Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

EventYou: Aplicación móvil con Flutter y FlutterFire

Autor: Álvaro Menacho Rodríguez
Tutor: José Manuel Fornés Rumbao

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

EventYou: Aplicación móvil con Flutter y FlutterFire

Autor:
Álvaro Menacho Rodríguez

Tutor:
José Manuel Fornés Rumbao
Profesor titular

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: EventYou: Aplicación móvil con Flutter y FlutterFire

Autor: Álvaro Menacho Rodríguez

Tutor: José Manuel Fornés Rumbao

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia y mis amigos

Agradecimientos

Quiero agradecer en estas líneas a todos los que me apoyan día a día y me han apoyado durante estos años para llegar aquí. A mi familia por estar siempre conmigo, confiar en mí, apoyarme y animarme siempre para seguir luchando. A ellos y a mis amigos por soportar mis ausencias. A mis amigos por convertirse en mi familia en Sevilla. A mis compañeros por hacer más ameno el día a día y haber estado en los buenos y en los malos momentos luchando juntos. A mi tutor por dejarme hacer realidad este proyecto personal. A todas las personas que ahora son parte de mí.

Álvaro Menacho Rodríguez

Sevilla, 2020

Resumen

La expansión imparable en la última década de los smartphones a nivel mundial ha propiciado el desarrollo de millones de aplicaciones móviles que han ido ampliando sus funcionalidades hasta límites insospechados. La cada vez mayor capacidad de procesamiento y la conectividad permanente de los dispositivos ha permitido la casi sustitución en muchos casos y ámbitos de pesados ordenadores portátiles para realizar una amplia variedad de tareas. Por lo que, en definitiva, están suponiendo un cambio en nuestra forma de vida, tanto en nuestras relaciones sociales como laborales.

La aparición de nuevos entornos de desarrollo y tecnologías como Flutter ha permitido la expansión del mercado de aplicaciones a empresas más pequeñas o desarrolladores independientes que no disponen de grandes equipos de desarrollo con conocimientos en desarrollo nativo o en desarrollo web.

Este trabajo pretende desarrollar una aplicación móvil multiplataforma llamada EventYou utilizando el nuevo entorno de desarrollo móvil de Google denominado Flutter y el lenguaje Dart. Esta aplicación está enfocada en la organización y gestión de eventos sociales privados, y pretende ser una herramienta útil y fácil de utilizar tanto para los organizadores como para los asistentes a dichos eventos.

Abstract

The unstoppable expansion of smartphones worldwide in the last decade has driven the development of millions of mobile apps which have been augmenting their functionalities till unsuspected limits. Higher processing capacities and permanent connectivity in our devices have allowed to replace heavyweight laptops in many diverse tasks. So ultimately, they are changing our lifestyle, both in our social and work relationships.

The appearance of new frameworks and technologies like Flutter has allowed the expansion of the applications market to smaller companies and freelance developers which do not have big development teams with native or web development skills.

This project aims to develop a cross-platform application named EventYou using the new mobile development framework of Google called Flutter and the Dart language. This application focuse on manage and organize private social events and wants to be a useful and intuitive tool for both organizers and assistants of those events.

Índice

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

El uso de teléfonos móviles está ampliamente extendido en nuestra sociedad actual, y más concretamente el uso de smartphones. Según las últimas estimaciones el mundo cuenta con aproximadamente tres billones y medio de usuarios de smartphones [1]. A nivel nacional, según datos de la CNMC, el 86% de las personas que tienen un teleféfono móvil disponen realmente de un smartphone [2].

Se han consolidado como el dispositivo más utilizado para acceder a Internet entre los españoles. Las posibilidades que ofrecen en cuanto a prestaciones y utilidades no dejan de aumentar casi mensualmente. Acompañados en todo momento por el desarrollo de aplicaciones cada vez más complejas y variadas que permiten realizar acciones hasta hace unos años inimaginables.

No podemos imaginar nuestras vidas actuales sin el uso diario de aplicaciones utilizadas mundialmente como redes sociales o aplicaciones de mensajería, de las que somos inseparables. Prácticamente de cualquier tema que pensemos existe una aplicación en el mercado; y esto es gracias a una "democratización" del desarrollo, en la que es más sencillo que alguien sin idea previa pueda aprender a desarrollar aplicaciones e incluso herramientas que permiten diseñar una aplicación sin tener ni idea de programación.

Aunque en los primeros años existían más sistemas operativos, y, por ende, más empresas dedicadas a su desarrollo, en la actualidad el mercado se divide casi en su totalidad entre: Android de Google y iOS de Apple. Esto se traduce en la casi obligación, para un desarrollador que quiera tener un éxito extendido, de lanzar su aplicación en ambas plataformas.

La manera de alcanzar este objetivo es diversa y existen diversos enfoques: el desarrollo nativo para cada plataforma, el desarrollo web, el desarrollo híbrido, que combina las dos anteriores, y el desarrollo en lenguajes multiplataforma que son posteriormente compilados a código nativo. Este trabajo expone este último enfoque.

Estructura

Este trabajo se ha organizado en cuatro capítulos:

- Flutter: este capítulo tiene como objetivo poner en contexto el framework utilizado en el desarrollo de la aplicación.
- FlutterFire: este capítulo se centra en el backend utilizado para el funcionamiento de la aplicación.
- EventYou: este capítulo presenta la aplicación desarrollada y sus diferentes funcionalidades, aportando los detalles del frontend.
- Conclusiones: este capítulo presenta los resultados tras la realización de este trabajo.

Objetivos

La idea de este trabajo proviene de dos fuentes distintas. Por un lado, la detección de la necesidad de una aplicación destinada a eventos sociales privados y por otro el interés que me despertó la asignatura de Diseño de Aplicaciones Móviles al realizar el trabajo final. Por estos dos motivos tenía claro que este era el tipo de trabajo final de grado que quería realizar.

Los objetivos que persigue este trabajo son:

- Diseñar y desarrollar una aplicación móvil multiplataforma utilizando el nuevo framework de desarrollo móvil creado por Google llamado Flutter.
- Constituir la base de la idea EventYou para su posterior comercialización en Google Play Store y Apple App Store.

2 FLUTTER

Flutter es un kit de desarrollo software de interfaz gráfica open source creado por Google. Fue mencionado por primera vez en la cumbre de desarrolladores de Dart de 2015 con el nombre en clave "Sky" [3] y su primer lanzamiento antes de la versión estable sucedió en los Días de los Desarrolladores de Google de 2017 en Shanghai. El lanzamiento de la versión estable fue el 4 de diciembre de 2018 en el Flutter Live Event [4].

Es utilizado actualmente en el desarrollo de aplicaciones para Android, iOS, Windows, Mac, Linux, Google Fuchsia y web a partir de un único código base. El proceso utilizado es la compilación del código en lenguaje Dart a código nativo de cada plataforma.

Este framework escrito en C, C++ y Dart se encuentra bajo licencia New BSD. Este hecho hace que se apoye mucho en una comunidad cada vez más numerosa y que se esté convirtiendo en la habilidad con más expansión entre ingenieros de software en la red profesional Linkedin [5].

Los paquetes oficiales y todos aquellos desarrollados por la comunidad, tanto para Flutter como para Dart están disponibles en https://pub.dev/. Mientras que la página oficial con toda la documentación es https://flutter.dev/.

Framework

Este framework de estilo reactivo se complementa con un motor de renderizado 2D, widgets listos para utilizar y herramientas de desarrollo. La arquitectura de Flutter se divide en los siguientes componentes:

- Plataforma Dart: para dispositivos móviles se realiza compilación AOT que permite que Flutter tenga un rendimiento elevado. Además, permite *hot reload* durante el desarrollo, es decir, permite realizar cambios y ver su comportamiento sin necesidad de volver a compilar toda la aplicación.
- Motor de Flutter: escrito en C++ y con soporte de renderizado de bajo nivel gracias a las librerías de gráficos de Skia de Google. Interactúa con el SDK específico de cada plataforma para implementar gráficos, entrada y salida de ficheros y de red, así como el tiempo de ejecución de Dart.
- Widgets específicos para cada plataforma, concretamente widgets de *Material Design* que implementan el lenguaje de diseño de Google y widgets *Cupertino* que implementan las guías de interfaz humana de iOS de Apple. Además, existe la posibilidad de crear tus propios widgets ya que en Flutter todo es un widget.

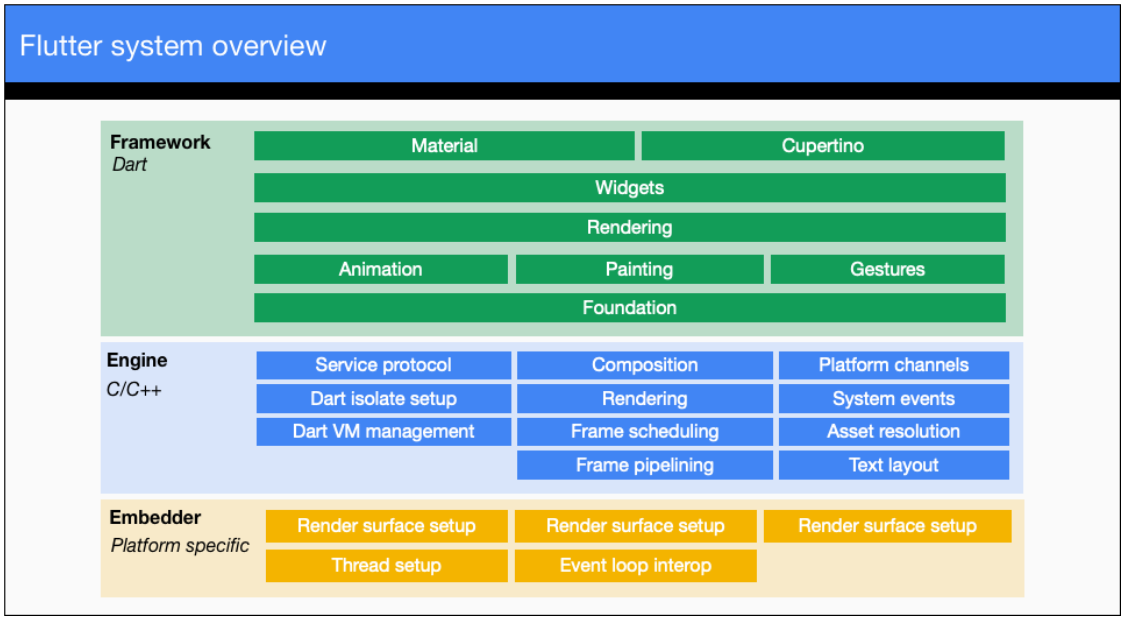

Ilustración 1: Visión del sistema Flutter [6]

