la ubicación de inicio, tamaño y zoom al inicializar y acceder a la pantalla de Actividad
- '**TileLayer'**: Encargado de dar formato al mapa, es decir, asignar colores y una estética a la visualización de este
- '**ChangeView'**: Si las coordenadas son editadas, este se encarga de volver a centrar el mapa (ejemplo: el usuario busca una ubicación en la barra de búsqueda). Ver Ilustración 32.
- '**ResizeMap'**: Se encarga de garantizar el correcto dimensionamiento del mapa en el espacio reservado para este.
- '**LocationMarker'**: Se trata de un icono, que indica la posición actual del usuario en el mapa. Ver Ilustración 33

```
// Correcto render del mapa
function ResizeMap () {
    const map = useMap();
    map._onResize();
    return null;
}

// Si editamos las coordenadas volvemos a centrar el mapa
// @ts-ignore
function ChangeView({ center, zoom }) {
    const map = useMap();
    map.setView(center, zoom);
    return null;
}
```

Ilustración 32: Centrado y dimensionamiento del mapa

```
const LocationMarker = () => {
    const [position, setPosition] = useState(null);

    const map = useMap();

    useEffect(() => {
      map.locate().on("locationfound", function (e) {
        setPosition(e.latlng);
      });
    }, [map]);

    return position === null ? null : (
      <Marker position={position} icon={new Icon({iconUrl: "../../assets/ownLocation.png", iconSize: [25,25]})}>
        <Popup>
          Estás aquí
        </Popup>
      </Marker>
    );
}
```

Ilustración 33: Ubicación del usuario

Por último, se ha mencionado una barra de búsqueda dentro del mapa, este componente se denomina '**SearchControl'** (ver Ilustración 34). Se trata de una funcionalidad que aporta **Leaflet**, en este caso se llama '**OpenStreetMapProvider'** que permite al usuario seleccionar entre las diferentes ubicaciones sugeridas mientras escribe la ubicación deseada.

```
// Barra de búsqueda en el mapa
const SearchControl = () => {

  const map = useMap();

  // Obtenemos las coordenadas de la ubicación buscada en la barra de búsqueda
  const handleLocationSelected = (location) => {
  console.log(location.location);
  setX(location.location.x)
  setY(location.location.y)
}

  useEffect(() => {
      const provider = new OpenStreetMapProvider();

      // @ts-ignore
      const searchControl = new GeoSearchControl({
          provider,
          showMarker: true,
          marker: {
              icon: new Icon({
                  iconUrl: '../../assets/alfiler.png',
                  iconSize: [35,35]
              }),
              draggable: false
          },
          style: 'bar',
          autoClose: true,
          animateZoom: true,
          keepResult: true,
          searchLabel: 'Introduzca una dirección',
          notFoundMessage: 'Dirección no encontrada',
          messageHideDelay: 3000,
          updateMap: true,
      });

      map.addControl(searchControl);

      map.on('geosearch/showlocation', handleLocationSelected);

      return () => {
          map.removeControl(searchControl);
      };
  }, [map]);
  return null;
}
```

Ilustración 34: Barra de búsqueda en el mapa

5.4.2 Base de datos

Para la base de datos, como se ha expresado previamente, se ha decidido utilizar Firebase y sus diferentes servicios [32]. A través de esta herramienta, se ha creado una estructura en la base de datos (Ilustración 35) que permite a los diferentes modelos de datos, y mediante claves foráneas (Ilustración 36), estar asignados a otros modelos de datos, lo que permite que una misma imagen pueda ser mostrada tanto en la galería de viaje como en la galería de viaje sin duplicar entradas.

