Escuela Politécnica Superior

Diseño y desarrollo de app móvil: red social de viajes

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Alberto Gómez Ferrández

Tutor/es:
Otto Colomina Pardo

Junio 2020

Universitat d'Alacant
Universidad de Alicante

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

Con la gran expansión de las redes sociales teniendo lugar hace ya más de diez años, siendo el intercambio de información y las relaciones interpersonales por internet completamente indispensables, unas pocas plataformas se han hecho con el monopolio de las redes sociales, a veces unidas bajo la misma marca. Estas grandes plataformas agrupan a la gran mayoría de consumidores de estos productos, dando poca oportunidad a nuevos competidores que, para triunfar, deben buscar funcionalidades y nichos de población concretos, ofreciéndoles una solución óptima y especializada para la necesidad que se busca solventar.

En este proyecto se ha llevado a cabo el diseño y desarrollo de una nueva aplicación móvil llamada TravelApp. Este es una aplicación que busca poner en contacto a personas que comparten destino de viaje, ya sea para compartir información sobre el lugar, sugerencias, o incluso hacer planes juntos.

Esta aplicación surge con la motivación de llenar un hueco vacío en el mercado de aplicaciones móviles en el sector turístico-social, aportando un producto especializado en el cumplimiento de su función.

Este documento se centrará en el proceso de ingeniería del software que rodea al desarrollo de un proyecto como el que se plantea. Ahondaremos en el análisis y especificación del proyecto a elaborar, el diseño de la interfaz gráfica y experiencia de usuario, y, también, haremos hincapié en las metodologías de desarrollo del software, claves para implementar una solución ordenada y fácilmente escalable con el tiempo.

1.1 Objetivos

Los objetivos del proyecto son:

1. Realizar un estudio de la situación actual de las aplicaciones turísticas-sociales.
2. Ofrecer una solución óptima a un problema real.
3. Hacer llegar un nuevo producto al mayor número de personas seleccionando las mejores tecnologías y plataformas para ello.
4. Implementar una solución incrementable y escalable con el tiempo.

1.2 Estado de la cuestión

TravelApp es un proyecto cuya característica principal a la hora de clasificarlo entre otras redes sociales, es su componente turístico. Es una solución que no tiene cabida si no se plantea una en la situación de realizar un viaje, por lo cual indagaré en las plataformas que hoy en día tengan un mayor uso con la intención que plantea nuestra aplicación.

No hay una aplicación clara cuya intención sea el socializar con personas que realicen el mismo viaje que tú, pero las hay que suplen esta necesidad de cierta manera. Nos centraremos en Erasmusu, CouchSurfing y Facebook:

Erasmusu

Erasmusu[1] es una plataforma online pensada para que los estudiantes que estén cerca de comenzar un intercambio por Europa encuentren facilidades para su estancia, tales como buscar alojamiento, compañeros de piso, profesores particulares, trabajo o simplemente encontrar y empezar a conocer otros estudiantes con los que comparten destino para entablar amistades.

Esta última característica es la que nos interesa. Esta plataforma se convierte de manera indirecta en un lugar en el que encontrar compañeros de viaje, de aventura, con los que poder compartir la experiencia y apoyar en la difícil situación de empezar una nueva vida en el extranjero.

Sin embargo, se centra exclusivamente en estudiantes de intercambio, con el consecuente rango de edad muy definido y con estancias muy largas. TravelApp busca no centrarse en un tipo específico de viajero, y que, sea cual sea su edad y motivo de su estancia, pueda encontrar personas afines.

[1] www.erasmusu.com

CouchSurfing

CouchSurfing[2] es una plataforma que permite ofrecer y encontrar alojamiento gratuito en casas de los usuarios registrados, con el objetivo intrínseco de fomentar el intercambio cultural y crear una comunidad de viajeros por el mundo. Los usuarios organizan eventos alrededor del mundo para conocer al resto de usuarios locales o viajeros actualmente en la zona.

La gran diferencia es que CouchSurfing se centra en conocer gente local, mientras que TravelApp busca juntar personas de cualquier lugar del mundo sin estar ligado a buscar alojamiento.

Facebook

Facebook[3] es la red social por excelencia, con una gran extensión de usuarios y funcionalidades, la cual no está pensada para encontrar gente viajando pero que, gracias a su amplia amalgama de funcionalidades, lo permite de cierto modo con los "grupos" de Facebook. Los grupos son comunidades de usuarios que se juntan en la red social con propósito de la temática del grupo. Esto permite que se creen comunidades de viajeros en cierto país, o de residentes extranjeros en cierto territorio, en las que se organizan eventos o actividades con los que socializar.

La diferencia que pretende marcar TravelApp es que, como ya hemos puntualizado, el punto de estos grupos es el de crear comunidades afianzadas para nuevos residentes en los lugares de destino, más que para juntar personas de forma más puntual.

[2] www.couchsurfing.com
[3] www.facebook.com

2. Tecnologías y herramientas

2.1 Tecnologías y herramientas en el Frontend

Cuando se habla de aplicaciones móviles podemos distinguir 3 tipos: aplicaciones web, aplicaciones nativas y aplicaciones híbridas.

Aplicaciones web

Las aplicaciones web son aquellas que se escriben en los lenguajes web estándar: JS, HTML y CSS. Estas aplicaciones requerirán del uso de un navegador web en el dispositivo para ejecutarlas.

Pros:

- Portabilidad, puede ejecutarse en cualquier dispositivo con navegador web
- Baja "barrera de entrada" para el desarrollador, no debe conocerse un lenguaje distinto para programar la aplicación para cada tipo de dispositivo.
- Se puede escapar al control de las tiendas de apps oficiales.

Contras:

- Rendimiento no aceptable para algunos tipos de aplicaciones (p.ej. juegos 3D)
- No tienen la apariencia de la plataforma móvil.
- Algunos APIs de componentes del teléfono no son accesibles (p.ej. acceso a la agenda de contactos)
- No se pueden vender en las tiendas de apps oficiales, ya que son sitios web.

Aplicaciones nativas

Las aplicaciones nativas son aquellas que son desarrolladas con el SDK nativo de la plataforma y en los lenguajes soportados por ellas.

Pros:

- Ofrecen el mejor rendimiento aprovechando al máximo el hardware.
- Acceso a todas las APIs nativas del teléfono.
- Apariencia de la plataforma.

Contras:

- Nula portabilidad, solo ejecutable desde dispositivos con el SDK nativo en cuestión.
- Desarrollo costoso.
- Requiere de programadores especializados en cada plataforma.

Aplicaciones híbridas

Las aplicaciones híbridas son aplicaciones web que se ejecutan dentro de un envoltorio que sí es una aplicación nativa.

Pros:

- Portabilidad entre plataformas.
- Al ser desde fuera una app clásica, se puede distribuir sin pegas en las *stores*.
- El envoltorio nativo suele dar acceso a APIs nativos desde Javascript.

Contras:

- Apariencia no del todo nativa.
- Rendimiento algo menor (relevante o no según el tipo de app).

Para este proyecto se ha desarrollado una aplicación híbrida atendiendo a su portabilidad y acceso a APIs nativas. Para construir la app a partir de la aplicación web se ha utilizado la herramienta llamada Cordova.

2.1.1 Cordova

Apache Cordova es un marco de desarrollo móvil de código abierto. Permite utilizar las tecnologías estándar web como HTML5, CSS3 y JavaScript para desarrollo multiplataforma, evitando el lenguaje de desarrollo nativo en cada plataforma móviles. Las aplicaciones ejecutan dentro de envolturas para cada plataforma y dependen de enlaces API para acceder a los sensores, datos y estado de la red de cada dispositivo (The Apache Software Foundation, 2020).

2.1.2 Ionic

Para el propio desarrollo de la aplicación web se ha utilizado el framework Ionic. Este es un marco que facilita la maquetación y *responsiveness* de las aplicaciones pensadas para móviles, otorgando bloques con cierto estilo predefinido en CSS. Ionic y Cordova tienen un gran acople y juntos ofrecen una gran solución para el desarrollo de apps móviles.

Como se ha nombrado anteriormente, además de proporcionar un marco básico para ejecutar una aplicación web dentro de una aplicación nativa, Cordova también proporciona API de JavaScript para permitir el acceso a una amplia variedad de funciones del dispositivo, como la base de datos de contactos. Estas capacidades se exponen mediante el uso de una colección de *plugins*. Los *plugins* proporcionan un puente entre la aplicación y los elementos nativos del dispositivo. El equipo de Ionic, además, creó Ionic Native, con interfaces TypeScript para más de 200 de los *plugins* más comunes (Ionic 2020).

2.2 Tecnologías y herramientas en el Backend

2.2.1 Node.JS

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones de red escalables. Pueden atenderse muchas conexiones simultáneamente. Por cada conexión, se activa la devolución de llamada o *callback*, pero si no hay trabajo que hacer, Node.js se dormirá. Esto contrasta con el modelo de concurrencia más común de hoy en día, en el que se emplean hilos del Sistema Operativo. HTTP es un elemento destacado en Node.js, diseñado teniendo en cuenta la transmisión de operaciones con *streaming* y baja latencia. Esto hace que Node.js sea muy adecuado para la base de una librería o un *framework* web (OpenJS Foundation, 2020).

En este caso, NodeJS ha sido utilizado para crear un API que servirá para ejecutar los métodos que la app móvil le mande a través de peticiones HTTP y servirá de nexo de unión entre esta y la base de datos.

2.2.2 MongoDB

Como sistema de base de datos se ha optado por MongoDB, un sistema de base de datos NoSQL, orientado a documentos y de código abierto.

MongoDB almacena datos en documentos flexibles similares a JSON, por lo que los campos pueden variar entre documentos y la estructura de datos puede cambiarse con el tiempo. El modelo de documento se asigna a los objetos en el código de su aplicación para facilitar el trabajo con los datos (MongoDB, Inc. 2020).

Se ha elegido este sistema por su gran integración con Javascript mediante librerías y el uso de JSON para estructurar lo datos, ya que este es el formato que se ha utilizado para enviar la información por red.

2.2.3 Servidor XMPP Openfire

Para implementar la mensajería instantánea se ha optado por el protocolo XMPP y el servidor Openfire (desarrollado por la compañía Ignite Realtime), fácil de utilizar y administrable desde su interfaz web.

El protocolo XMPP (Extensible Messaging and Presence Protocol) es una tecnología XML para la comunicación en tiempo real, que nutre una amplia gama de aplicaciones que incluyen mensajería instantánea.

Una tecnología clave de XMPP es la llamada BOSH (Bidirectional-streams Over Synchronous HTTP) para la comunicación bidireccional a través del protocolo HTTP, la cual se ha utilizado para el intercambio de mensajes entre clientes y servidor.

2.2.4 AWS EC2

Para lanzar la aplicación en producción se ha desplegado un servidor propio haciendo uso del producto EC2 de la plataforma de informática en la nube Amazon Web Services, también conocida como AWS.

La informática en la nube es la distribución de recursos de TI bajo demanda a través de Internet mediante un esquema de pago por uso. En vez de comprar, poseer y mantener servidores y centros de datos físicos, puede obtener acceso a servicios tecnológicos, como capacidad informática, almacenamiento y bases de datos, en función de sus necesidades a través de un proveedor de la nube como AWS (Amazon Web Services 2020a).

Amazon Elastic Compute Cloud (Amazon EC2) proporciona capacidad de computación escalable en la nube de AWS. El uso de Amazon EC2 elimina la necesidad de invertir inicialmente en hardware, de manera que puede desarrollar e implementar aplicaciones en menos tiempo. Se puede usar Amazon EC2 para lanzar servidores virtuales, configurar la seguridad, las redes y el almacenamiento. Permite escalar hacia arriba o hacia abajo para controlar cambios en los requisitos, con lo que se reduce la necesidad de prever el tráfico (Amazon Web Services 2020b).

2.3 Herramientas en el desarrollo

2.3.1 Visual Studio Code

Como editor de código se escogió Visual Studio Code ya que permite la instalación de extensiones creadas por terceros para una mejor experiencia con la sintaxis de ciertos lenguajes de programación o nuevas características de estos.

2.3.2 Git / GitHub

Se escogió Git como sistema de control de versiones y la plataforma web Github para alojar el repositorio remoto. En esta plataforma también se utilizó una funcionalidad para la gestión de proyectos siguiendo una metodología de desarrollo iterativo y ágil, ofreciendo la posibilidad de crear *issues*, tableros y etiquetas para marcar las versiones del programa, entre otras cosas.

El repositorio se encuentra en el siguiente enlace: https://github.com/AlbertoGF12/travelapp

2.3.3 Balsamiq Wireframes

Balsamiq es un producto diseñado para la creación de *wireframes* de interfaces de usuario. Ha sido utilizado para realizar los bocetos iniciales de la interfaz gráfica de la aplicación.

2.3.4 AdobeXD

AdobeXD es un software destinado al diseño y creación de prototipos y experiencias de usuario. Se ha utilizado para la creación de los *mockups* de la aplicación.

3. Arquitectura

3.1 Modelo Cliente-Servidor

La arquitectura de la aplicación se basa en el patrón cliente-servidor Este es un modelo de comunicación que vincula a varios dispositivos a través de una red donde, el cliente, realiza peticiones de servicios al servidor, que se encarga de satisfacer dichos requerimientos (Pérez Porto & Gardey, 2018).

Con esta arquitectura, las tareas se distribuyen entre los servidores (que proveen los servicios) y los clientes (que demandan dichos servicios). Este tipo de modelos permite repartir de la capacidad de procesamiento (Pérez Porto & Gardey, 2018).

De acuerdo con Sommerville (2005, p. 226) los clientes pueden conocer los nombres de los servidores disponibles y los servicios que proporcionan, mientras que los servidores no necesitan conocer la identidad de los clientes. Estos acceden a los servicios proporcionados a través de llamadas utilizando un protocolo de petición-respuesta como puede ser el protocolo HTTP (Sommerville 2005, p. 226).

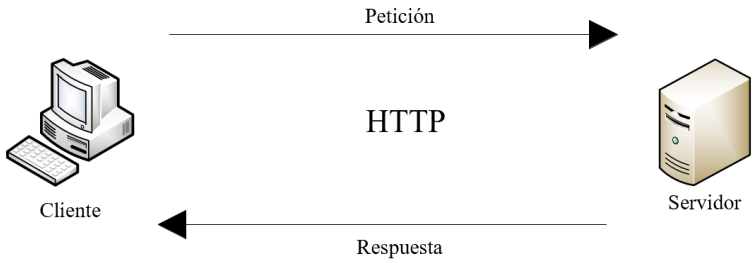

Figura 1: Modelo cliente-servidor

Una ventaja importante de este modelo es que es una arquitectura distribuida, se puede hacer uso de los sistemas de red con muchos procesadores distribuidos y es fácil añadir un nuevo servidor e integrarlo con el resto del sistema o actualizar los servidores de forma transparente sin afectar a este (Sommerville 2005, p. 227).

3.2 Arquitectura de la aplicación

En la parte del servidor disponemos de un servidor de aplicaciones que aloja tanto la aplicación *backend* de TravelApp, escrita en NodeJS, como el servicio servidor del chat o mensajería instantánea, implementado con Openfire. Disponemos también de un servidor de base de datos, al cual se conecta el servidor de aplicaciones para recoger la información solicitada por el cliente o escribir datos provenientes de este. La base de datos utilizada es MongoDB.

En el lado del cliente disponemos de una aplicación que se ejecutará en dispositivos móviles.

La comunicación entre el cliente y el servidor de aplicaciones se realiza mediante el protocolo HTTP. A pesar de que la mensajería instantánea utiliza el protocolo de comunicación XMPP, como ya se ha explicado en el apartado de Tecnologías y Herramientas, este protocolo dispone de una tecnología denominada BOSH, la cual permite utilizar el protocolo XMPP a través de comunicaciones con HTTP.

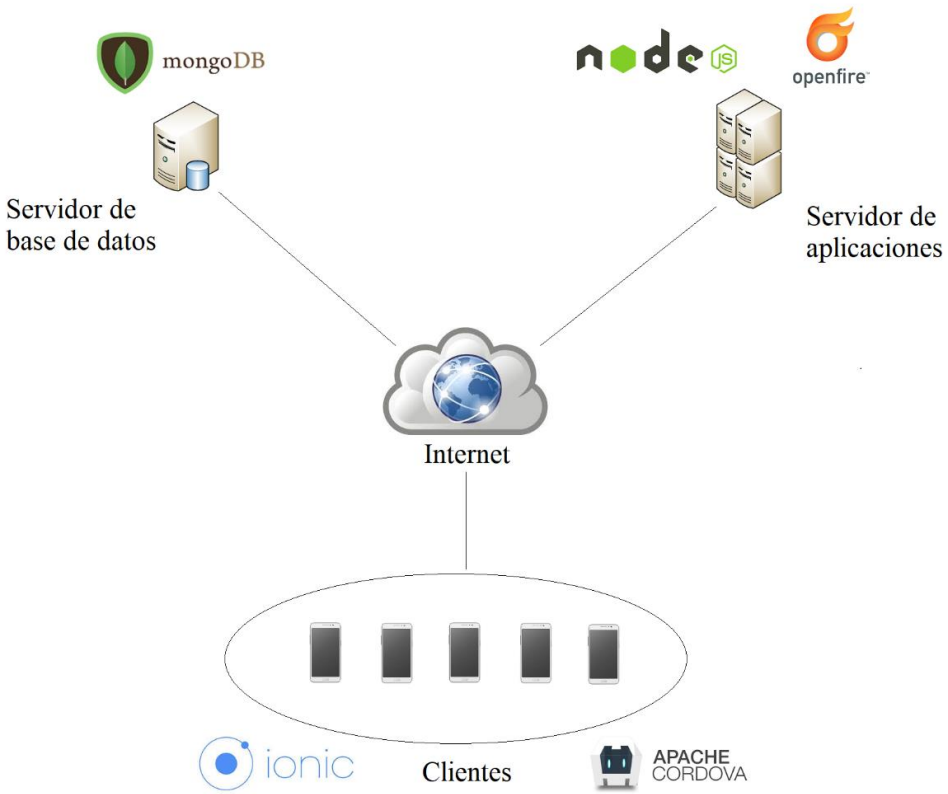

Figura 2: Arquitectura de la aplicación

4. Diseño de la aplicación

En el proceso de especificación y diseño de la aplicación primero se definieron los requisitos o requerimientos funcionales de la aplicación. Seguidamente veremos el diagrama de clases y los casos de uso. Por último, repasaremos el diseño inicial de la interfaz de la aplicación.