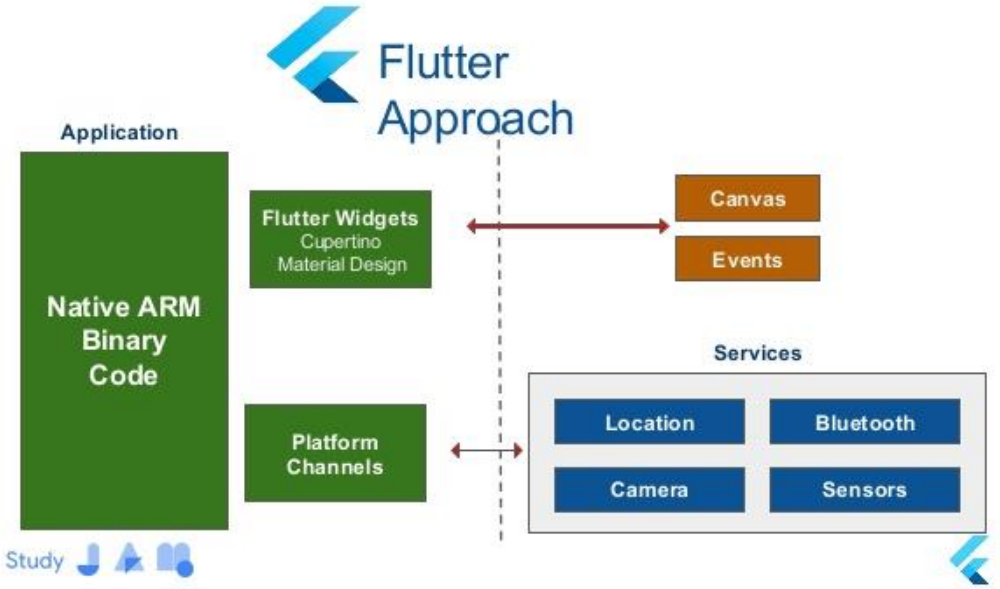

Ilustración 2: Arquitectura Flutter en un terminal

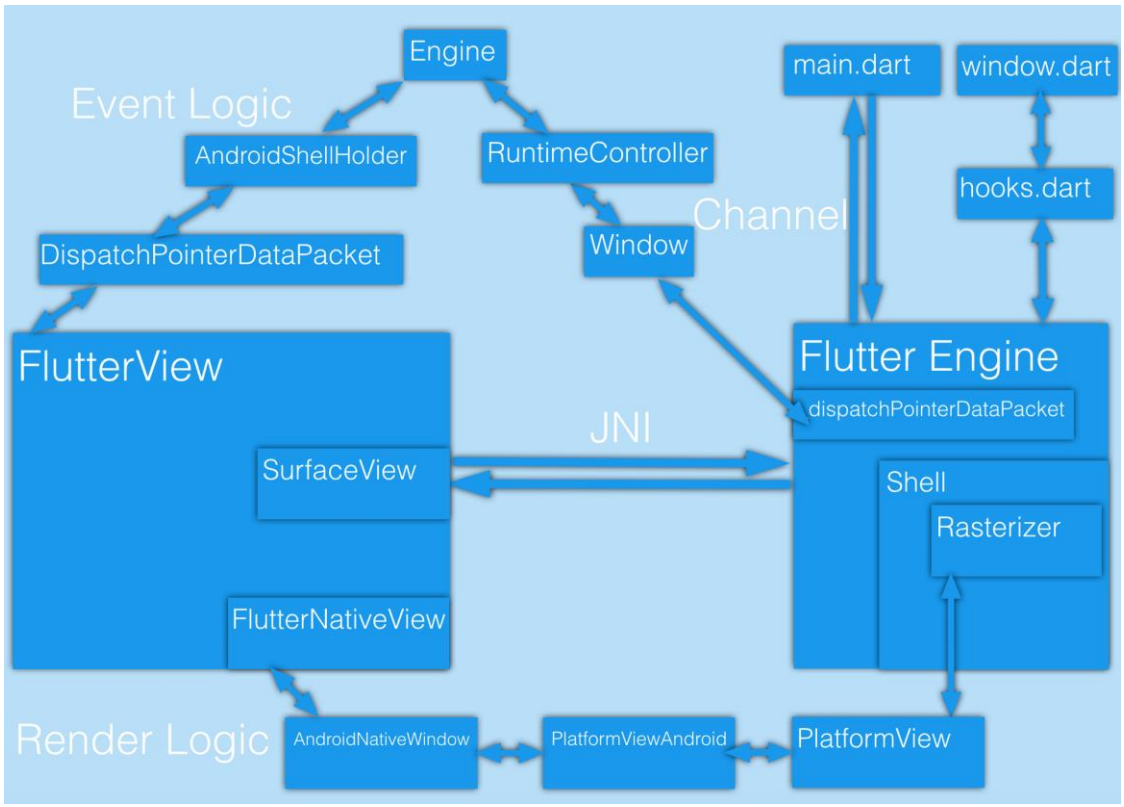

Ilustración 3: Lógica de ejecución de Flutter en Android [7]

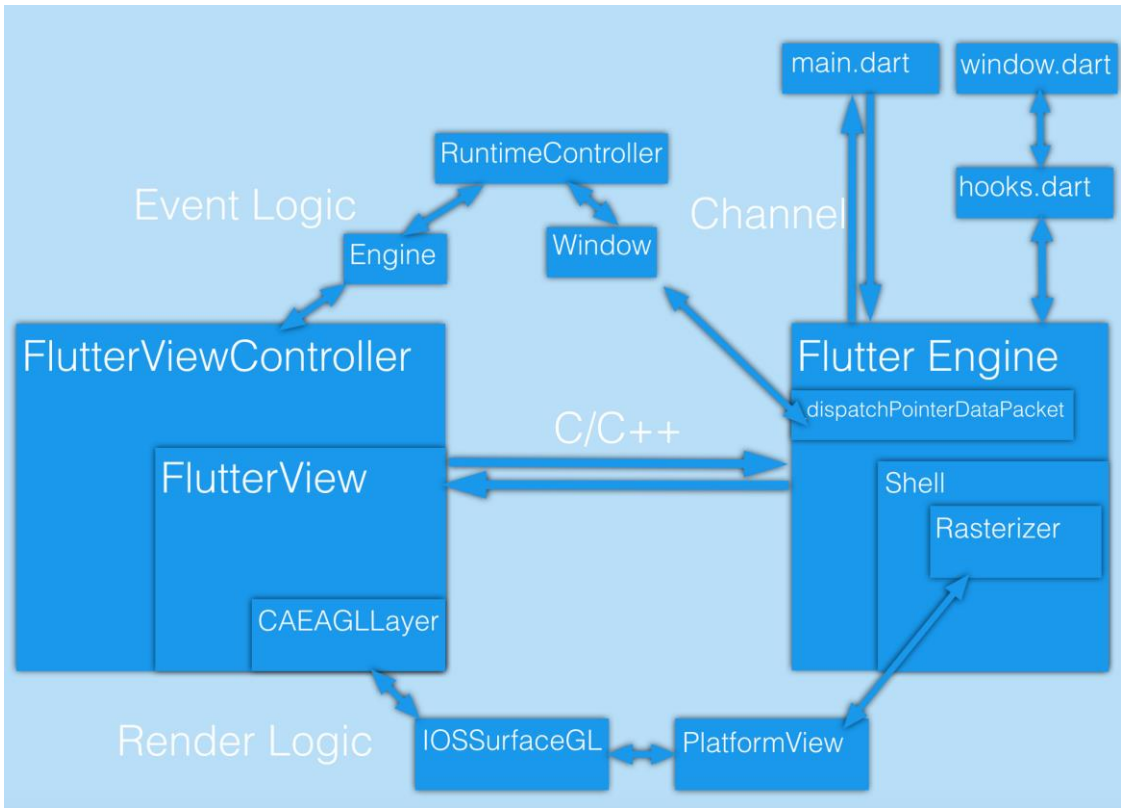

Ilustración 4: Lógica de ejecución de Flutter en iOS [7]

El framework se puede utilizar actualmente desde los editores Android Studio/ IntelliJ o Visual Studio Code. En el primero mediante la instalación de los plugins de Flutter y de Dart; mientras que en segundo mediante la instalación del complemento para Flutter.

Instalación

Para este trabajo el framework se ha instalado en Windows 10 para utilizarlo mediante el plugin para Android Studio. Para su funcionamiento es necesario PoweShell 5.0 y Git 2.x para Windows, por lo que debemos asegurarnos de tenerlos instalados previamente.

En primer lugar, se ha descargado el SDK de Flutter desde https://flutter.dev/docs/get-started/install y se ha descomprimido en la carpeta elegida. A continuación, se ha actualizado la variable de entorno PATH para incluir la ruta hacia el SDK.

Finalmente se han instalalo los plugins en Android Studio desde Opciones > Plugins.

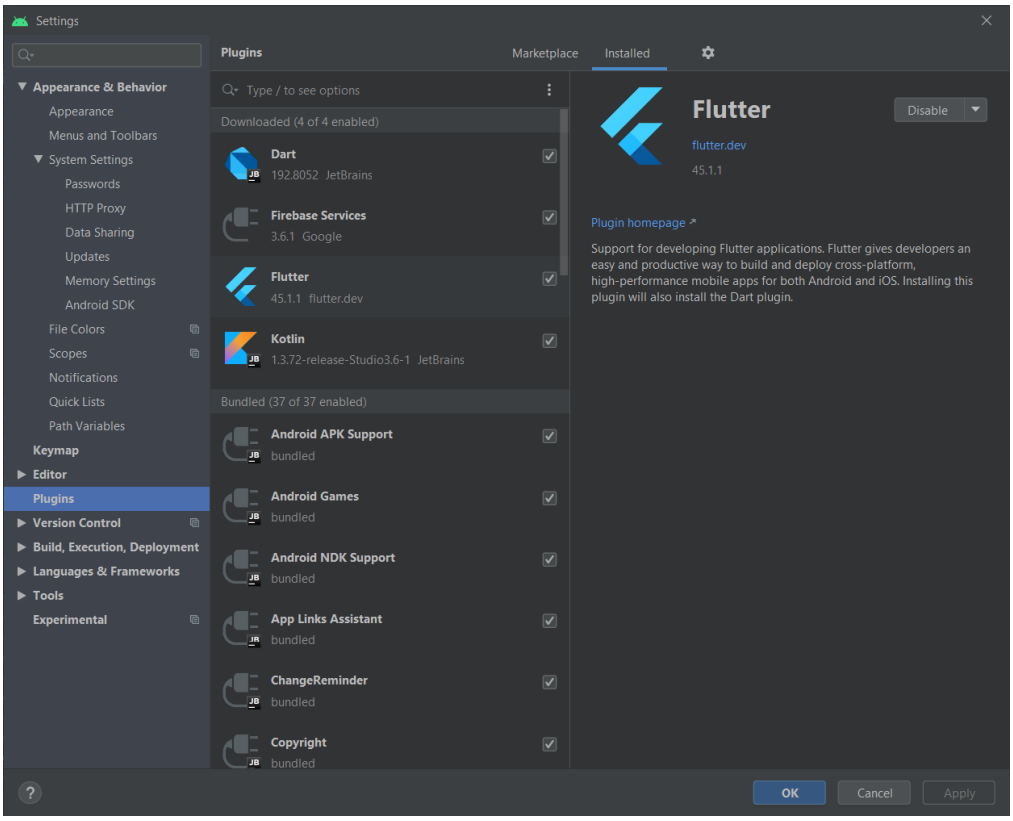

Ilustración 5: Instalación de plugins en Android Studio

Uso

Una vez instalados los plugins necesarios se podrá comenzar un nuevo proyecto de Flutter desde Android Studio.