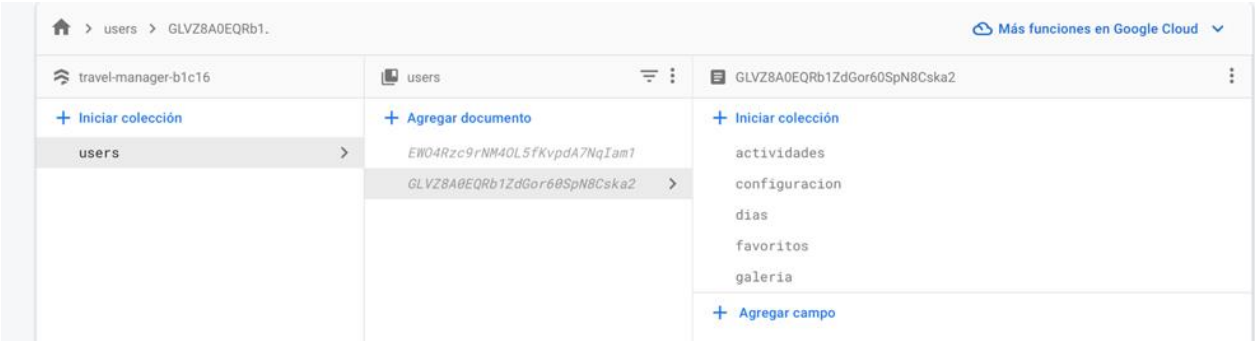

Ilustración 35: Estructura del proyecto en Firebase

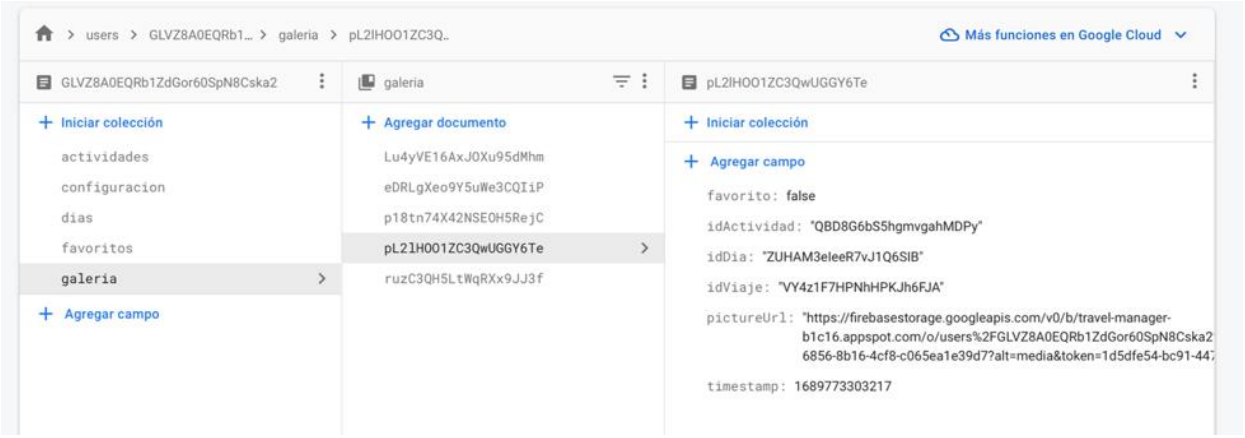

Ilustración 36: Estructura de los componentes en Firebase

Se comprueba que la estructura de la aplicación pasa por almacenar las entradas para cada usuario dentro del mismo identificador de usuario (que solo se puede obtener si estás registrado e iniciado sesión en el sistema).

Vemos también en la galería (Ilustración 36), una imagen con identificadores de viaje, día y actividad para poder ser mostrada en diferentes pantallas de la aplicación.

6. Presentación de la aplicación

En este apartado se va a realizar una presentación de la aplicación. Se incluirá un breve resumen sobre cada una de las pantallas de la aplicación, es decir, con los aspectos más importantes a conocer por el usuario para sacar el máximo provecho a la misma. Este apartado será un resumen de lo que se puede encontrar más en detalle en el Anexo V.

Ilustración 37: Logo Travel Diaryr

6.1 Menú Principal

En esta pantalla encontramos los siguientes componentes:

- **Barra de herramientas**: permite al usuario acceder a los ajustes de la aplicación.

- **Barra de navegación**: permite al usuario elegir que listado de viajes mostrarle dependiendo de donde se coloque el cursor, si el usuario selecciona '***PRÓXIMOS VIAJES'***, la aplicación mostrará los viajes que el usuario tiene planeados, por el contrario, si el usuario selecciona '***VIAJES PASADOS***', la aplicación mostrará viajes del usuario ya finalizados.
- **Contenido de la pantalla**: Muestra al usuario el listado de viajes dependiendo de lo seleccionado en la barra de navegación, estos viajes pueden ser pasados o futuros (Ilustración 38).