4.1 Requisitos funcionales

De acuerdo con Sommerville (2005, p. 108), los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayuda a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información (Sommerville 2005, p. 108). El proceso de descubrir, analizar, documentar y verificar estos servicios o restricciones se denomina ingeniería de requerimientos (Sommerville 2005, p. 108).

Según Sommerville (2005, p. 109) los requerimientos funcionales, en concreto, son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Sommerville 2005, p. 109).

Los requerimientos funcionales de TravelApp son:

Registrar usuario
Justificación: Un usuario debe tener una cuenta en la aplicación.
Precondiciones: El usuario debe iniciar la app por primera vez o haber cerrado su antigua sesión.
Datos de entrada: Nombre de usuario, email, contraseña, confirmación de contraseña, biografía, fecha de nacimiento y foto de perfil.
Descripción: El usuario introduce sus datos para crear una cuenta.
Pruebas de Aceptación: email no repetido en la base de datos, campos de email, contraseña, fecha de nacimiento y nombre no pueden estar vacíos.
Postcondiciones: Los datos del usuario se guardarán en la base de datos, la foto en el sistema de archivos del servidor y la sesión se iniciará automáticamente accediendo a la pantalla principal.

Tabla 1: Requisito funcional 1 - Registrar usuario

Subir foto de perfil
Justificación: Un usuario puede subir una foto de perfil al registrarse.
Precondiciones: El usuario debe tener la app instalada, no tener cuenta y estar registrándose.
Datos de entrada: Foto almacenada en el dispositivo.
Descripción: Se selecciona una foto de la galería para subirla a la aplicación.
Pruebas de Aceptación: el archivo seleccionado debe ser una imagen.
Postcondiciones: La imagen se sube al servidor, se guarda en su sistema de archivos y se modifica el registro del usuario en la BD, añadiendo la ruta a su foto de perfil.

Tabla 2: Requisito funcional 2 – Subir foto de perfil

Recortar foto
Justificación: Un usuario quiere recortar la foto para subirla.
Precondiciones: El usuario debe haber seleccionado una foto desde la galería.
Datos de entrada: Foto almacenada en el dispositivo.
Descripción: Cuando el usuario selecciona una foto de su dispositivo, esta se abre con un marco para recortar la imagen con una relación de aspecto 1:1.
Pruebas de Aceptación: el archivo seleccionado debe ser una imagen.
Postcondiciones: La imagen resultado será la foto disponible para las siguientes acciones.

Tabla 3: Requisito funcional 3 – Recortar foto

Iniciar sesión
Justificación: Un usuario registrado quiere iniciar sesión en su cuenta desde cierto dispositivo.
Precondiciones: El usuario debe estar registrado en el sistema y no debe haber una sesión abierta en el dispositivo.
Datos de entrada: Correo electrónico y contraseña.
Descripción: El usuario introduce sus datos para acceder a su cuenta.
Pruebas de Aceptación: El correo electrónico y la contraseña son válidos.
Postcondiciones: La sesión se iniciará accediendo a la pantalla principal, el token de sesión se almacenará.

Tabla 4: Requisito funcional 4 – Iniciar sesión

Cerrar sesión
Justificación: Un usuario quiere cerrar su sesión en su cierto dispositivo.
Precondiciones: El usuario debe estar loggeado en el sistema.
Datos de entrada: Token de sesión.
Descripción: Se elimina el token de sesión del dispositivo.
Pruebas de Aceptación: Ninguna.
Postcondiciones: La sesión se cerrará, borrando el token almacenado y accediendo a la pantalla de inicio.

Tabla 5: requisito funcional 5 – Cerrar sesión

Ver perfil personal
Justificación: Un usuario quiere ver el estado actual de su perfil.
Precondiciones: El usuario debe estar loggeado en el sistema.
Datos de entrada: Ninguno.
Descripción: Un usuario pulsa el botón de enlace a su perfil personal.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se muestra por pantalla el perfil personal del usuario.

Tabla 6: Requisito funcional 6 – Ver perfil personal

Modificar foto de perfil
Justificación: Un usuario quiere cambiar su foto de perfil una vez estando registrado.
Precondiciones: El usuario debe tener la sesión iniciada en la app.
Datos de entrada: Foto almacenada en el dispositivo, ruta a la foto de perfil actual.
Descripción: El usuario selecciona una foto de la galería, la recorta y la sube.
Pruebas de Aceptación: el archivo seleccionado debe ser una imagen.
Postcondiciones: La imagen se sube al servidor, se guarda en su sistema de archivos, se borra la foto anterior y se modifica el registro del usuario en la BD, añadiendo o modificando la ruta a su foto de perfil.

Tabla 7: Requisito funcional 7 – Modificar foto de perfil

Subir foto
Justificación: Un usuario quiere subir una foto a su perfil.
Precondiciones: El usuario debe estar loggeado en la app.
Datos de entrada: Foto almacenada en el dispositivo.
Descripción: Además de la foto de perfil, el usuario puede tener más fotos en la aplicación. La selecciona de la galería, la recorta y la sube al servidor.
Pruebas de Aceptación: el archivo seleccionado debe ser una imagen y el usuario no debe de tener más de 6 fotos subidas.
Postcondiciones: La imagen se sube al servidor, se guarda en su sistema de archivos y se modifica el registro del usuario en la BD, añadiendo la ruta al array de rutas de sus fotos. Se redirige a su perfil personal con la imagen ya en pantalla.

Tabla 8: Requisito funcional 8 – Subir foto

Eliminar foto
Justificación: Un usuario quiere eliminar una foto de su perfil.
Precondiciones: El usuario debe estar loggeado en la app y tener al menos una foto.
Datos de entrada: ruta de la foto en el servidor.
Descripción: El usuario puede seleccionar la foto que desea borrar de su perfil para eliminarla del servidor.
Pruebas de Aceptación: Ninguna.
Postcondiciones: La imagen se elimina del servidor, se modifica el registro del usuario en la BD, eliminando la ruta de array de rutas de sus fotos. La foto se elimina automáticamente de la pantalla de su perfil personal.

Tabla 9: Requisito funcional 9 – Eliminar foto

Modificar perfil
Justificación: El usuario quiere modificar la información de su perfil personal.
Precondiciones: El usuario debe estar loggeado en el sistema en la página de su perfil personal.
Datos de entrada: nombre de usuario, descripción, fotos, rango de edad de gente que quiere encontrar.
Descripción: El usuario accede a la opción de editar perfil, introduce los datos que quiera cambiar y acepta los cambios.
Pruebas de Aceptación: Rango de edad válido entre 18 y 100 años.
Postcondiciones: Se redireccionará al usuario a su propio perfil con la información actualizada.

Tabla 10: Requisito funcional 10 – Modificar perfil

Modificar contraseña
Justificación: El usuario quiere modificar la contraseña de su cuenta.
Precondiciones: El usuario debe estar loggeado en el sistema en la página de su perfil personal.
Datos de entrada: contraseña nueva y actual
Descripción: El usuario debe introducir su contraseña actual, dos veces la nueva y aceptar el cambio.
Pruebas de Aceptación: Contraseña actual correcta, las dos contraseñas nuevas deben coincidir y contener más de 6 caracteres.
Postcondiciones: Se redireccionará al usuario a su propio perfil con la contraseña actualizada.

Tabla 11: Requisito funcional 11 – Modificar contraseña

Añadir nuevo destino
Justificación: El usuario quiere añadir un nuevo destino para encontrar personas y eventos.
Precondiciones: El usuario debe estar loggeado en el sistema.
Datos de entrada: ciudad de destino, fecha de inicio y de fin.
Descripción: El usuario accede a la pantalla de nuevo destino, introduce los datos y pulsa el botón de aceptar.
Pruebas de Aceptación: El destino debe estar guardado en la base de datos, las fechas deben ser válidas.
Postcondiciones: Se redireccionará al usuario a la pantalla de destinos guardados.

Tabla 12: Requisito funcional 12 – Añadir nuevo destino

Eliminar destino
Justificación: El usuario quiere eliminar un destino de su lista.
Precondiciones: El usuario debe estar loggeado en el sistema y tener al menos un destino.
Datos de entrada: Id del destino.
Descripción: El usuario elimina uno de los destinos que tiene guardados para dejar de ver los eventos y las personas del lugar.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se eliminará el destino de la BD y se dejará de mostrar el destino en cuestión. Se eliminarán también los chats asociados a tal destino.

Tabla 13: Requisito funcional 13 – Eliminar destino

Ver eventos
Justificación: El usuario quiere ver los eventos que tienen lugar en sus destinos.
Precondiciones: El usuario debe estar loggeado en el sistema y tener al menos un destino guardado.
Datos de entrada: id del destino.
Descripción: El usuario entra en la página de eventos y selecciona un destino.
Pruebas de Aceptación: Ninguna.
Postcondiciones: La lista de eventos que tienen lugar en su destino se muestran por pantalla.

Tabla 14: Requisito funcional 14 – Ver eventos

Filtrar eventos
Justificación: El usuario quiere filtrar los eventos de cierto destino según categoría.
Precondiciones: El usuario debe estar loggeado en el sistema y tener al menos un destino guardado.
Datos de entrada: Categoría de la que quiere buscar eventos.
Descripción: El usuario puede seleccionar una categoría de una lista para filtrar los resultados de eventos.
Pruebas de Aceptación: Ninguna
Postcondiciones: La lista de eventos que tienen lugar en ese destino y con esa categoría se muestran por pantalla.

Tabla 15: Requisito funcional 15 – Filtrar eventos

Ver perfil de evento
Justificación: El usuario quiere ver la información de un evento.
Precondiciones: El usuario debe estar loggeado en la aplicación y tener al menos un destino guardado.
Datos de entrada: El id del evento seleccionado
Descripción: El usuario pulsa en el evento de la lista del que quiere más información.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se redireccionará a la página perfil de ese evento mostrando sus datos.

Tabla 16: requisito funcional 16 – Ver perfil de evento

Ver página oficial de evento
Justificación: El usuario quiere ver un evento con más detalle.
Precondiciones: El usuario debe estar loggeado en la aplicación y tener al menos un destino guardado.
Datos de entrada: url de la página oficial del evento.
Descripción: El usuario pulsa en el botón de visitar página oficial desde el perfil de un evento.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se redireccionará a la página oficial de ese evento abriendo el navegador del dispositivo desde la aplicación.

Tabla 17: Requisito funcional 17 – Ver página de evento

Ver lista de usuarios
Justificación: El usuario quiere ver qué personas viajan a su mismo destino.
Precondiciones: El usuario debe estar loggeado y tener al menos un destino guardado.
Datos de entrada: Id del destino.
Descripción: El usuario seleccionará un destino del desplegable para ver las personas que viajan a este.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se mostrarán por pantalla los usuarios con los que coincida en lugar y fechas del destino.

Tabla 18: Requisito funcional 18 – Ver lista de usuarios

Ver perfil de otro usuario
Justificación: El usuario quiere ver la información de otro usuario.
Precondiciones: Estar loggeado en el sistema, tener al menos un destino guardado y haber buscado personas con ese destino.
Datos de entrada: Id del usuario.
Descripción: El usuario pulsará sobre un usuario de la lista.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se redireccionará al usuario al perfil del usuario seleccionado.

Tabla 19: Requisito funcional 19 – Ver perfil de otro usuario

Ampliar foto
Justificación: El usuario quiere ver una foto de un perfil más de cerca.
Precondiciones: Estar loggeado en la aplicación.
Datos de entrada: foto a ampliar.
Descripción: El usuario puede pulsar sobre una foto, ya sea en su perfil personal, el perfil de otro usuario o el perfil de un evento para verla más de cerca.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se abre una página modal con la imagen ampliada.

Tabla 20: Requisito funcional 20 – Ampliar foto

Navegar entre fotos
Justificación: El usuario quiere poder navegar entre las fotos ampliadas.
Precondiciones: Estar en un perfil y aumentar una foto.
Datos de entrada: Fotos del perfil.
Descripción: Desde una foto ampliada en un perfil, el usuario podrá navegar entre las fotos sin salir de la página modal.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se muestra la siguiente o anterior foto en la página.

Tabla 21: Requisito funcional 21 – Navegar entre fotos

Ver lista de chats
Justificación: El usuario quiere ver los chats que tiene abiertos para poder entrar a ellos.
Precondiciones: El usuario debe estar loggeado y tener al menos un destino y chat.
Datos de entrada: Ninguno.
Descripción: El usuario accede a la página de chats.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se muestra por pantalla una lista de los chats que la persona tiene abiertos.

Tabla 22: Requisito funcional 22 – Ver lista de chats

Iniciar chat
Justificación: El usuario quiere entrar en contacto con otro usuario.
Precondiciones: El usuario debe estar loggeado y tener al menos un destino guardado.
Datos de entrada: Usuario personal y el usuario objetivo.
Descripción: Desde el perfil del usuario objetivo, el usuario pulsará en el botón de iniciar chat.
Pruebas de Aceptación: Que el chat entre ambos usuarios no exista ya.
Postcondiciones: Se creará un chat en la BD y se redireccionará al usuario a la pantalla del chat.

Tabla 23: Requisito funcional 23 – Iniciar chat

Filtrar chats
Justificación: El usuario quiere ver los chats que tiene en cierto destino.
Precondiciones: El usuario debe estar loggeado y tener al menos un destino guardado.
Datos de entrada: Id del destino.
Descripción: El usuario selecciona un destino del desplegable.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se muestra por pantalla una lista de los chats que la persona tiene abiertos para ese destino.

Tabla 24: Requisito funcional 24 – Filtrar chats

Entrar en un chat
Justificación: El usuario quiere entrar en un chat para hablar con la otra persona.
Precondiciones: El usuario debe estar loggeado, tener al menos un destino guardado y un chat abierto.
Datos de entrada: Id del usuario con el que tiene el chat.
Descripción: El usuario pulsa sobre un chat de su lista.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se redirecciona al usuario a la sala de chat con el otro usuario.

Tabla 25: Requisito funcional 25 – Entrar en un chat

Enviar mensaje
Justificación: El usuario quiere enviar un mensaje a otro con el que tiene un chat abierto.
Precondiciones: El usuario debe estar loggeado, tener al menos un destino guardado y un chat abierto.
Datos de entrada: Id del usuario receptor y cuerpo del mensaje.
Descripción: El usuario, desde dentro de un chat, escribe el mensaje y lo envía.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Se envía el mensaje al otro usuario y aparece en la caja de mensajes del chat. Se almacena en un array de mensajes para ese chat.

Tabla 26: Requisito funcional 26 – Enviar mensaje

Recibir mensaje
Justificación: El usuario recibe un mensaje de otro usuario.
Precondiciones: El usuario debe estar loggeado y tener al menos un destino guardado.
Datos de entrada: Cuerpo del mensaje y usuario emisor.
Descripción: Si el usuario receptor ya tiene el chat creado con el usuario emisor, el mensaje se guarda en el array de mensajes del chat en cuestión. Si es el primer mensaje del chat, se creará un chat en la lista de chats con el usuario emisor, se creará un array de mensajes para ese chat y el mensaje se almacenará de la misma manera.
Pruebas de Aceptación: Ninguna.
Postcondiciones: Si se ha creado un nuevo chat, este se mostrará automáticamente en la lista de chats. El mensaje se mostrará automáticamente en la pantalla del chat.

Tabla 27: Requisito funcional 27 – Recibir mensaje

Eliminar chat
Justificación: El usuario quiere dejar de hablar con un usuario.
Precondiciones: El usuario debe estar loggeado, con al menos un destino guardado y un chat abierto.
Datos de entrada: Chat seleccionado.
Descripción: Desde la página del chat objetivo, el usuario pulsará en el botón de eliminar chat.
Pruebas de Aceptación:
Postcondiciones: Se eliminará el chat de la BD y los mensajes del teléfono. Se redireccionará al usuario a la pantalla de lista de chats.

Tabla 28: Requisito funcional 28 – Eliminar chat

4.2 Diagrama de clases

Según Kimmel (2008, p.101), el diagrama de clases constituye la vista más común e importante del diseño. Se les llama estáticos porque no describen acción, sino cosas y sus relaciones. Se diseñan para mostrar todas las piezas de la solución y deben transmitir un sentido del sistema que se estructurará en reposo (Kimmel 2008, p. 101).

La aplicación tiene 5 entidades: Usuario, Localización, Destino, Evento y Chat.

- Las localizaciones son los lugares a los que el usuario puede indicar que va a viajar al crear un destino.
- Un destino está formado por la localización del viaje, el usuario que lo crea y las fechas de inicio y fin de este.
- Un evento tendrá siempre asociada la localización en la que tiene lugar.
- Un chat está formado por los dos usuarios integrantes y la localización del destino por el que se han encontrado.