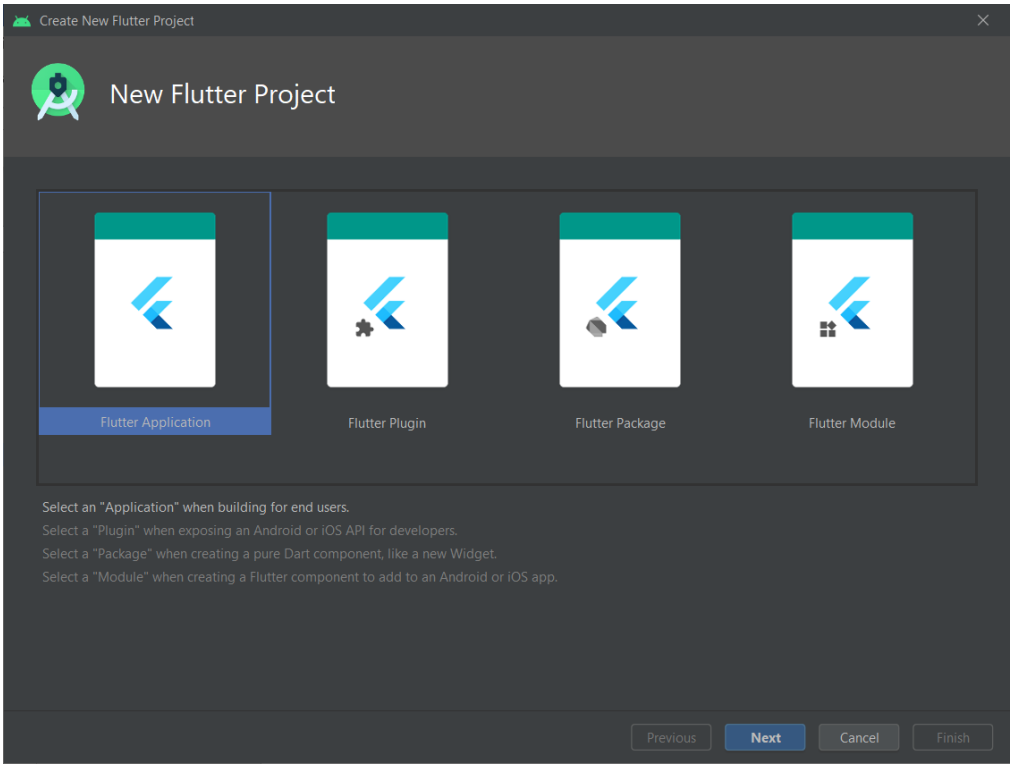

Ilustración 6: Creación de un proyecto Flutter

El proyecto se organiza en: módulo de Android, módulo de iOS y carpeta lib. Como sus nombres indican en el primero encontramos toda la estructura de un proyecto Android y en el segundo la estructura de un proyecto de Xcode. Mientras que en la carpeta lib situaremos todo el código Dart de la aplicación. Recién creado el proyecto esta carpeta solo contendrá el *main.dart* con el método main para iniciar la aplicación.

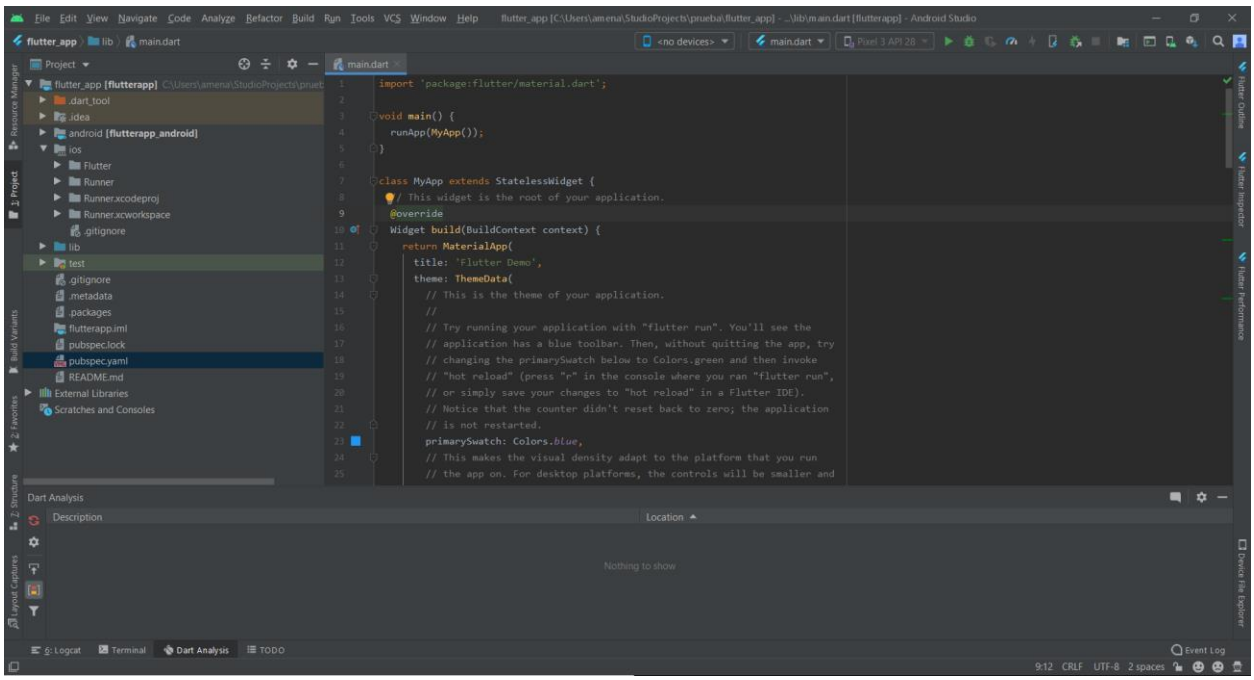

Ilustración 7: Estructura de un proyecto Flutter

Es muy importante también el fichero *pubspec.yaml* ya que es el encargado de gestionar las versiones y las dependencias de otros paquetes del proyecto. Mediante los comandos *flutter pub get*, *flutter pub upgrade* y *flutter doctor* se puede descargar y actualizar las dependencias, actualizar el SDK y comprobar que la configuración del SDK es correcta respectivamente.

El plugin incorpora a Android Studio cuatro elementos muy útiles durante el desarrollo: Dart Analysis, Flutter Outline, Flutter Inspector y Flutter Perfomance.

El primero es una herramienta de análisis sintactico para el lenguaje Dart. El segundo es una herramienta para visualizar e interactuar con las clases de un fichero concreto, con todas las variables y métodos declarados, permitiendo la edición de propiedades y la envoltura automática de un widget dentro de otro. El tercero es una herramienta para visualizar el árbol de widgets que se está renderizando en una ejecución en marcha, y el cuarto es un panel de información sobre los tiempos de renderizado de las pantallas y sobre el rendimiento en memoria de la aplicación.

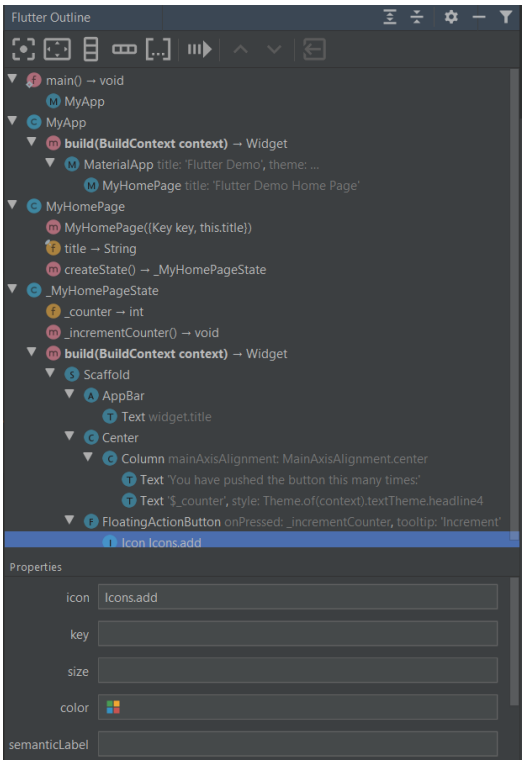

Ilustración 8: Flutter Outline

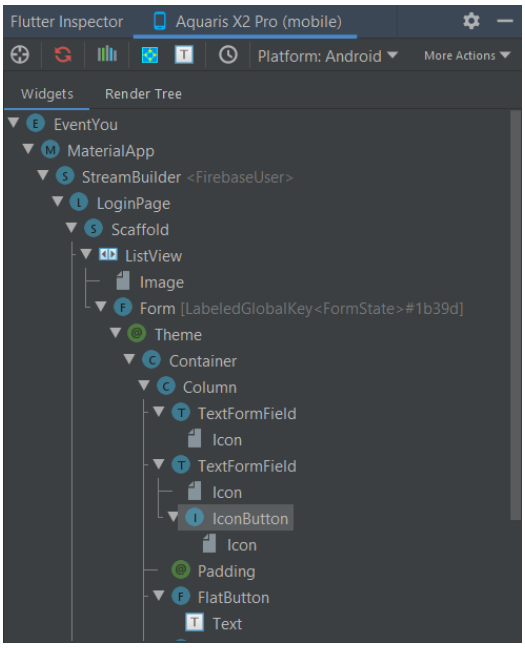

Ilustración 9: Flutter Inspector

Ilustración 10: Flutter Performance

Lenguaje Dart

Dart es un lenguaje optimizado para el lado cliente desarrollado por Lars Bak y Kasper Lund para Google en 2011. Su primera versión estable 1.0 llegaría no obstante en 2013 y su última versión es la 2.8.2, publicada en mayo de 2020.

Es un lenguaje orientado a objetos, basado en clases, reflexivo, con recolección de basura autogestionada e imperativo. Tiene una sintaxis muy parecida a C y está influenciado por lenguajes como Java, Javascript, Erlang, Kotlin, Ruby o Strongtalk.

Por otro lado, puede ser compilado a código nativo o a Javascript. Fue estandarizado por el comité técnico TC52 de ECMA Internacional y aprobado en julio de 2014 durante la 107ª asamblea general de dicha organización. Siendo la segunda edición de la especificación del lenguaje aprobado en diciembre de ese mismo año [8].

El lenguaje está optimizado para el diseño de interfaces gráficas y cuenta con mecanismos para escribir código basado en eventos y un sistema de tipado flexible. Además, permite la iteración mediante cambios en caliente que se ven reflejados inmediatamente en las aplicaciones.

Su rapidez en todas las plataformas y su flexibilidad para escribir tanto el frontend como el backend hacen de este lenguaje una auténtica apuesta ganadora en estos momentos.

Existe un editor online oficial para el lenguaje disponible en https://dartpad.dev/.

3 FLUTTERFIRE

Firebase es una plataforma de desarrollo móvil y web desarrollada por Firebase Inc. en 2011 y que fue adquirida por Google en 2014. Hoy en día, la plataforma tiene 19 productos y es utilizada por aproximadamente 1,5 millones de aplicaciones.

Es una plataforma que ofrece backend como servicio y que divide sus productos entre los enfocados al desarrollo, al análisis de datos y al crecimiento. Su principal objetivo es quitar a los desarrolladores la preocupación de contar con una infraestructura escalable y compatible con varias plataformas, adaptando su operativa al pago por uso.