- **Botón '+'**: Permite al usuario acceder a una nueva pantalla donde pueda crear un nuevo viaje.

Ilustración 38: Menú principal

6.2 Pantalla de Viaje

En esta pantalla, vamos a entrar en el menú principal del Viaje, en este caso, disponemos, como en el menú principal, de una barra de navegación superior, que permitirá al usuario elegir qué información mostrar.

Los componentes principales que encontramos en esta pantalla son los siguientes (ver Ilustración 39 e Ilustración 41):

- **Botón de vuelta atrás**: Se trata de un botón con forma de flecha, que una vez seleccionado nos permite retroceder a la pantalla anterior.

- **Opciones del viaje**: Se trata del botón ubicado en la esquina superior izquierda, con forma de tres puntos verticales, que nos permitirá abrir un pequeño desplegable con una serie de opciones para el viaje.
- **Barra de navegación:** Esta barra de navegación nos permitirá acceder a la información general del viaje, a la galería de fotos que el usuario haya decidido incluir al viaje, y también un listado con los momentos destacados del viaje.

- **Imagen de perfil del viaje**: Imagen que se muestra en la pantalla principal, como se veía en el ejemplo anteriormente expuesto de Milán.

- **Listado de días (calendario de viaje)**: Se trata de un listado con los días del viaje, el usuario podrá seleccionar cualquiera de estos días para acceder a las actividades planeadas para cada día (ver Ilustración 39).

- **Galería de viaje:** Se trata de todas las imágenes que el usuario ha incluido en las actividades de las que está compuesto el viaje (ver Ilustración 41).

- **Momentos destacados del viaje**: Aquellas imágenes marcadas como 'favoritas' por el usuario al visualizar la galería, son recogidas en este apartado de la 'Barra de navegación'. Por lo tanto, se trata de un listado de mejores momentos del viaje, donde el usuario además habrá podido incluir alguna nota o mensaje de voz. [33] (ver Ilustración 40)

Ilustración 39: Pantalla de viaje - Información general

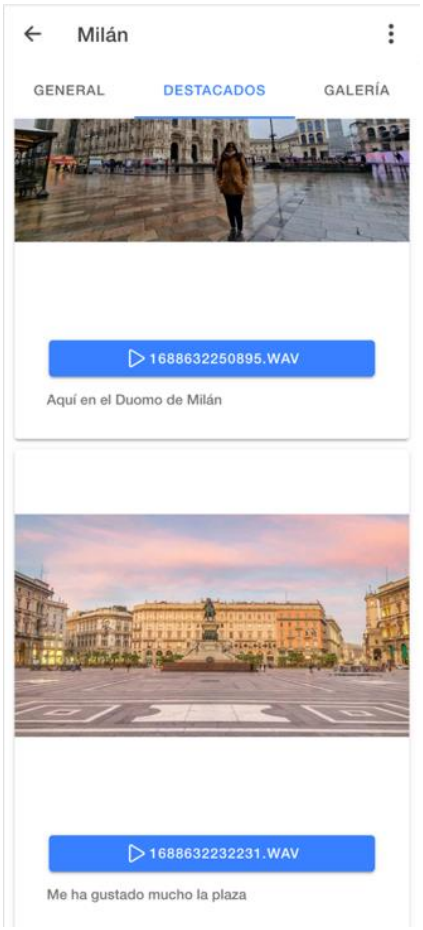

Ilustración 40: Momentos destacados del viaje

Ilustración 41: Pantalla de viaje – Galería

6.3 Pantalla de Actividad

En esta pantalla el usuario podrá acceder a toda la información referente a la actividad que va a realizar (ver Ilustración 42).

- **Barra de navegación**: Esta barra tiene el mismo funcionamiento que en la 'Pantalla de Viaje'. En este caso solo puedes acceder a dos botones, el primero '***INFORMACIÓN GENERAL***' permite al usuario acceder a la información de su actividad, el botón '***GALERÍA***' permite acceder al usuario a las fotos que haya hecho durante la actividad.