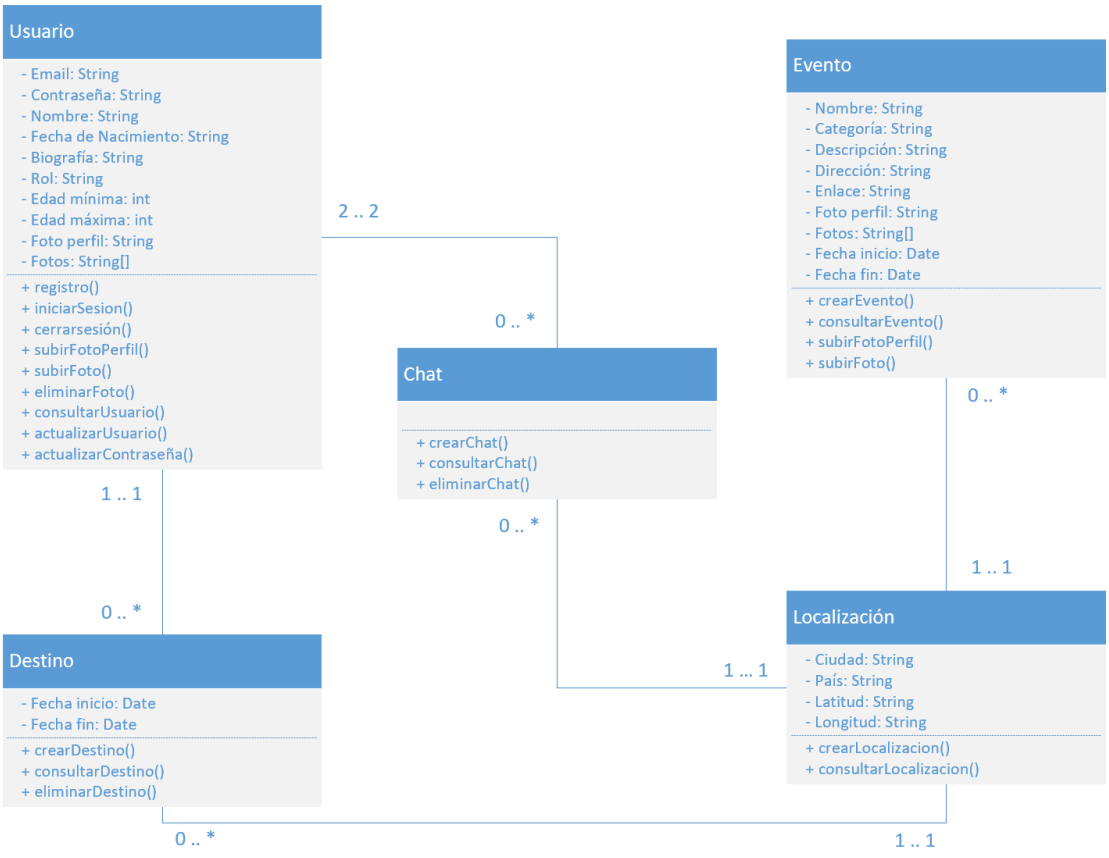

Figura 3: Diagrama de clases de la aplicación

4.3 Casos de uso

4.3.1 Contexto teórico

Según Cevallos (2019), los casos de uso son descripciones de las acciones de un sistema desde el punto de vista del usuario. Los diagramas de caso de uso modelan la funcionalidad del sistema usando actores y casos de uso.

Elementos de los diagramas de casos de uso

Los diagramas básicos de casos de uso constan de sólo unos cuantos símbolos: actores, casos de uso y conectores.

La figura humana representa al actor, es decir, al que lleva a cabo la acción. El símbolo del caso de uso se utiliza para representar las funciones del sistema y los conectores sirven para indicar cómo los actores y casos de uso se relacionan. El estilo de los conectores varía para transmitir información más concreta sobre la relación entre actores y casos de uso. El conector de línea simple se llama asociación y une a los actores con los casos de uso que pueden llevar a cabo. El conector de línea punteada se conoce como dependencia.

Estereotipado de los conectores

Como dice Kimmel (2008, pp. 24-25) los estereotipos agregan detalles a la relación entre los elementos en un diagrama de caso de uso. Una relación de dependencia entre dos casos de uso significa que, de alguna manera, el caso dependiente necesita al caso del que depende. Dos estereotipos que refinan las dependencias en los casos de uso son el incluir y el extender.

Una dependencia rotulada con el estereotipo incluir significa que el caso de uso dependiente necesita la realización del que depende, pero lo opuesto no es cierto. El caso de uso del que se depende es una entidad completa y distinta que no debe depender del caso dependiente (Kimmel 2008, p. 25).

El estereotipo extender se usa para agregar más detalle a una dependencia, lo cual significa que estamos agregando más capacidades, de manera opcional (Kimmel 2008, p. 26).

4.3.2 Casos de uso de la aplicación

Dividiremos los casos de uso por secciones para una mejor visibilidad y entendimiento de los diagramas.

Primero mostramos los casos de uso de un usuario que entra en contacto con la aplicación por primera vez:

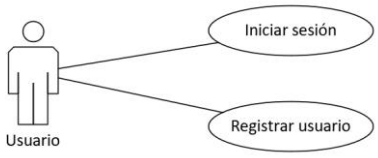

Figura 4: Casos de uso de sesión

Caso de uso: Registrar usuario	
Actor: Usuario	
Curso Normal	**Alternativas**
1) Desde la pantalla de inicio, pulsar en el botón de registro	
2) El usuario introduce los datos de entrada y elige una foto de perfil desde su dispositivo.	
3) El sistema comprueba que los datos son válidos.	**2.1)** Si no lo son se muestra un mensaje de error en pantalla.
4) Se registra al usuario en la base de datos.	
5) Se guarda la imagen en el sistema de archivos del servidor.	
6) Se actualiza el registro de BD del usuario añadiendo la ruta a su foto de perfil.	

Tabla 29: Caso de uso 1 – Registrar usuario

Caso de uso: Iniciar sesión	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario pulsa en iniciar sesión.	
2) El usuario introduce sus datos	
2) El sistema comprueba que los datos son correctos.	**2.1)** Si no lo son se muestra un mensaje de error en pantalla.
3) Se inicia sesión y se muestra la pantalla principal.	

Tabla 30: Caso de uso 2 – Iniciar sesión

En todos los demás casos de uso a continuación, damos por hecho que todo incluyen el caso de uso "Iniciar sesión", puesto que todos son funcionalidades para usuarios registrados en la aplicación.

A continuación, mostramos los casos de uso relacionados con los destinos del usuario y los perfiles de otros usuarios:

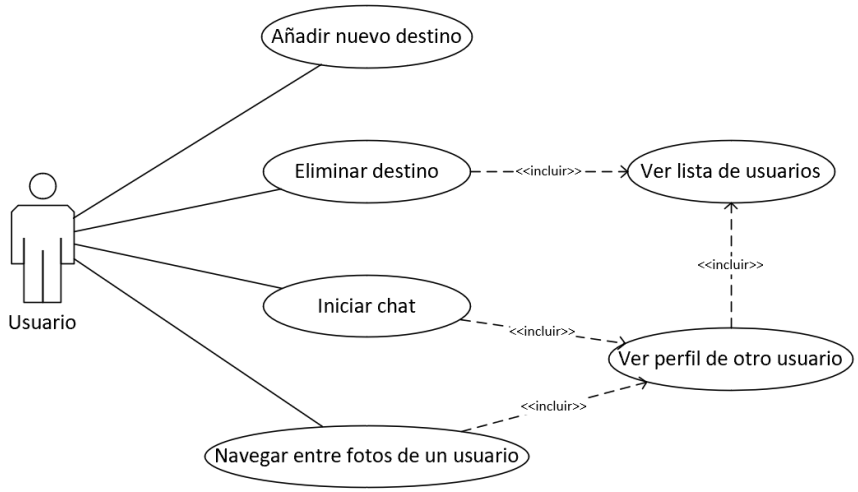

Figura 5: Casos de uso de destinos y usuarios

Caso de uso: Añadir nuevo destino	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario pulsa el botón de añadir destino.	
2) El usuario introduce los datos del destino.	
2) El sistema comprueba que los datos sean válidos.	**2.1)** Si no lo son, se muestra un mensaje de error por pantalla.
3) Se le guarda el destino en la base de datos.	
4) Se redirige al usuario a la página de destinos.	

Tabla 31: Caso de uso 3 – Añadir nuevo destino

Caso de uso: Ver lista de usuarios	
Actor: Usuario	
Curso Normal	**Alternativas**
1) En la página de destinos, el usuario selecciona un destino del desplegable.	
2) El sistema muestra por pantalla las personas con los que el usuario comparte destino.	

Tabla 32: Caso de uso 4 – Ver lista de usuarios

Caso de uso: Ver perfil de otro usuario	
Actor: Usuario	
Curso Normal	**Alternativas**
1) En la página de destinos, el usuario selecciona un destino del desplegable.	
2) El sistema muestra por pantalla las personas con los que el usuario comparte destino.	
3) El usuario pulsa sobre un usuario de la lista.	
4) Se redirige al usuario al perfil de dicho usuario mostrando su información por pantalla.	

Tabla 33: Caso de uso 5 – Ver perfil de otro usuario

Caso de uso: Navegar entre fotos de un usuario	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en el perfil de un usuario.	
2) El usuario pulsa sobre una foto.	
2) Se abre una página modal con la imagen en un tamaño mayor.	
3) El usuario pulsa en la flecha hacia la izquierda o derecha.	
4) Se muestra en la página modal la foto adyacente con tamaño ampliado.	

Tabla 34: Caso de uso 6 – Navegar entre fotos de un usuario

Caso de uso: Iniciar chat	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en el perfil de otro usuario.	
2) El usuario pulsa el botón de iniciar chat.	
3) Se crea un chat en la BD y se redirige al usuario a la página del chat.	

Tabla 35: Caso de uso 7 – Iniciar chat

Caso de uso: Eliminar destino	
Actor: Usuario	
Curso Normal	**Alternativas**
1) En la pantalla principal el usuario selecciona un destino del desplegable	
2) El usuario pulsa el botón de eliminar destino.	
3) El destino se elimina de la base de datos. También se eliminan los chats asociados a ese destino.	**3.1)** Si ocurre algún error en el servidor, se muestra un mensaje de error por pantalla.
4) Se elimina el destino del desplegable y desaparece la información del destino de la pantalla.	

Tabla 36: Caso de uso 8 – Eliminar destino

Seguidamente, mostramos los casos de uso relacionados con los eventos:

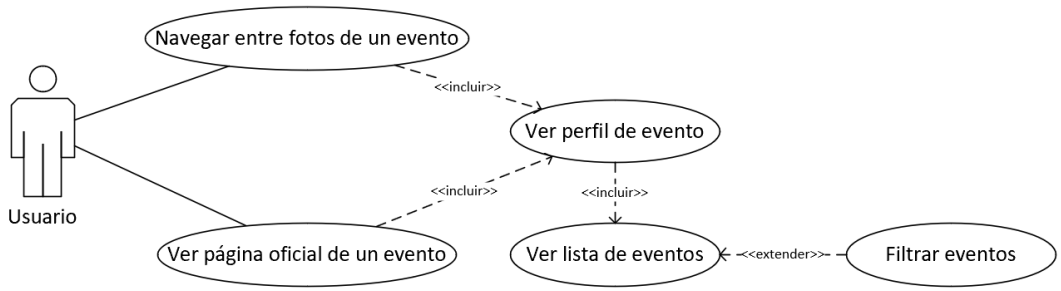

Figura 6: Casos de uso de eventos

Caso de uso: Ver lista de eventos	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en la página de eventos.	
2) El usuario selecciona un destino.	
3) Se muestran por pantalla los eventos que tienen lugar en el lugar y fecha del destino indicado.	

Tabla 37: Caso de uso 9 – Ver lista de eventos

Caso de uso: Ver perfil de evento	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario pulsa sobre un evento.	
2) Se redirige al usuario al perfil del evento mostrando por pantalla la información de dicho evento.	

Tabla 38: Caso de uso 10 – Ver perfil de evento

Caso de uso: Filtrar eventos	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en la página de eventos.	
2) El usuario selecciona un destino.	
3) Se muestran por pantalla los eventos que tienen lugar en el lugar y fecha del destino indicado.	
4) El usuario selecciona una categoría.	
5) Se muestran por pantalla los eventos que tienen lugar en el lugar y fecha del destino indicado que pertenecen a la categoría seleccionada.	

Tabla 39: Caso de uso 11 – Filtrar eventos

Caso de uso: Navegar entre fotos de un evento	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en el perfil de un evento.	
2) El usuario pulsa sobre una foto.	
2) Se abre una página modal con la imagen en un tamaño mayor.	
3) El usuario pulsa en la flecha hacia la izquierda o derecha.	
4) Se muestra en la página modal la foto adyacente con tamaño ampliado.	

Tabla 40: Caso de uso 12 – Navegar entre fotos de un evento

Caso de uso: Ver página oficial de un evento	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario pulsa sobre un evento.	
2) Se redirige al usuario al perfil del evento mostrando por pantalla la información de dicho evento.	
3) El usuario pulsa el enlace a la página oficial del evento.	

Tabla 41: Caso de uso 13 – Ver página oficial de un evento

Ahora los casos de uso relacionados con los chats:

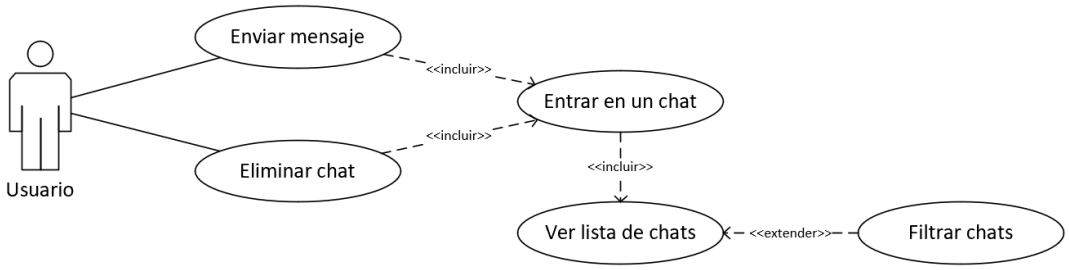

Figura 7: Casos de uso de chats

Caso de uso: Ver lista de chats	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en la página de chats.	
2) Se muestran por pantalla todos los chats que el usuario tenga abiertos.	

Tabla 42: Caso de uso 14 – Ver lista de chats

Caso de uso: Filtrar chats	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en la página de chats.	
2) El usuario selecciona un destino del desplegable.	
3) Se muestran por pantalla los chats abiertos con usuarios con los que comparte dicho destino.	

Tabla 43: Caso de uso 15 – Filtrar chats

Caso de uso: Entrar en un chat	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en la página de chats.	
2) El usuario pulsa sobre un chat de la lista.	
3) Se redirige al usuario a la página de chat donde se muestran los mensajes que ha intercambiado con dicho usuario.	

Tabla 44: Caso de uso 16 – Entrar en un chat

Caso de uso: Enviar mensaje	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en un chat.	
2) El usuario escribe y envía un mensaje.	
3) El mensaje se guarda en un array de mensajes para ese chat. Los mensajes de los chats abiertos se almacenan en el dispositivo.	
4) El mensaje se muestra automáticamente en la página del chat.	

Tabla 45: Caso de uso 17 – Enviar mensaje

Caso de uso: Eliminar chat	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en un chat.	
2) El usuario pulsa el botón de eliminar chat.	
3) Se elimina el chat de la BD y los mensajes del teléfono.	
4) Se redirige al usuario a la página de chats con la lista actualizada.	

Tabla 46: Caso de uso 18 – Eliminar chat

Y por último los casos de uso relacionados con el perfil personal del usuario:

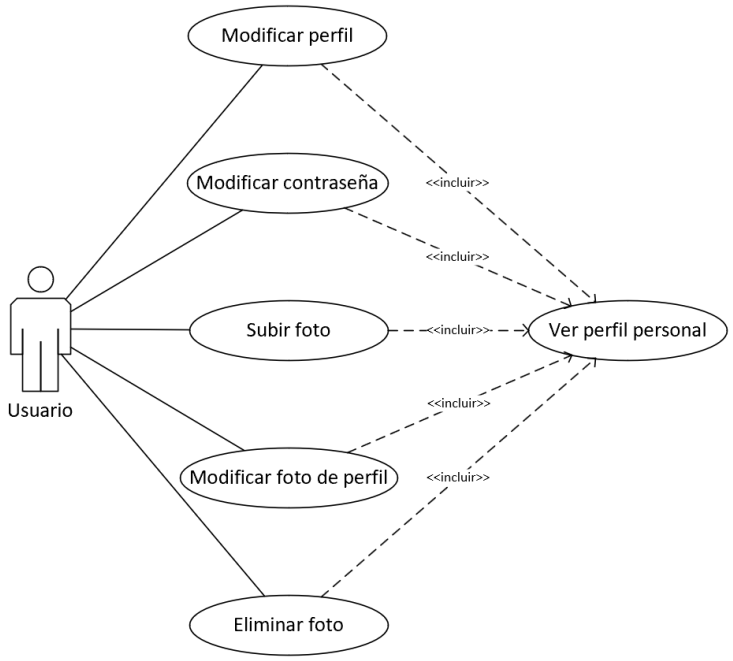

Figura 8: Casos de uso de perfil personal

Caso de uso: Ver perfil personal	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario pulsa en el botón de perfil personal.	
2) El sistema pide a la BD los datos actualizados del usuario.	
3) Se redirige al usuario a la página de su perfil con sus datos en pantalla.	

Tabla 47: Caso de uso 19 – Ver perfil personal

Caso de uso: Subir foto	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entre en su perfil personal.	
2) El usuario pulsa el botón de subir fotos.	**2.1)** Si el usuario ya tiene 6 fotos, se muestra un mensaje de error.
3) Se abre una pantalla modal para subir una foto.	
4) El usuario selecciona una foto de su galería y recorta la foto.	
5) El usuario pulsa en el botón de subir foto.	
6) Se guarda la foto en el sistema de archivos del servidor, se actualiza el registro del usuario de BD añadiendo la ruta a la foto.	**6.1)** Si algo falla en el servidor se muestra un mensaje de error.
7) Se actualiza la vista del perfil personal.	

Tabla 48: Caso de uso 20 – Subir foto

Caso de uso: Modificar perfil	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en su perfil personal.	
2) El usuario entra en la opción de editar perfil.	
3) El usuario modifica los campos que quiera.	
4) El sistema comprueba que los datos introducidos son válidos.	**4.1)** Si no lo son, se muestra un mensaje de error por pantalla.
5) Se modifican los datos en la base de datos y se muestra por pantalla el perfil personal con los nuevos datos.	

Tabla 49: Caso de uso 21 – Modificar perfil

Caso de uso: Modificar contraseña	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en su perfil.	
2) El usuario entra en la opción de editar perfil.	
3) El usuario introduce su contraseña actual y dos veces la nueva.	
4) El sistema comprueba que los datos introducidos son válidos.	**4.1)** Si no lo son, se muestra un mensaje de error por pantalla.
5) Se modifica la contraseña en la base de datos y se redirige al usuario a su perfil personal.	