La plataforma está completamente integrada con Google Cloud Platform lo que permite disfrutar de todas las ventajas de la computación en la nube. La más formidable es que las aplicaciones son escalables y tienen todo el respaldo de mantenimiento de Google.

Con el paso de los años la comunidad de Firebase ha ido aumentado exponencialmente y esto se ha traducido en aportaciones que han ampliado considerablemente las librerías, ejemplos y SDK's disponibles. Con el objetivo de que todo ese conocimiento revierta en la comunidad se publican las aportaciones en formato open source en GitHub [9], tanto por parte de la comunidad como por parte del equipo de Firebase.

FlutterFire es un conjunto de plugins de Flutter que permiten a las aplicaciones desarrolladas en dicho framework utilizar los servicios de Firebase y que es fruto de la colaboración entre la comunidad y Google. Actualmente hay disponible 15 plugins correspondientes a 15 de los servicios que ofrece Firebase.

En este trabajo se han utilizado los plugins desarrollados a continuación. [10]

Firebase Authentication

Este plugin [11] implementa el sistema de autenticación de Firebase y permite la gestión de un sistema completo de usuarios y sesiones. De los métodos de inicio de sesión disponibles se ha utilizado el de correo electrónico y contraseña con el objetivo de que los usuarios tengan una cuenta en exclusiva para esta aplicación y así no utilicen sistemas de autenticación de terceros como Google o Facebook.

En este trabajo se ha utilizado para el registro de los usuarios, la verificación por email de las cuentas, la recuperación de contraseñas olvidadas, la eliminación de los usuarios y sobretodo para gestionar todo el flujo de autenticación. Gracias a estas cuentas de usuario se puede identificar de forma unívoca a los clientes y es la base de toda la arquitectura de la aplicación y del backend asociado.

Cloud Firestore

Este producto de Firebase ofrece una base de datos No-SQL flexible y escalable que permite organizar estructuras de datos jerarquizadas en colecciones y documentos. Admite consultas simples y complejas a datos que son almacenados en caché cuando el dispositivo está sin conexión.

El plugin [12] permite realizar toda la comunicación necesaria con este producto y ha permitido establecer el modelo de datos que soporta todo el funcionamiento de la aplicación.

El modelo de datos se ha estructurado de la siguiente forma, recogida gráficamente en el Anexo A:

- Eventos: colección que recoge todos los eventos organizados mediante la aplicación y su información asociada. Cada documento se corresponde con un evento y utiliza un identificar único generado aleatoriamente. Cada evento se compone de:
    - 5 subcolecciones
        - Organizadores: colección compuesta por documentos identificados por el identificador de cada usuario. Cada documento tiene el siguiente campo:

        - ‘uid’: campo que almacena el identificar del usuario correspondiente.
    - Asistentes: colección compuesta por documentos identificados por el identificador de cada usuario. Cada documento tiene el siguiente campo:
        - ‘uid: campo que almacena el identificar del usuario correspondiente.
    - Tareas: colección compuesta por documentos con identificador generado aleatoriamente. Cada documento se corresponde con una tarea y dispone de los siguientes campos:
        - ‘id’: identificador único de la tarea.
        - ‘título’: título de la tarea.
        - ‘descripcion’: descripción con los detalles de la tarea.
        - ‘fecha’: marca temporal de la tarea.
        - ‘encargados’: vector de identificador de usuarios a los que se le ha asignado la tarea.
        - ‘completada’: booleano que indica si la tarea se ha completado.
    - Chat: colección compuesta por documentos identificados por su marca temporal y que corresponden a los mensajes del chat del evento. Cada mensaje tiene los siguientes campos:
        - ‘id’: identificador único del mensaje.
        - ‘createdAt’: marca temporal de creación del mensaje.
        - ‘customProperties’: objeto para englobar más propiedades.
        - ‘image’: url de Firebase Cloud Storage donde está almacenada la imagen si el mensaje la tuviera.
        - ‘quickReplies’: mensajes de respuesta rápida a los mensajes.
        - ‘text’: texto del mensaje.
        - ‘video’: url de Firebase Cloud Storage donde está almacenado el vídeo si el mensaje lo tuviera.
        - ‘user’: objeto que modela el usuario que envía el mensaje. Compuesto por:
            - ‘avatar’: url de Firebase Cloud Storage donde está almacenada la foto de perfil del usuario.
            - ‘color’: color del usuario en el chat.
            - ‘containerColor’: color de la burbuja del chat.
            - ‘name’: nombre de usuario.
            - ‘uid’: identificador del usuario.
    - Galeria: colección compuesta por documentos identificados de forma aleatoria y que corresponden a cada foto o vídeo incluido en la galería del evento de forma directa o tras haber sido enviado por el chat. Cada elemento tiene los siguientes campos:
        - ‘id’: marca temporal que sirve de identificador único del archivo.
        - ‘type’: tipo de elemento, foto o vídeo.
        - ‘url’: url de Firebase Cloud Storage donde está almacenado el archivo.
- 9 campos
    - ‘id’: identificador único del evento.

        - ‘date’: marca temporal que indica cuándo está programado el evento.
        - ‘descripcion’: descripción con los detalles del evento.
        - ‘linkInvite: enlace de invitación al evento.
        - ‘localizacion’: vector de ubicaciones donde se va a desarrollar el evento.
        - ‘nombre’: nombre del evento.
        - ‘public’: booleano que indica si el evento es público o privado.
        - ‘terminado’: booleano que indica si el evento ya ha finalizado.
        - ‘tipo’: tipo del evento. Los tipos disponibles actualmente son: cumpleaños, almuerzo, cena, desayuno, merienda, barbacoa, fiesta, quedada y otro.

- Nicknames: colección que almacena los nombres de usuario. Dispone de reglas de seguridad distintas, ya que es necesario que durante la fase de registro se compruebe la disponibilidad del nombre sin tener aún cuenta de usuario. Cada documento está identificado por el identificador único del usuario y tiene el campo siguiente:
    - ‘nickname’: nombre de usuario.
- Usuarios: colección que recoge todos los usuarios de la aplicación y la información asociada a cada uno de ellos. Cada documento representa a un usuario y se encuentra identificado unívocamente por el identificador del usuario. Cada usuario se compone de:
    - 3 subcolecciones
        - Grupos: colección compuesta por documentos con un identificador generado aleatoriamente. Cada documento corresponde por un grupo y tiene los siguientes campos:
            - ‘id’: identificador único de cada grupo.
            - ‘nombre’: nombre del grupo.
            - ‘contactos’: vector que contiene los identificadores de los usuarios que forman el grupo.
        - Notificaciones: colección compuesta por documentos con un identificador generado aleatoriamente. Cada documento corresponde con una notificación y tiene los siguientes campos:
            - ‘body’: texto del cuerpo de la notificación.
            - ‘date’: marca temporal de la notificación.
            - ‘idEvento’: identificador del evento al que corresponde la notificación.
            - ‘leida’: booleano que indica si la notificación ha sido leída por el usuario.
            - ‘payload’: carga de datos de la notificación.
            - ‘title’: nombre del evento al que pertenece la notificación.
            - ‘type’: tipo de la notificación. Los tipos disponibles actualmente son info, chat y tareas.
        - Invitations: colección compuesta por documentos con un identificador generado aleatoriamente. Cada documento corresponde con una invitación y tiene el siguiente campo:
            - ‘eventoID’: identificador del evento al que pertenece la invitación.
    - 13 campos
        - ‘badges’: vector para almacenar los identificadores de los logros conseguidos.

- ‘email’: correo electrónico del usuario.
- ‘eventosAsistidos’: vector de identificadores de los eventos a los que ha asistido el usuario.
- ‘eventosOrganizados’: vector de identificadores de los eventos que ha organizado el usuario.
- ‘notificacionesChat’: booleano que indica si el usuario desea recibir notificaciones sobre mensajes nuevos de los chats de los eventos en los que participa.
- ‘notificacionesInfo’: booleano que indica si el usuario desea recibir notificaciones sobre cambios en la información de los eventos en los que participa.
- ‘notificacionesTareas’: booleano que indica si el usuario desea recibir notificaciones sobre asignaciones de nuevas tareas.
- ‘proAccount’: booleano que indica si el usuario es del segmento gratuito o del segmento premium.
- ‘profilePhoto’: url de Firebase Cloud Storage donde está almacenada la foto de perfil del usuario.
- ‘telefono’: número de teléfono del usuario en formato internacional.
- ‘token’: token de Firebase Cloud Messaging de Google del dispositivo del usuario que permite recibir las notificaciones de este sistema.
- ‘uid’: identificador único del usuario.
- ‘username’: nombre de usuario.

Firebase Cloud Storage

Este plugin [13] ofrece la comunicación con el sistema de almacenamiento de Firebase. Un sistema escalable y robusto que permite reiniciar las subidas y las descargas en el punto en el que se interrumpieron. Además, realiza dichas operaciones sin importar la calidad de la red.

El almacenamiento consta de un único segmento y está organizado en dos directorios:

- Eventos: directorio que contiene a su vez los directorios de cada evento. Cada directorio, con el nombre del identificador del evento, tiene la siguiente estructura:
    - Galería: directorio que contiene todos los archivos multimedia subidos a la galería del evento.
    - ChatImages: directorio que contiene todas las imágenes enviadas por el chat del evento.
- Usuarios: directorio que contiene a su vez los directorios de cada usuario. Cada directorio, con el nombre del identificador del usuario, contiene, de momento, la foto de perfil de este.

Cloud Functions

Este producto de Firebase [14] permite ejecutar, en un entorno administrado y escalable, código de backend en respuesta a eventos activados por otros productos de Firebase o peticiones https. Firebase se encarga de aumentar los recursos de procesamiento según los patrones de uso de los usuarios. Es una solución aislada del cliente y que admite código en Node.js y TypeScript. Las funciones implementadas pueden consultarse en el Anexo B.

En EventYou se ha utilizado para las siguientes tareas:

- Implementar un disparador de Cloud Firestore que responde a la creación de documentos en la subcolección ‘notificaciones’ de cada usuario. Una vez detectado lee los campos de dichos objetos para enviar las notificaciones al usuario utilizando Firebase Cloud Messaging.

- Implementar un disparador de Cloud Firestore que responde a la eliminación de un usuario y realiza la eliminación del identificador de este de los grupos de otros usuarios en los que figure.
- Implementar un disparador de Cloud Firestore que responde a la eliminación de un usuario y lo quita del conjunto de asistentes de todos los eventos en los que figure como tal.
- Implementar un disparador de Cloud Firestore que responde a la eliminación de un usuario y lo quita del conjunto de organizadores de todos los eventos en los que figure como tal.
- Implementar un disparador de Cloud Firestore que responde a la eliminación de un usuario y lo quita de la lista de encargados de una tarea que tuviera asignada en cualquiera de los eventos en los que esté.

Firebase Analytics

Este plugin [15] integra Google Analytics en la aplicación aportando estadísticas detalladas de uso y participación de los usuarios. Permite hacer un seguimiento exhaustivo de las pantallas más frecuentadas y de los usuarios activos en tiempo real.

Por otro lado, permite establecer propiedades para los usuarios y embudos de conversión de los más de 200 eventos que se pueden registrar de la aplicación.