- **Fecha de Inicio y descripción**: Información básica para la actividad, hora de comienzo, que se tendrá en cuenta a la hora de notificar al usuario del comienzo de dicha actividad, y una pequeña descripción donde el usuario pueda escribir en que consiste la actividad.

- **Mapa**: Permite al usuario ubicar la localización de la actividad. Es un mapa que muestra con un icono rojo la ubicación exacta. Además, el usuario puede desplazarse por el mismo libremente, e incluso podrá ver su ubicación. [34]

- **Documento adjunto**: En caso de que para esta actividad se necesite un documento, el usuario podrá adjuntarlo a la actividad, para una vez se solicite, el usuario pueda clicar en el para abrir el documento PDF.

- **Barra superior de herramientas**: Esta barra, como en otras pantallas, permite al usuario volver hacia la pantalla anterior, o acceder a las opciones como ya ocurría en la 'Pantalla de Viaje'

Ilustración 42: Pantalla de actividad

6.4 Galería

En esta pantalla el usuario podrá visualizar las imágenes que haya decidido subir previamente a la actividad, si el usuario accede a la misma desde la pantalla de actividad, o al viaje si el usuario accede desde la pantalla de viaje.

El funcionamiento es exactamente el mismo que tiene la galería de un dispositivo móvil corriente. Si desea acceder a la imagen, el usuario únicamente debe pulsar encima de la imagen que desee visualizar.

Adicionalmente, los siguientes componentes también serán visualizados por el usuario al pulsar sobre una imagen (ver Ilustración 43 e Ilustración 44).

- **Barra de cerrar:** Se trata de un botón con forma de cruz, que permitirá volver a la galería de imágenes al usuario
- **Botón favoritos:** Botón con forma de estrella, que permitirá al usuario crear un momento destacado con la foto en cuestión.
- **Botones de navegación:** Se trata de botones con forma de flecha, con direcciones izquierda y derecha, que tras pulsarlos permitirán acceder a la visualización de la imagen inmediatamente anterior o posterior
- **Botón eliminar:** Botón con forma de basura que permitirá eliminar la foto de la galería de imágenes.

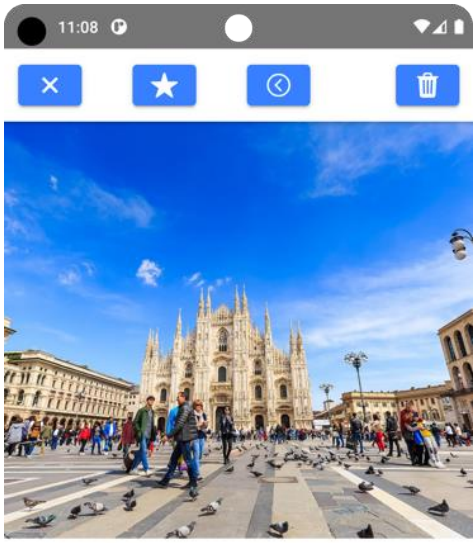

Ilustración 43: Ver Imagen 1

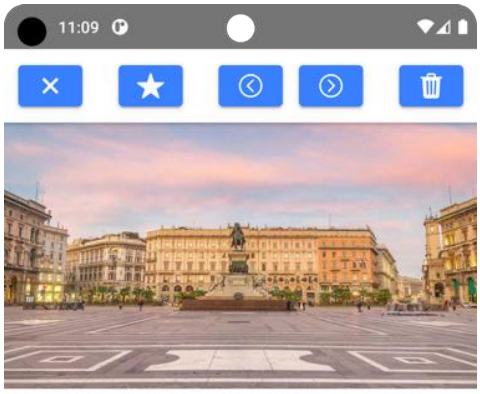

Ilustración 44: Ver imagen 2

6.5 Notificaciones [35]

Previamente se mencionó como opción de configuración la antelación con la que el usuario recibiría la notificación de que su actividad iba a comenzar pronto. En este caso, y como se muestra en la Ilustración 45 e Ilustración 46, existen dos tipos de notificación. Aquella para las actividades con documento adjunto, que permitirá abrir el PDF al seleccionar la notificación, y aquella para las actividades sin documento adjunto, que únicamente te avisarán de que la actividad está a punto de comenzar.