Tabla 50: Caso de uso 22 – Modificar contraseña

Caso de uso: Eliminar foto	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en su perfil personal.	
2) El usuario pulsa sobre la foto que desea eliminar.	
3) Se pulsa el botón de eliminar foto.	
4) La foto se borra del sistema de archivos del servidor, se actualiza el registro del usuario de BD eliminando la ruta a la foto.	
5) La imagen desaparece automáticamente en la sección de fotos del perfil personal.	

Tabla 51: Caso de uso 23 – Eliminar foto

Caso de uso: Modificar foto de perfil	
Actor: Usuario	
Curso Normal	**Alternativas**
1) El usuario entra en su perfil personal.	
2) El usuario pulsa sobre su foto de perfil.	
3) El usuario pulsa sobre el botón de subir foto.	
4) Se muestra al usuario una pantalla modal para subir una foto.	
5) El usuario selecciona una foto de su galería y recorta la foto.	
6) El usuario pulsa en el botón de subir foto.	
7) Se guarda la foto en el sistema de archivos del servidor, se actualiza el registro del usuario de BD añadiendo la ruta a la foto.	7.1) Si algo falla en el servidor se muestra un mensaje de error.
8) La imagen aparece automáticamente como foto de perfil en el perfil personal.	

Tabla 52: Caso de uso 24 – Modificar foto de perfil

4.4 Diseño de la Interfaz

En primer lugar, se realizaron unos *wireframes* a mano alzada imaginando cómo luciría la aplicación. Posteriormente fueron digitalizados con la herramienta Balsamiq Wireframes.

Con estos bocetos iniciales en mano, se realizó una ronda de contactos con tres personas para comentarlos y recibir *feedback*:

- Todos coincidieron en que el método de navegación principal de la aplicación debería ser mediante tres pestañas en la parte inferior de la pantalla.
- Se eligió el color azul como tono principal de la aplicación.

Posteriormente se hicieron los mockups pasando la aplicación a color y añadiendo la barra de navegación entre páginas.