Firebase Crashlytics

Este plugin [16] conecta la aplicación con la herramienta Crashlytics. Es una herramienta que informa en tiempo real de errores y bloqueos en la aplicación de forma detallada aportando las circunstancias en los que se produjeron, incluidas las líneas de código concretas.

Permite realizar un seguimiento de los problemas de estabilidad e informar de la cantidad de usuarios afectados por los errores.

Firebase Dynamic Links

Este plugin [17] permite realizar la gestión y la creación de enlaces dinámicos en la aplicación. Funcionan en todas las plataformas incluso cuando hay que realizar primero la instalación de la aplicación y permiten conducir al usuario al punto deseado dentro de la aplicación. Además, se pueden configurar los metadatos del enlace como son el título, la descripción o la imagen a mostrar.

En la aplicación se han utilizado en dos situaciones:

- Enlace que invita a la descarga de la aplicación y la abre si ya está instalada.

Ilustración 11: Enlace de descarga de la aplicación.

- Enlaces de invitación a los eventos que permiten que un usuario sea agregado automáticamente a la lista de asistentes del evento.

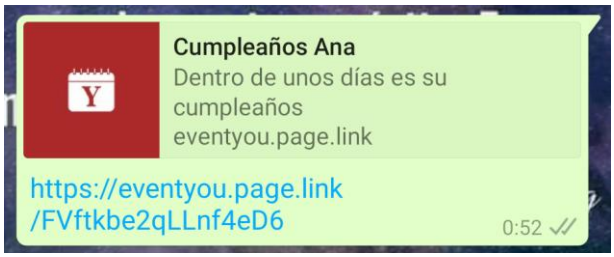

Ilustración 12: Enlace de invitación a un evento.

Firebase Cloud Messaging

Este producto de Firebase [18] implementa un servicio de mensajería multiplataforma que permite enviar desde un entorno de confianza (Firebase Cloud Functions o un servidor con Admin SDK) mensajes de notificación o mensajes de datos a las aplicaciones clientes. La distribución de estos mensajes puede ser a dispositivos individuales, grupos de dispositivos o a dispositivos subscritos a temas.

En la aplicación se hace uso de este producto [19] para notificar a los usuarios tres situaciones principalmente:

- Cambios en la información de un evento en el que figuremos como organizador o asistente.
- Nuevos mensajes en el chat de un evento.
- Asignación al usuario de una nueva tarea en un evento en el que figure como organizador.

El entorno de confianza seleccionado y que ejecuta la lógica es, como ya se ha mencionado anteriormente, Firebase Cloud Functions. La función utilizada puede consultarse en el Anexo B.

Firebase In-app messaging

Esta solución de Firebase [20] permite enviar mensajes contextuales a los usuarios de la aplicación en función de disparadores o de forma manual para interactuar con ellos cuando están usando la aplicación. Se puede utilizar para informar, ofrecer descuentos y promociones, solicitar feedback o dirigir a sitios concretos de la aplicación.

En la aplicación se ha utilizado [21] para instar a los usuarios a que instalen la última actualización disponible.

Firebase Extensions

Las extensions de Firebase [22] son soluciones empaquetadas que utilizan otros productos de Firebase para ofrecer una funcionalidad completa capaz de operar autónomamente después de ser configurada. Actualmente las extensiones disponibles son: modificación del tamaño de imágenes, traducción de texto, sincronización con Mailchimp, activación de correo electrónico, exportación de colecciones a BigQuery, acortamiento de URLs, contadores distribuidos, limitación de nodos secuendarios en Realtime Databse y eliminación de datos de usuario.

En la aplicación se ha utilizado la última extensión mencionada para automatizar la eliminación completa de un usuario, tanto si esta ocurre desde la consola de Firebase como desde la aplicación al solicitarla el usuario. El proceso configurado elimina al usuario de la colección de *usuarios* de Cloud Firestore, su nombre de usuario de la colección de *nicknames* y su carpeta de Firebase Cloud Storage.

4 EventYou

Actualmente la organización de un evento social privado puede resultar tediosa y llevar mucho tiempo. El uso habitual de aplicaciones de mensajería para este cometido origina pérdidas de información además de no contar de herramientas e interfaz nativa para este cometido, mientras que el uso de herramientas especializadas en eventos públicos se antoja excesiva al estar dedicadas a un público elevado.

Las aplicaciones de mensajería no permiten tener toda la información concentrada y no garantizan llegar a todos los invitados, ya que algunos puede que no se encuentren en dicha aplicación o no hayan leído un determinado mensaje dentro de la vorágine de mensajes que con asiduidad acompañan a un grupo creado con este fin. Es una oportunidad dotar a los organizadores de una herramienta que les permita mantener el control de la situación y contar con un canal de comunicación fácil de usar y ameno.

EventYou pretense ser esa herramienta fácil de usar y de uso masivo que permita organizar y gestionar, abarcando todo el ciclo de vida del evento.

Para conseguirlo incorpora las siguientes características:

- Un calendario personal con los eventos organizados y a los que se ha asistido o se va a asistir.
- Un sistema de invitaciones que permite aceptar o rechazar la asistencia a un evento.
- Una pantalla de creación de eventos fácil de usar y que permite establecer toda la información básica de una sola vez.
- Un apartado de perfil personal en el que se puede modificar el nombre de usuario o el número de teléfono asociado y realizar acciones como eliminar la cuenta o cambiar la contraseña de acceso.
- Un apartado que presenta un histórico de los eventos organizados, asistidos y a los que se va a asistir.
- Un apartado de gestión de contactos que permite ver qué contactos de la agenda del teléfono están registrados en la aplicación y organizarlos en grupos para agilizar las invitaciones a eventos.
- Una pantalla de visualización de notificaciones asociadas a los eventos en los que figura el usuario.
- Un apartado de ajustes en el que activar o desactivar alguno de los tipos de notificación disponible.
- Pantallas dentro de cada evento para visualizar la información, gestionar los organizadores y los asistentes, hablar por chat, subir contenido multimedia a la galería del evento o asignar tareas entre los organizadores.

Desarrollo

4.1.1 Inicio de sesión y registro

La primera pantalla disponible en la aplicación tras la pantalla de carga es la correspondiente al inicio de sesión. Esta pantalla permite realizar cuatro acciones: iniciar sesión, acceder a la pantalla de registro, acceder a la pantalla de recuperación de contraseña y enviar un email a la dirección de correo de soporte técnico.

En el diseño de la aplicación se ha decidido utilizar como método de autenticación la combinación correo electrónico/contraseña. Se establece una sesión permanente en el dispositivo de forma que el usuario por regla general no tiene que volver a autenticarse.

Por otro lado, se han diseñado dos pantallas de registro en las que se pide al usuario que introduzca el correo electrónico que desea utilizar, una contraseña, el nombre de usuario que quiere utilizar, previa comprobación de su disponibilidad y un número de teléfono móvil. Tras aceptar los términos de uso y la política de privacidad y comprobado que el email no está en uso y que la contraseña cumple la política de contraseñas, se envía un correo electrónico de verificación con un enlace de activación de la cuenta. Una vez realizado este paso se podrá iniciar sesión.

En cuanto a la pantalla de recuperación de contraseña se realiza una comprobación previa de si el email indicado está registrado, ya que se ha establecido que ha de usarse como correo de recuperación el mismo utilizado en el registro. Una vez hecho se envía un correo electrónico con un enlace para restablecer la contraseña.

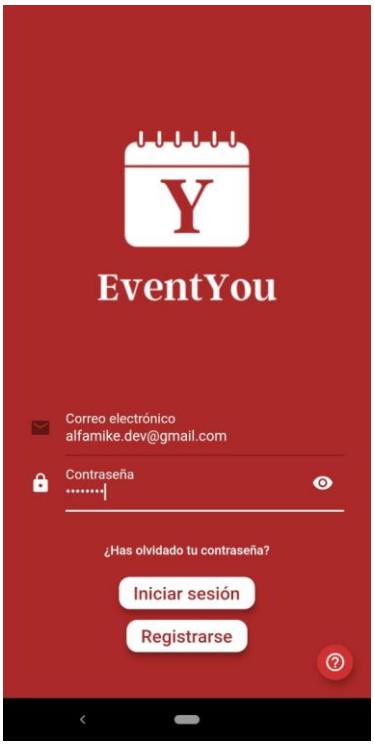

Ilustración 13: Pantalla de inicio de sesión

Ilustración 14: Pantalla de recuperación de contraseña

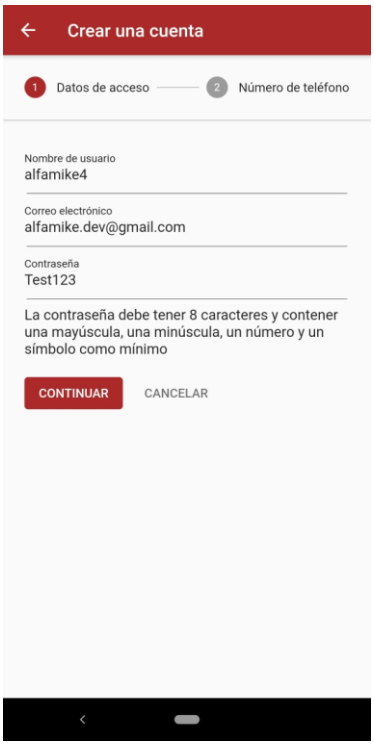

Ilustración 15: Pantalla de registro (I)

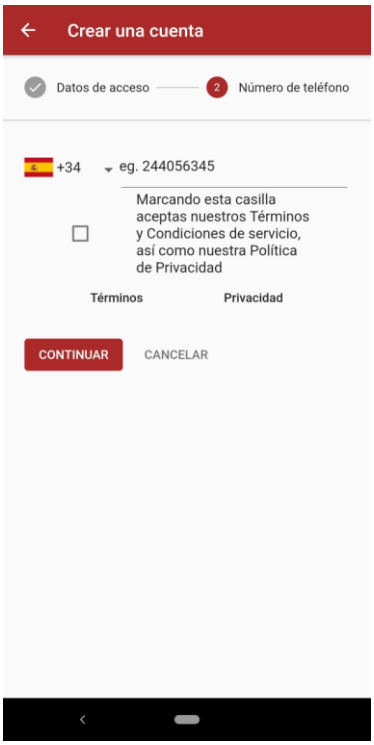

Ilustración 16: Pantalla de registro (II)

4.1.2 Pantalla principal

La pantalla principal de la aplicación está compuesta por cinco elementos: un menú desplegable lateral de navegación, un botón flotante para la creación de eventos, un botón de refresco, una página con el calendario personal y una página para gestionar las invitaciones de los eventos.

El menú lateral dirige la navegación hacia todas las secciones de la aplicación y es por tanto esencial en la fluidez de uso. Mientras que el botón de refresco es vital para sincronizar la aplicación con el backend.