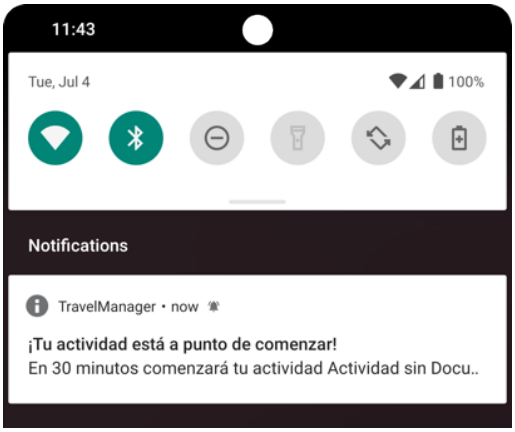

Ilustración 45: Notificación sin documento adjunto

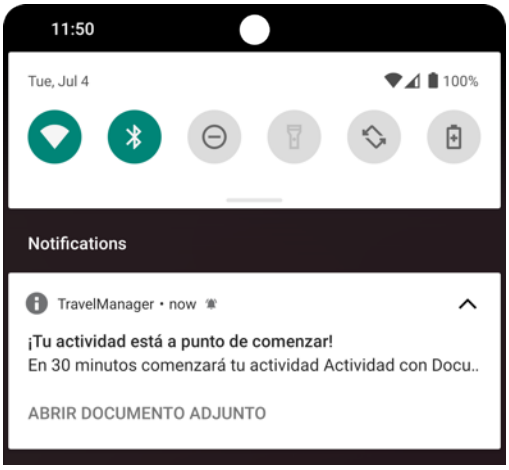

Ilustración 46: Notificación con documento

7. Conclusiones y futuras líneas de trabajo

7.1 Conclusiones

Con el proyecto finalizado se pueden sacar diferentes conclusiones.

En cuanto a los objetivos software:

- Se ha conseguido desarrollar una aplicación Android completamente funcional, que podría ser incluida en la tienda Google Play Store.
- Gestión satisfactoria de base de datos, autenticación y almacenamiento de archivos.
- Creación de notificaciones en demanda, incluyendo archivos con estas.
- Obtención de geolocalización del usuario integrada en la aplicación.
- Grabación de mensajes de voz, junto a una correcta gestión de los datos grabados.
- Control de la aplicación 'Cámara' del dispositivo para incluir imágenes tomadas al momento.

Gracias a las tecnologías empleadas, el desarrollo de la aplicación ha sido sencillo. El aprendizaje y uso del framework de **Ionic** con **React** ha permitido que las tareas de desarrollo no tuviesen una gran dificultad, ya que esta nos permite crear aplicaciones de forma intuitiva y con gran funcionalidad debido a las características de este. Por otro lado, Capacitor ha sido un gran factor adicional debido a que se ha podido hacer uso de las funcionalidades propias de los dispositivos móviles (**Plugins** de **Capacitor**) de forma trivial y ha permitido hacer la portabilidad del código escrito en TypeScript, React y JavaScript a Android con facilidad.

Por último, es importante comentar la facilidad de uso que permite **Firebase** como base de datos, de almacenamiento de imágenes y archivos, además de proporcionar el mecanismo de autenticación de los usuarios. Hay que comentar que, a pesar de contar con una documentación extensa, fácil de comprender y actualizada, la recuperación de datos, en algunas situaciones específicas no ha sido la más intuitiva debido a la ambigüedad de su documentación.

Durante la realización de este proyecto, se ha podido poner en valor el aprendizaje obtenido durante estos seis años cursando este grado, se han podido aplicar los conocimientos de gestión de proyectos, ingeniería del software, etc. Además, se ha disfrutado de la posibilidad de realizar un desarrollar una aplicación completa y funcional con una metodología ágil.

He de comentar también, que he aprendido tanto nuevos lenguajes de programación, como a utilizar herramientas que facilitan en gran medida el desarrollo de aplicaciones móviles, las cuales, se podrían utilizar en futuros proyectos personales o profesionales.