4.4.1 Wireframes

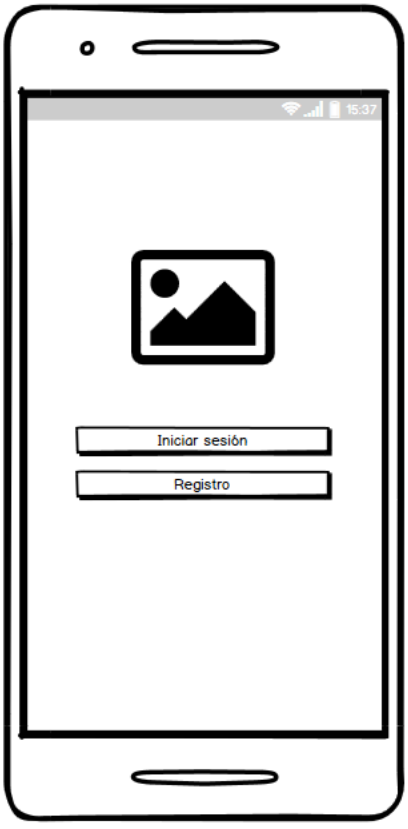

Figura 9: Wireframe de inicio

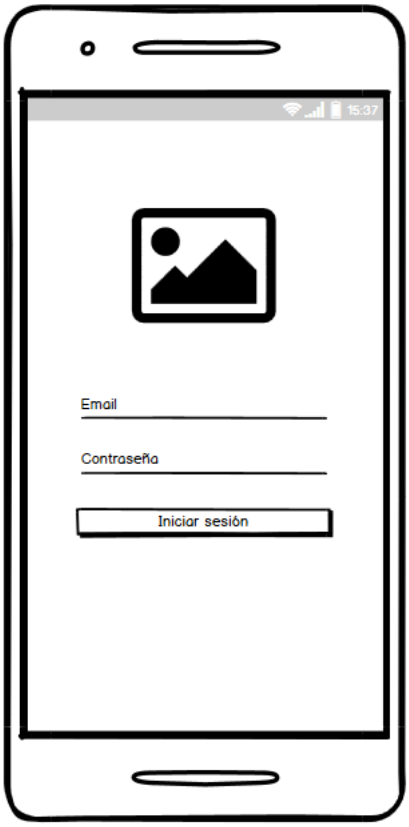

Figura 10: Wireframe de iniciar sesión

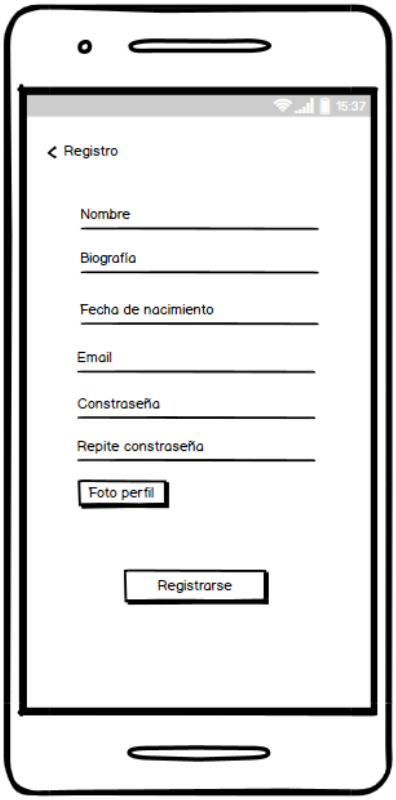

Figura 11: Wireframe de registro

Figura 13: Wifreframe de subir foto perfil

Figura 12: Wireframe de mis destinos

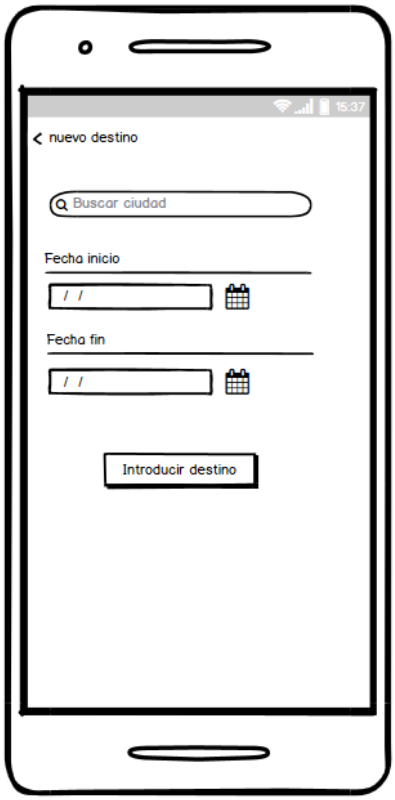

Figura 14: Wireframe de nuevo destino

Figura 15: Wireframe de eventos

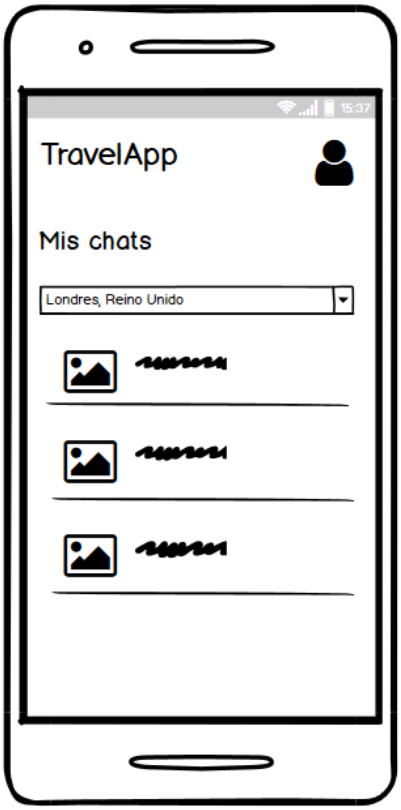

Figura 17: Wireframe de mis chats

Figura 16: Wireframe de perfil de usuario

Figura 18: Wireframe de perfil de evento

Figura 19: Wireframe de perfil personal

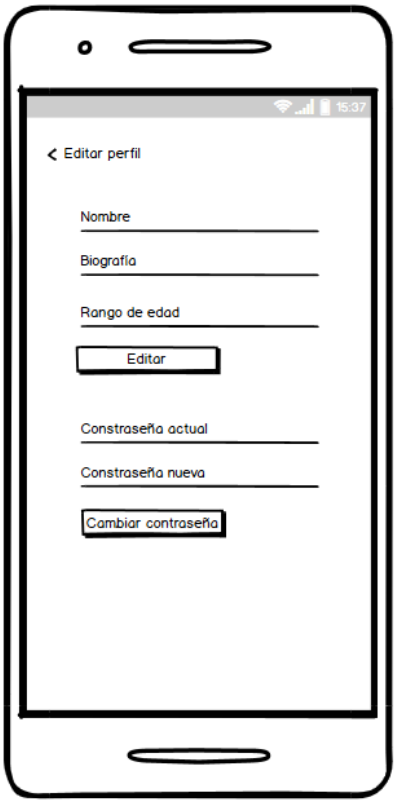

Figura 20: Wireframe de modificar perfil

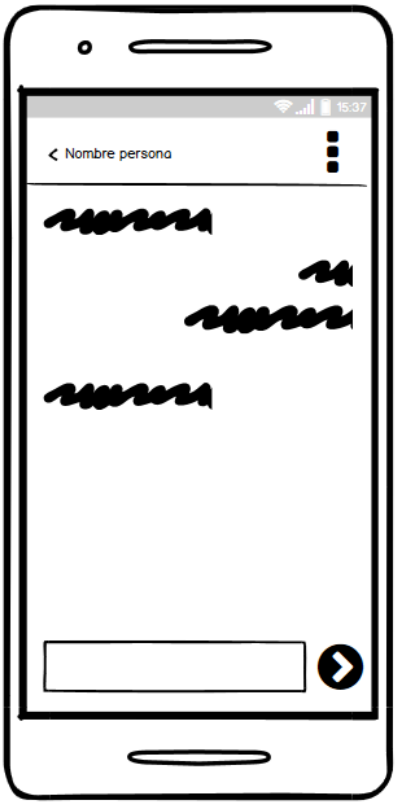

Figura 21: Wireframe de chat

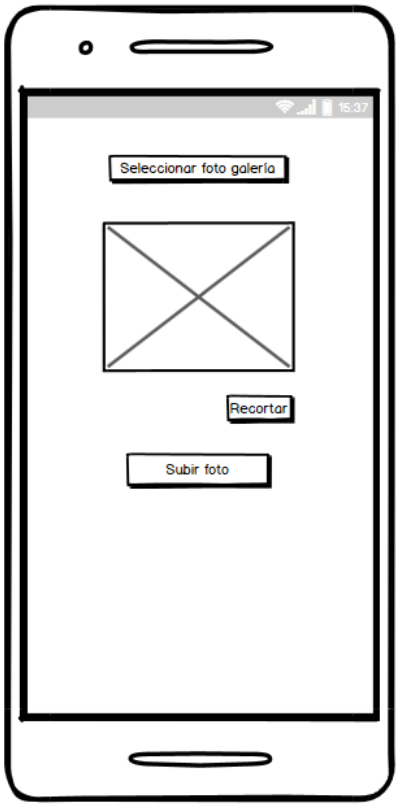

Figura 22: Wireframe de subir foto

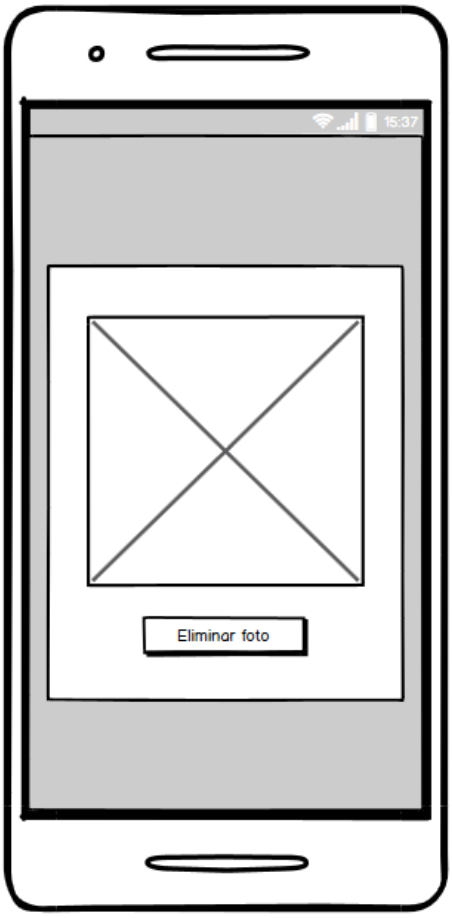

Figura 23: Wireframe de eliminar foto

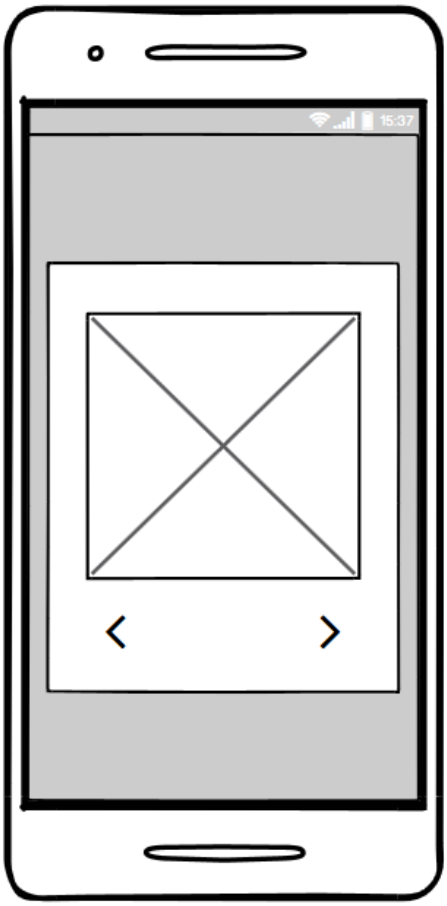

Figura 24: Wireframe de navegar entre fotos

4.4.2 Mockups

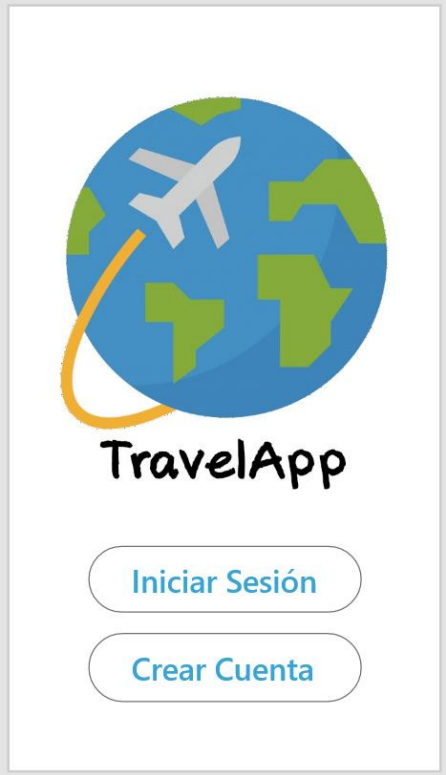

Figura 25: Mockup de inicio

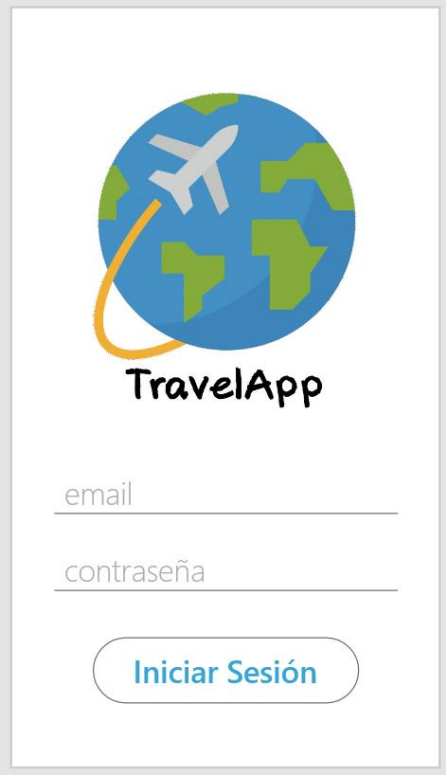

Figura 26: Mockup de iniciar sesión

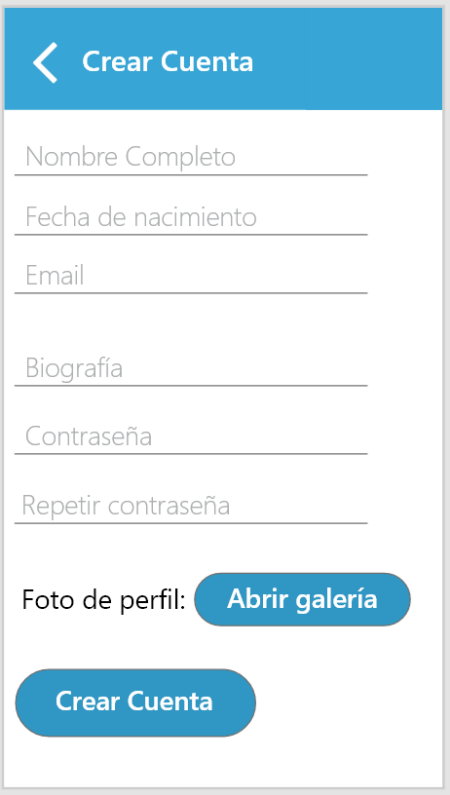

Figura 27: Mockup de registro

Figura 28: Mockup de subir foto perfil

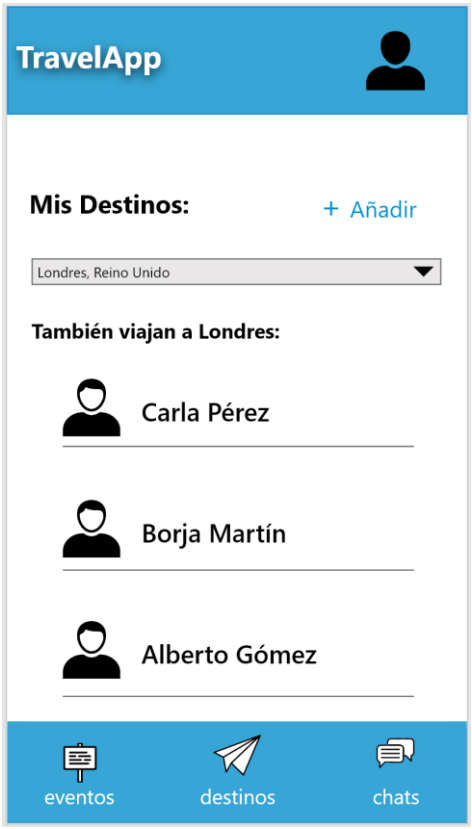

Figura 29: Mockup de mis destinos

Figura 30: Mockup de nuevo destino

Figura 31: Mockup de eventos

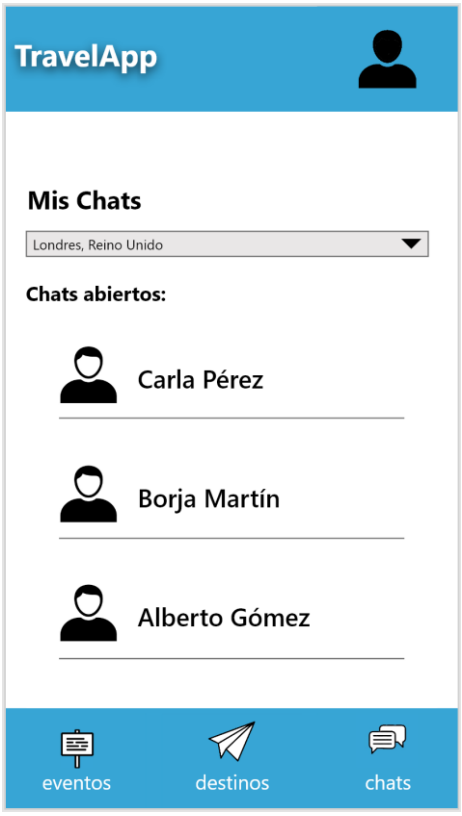

Figura 32: Mockup de mis chats

Figura 33: Mockup de perfil de evento

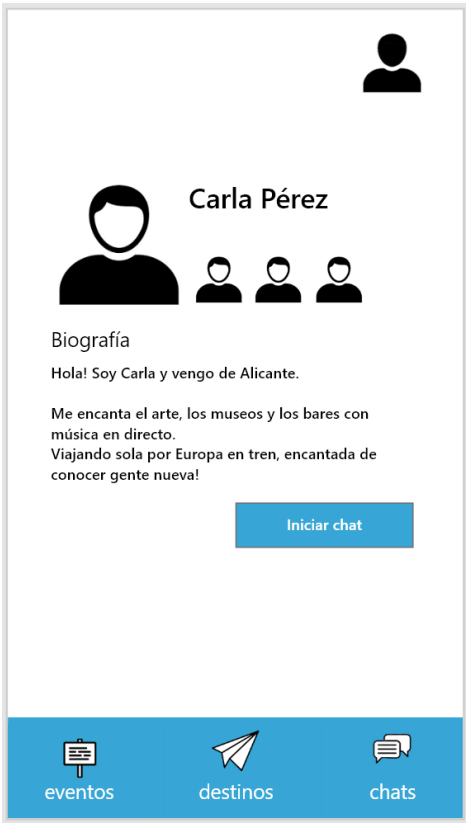

Figura 35: Mockup de perfil de usuario

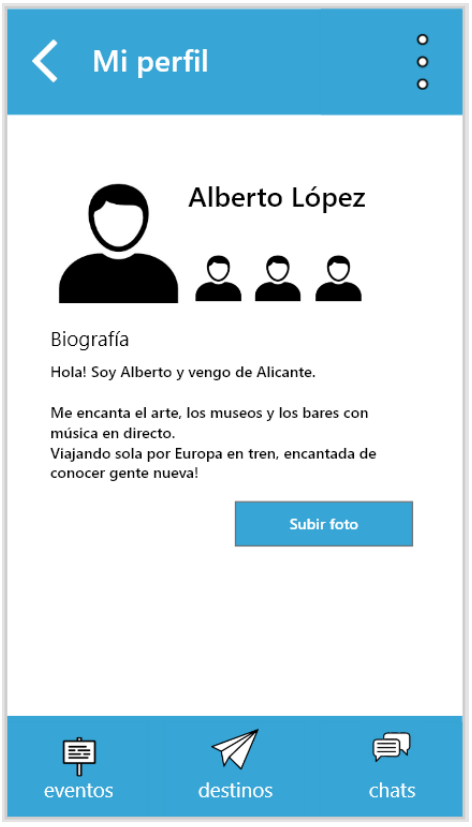

Figura 34: Mockup de perfil personal

Figura 36: Mockup de modificar perfil

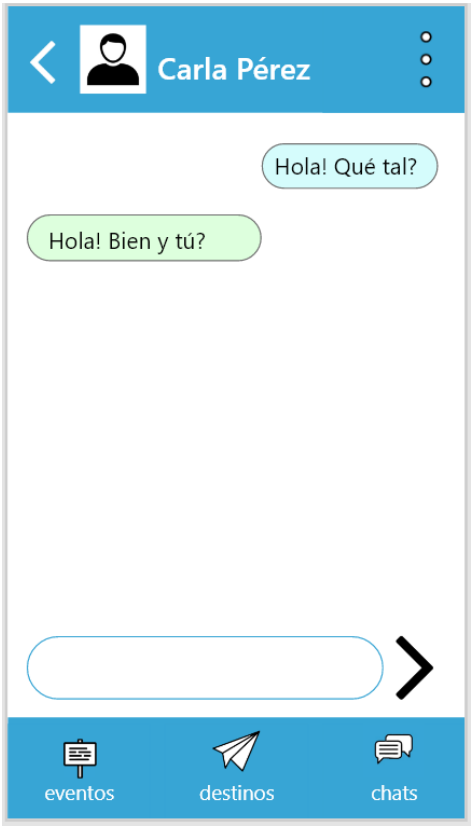

Figura 37: Mockup de chat

Figura 38: Mockup de subir foto

Figura 39: Mockup de navegar entre fotos

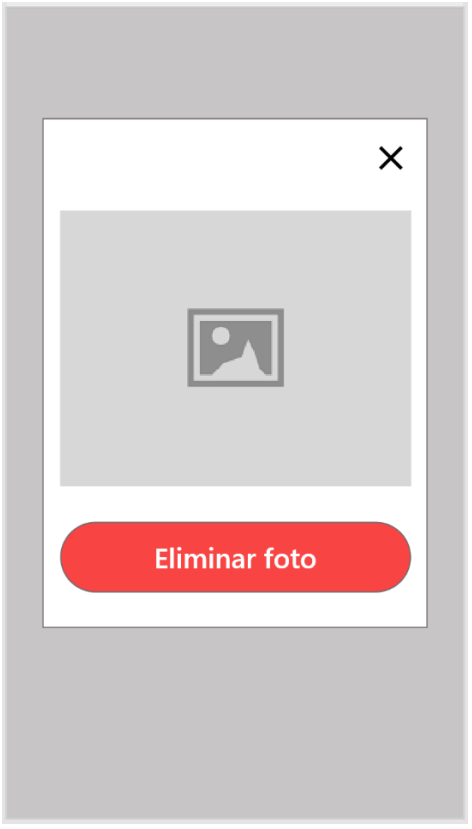

Figura 40: Mockup de eliminar foto

5. Metodología de desarrollo

Como explica Sommerville (2005, pp. 60-61), los modelos del software son representaciones abstractas de procesos del software, abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo del software.

5.1 Modelos del software

Desarrollo iterativo e incremental

Los cambios son inevitables en todos los proyectos de software grandes. Acorde con Sommerville (2005, p. 66), esto significa que el software no es un proceso único; más bien, las actividades del proceso se repiten regularmente conforme el sistema se rehace en respuesta a peticiones de cambios. La esencia de los procesos iterativos es que la especificación se desarrolla junto con el software (Sommerville 2005, p. 66).

Según Sommerville (2005, p. 66), en la entrega incremental, la especificación, el diseño y la implementación del software se dividen en una serie de incrementos, los cuales se desarrollan por turnos. Los clientes identifican a priori los servicios que proporcionará el sistema, otorgan prioridad a unos servicios sobre otros y se definen incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema (Sommerville 2005, p. 66). Así, los clientes no tienen que esperar hasta que se termine el sistema para sacarle provecho, y también pueden utilizar los incrementos iniciales como prototipos y obtener experiencia (Sommerville 2005, p. 67).

Metodologías ágiles

Como dice Sommerville (2005, p. 362), Los métodos ágiles dependen de un enfoque iterativo para la especificación, desarrollo y entrega del software y principalmente fueron diseñados para apoyar el desarrollo de aplicaciones donde los requisitos del sistema con normalidad cambian rápidamente durante el proceso de desarrollo. Están pensados para entregar el software funcional de forma rápida a los clientes, quienes pueden entonces proponer que se incluyen en iteraciones posteriores del sistema nuevos requerimientos o cambios en los mismos (Sommerville 2005, p. 362).

5.2 Scrum

Scrum es un marco de trabajo en equipo para el desarrollo ágil de soluciones (no solamente software) estableciendo diferentes roles y responsabilidades a los integrantes y presentando una serie de conceptos y utilidades para ayudar con el proceso.

Al ser el desarrollo de este proyecto un proceso individual, el framework ha sido adaptado de manera que se han utilizado los elementos y utilidades que otorga, tales como backlogs, sprints, historias de usuario, los cuales presentaremos a continuación.

Un backlog es una lista del trabajo pendiente para el producto, con lo que primero se especificó un backlog del producto inicial incluyendo las historias de usuario:

1. El usuario podrá tener una cuenta en el sistema.
2. El usuario podrá modificar su perfil.
3. El usuario puede introducir destinos con lugar y fecha.
4. El usuario podrá ver al resto de usuarios que viajan a sus mismos destinos.
5. El usuario podrá filtrar las personas que ve en pantalla según el destino.
6. El usuario podrá ver eventos que ocurren en su destino.
7. El usuario podrá filtrar los eventos que ve en pantalla según destino y categoría.
8. Los eventos tendrán una página o perfil con su información.
9. El usuario podrá acceder a la página oficial del evento desde un enlace en el perfil de este.
10. El usuario podrá acceder a el perfil de un evento pulsando sobre él.
11. El usuario podrá ver el perfil de los usuarios con los que comparte destino.
12. El usuario podrá iniciar un chat con otro usuario desde el perfil de este.
13. El usuario podrá filtrar los chats que ve en pantalla según el destino.
14. El usuario podrá acceder al chat pulsando en él.
15. El usuario podrá enviar mensajes por un chat.
16. El usuario podrá eliminar chats.
17. El usuario podrá eliminar destinos.

Estas historias de usuario fueron refinadas y repartidas en sprints. Antes de iniciar cada sprints, las historias de usuario eran refinadas y divididas en tareas específicas. Este conjunto de tareas formaba el backlog del sprint, como se llama en Scrum. Se especificaron 6 sprints, los cuales finalmente se estructuraron de la siguiente manera:

Sprint 1: Como usuario quiero poder introducir mis próximos destinos en la aplicación y filtrar destinos para buscar gente.

Tareas:

- Crear estructura básica del backend.
- Crear modelos de Usuario, Localización y Destino.
- Crear controlador para Usuario y Localización.
- Crear controlador para Destino.
- Crear estructura básica del frontend.
- Crear página de nuevo destino y mis destinos.
- Crear componente de lista usuarios.

Sprint 2: Como usuario quiero poder buscar y filtrar eventos según mis destinos.

Tareas:

- Crear modelo y controlador de Evento.
- Crear función para filtrar eventos según categoría.
- Crear servicio de eventos.
- Crear página de eventos.
- Crear componente de lista eventos.

Sprint 3: Como usuario quiero poder registrarme en la aplicación e iniciar sesión en ella para acceder a mis datos guardados.

Tareas:

- Implementar creación de token JWT.
- Implementar registro y login en el backend.
- Crear sistema de subida de archivos en el backend.
- Crear vista de registro y login.
- Implementar subida de imágenes en frontend.

Sprint 4: Como usuario quiero poder ver información de los eventos, perfiles de los usuarios y mi perfil personal.

Tareas:

- Crear componente de perfil de usuario.
- Crear vista de perfil usuario y perfil personal.
- Crear función para cerrar sesión.
- Crear vista de perfil evento.
- Crear página modal para ampliar imágenes.

Sprint 5: Como usuario quiero poder modificar mi información y mi contraseña.

Tareas:

- Crear función para modificar información de usuario.
- Crear función para cambiar contraseña.
- Crear función para subir fotos de usuario y evento.
- Crear vista de editar perfil y cambiar contraseña.
- Crear vista de subir fotos.
- Crear función de eliminar fotos.
- Modificar página modal para mostrar opción de cambiar foto de perfil y eliminar fotos.

Sprint 6: Como usuario quiero poder iniciar chats con otros usuarios. Como usuario quiero poder intercambiar mensajes en los chats. Como usuario quiero poder eliminar chats.

Tareas:

- Crear modelo de Chat.
- Crear función para crear chat.
- Crear componente de lista chats.
- Crear página de Chats y Chat.
- Implementar llamadas para registro y login en el servidor XMPP.
- Implementar envío de mensajes.
- Crear función para eliminar destinos y chats.
- Implementar actualización automática de lista de chats cuando se recibe un nuevo mensaje.

Como hemos nombrado en la sección de Tecnologías y Herramientas, se utilizó GitHub como herramienta para la gestión del proyecto. En esta plataforma se pueden crear las llamadas *issues*, las cuales representan tareas pendientes. Por lo tanto, se creó una *issue* por cada tarea especificada para el sprint, pero estas *issues* también son válidas para crear nuevas tareas de otro tipo, como por ejemplo corrección de bugs. En cada *issue* se puede especificar el sprint al que pertenece, y también un *milestone*, que representa la versión del proyecto para la cual la *issue* debe estar implementada.

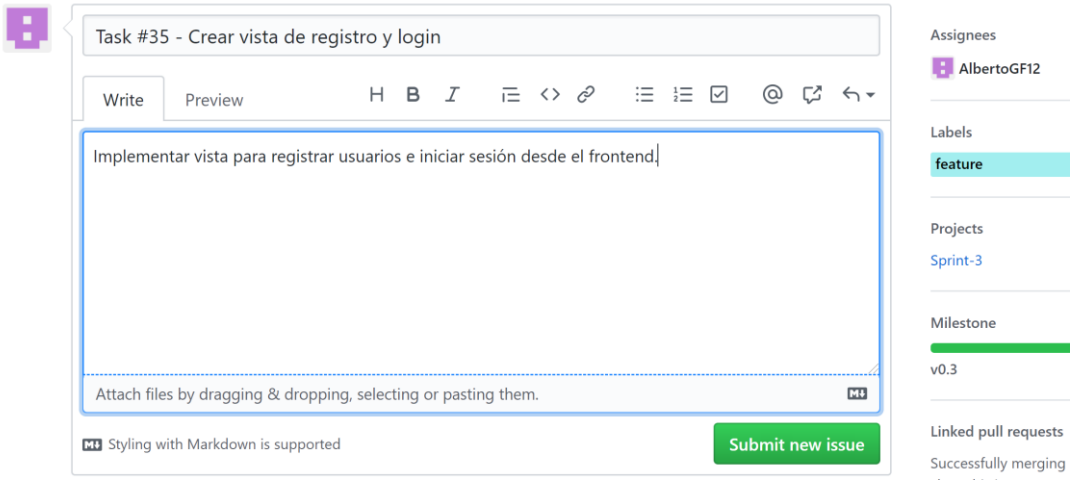

Figura 41: Creación de issue

Por cada sprint se creaba un tablero. Un tablero es una superficie divisible en columnas para plasmar el estado actual del sprint. Cada columna representa un estado y las *issues* van cambiando de una a otra conforme avanza el sprint.

Las diferentes columnas establecidas para los sprints fueron: *To do* (Por hacer), *In Progress* (en desarrollo), *Testing* y *Done* (Terminado). En la columna *To do* se almacenaban las *issues* pendientes de empezar, en *In Progress* las que habían comenzado a desarrollarse, en *Testing* las que estaban siendo sometidas a pruebas y en *Done* las que habían sido dadas por terminadas.

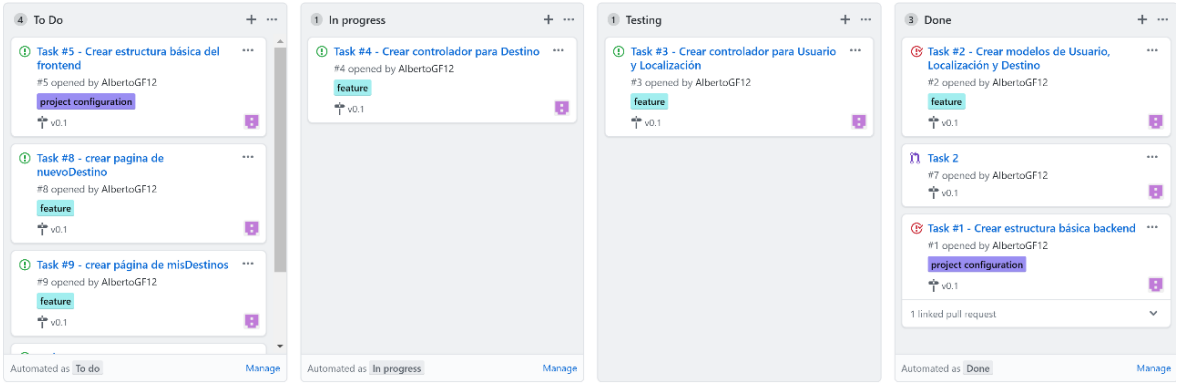

Figura 42: Tablero de un sprint de Scrum

5.3 Git Flow

El flujo de trabajo de Git Flow es un diseño de flujo de trabajo de Git que fue publicado por primera vez y popularizado por Vincent Driessen. El flujo de trabajo de Git Flow define un modelo estricto de ramificación diseñado alrededor de la publicación del proyecto. Proporciona un marco sólido para gestionar proyectos más grandes y es ideal para los proyectos que tienen un ciclo de publicación programado. En realidad, Git Flow es una especie de idea abstracta de un flujo de trabajo de Git. Esto quiere decir que ordena qué tipo de ramas se deben configurar y cómo fusionarlas (Altassian, 2020).