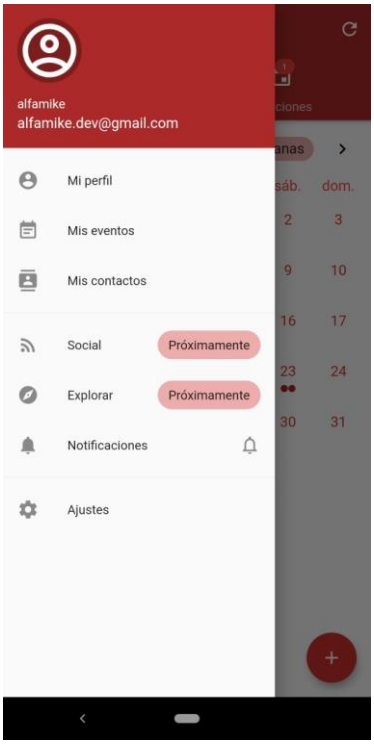

Ilustración 17: Menú lateral de navegación

4.1.2.1 Calendario

Esta página presenta un calendario personal que refleja los eventos del usuario, tanto los organizados como aquellos en los que figura como asistente. Disponiendo de tres vistas diferentes de las fechas: mes, dos semanas y una semana.

Los eventos del día seleccionado son desplegados debajo del calendario ofreciendo los detalles y la oportunidad de acceder a las pantallas de dicho evento.

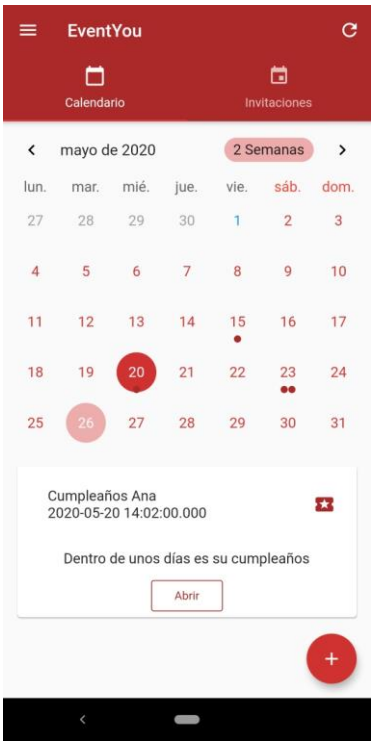

Ilustración 18: Pantalla del calendario

4.1.2.2 Invitaciones

Esta página presenta una lista de las invitaciones de eventos pendientes de gestionar por parte del usuario. Cada invitación refleja la información principal de cada evento y permite aceptar o declinar. Una vez seleccionada la opción la invitación desaparecerá y el evento será accesible desde el calendario y desde la sección de *Mis eventos*. Además, el usuario será agregado a la lista de asistentes en caso de haber sido aceptada.

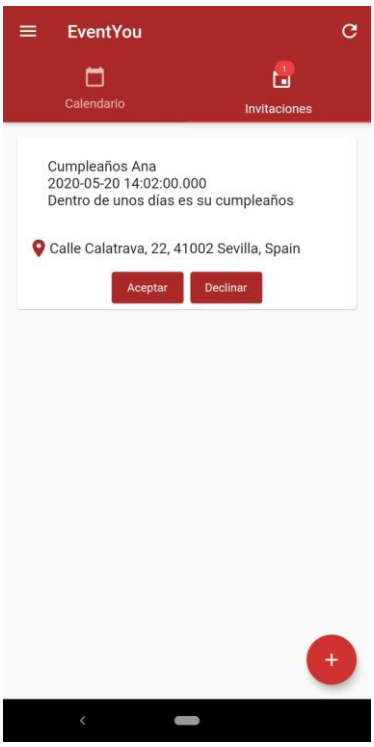

Ilustración 19: Pantalla de invitaciones

4.1.2.3 Creación de eventos

La pantalla de creación de los eventos pretende ofrecer una interfaz sencilla muy desacoplada en cada paso para aportar toda la información de forma amena. Los pasos que seguir durante la configuración son:

- Introducción del nombre, descripción y tipo del evento. Los tipos disponibles actualmente son: cumpleaños, almuerzo, cena, desayuno, merienda, barbacoa, fiesta, reunión u otro.
- Introducción de la hora y la fecha.
- Introducción opcional de una ubicación inicial para el evento. Posteriormente es posible añadir varias localizaciones una vez dentro de la pantalla del evento.
- Introducción de los organizadores iniciales. Posteriormente es posible agregar más organizadores y a todos los asistentes.
- Generación del enlace de invitación al evento y botón para compartirlo en redes sociales o aplicaciones de mensajería.

Ilustración 20: Pantalla de creación de evento (I)

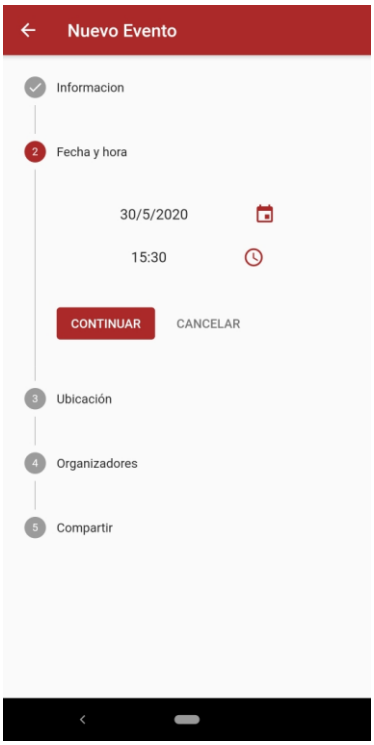

Ilustración 21: Pantalla de creación de evento (II)

Ilustración 22: Pantalla de creación de evento (III)

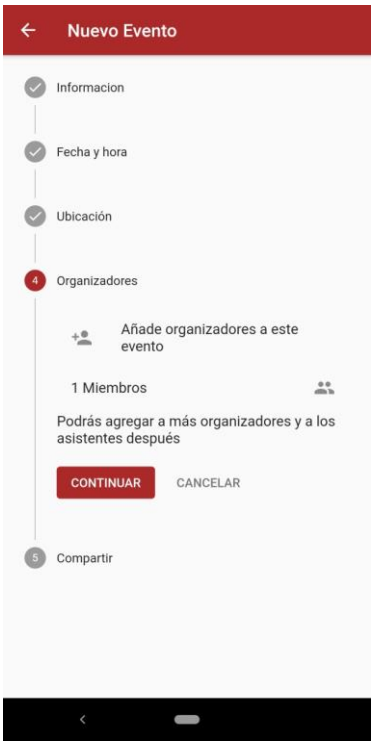

Ilustración 23: Pantalla de creación de evento (IV)

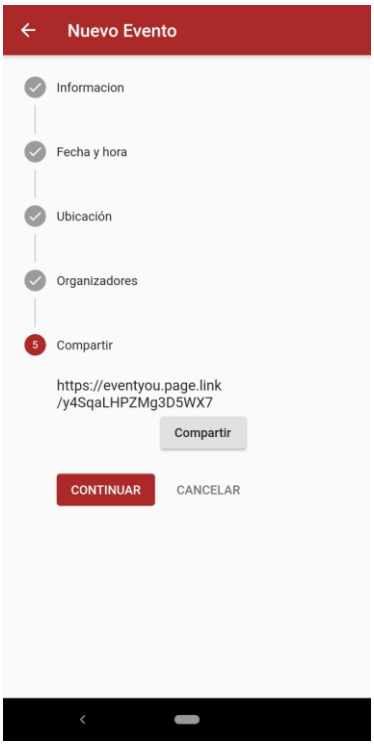

Ilustración 24: Pantalla de creación de evento (V)

4.1.3 Evento

Los eventos de los que forma parte el usuario son accesibles desde la pantalla del calendario o desde la sección *Mis eventos*. El conjunto de información, acciones, gestión y comunicación del evento se realizan desde las pantallas que se van a comentar a continuación. Estas pantallas varían dependiendo del rol que ejerza el usuario en el evento: organizador o asistente.

4.1.3.1 Información

La pantalla de información presenta los campos: nombre, descripción, tipo de evento, fecha y hora, y ubicaciones. Con el rol de organizador se pueden modificar todos los campos y en el caso concreto de ubicaciones se pueden agregar nuevas o eliminarlas. Las ubicaciones al pulsarlas abrirán la aplicación de mapas disponible en el dispositivo. Además, también se puede cancelar el evento.

Mientras que con el rol de asistente solo se pueden ver. Con independencia del rol cualquier cambio realizado en alguno de los campos será motivo de notificación a todos los participantes.

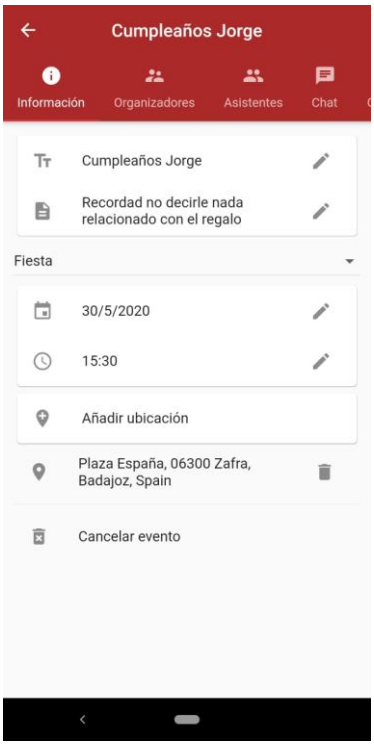

Ilustración 25: Pantalla de información del evento. Rol organizador

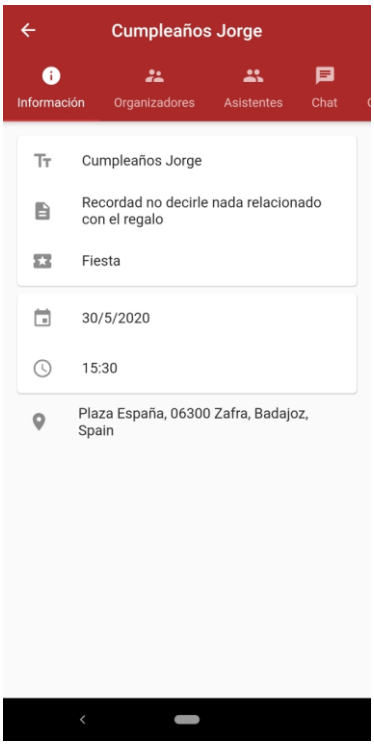

Ilustración 26: Pantalla de información del evento. Rol asistente

4.1.3.2 Organizadores

Esta pantalla permite agregar organizadores al evento mediante una búsqueda de los usuarios registrados en la aplicación. Si este usuario se encontraba entre los asistentes del evento causará baja en ellos y será incorporado a organizadores. De igual modo, si un organizador es eliminado se incorporará automáticamente a la lista de asistentes. Estos cambios solo implican un cambio de rol.