Desde mi punto de vista, la experiencia vivida estos meses trabajando en el proyecto ha sido altamente satisfactoria, si es cierto que han existido momentos tensos por la frustración de no entender que fallaba, o no tener clara alguna idea de cómo implementar alguna funcionalidad. Pero habiendo ya experimentado lo que es el mundo laboral, no deja de ser algo común tanto en nuestro sector, como en muchos otros. Trabajar solo ha tenido sus ventajas, debido a que la gran mayoría de trabajos realizados durante el grado se realizan por parejas, y siempre anhelas la independencia de tener tus propios tiempos para trabajar en el proyecto.

Por último, se han observado ciertos aspectos a mejorar en la implementación y gestión de proyectos. Estos van a servir para aprender y evolucionar en el desarrollo y gestión de futuros retos.

7.2 Futuras líneas de trabajo

El proyecto cuenta con varias líneas de trabajo futuras tanto en la ampliación de este como en el perfeccionamiento de ciertos aspectos que ayudarían a que la aplicación fuera mejor, estas líneas de trabajo son:

- Eliminar la dependencia de Capacitor para realizar la portabilidad a Android, es decir, realizar el desarrollo utilizando los lenguajes nativos (Java/Kotlin) directamente en Android Studio, lo que permitiría no depender de los Plugin de Capacitor, que no terminan de funcionar de la forma óptima esperada. Este cambio podría resultar en:

    - Correcta gestión de las notificaciones. Como ya se comentó en el Anexo IV, la gestión debido al Plugin proporcionado por Capacitor es 'justita', trabajando en nativo permitiría realizar una gestión más optimizada que permitiese añadir diferentes funcionalidades y pulir el resultado final.

    - Más opciones para subir imágenes a la galería del viaje/actividad. El Plugin funciona correctamente, pero si es cierto que, trabajando de forma nativa, podríamos elegir entre los diferentes álbumes creados en nuestra galería, como se hace con otras aplicaciones que permite al usuario encontrar de forma más sencilla la imagen deseada.

    - Lector del documento PDF adjunto. Actualmente se utiliza un Plugin por el cual el documento, una vez, seleccionado por el usuario, se abre en el navegador del dispositivo móvil, esta fue la solución proporcionada por el framework y Capacitor, puesto que las soluciones ofrecidas por la comunidad no funcionaban debido a su incompatibilidad con Capacitor al realizar la portabilidad. Lo ideal sería trabajar en nativo para realizar el visionado del documento en la propia aplicación.

- Permitir al usuario escribir libremente sus notas a la hora de crear sus momentos destacados, que pudiesen crear listas, remarcar palabras o frases, etc. Es decir, dotar al usuario de mayor libertad a la hora de crear sus notas.

- Permitir a los usuarios compartir el viaje con sus amigos, para que todos puedan tener acceso a este, puedan añadir actividades, billetes, imágenes para la galería, etc.

- Añadir la posibilidad de incluir en el mapa el funcionamiento 'tipo' GPS, es decir, que dentro de la aplicación se genere una ruta para que el usuario llegue de la forma más eficiente a su actividad.

8. Bibliografía

Para la bibliografía se ha utilizado en todo momento el estándar IEEE.

[1] L. Herazo, «Anincubator,» [En línea]. Available: https://anincubator.com/que-es-una-aplicacion-movil/.

[2] Developers Android, «Aspectos fundamentales de la aplicación,» [En línea]. Available: https://developer.android.com/guide/components/fundamentals?hl=es-419.

[3] S. G. Sotomayor, «IEBS,» [En línea]. Available: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/.

[4] K. B. M. B. A. C. A. v. B. W. C. A. H. M. F. D. T. J. H. R. J. J. K. J. G. R. C. M. S. M. K. S. y. J. S. Brian Marick, «Manifiesto ágil,» [En línea].

[5] Sentrio, «Los 4 valores y 12 principios del 'Manifiesto Ágil',» [En línea]. Available: https://sentrio.io/blog/valores-principios-agile-manifiesto-agil/.

[6] K. S. y. J. Sutherland, «Guía Scrum,» [En línea]. Available: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf.

[7] The Redbooth Team, «The 3 main roles in an Agile Team,» [En línea]. Available: https://redbooth.com/blog/main-roles-agile-team.

[8] Jerónimo Palacios, «Facilitación gráfica - Eventos en Scrum,» [En línea]. Available: https://jeronimopalacios.com/agile/facilitacion-grafica-eventos-en-scrum/.