Tendremos dos ramas principales, "*Master"* y "*Develop"*. La rama Master servirá para almacenar las diferentes versiones del proyecto. No se desarrollará en esta rama y solo se mergeará código en ella cuando esté listo para ser etiquetado como una versión funcional del proyecto.

La rama *Develop* será la rama base para el desarrollo del código. De esta rama saldrán las ramas de *features* (funcionalidades) y las ramas de *release* (lanzamiento).

Las ramas de funcionalidades son en las que se desarrollará todo el código. Crearemos una rama por cada tarea del sprint y se mergeará a *Develop* cuando su desarrollo haya terminado. La nomenclatura de las ramas de *release* es libre, evitando nombres clave como *master*, *develop*, *release* o *hotfix* (término explicado a continuación). En mi caso la nomenclatura ha mantenido relación con el formato que he utilizado con las *issues*. Si la *issue* se tiene el nombre: "Task#10 - *Título de la tarea*", su respectiva rama se denomina "task-10".

Las ramas de *release* son aquellas que se crean cuando el desarrollo de cierta versión del producto ha finalizado (por ejemplo, al terminar un sprint). En estas ramas no se implementan más tareas, solo se corrigen errores y se prepara la versión para su lanzamiento.

La nomenclatura que siguen estas ramas es "release-" seguido de la versión del producto (p.e.: release-v0.2).

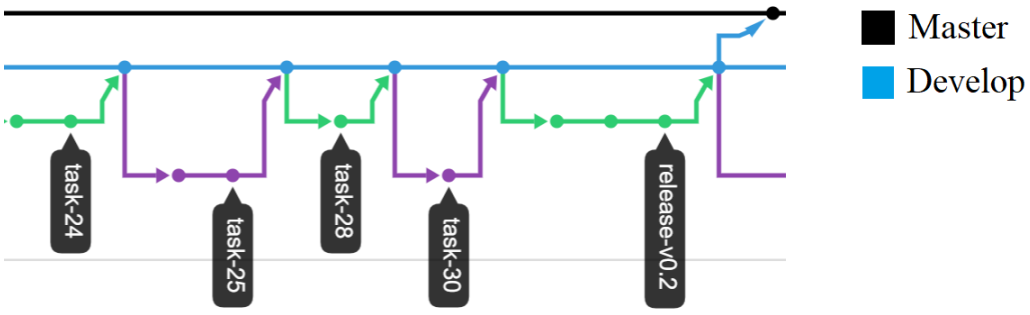

Figura 43: Ejemplo de Git Flow

Por último, las ramas denominadas “de *hotfix*”, son ramas específicas para corregir errores de código ya desplegado en producción. Estas ramas salen de la rama máster, en ellas se realizan los cambios pertinentes en el código para solucionar el problema.

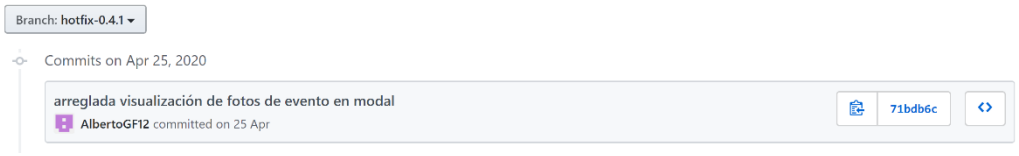

Figura 44: Commit con error solucionado en un hotfix

La rama se mergea de vuelta a la rama *Master* y a la rama *Develop* para que la nueva versión en desarrollo se lance con el fallo ya corregido.

Figura 45: Merge de rama hotfix

En la rama *Master* se crea una etiqueta para marcar la nueva versión del proyecto con el error corregido.

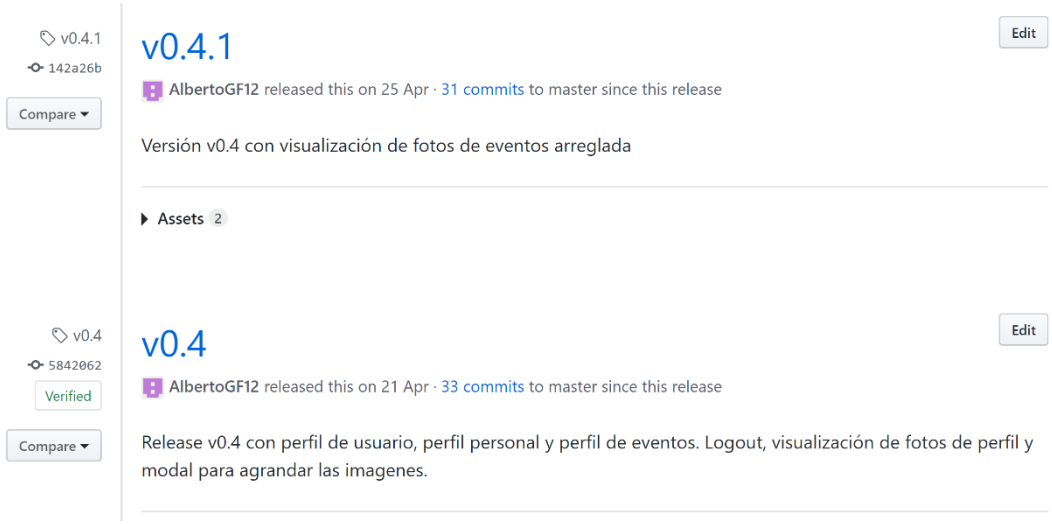

Figura 46: Creación de nueva versión tras finalizar un hotfix

5.4 Entrega continua

Como se ha explicado en la introducción de la sección, las metodologías iterativas cuentan con una entrega continua de incrementos del producto.

Cada vez que un sprint terminaba y el código de la versión era enviado a la rama *Master*, la aplicación era desplegada en producción. El código del API se subía y se desplegaba en el servidor, mientras que el código del cliente era construido para generar la aplicación instalable en dispositivos móviles (véase Anexo 1).

Haciendo alusión a las ventajas principales de este tipo de metodologías, los usuarios utilizaban estas versiones incrementales de la aplicación para testear el funcionamiento e interfaz de esta.

Surgieron algunos cambios en la interfaz durante el desarrollo del producto, siendo el más grande la disposición de las fotos de los usuarios dentro de su perfil:

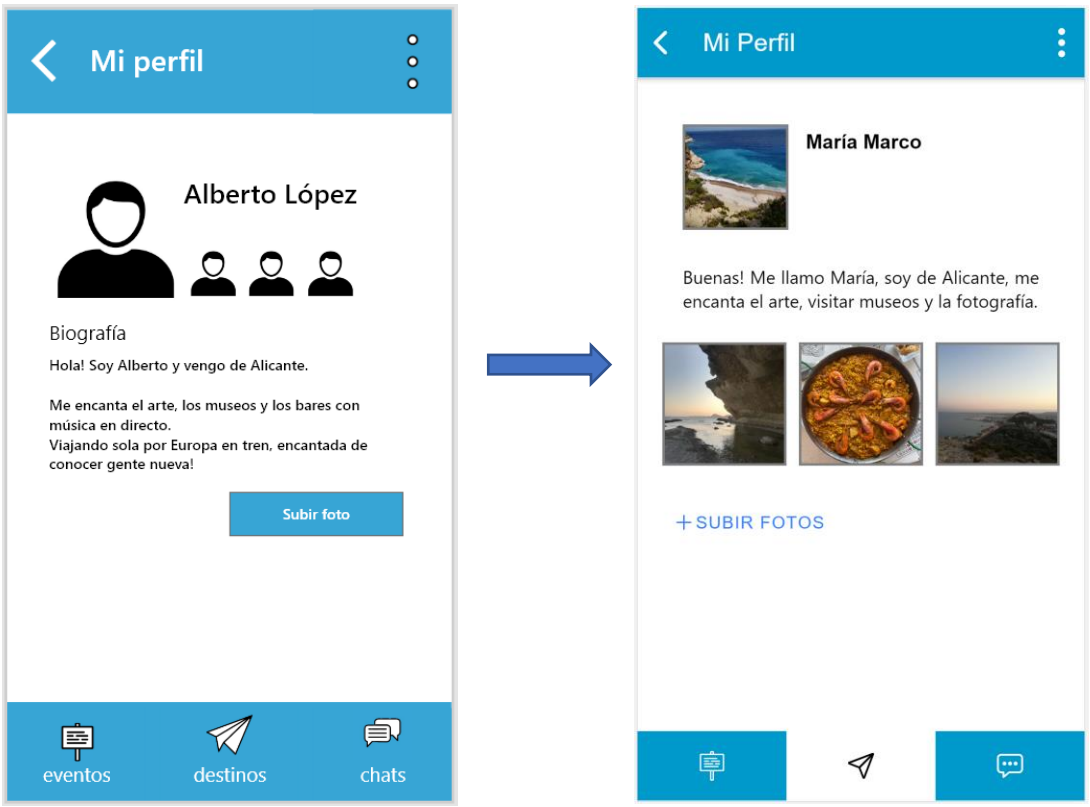

Figura 47: Modificación de la interfaz del perfil personal a mitad del desarrollo

Y otros cambios más pequeños relacionados con el tamaño de los elementos o con texto:

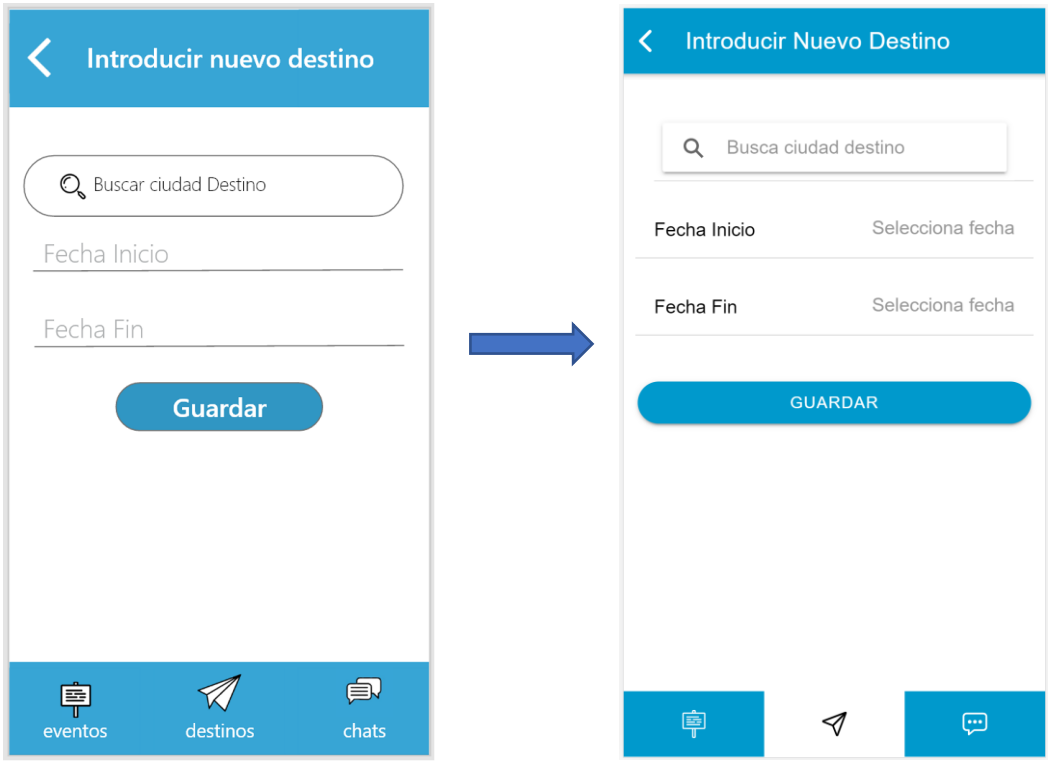

Figura 48: Modificación de la interfaz de nuevo destino a mitad del desarrollo

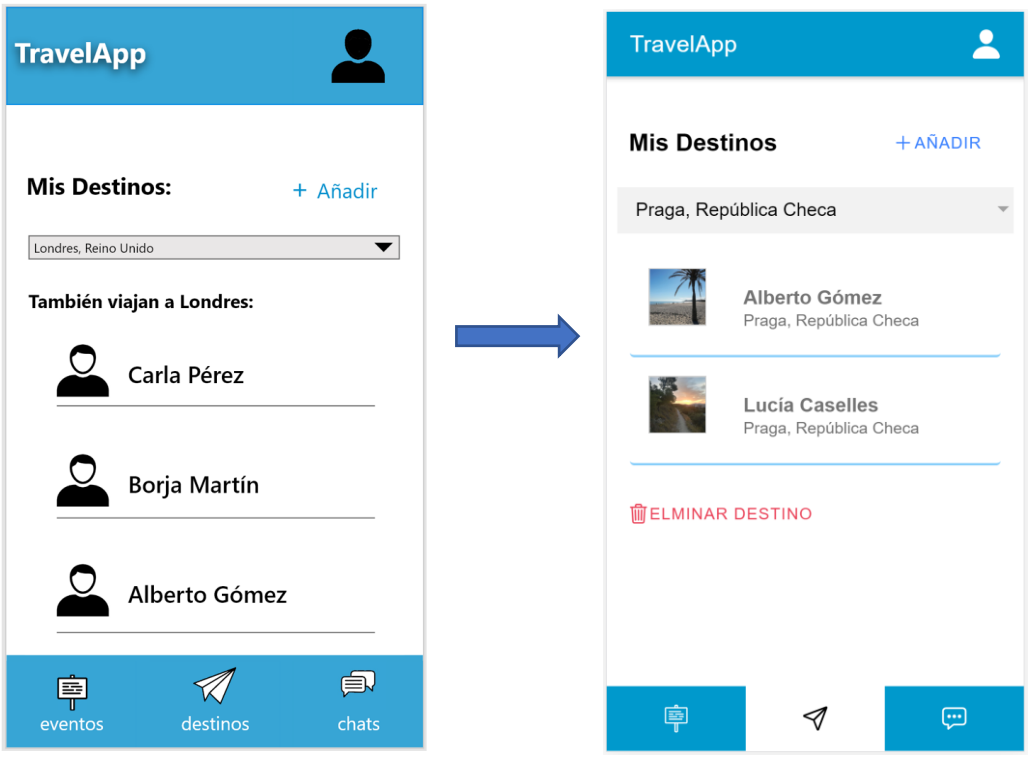

Figura 49: Modificación de la interfaz de mis destinos a mitad del desarrollo

6. Testing

En este apartado mostraremos las pruebas realizadas sobre la API de la aplicación. Para realizar estas pruebas se ha estado utilizando la herramienta Postman, presentada en el apartado de Tecnologías y Herramientas de este mismo documento.

Esta herramienta nos permite realizar llamadas a la dirección que queramos con los distintos métodos HTTP disponibles para probar todos los recursos que la API ofrece.

Conforme avanzaba el desarrollo y los sprints, se fue incrementando el catálogo de recursos disponibles en la API, y, a su vez, la cantidad de pruebas en Postman, las cuales se iban almacenando en una misma colección.

Figura 50: Conjunto de pruebas realizadas sobre la API

Estas pruebas pueden catalogarse como pruebas funcionales, las cuales comprueban que el código que se ejecuta tiene el comportamiento esperado.

Cada vez que se realizaba un incremento en la aplicación, se añadían las pruebas necesarias para probar el nuevo código, integrándose con el anterior, por lo que se volvían a ejecutar todos los tests para comprobar que el comportamiento del código anterior no había sido alterado por los nuevos incrementos. Este tipo de pruebas son las catalogadas como pruebas de regresión.

A continuación, se mostrarán los distintos tipos de respuesta que se esperan del API.