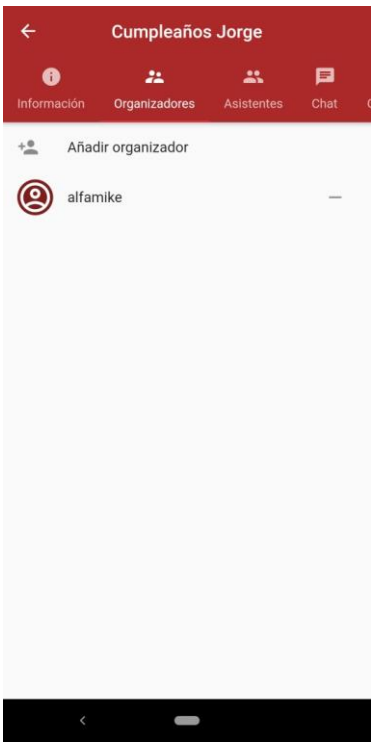

Ilustración 27: Pantalla de organizadores de evento. Rol organizador

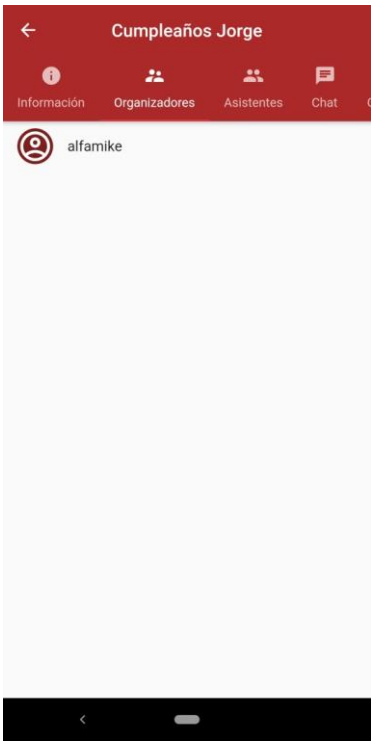

Ilustración 28: Pantalla de organizadores del evento. Rol asistente

4.1.3.3 Asistentes

Esta pantalla permite gestionar los asistentes al evento. Mientras que un asistente solo puede ver, el rol de organizador puede agregar y eliminar asistentes. Para agregar nuevos asistentes existen tres formas:

- Invitación a grupos creados por el usuario mediante la sección *Mis contactos*.
- Botón para agregar asistentes mediante una búsqueda entre todos los usuarios de la aplicación. Esta acción se traduce en el envío de invitaciones a los usuarios indicados.
- Botón flotante para compartir el enlace de invitación en aplicaciones de mensajería o en redes sociales.

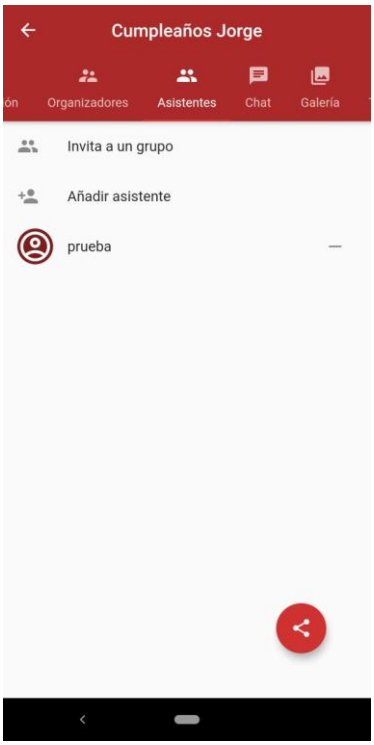

Ilustración 29: Pantalla de asistentes del evento. Rol organizador

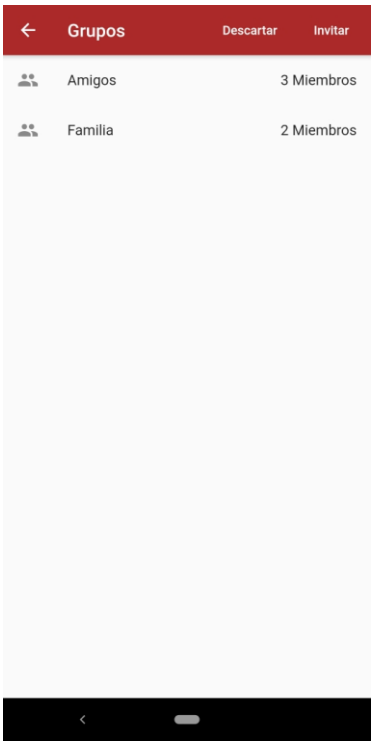

Ilustración 30: Pantalla de invitación mediante grupos

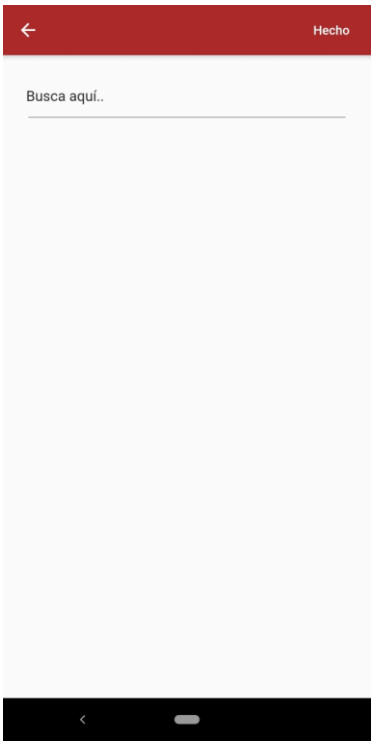

Ilustración 31: Pantalla de búsqueda de usuarios

4.1.3.4 Chat

Esta pantalla ofrece un chat para la comunicación entre los participantes del evento. Admite el envío de imágenes y genera notificaciones cuando se envían nuevos mensajes. Los usuarios se identifican mediante su foto de perfil.

Ilustración 32: Pantalla del chat del evento

4.1.3.5 Galería

Esta pantalla ofrece la gestión de la galería del evento. Permite subir fotos y vídeos desde el almacenamiento del dispositivo y también recopila las imágenes enviadas por el chat. Todo el contenido multimedia puede ser descargado o eliminado, y los vídeos pueden ser reproducidos con un reproductor integrado.

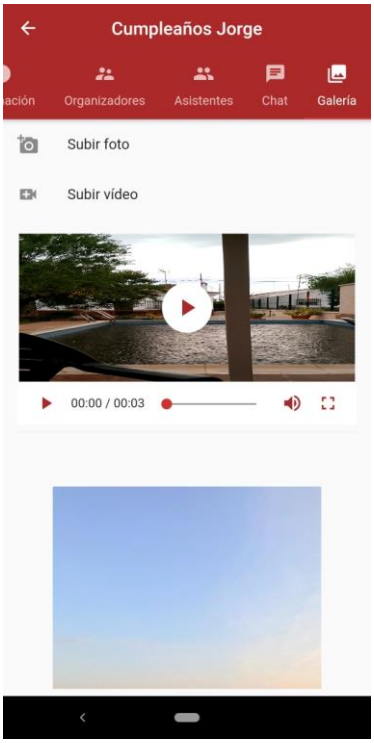

Ilustración 33: Pantalla de galería del evento

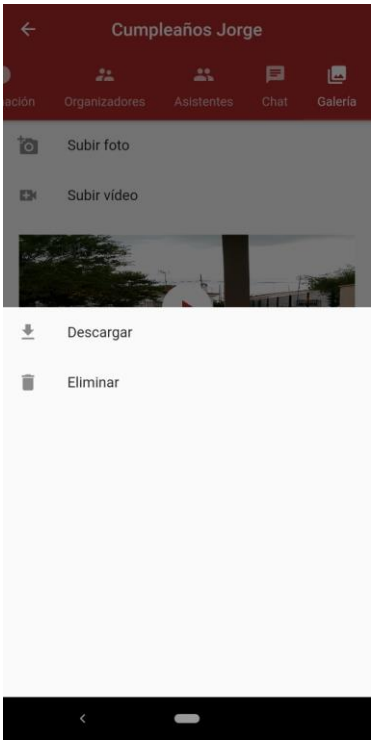

Ilustración 34: Opciones del contenido multimedia

4.1.3.6 Tareas

La pantalla de tareas solo esta disponible con el rol de organizador. Permite gestionar la creación y edición de tareas para los organizadores del evento. La asignación de una nueva tarea genera una notificación para el usuario correspondiente.

Ilustración 35: Pantalla de creación de una tarea

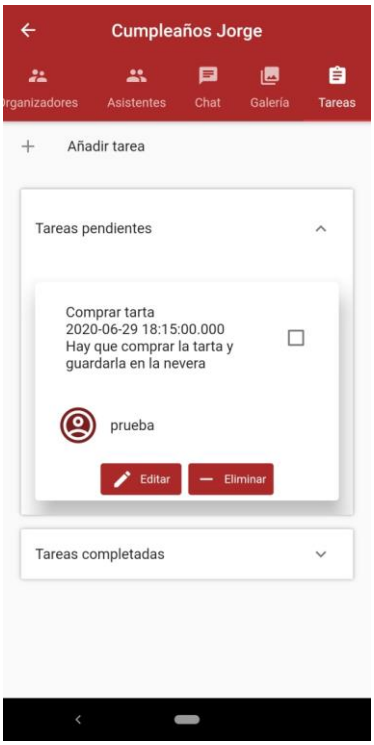

Ilustración 36: Pantalla principal de tareas del evento (I)

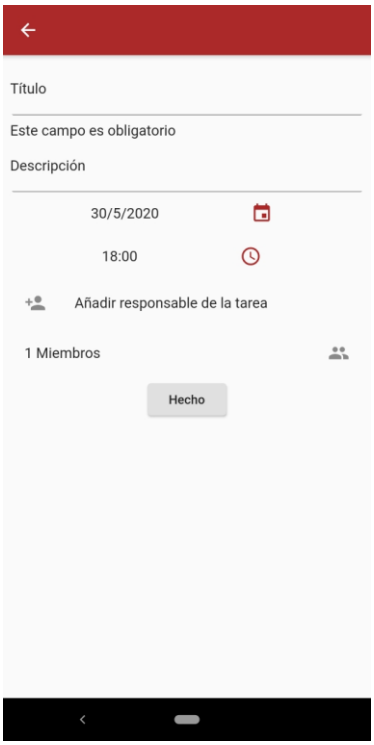

Ilustración 37: Pantalla de modificación de una tarea

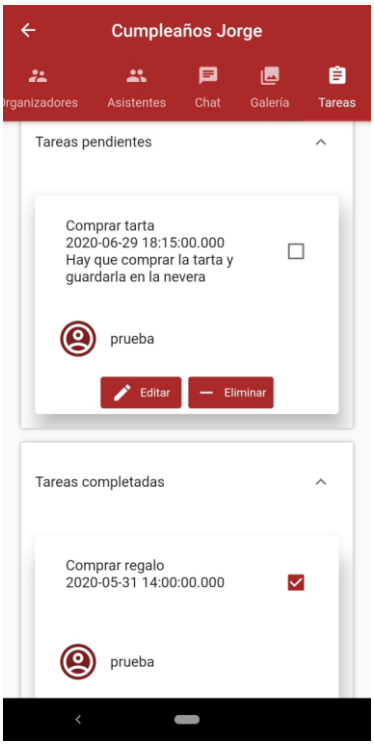

Ilustración 38: Pantalla principal de tareas del evento (II)

4.1.4 Perfil

La sección *Mi perfil* permite gestionar la información personal del usuario. En concreto, cambiar la foto de perfil seleccionando una imagen desde la galería o utilizando la cámara, cambiar el nombre de usuario, previa comprobación de la disponibilidad, y cambiar el número de teléfono asociado a la cuenta.

Por otro lado, esta pantalla permite realizar tres acciones de control sobre la cuenta de usuario: cambiar la contraseña, cerrar sesión y eliminar la cuenta.