[9] J. Roche, «Deloitte - Artefactos Scrum: las 3 herramientas clave de gestión,» [En línea]. Available: https://www2.deloitte.com/es/es/pages/technology/articles/artefactos-scrum.html.

[10] Editorial Etecé, «¿Qué es turismo?,» [En línea]. Available: https://concepto.de/turismo/.

[11] P. R. Bernardí Cabrer-Borrás, «Impacto económico del sector del turismo en España,» [En línea]. Available: https://roderic.uv.es/bitstream/handle/10550/80528/148089.pdf?sequence=1.

[12] C. Iglesias, «Contribución Social del Sector Turístico Español,» [En línea]. Available: https://www2.deloitte.com/es/es/pages/risk/articles/contribucion-social-sector-turistico.html.

[13] VIAJA. VIVE. VINCCI., «Diario de viaje: qué es y cómo hacer un cuaderno de viajero,» [En línea]. Available: https://www.vinccihoteles.com/blog/diario-de-viaje-que-es-y-como-hacer-un-diario-de-viaje/.

[14] Wikipedia, «Visual Studio Code,» [En línea]. Available: https://es.wikipedia.org/wiki/Visual_Studio_Code.

[15] J. M. A. Atmitim, «Qué es Ionic: ventajas y desventajas de usarlo para desarrollar apps móviles híbridas,» [En línea]. Available: https://profile.es/blog/que-es-ionic/.

[16] Ionic Framework, «Ionic Docs,» [En línea]. Available: https://ionicframework.com/docs.

[17] «Apuntes.de,» [En línea]. Available: https://apuntes.de/ionic/crear-aplicaciones-nativas-con-capacitor/#gsc.tab=0.

[18] Capacitor, «Capacitor Docs - Local Notifications,» [En línea]. Available: https://capacitorjs.com/docs/apis/local-notifications.

[19] Capacitor Docs, «@capacitor/camera,» [En línea]. Available: https://capacitorjs.com/docs/apis/camera.

[20] J. Lucas, «Qué es NodeJS y para qué sirve,» [En línea]. Available: https://openwebinars.net/blog/que-es-nodejs/.

[21] Google, «Firebase Documentation,» [En línea]. Available: https://firebase.google.com/docs/reference/js?hl=es-419.

[22] V. Agafonkin, «Leaflet,» [En línea]. Available: https://leafletjs.com.

[23] J. Yesares, «¿Qué es Leaflet?,» [En línea]. Available: https://wpcarto.com/blog/webmapping/que-es-leaflet-libreria-javascript/.

[24] Meta Open Source, «React,» [En línea]. Available: https://es.react.dev.

[25] D. A., «Qué es React: definición, características y funcionamiento,» [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-react.

[26] Wikipedia, «JavaScript,» [En línea]. Available: https://es.wikipedia.org/wiki/JavaScript.

[27] Wikipedia, «CSS,» [En línea]. Available: https://es.wikipedia.org/wiki/CSS.

[28] «Wrike,» [En línea]. Available: https://www.wrike.com/es/.

[29] Danysoft, «Wrike,» [En línea]. Available: https://www.danysoft.com/wrike/.

[30] N. Bouchard, «Wrike Beginners Guide,» [En línea]. Available: https://unito.io/blog/a-beginners-guide-to-gitlab/.

[31] J. M. Aguilar, «¿Qué es el patrón MVC en programación y por qué es útil?,» [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx.

[32] M. Nasato, «Ionic React: Cross-Platform Mobile Development with Ionic,» [En línea]. Available: https://www.udemy.com/share/10395o3@cGI5DcWUwOGEllh5QrUOuS1nxXLfw5FDmXPC7Y3VRdnc-mZCLRzshDizA9T8MWZw/.

[33] S. Grimm, «Ionic Audio Recording like WhatsApp with Capacitor,» [En línea]. Available: https://youtu.be/_OMpguY5uWg.

[34] GeoDev, «Leaflet crash course | All you need to know about leaflet | Leaflet | Tekson,» [En línea]. Available: https://youtu.be/ls_Eue1xUtY.

[35] A. Rathore, «Implement Local Notifications in Ionic 5 with Capacitor,» [En línea]. Available: https://enappd.com/blog/local-notifications-in-ionic-5-capacitor/132/.