Cuando se accede a un recurso válido, se devolverá un código 200 con el cuerpo de la respuesta correspondiente. Por ejemplo, cuando consultamos la información de un usuario:

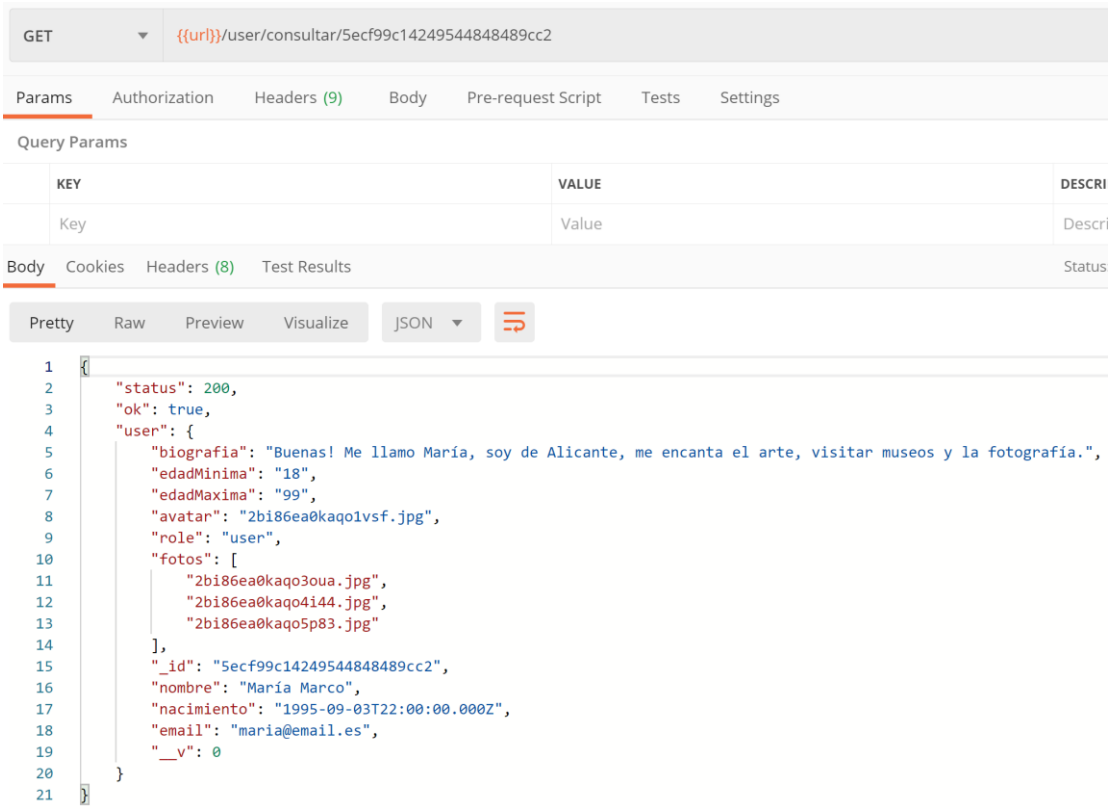

Figura 51: Respuesta de la API con código 200

Sin embargo, si intentamos acceder a un recurso no disponible, como a la información de un usuario inexistente, se devuelve un 404:

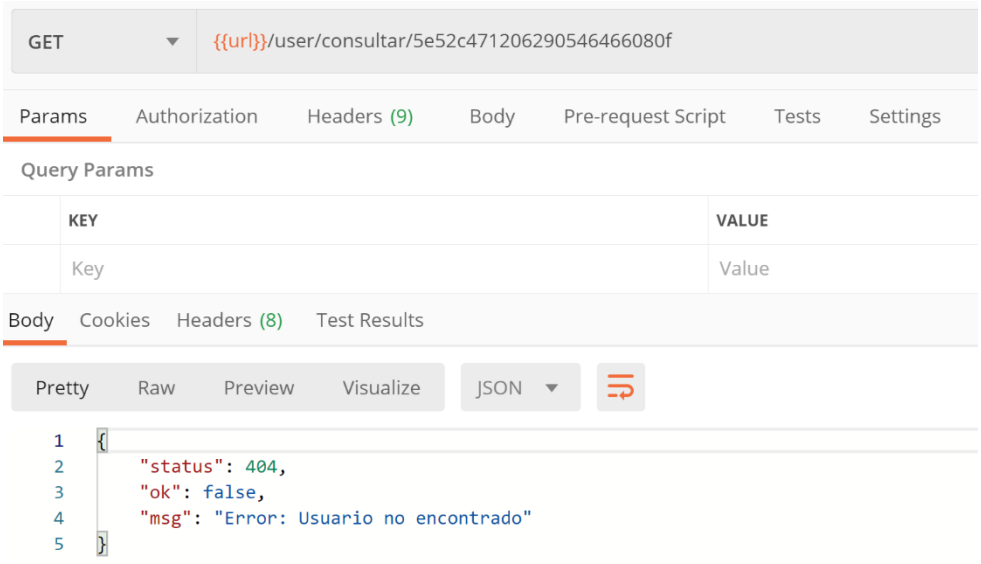

Figura 52: Respuesta de la API con código 404

En caso de que accedamos a un recurso POST, como crear un nuevo destino para cierto usuario y fechas, se devolverá un estado 201:

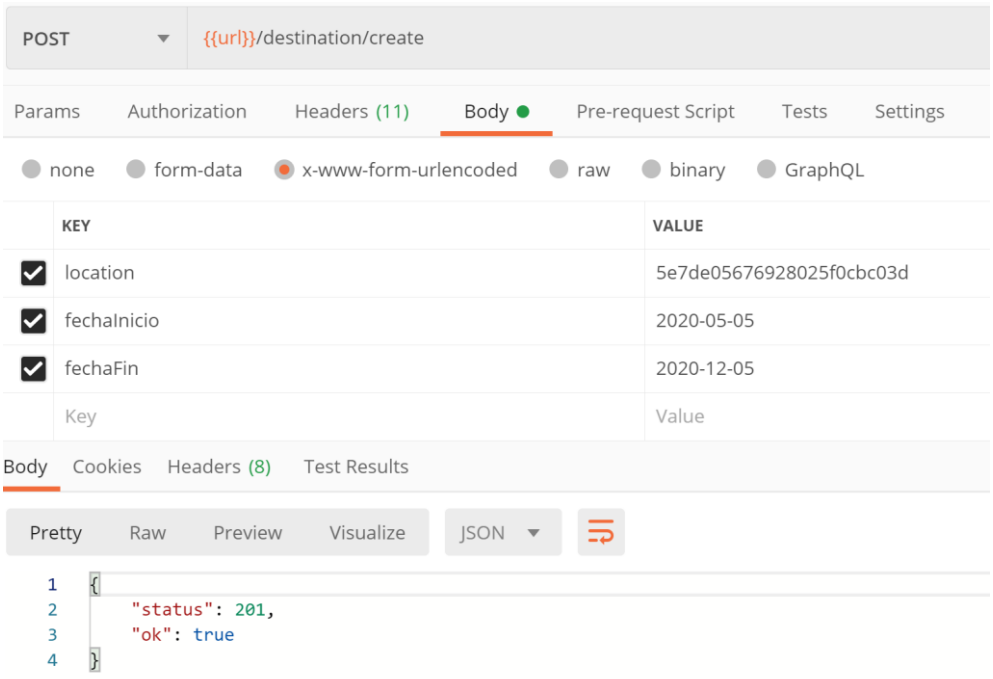

Figura 53: Respuesta de la API con código 201

Sin embargo, cuando se intenta acceder al mismo recurso sin el token de autorización se devuelve un estado 401:

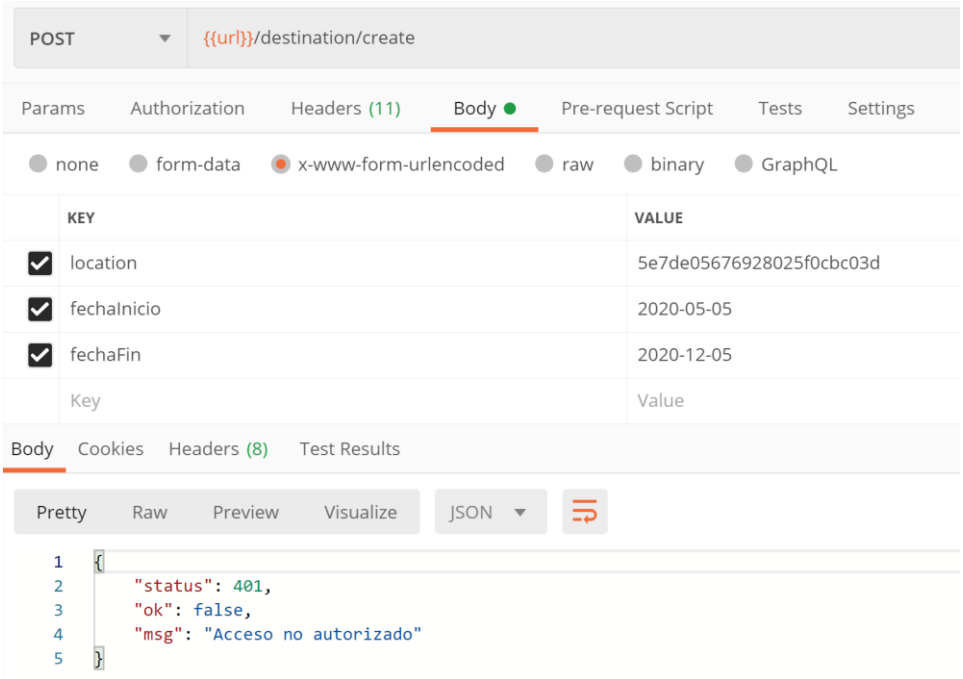

Figura 54: Respuesta de la API con código 401

Para testear la aplicación cliente Ionic cuenta con el comando *Ionic serve*, el cual ejecuta la aplicación en un entorno de depuración utilizando el navegador web en el que se puede utilizar la aplicación utilizando el ratón para sustituir el dedo.

De este modo se puede testear cada nueva funcionalidad, el diseño o la interacción entre los elementos gráficamente. Además, puede elegirse el dispositivo que queremos emular (entre unas cuantas opciones) y su orientación.

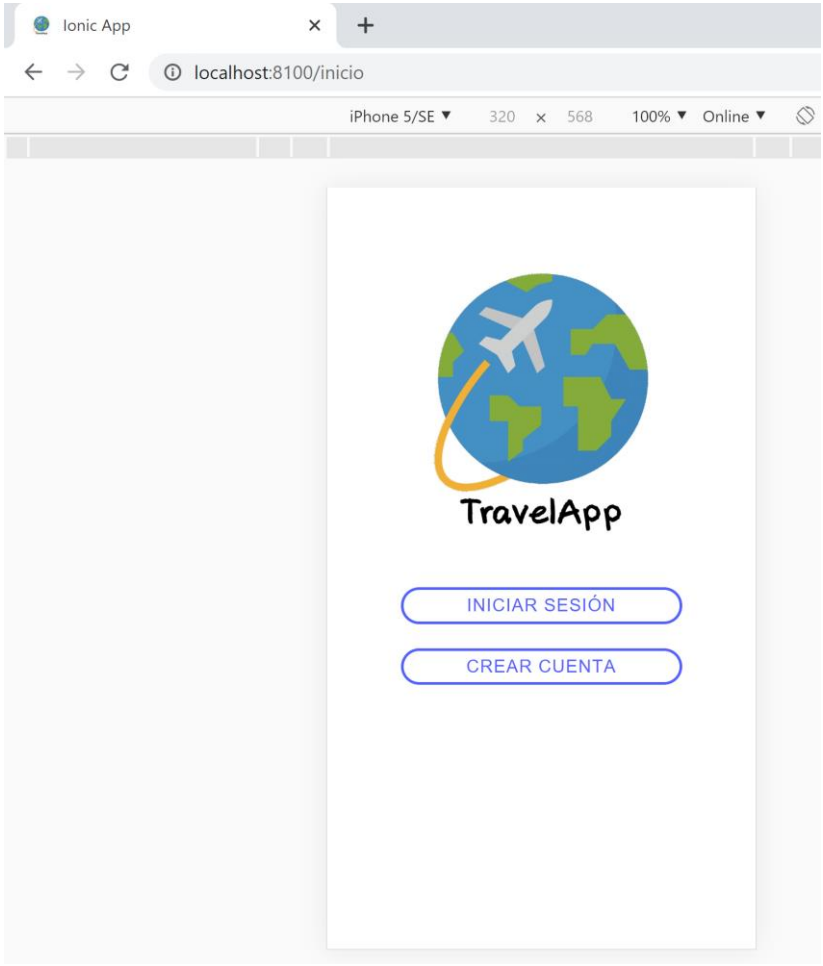

Figura 55: Depuración de aplicación con Ionic

Gracias a las opciones de desarrollador de los dispositivos móviles, también puede ejecutarse la aplicación dentro de un dispositivo móvil real conectado al ordenador vía USB. Con el comando *ionic cordova run android --device* (para dispositivos Android) se crea una *build* de la aplicación en su estado actual que se instala en el dispositivo conectado para poder probarla en un entorno real durante el proceso de desarrollo.

7. Conclusiones y líneas futuras

En este apartado estudiaremos cómo las decisiones tomadas respecto el desarrollo y despliegue del proyecto han sido determinantes para el cumplimiento de los objetivos principales del mismo.

Estos objetivos eran ofrecer una solución óptima a un problema real, crear un producto accesible para el mayor número de personas posibles y que este fuera incrementable y escalable a lo largo del tiempo.

La selección de tecnologías móviles es indispensable para asegurar la accesibilidad del producto en un mundo en el que los dispositivos móviles son la orden del día, dejando a un claro segundo plano las aplicaciones de escritorio.

El uso en específico de una tecnología híbrida facilita una mayor accesibilidad y que los futuros incrementos de la aplicación lleguen lo antes posible a los usuarios. Esto es debido a que se utiliza un conjunto de tecnologías específicas independientemente de la plataforma en la que vaya a ejecutar la aplicación, con lo que el proceso de desarrollo es único.

La utilización de computación en la nube para el despliegue del servidor es una decisión que facilita la sostenibilidad, mantenimiento y escalabilidad de los recursos. Estas compañías, como AWS, ofrecen amplias posibilidades de incremento en las prestaciones de hardware y de red para la infraestructura, incluso de manera dinámica, según las necesidades de la solución desplegada.

Como líneas futuras se plantean posibles ampliaciones como:

- Soporte a la creación de eventos desde la aplicación, pasando la responsabilidad a los propios usuarios o a posibles perfiles para empresas u organizaciones, con vistas personalizadas para ellas.
- Ampliación de las funciones de chat, como creación de conversaciones grupales o envío de contenido multimedia.
- Posibilidad de compartir enlaces a eventos a través del propio chat de la aplicación o mediante aplicaciones de mensajería externas.
- Implementación de un sistema de financiación mediante la promoción de eventos.

8. Referencias bibliográficas

The Apache Software Foundation. (2020). Apache Cordova. Obtenido de https://cordova.apache.org/docs/es/latest/guide/overview/

Ionic. (2020). ¿Qué es Apache Cordova? Obtenido de Ionic Framework: https://ionicframework.com/resources/articles/what-is-apache-cordova

OpenJS Foundation. (2020). Acerca de Node.js. Obtenido de NodeJS: https://nodejs.org/es/about/

MongoDB. (2020). ¿Qué es MongoDB? Obtenido de MongoDB: https://www.mongodb.com/es/what-is-mongodb

Amazon Web Services. (2020a). ¿Qué es la informática en la nube? Obtenido de https://aws.amazon.com/es/what-is-cloud-computing/.

Amazon Web Services. (2020b). ¿Qué es Amazon EC2? Obtenido de: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html

Atlassian. (2020). Flujo de trabajo de Gitflow. Obtenido de Atlassian: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow

Cevallos, K. (2015). Ingeniería del Software. Obtenido de https://ingsotfwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/

Pérez Porto, J., & Gardey, A. (2018). Definición de Cliente Servidor. Obtenido de Definicion.de: https://definicion.de/cliente-servidor/

Kimmel, P. (2008). Manual de UML. México D.F.: McGraw-Hill Interamericana Editores.

Sommerville, I. (2005). Ingeniería del Software 7ma. Ed. Madrid: Pearson Educación S.A.

Anexo 1: Despliegue en producción

Para la puesta en marcha del servidor, primero se creó una instancia en EC2. Una máquina virtual altamente configurable. A la hora de la creación se puede elegir la capacidad del procesador, disco duro, memoria RAM y capacidad de red, entre otros.

AWS permite construir nubes virtuales privadas (VPC) en las que poder configurar todos los vectores de entrada, enrutamiento, cortafuegos, DNS propio, etc. Para el propósito de este proyecto se ha utilizado una VPC por defecto que AWS permite utilizar, la cual proporciona a las máquinas una dirección IP y nombre de dominio públicos.

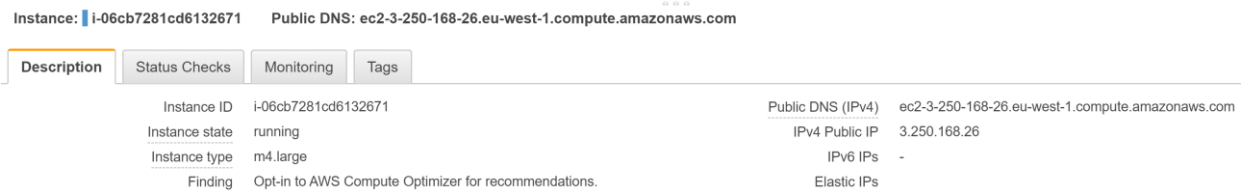

Figura 56: Panel de información de la instancia EC2

También se debe configurar un *Security Group*, un elemento con opciones de firewall que permite escoger qué puertos de la instancia son accesibles desde el exterior y desde qué direcciones IP.

Inbound rules | Outbound rules | Tags

Inbound rules

Type	Protocol	Port range	Source
Custom TCP	TCP	3000	0.0.0.0/0
Custom TCP	TCP	3000	::/0
RDP	TCP	3389	0.0.0.0/0
RDP	TCP	3389	::/0
Custom TCP	TCP	7070	0.0.0.0/0
Custom TCP	TCP	7070	::/0

Figura 57: Reglas de entrada en un Service Group

En el caso de TravelApp se abrieron los puertos 3000 y 7070, en los cuales escuchan los servicios del API y del servidor Openfire. Además, el puerto 3389 está abierto para la administración remota.

En la instancia se instalaron los programas necesarios para el despliegue y ejecución de los servicios, haciéndolos escuchar en los puertos indicados.

Para la generación del archivo APK instalable en dispositivos, se siguió la guía oficial[4] de Ionic de su página web. Para generar una aplicación instalable en iOS, es necesario disponer de un ordenador Mac para utilizar ciertas herramientas específicas, por lo que en entorno de producción se ha utilizado únicamente Android.

El proceso es el siguiente:

Antes de nada, debemos tener el SDK de Android instalado. Una vez lo tenemos, en el directorio de nuestro proyecto de Ionic debemos utilizar el siguiente comando para que Cordova cargue los elementos necesarios para poder construir la aplicación en Android:

```
ionic cordova platform add android
```

Seguidamente debemos hacer una *build* del código con el siguiente comando:

```
ionic cordova build android --release --prod
```

Una consideración a tener en cuenta para entender lo siguiente, es que en Ionic existen dos archivos que definen las URL de los entornos de desarrollo y producción.

Figura 58: Diferencia entre archivos de entorno de desarrollo y producción

En el código, el archivo *environment.ts* es siempre referenciado para acceder a la dirección de la máquina a la que queremos llamar. Ejecutar el comando anterior con la opción --prod hace que el código del archivo *environment.prod.ts* sustituya el anterior, consiguiendo así que todas las llamadas del código se dirijan a la máquina de producción correspondiente.

[4] https://ionicframework.com/docs/v1/guide/publishing.html

Con el comando anterior se genera un archivo APK sin firmar (*travelapp-release-unsigned.apk*) que no es instalable en ningún dispositivo. Para firmar la APK es necesaria una clave privada que se consigue con el comando *keytool*, disponible con el JDK de Java.

```
keytool -genkey -v -keystore travelapp-key.keystore -alias alias_travelapp -keyalg RSA -keysize 2048 -validity 10000
```

Seguidamente firmamos el archivo APK con la clave generada utilizando el comando *jarsigner*, también disponible con el JDK:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore travelapp-key.keystore travelapp-release-unsigned.apk alias_travelapp
```

Por último, este archivo APK es optimizado utilizando el comando *zipalign*, dándole también el nombre definitivo al archivo:

```
zipalign -v 4 travelapp-release-unsigned.apk travelapp.apk
```

El archivo generado es perfectamente funcional, instalable en dispositivos Android y se puede publicar en la Play Store.

Anexo 2: Interfaz gráfica final

En esta sección mostraremos capturas del resultado final de la aplicación.

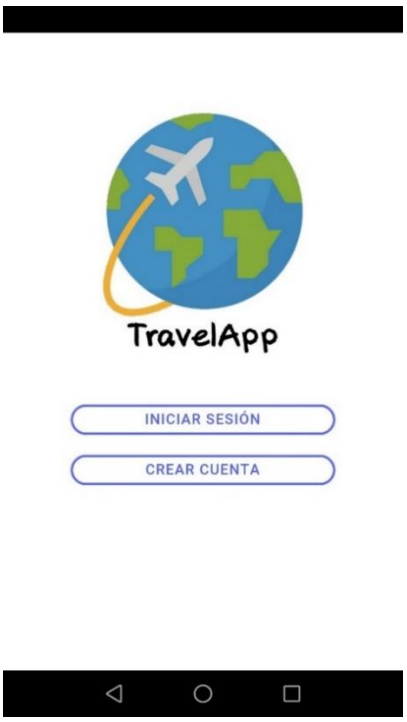

Figura 59: Interfaz final de inicio

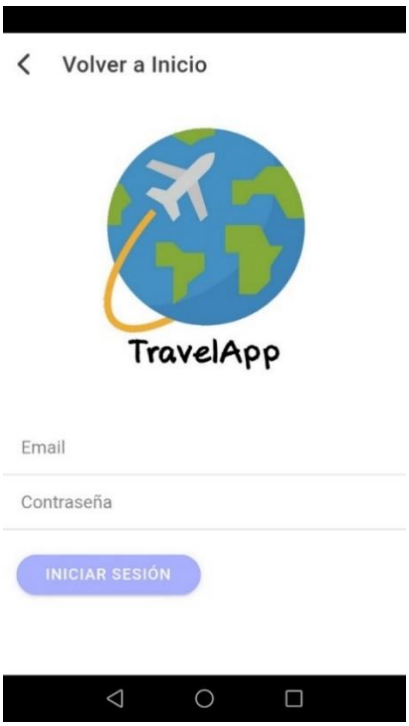

Figura 60: Interfaz final de iniciar sesión

Figura 61: Interfaz final de registro

Figura 62: Interfaz final de recortar foto de perfil

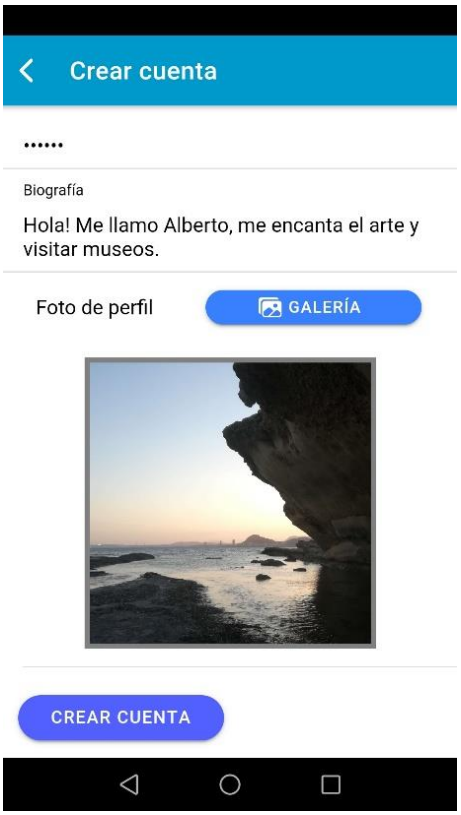

Figura 63: Interfaz final de subir foto de perfil

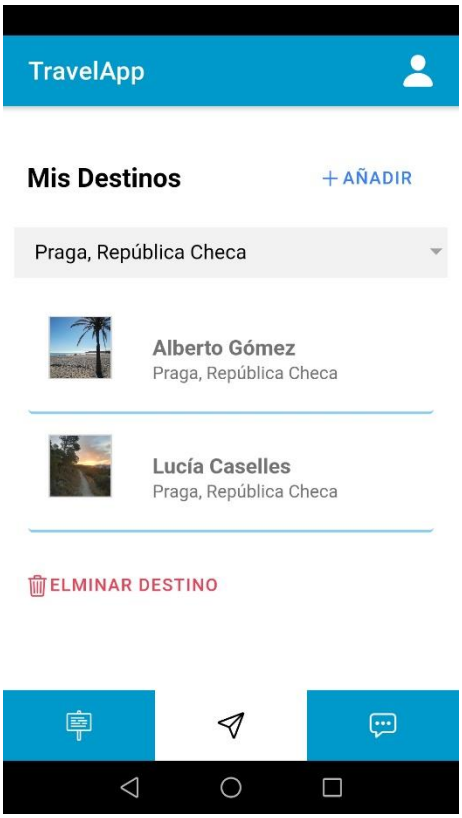

Figura 64: Interfaz final de mis destinos

Figura 65: Interfaz final de nuevo destino

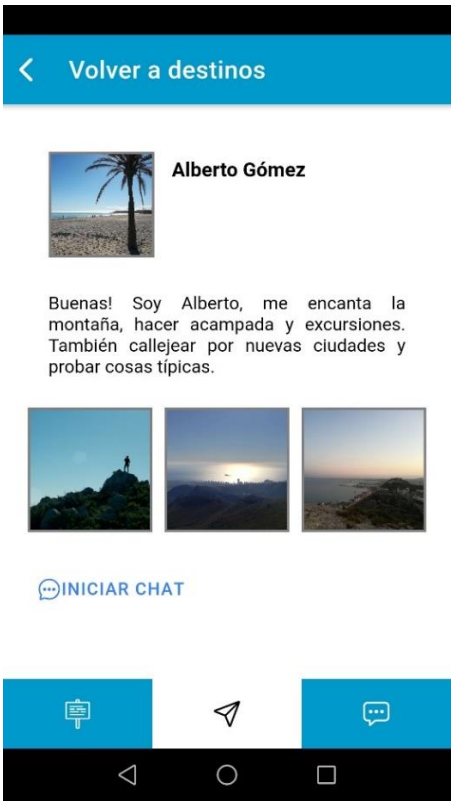

Figura 66: Interfaz final de perfil de usuario

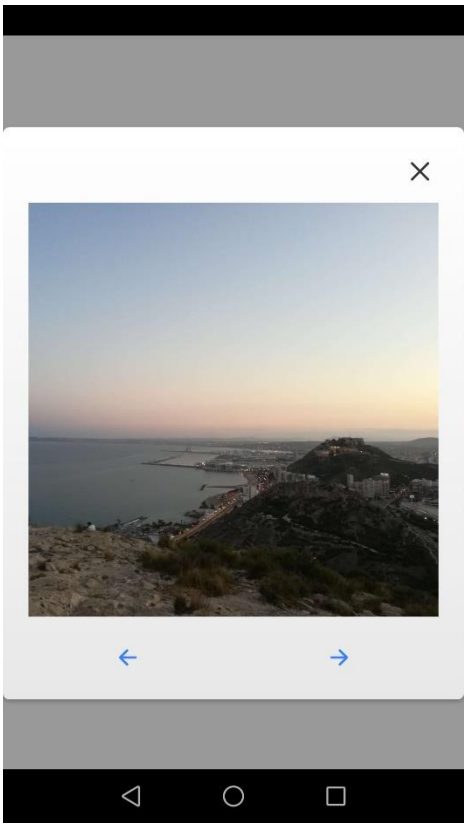

Figura 67: Interfaz final de navegar entre fotos

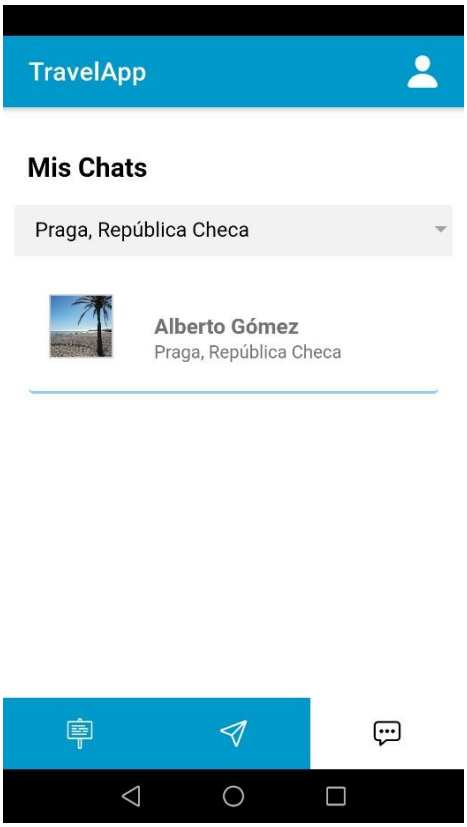

Figura 68: Interfaz final de mis chats

Figura 69: Interfaz final de chat

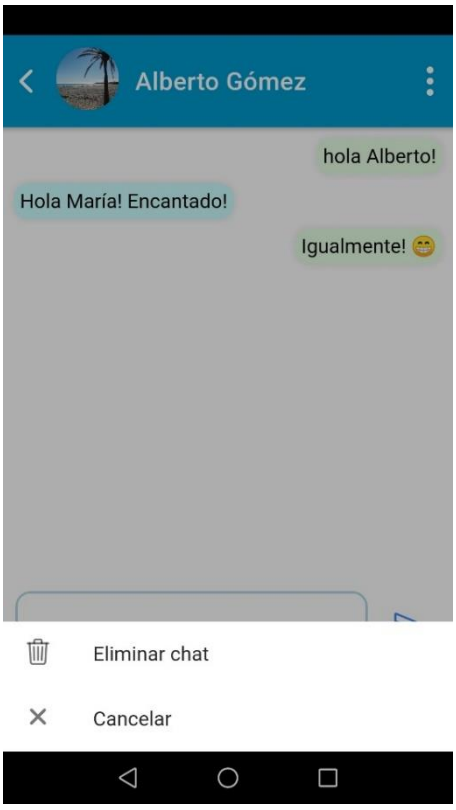

Figura 70: Interfaz final de eliminar chat

Figura 71: Interfaz final de perfil personal

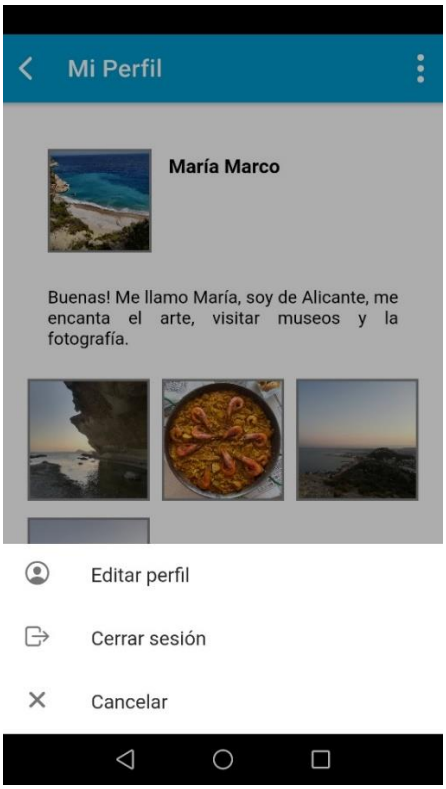

Figura 72: Interfaz final de opciones en perfil personal

Figura 74: Interfaz final de modificar perfil

Figura 73: Interfaz final de modificar contraseña

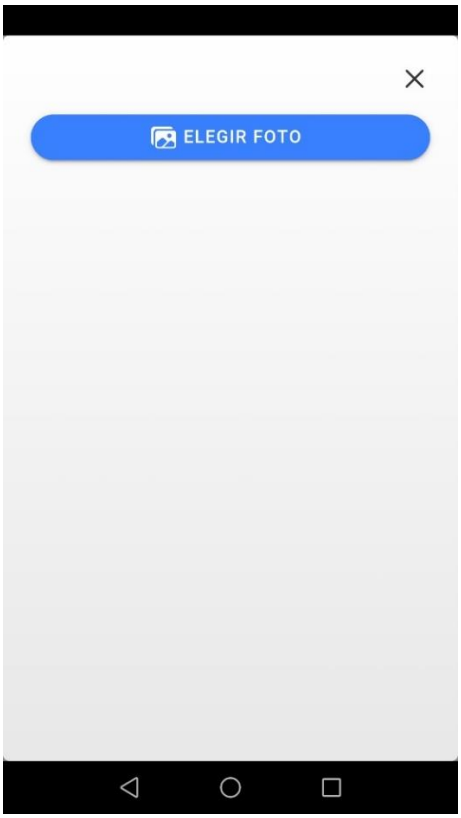

Figura 73: Interfaz final de subir foto

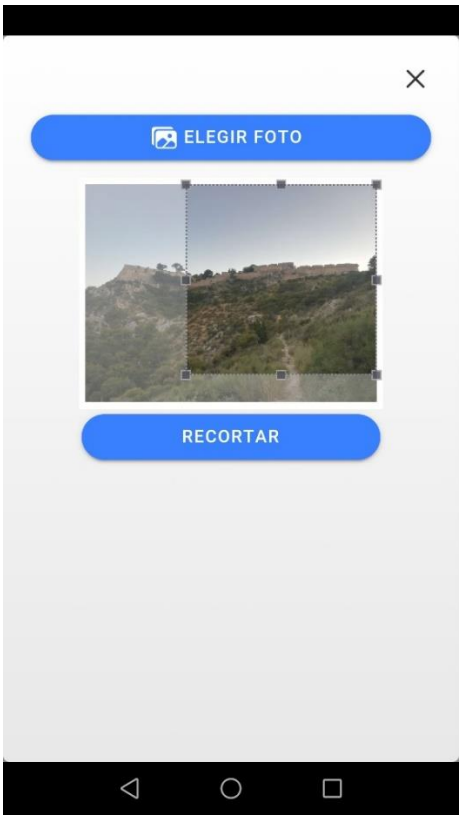

Figura 74: Interfaz final de recortar foto

Figura 75: Interfaz final de subir foto - 2

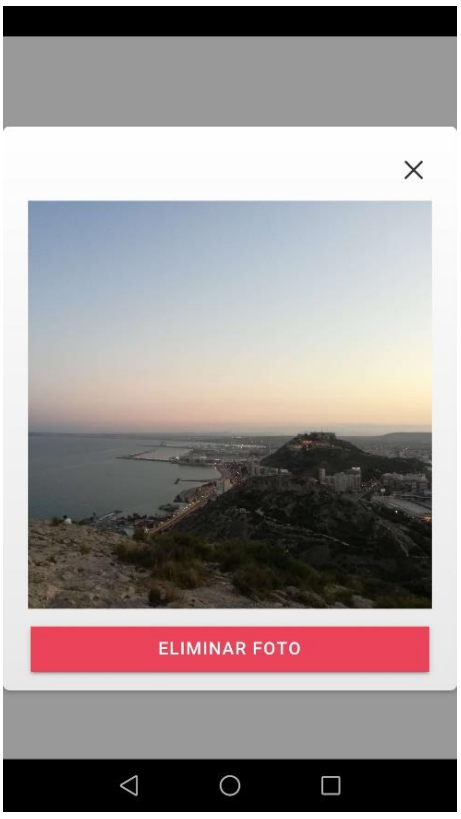

Figura 76: Interfaz final de eliminar foto

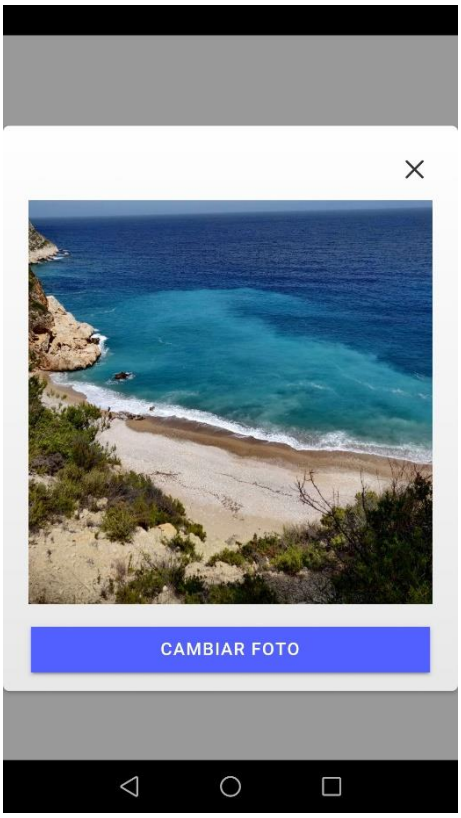

Figura 77: Interfaz final de modificar foto de perfil

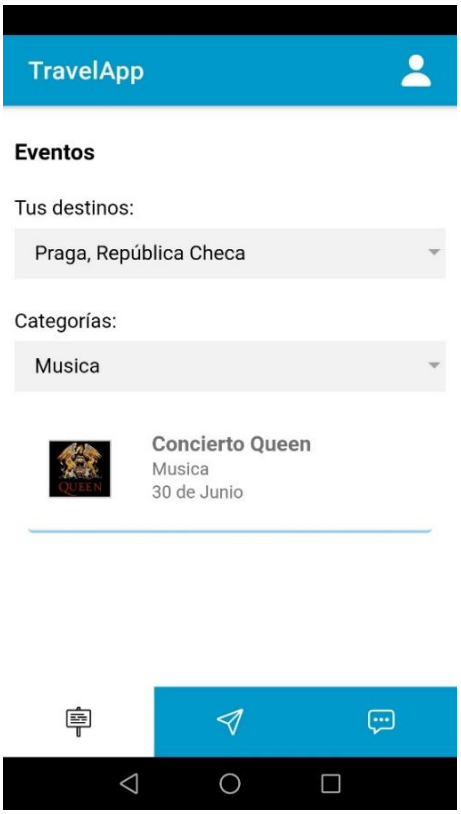

Figura 78: Interfaz final de eventos

Figura 79: Interfaz final de perfil de evento