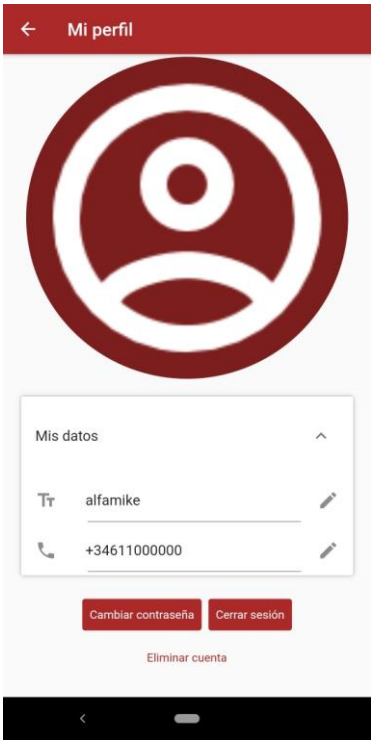

Ilustración 39: Pantalla de perfil (I)

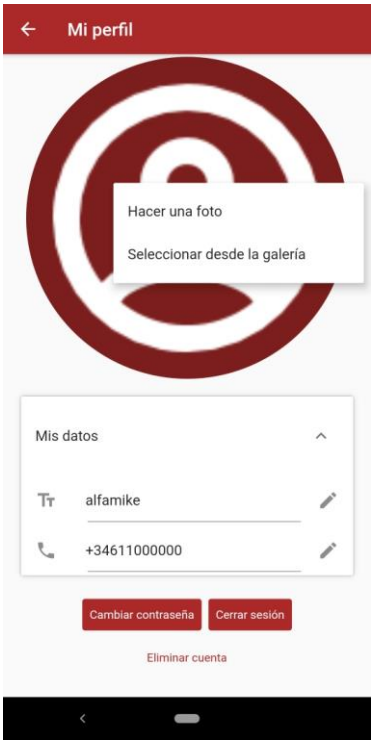

Ilustración 40: Pantalla de perfil (II)

4.1.5 Eventos organizados y asistidos

La sección *Mis eventos* consta de dos pantallas que despliegan los eventos organizados y los eventos asistidos o a los que se va a asistir respectivamente. Estas listas ordenadas cronológicamente pretenden servir de histórico y también proporcionar un rápido acceso a los eventos sin tener que desplazarnos por el calendario de la pantalla principal hasta la fecha correspondiente. El acceso a los eventos se realizará mediante la pulsación sobre su tarjeta y acorde al rol que tenga el usuario en ese evento.

Ilustración 41: Pantalla de eventos organizados

Ilustración 42: Pantalla de eventos asistidos

4.1.6 Contactos

La sección *Mis contactos* pretende ayudar al usuario a encontrar a familiares, amigos y conocidos que también tengan cuenta en la aplicación. Además, los contactos que estén pueden ser organizados en grupos para facilitar el proceso de invitación a los eventos que organice el usuario.

Siempre que el usuario haya dado permiso para acceder a los contactos de su agenda se desplegará una lista con todos ellos que mostrará un botón de invitación para aquellos que no tengan cuenta.

Por otro lado, el apartado de grupos consta de una pantalla para la creación o eliminación de ellos. La creación requerirá de un nombre y de los integrantes, que deberán ser mínimo dos y no podrá incluirse uno mismo. Una vez creado, se podrá acceder, pulsando en ellos, a una pantalla de detalle de los integrantes en la que se podrá agregar o quitar a cualquiera de ellos.

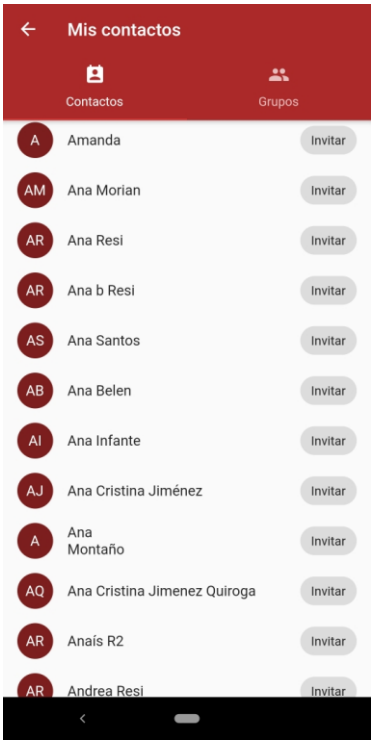

Ilustración 43: Pantalla de contactos de la agenda

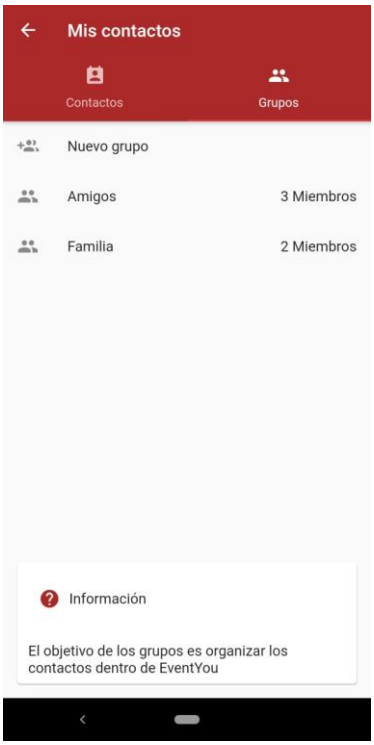

Ilustración 44: Pantalla de gestión de grupos

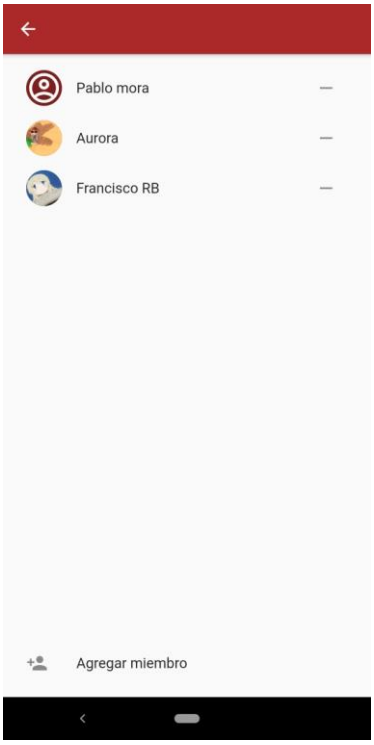

Ilustración 45: Pantalla de detalle de un grupo

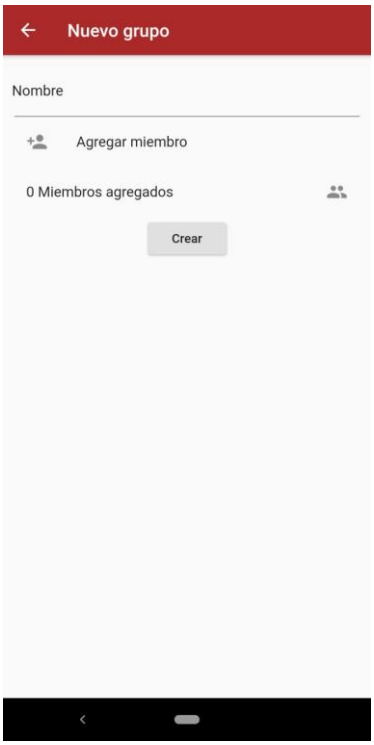

Ilustración 46: Pantalla de creación de un nuevo grupo

4.1.7 Notificaciones

En la aplicación se han contemplado tres tipos de notificaciones: cambios en la información de un evento, nuevos mensajes en el chat y la asignación de una tarea al usuario. Las notificaciones se mostrarán en la bandeja de entrada del sistema cuando la aplicación se encuentre en segundo plano o cerrada; mientras que en primer plano solo las notificaciones de información entrarán a la sección de *Notificaciones*.

En dicha sección se presenta una pantalla con las notificaciones de información en una lista con las más recientes en primer lugar. Cada notificación, encabezada por el nombre del evento al que pertenece, contiene el tipo de cambio que se ha realizado junto con el organizador que lo ha ejecutado.

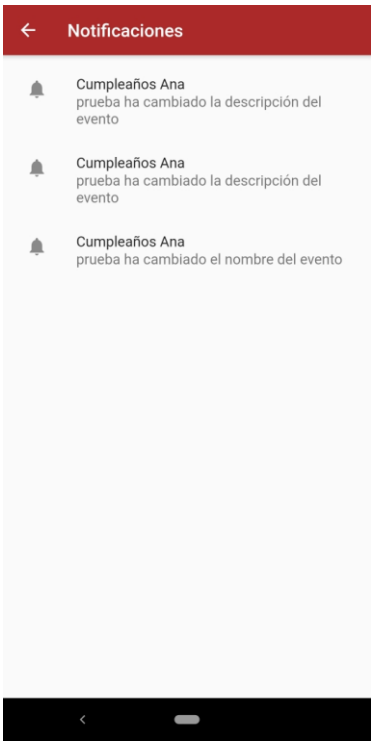

Ilustración 47: Pantalla de notificaciones

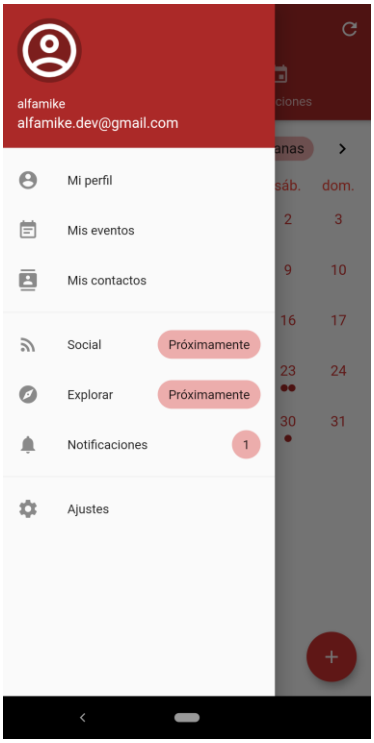

Ilustración 48: Indicador del número de notificaciones sin leer

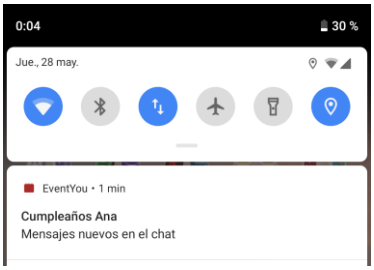

Ilustración 49: Notificación de nuevos mensajes en el chat

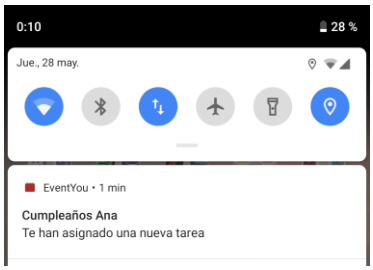

Ilustración 50: Notificación de asignación de tarea

4.1.8 Ajustes

La sección de ajustes está compuesta por los siguientes elementos:

- Tres interruptores para activar o desactivar la recepción de notificaciones de cada uno de los tipos disponibles.

- Una pantalla que muestra todas las licencias de los paquetes de terceros utilizados en el desarrollo.
- La versión de la aplicación instalada en el dispositivo.

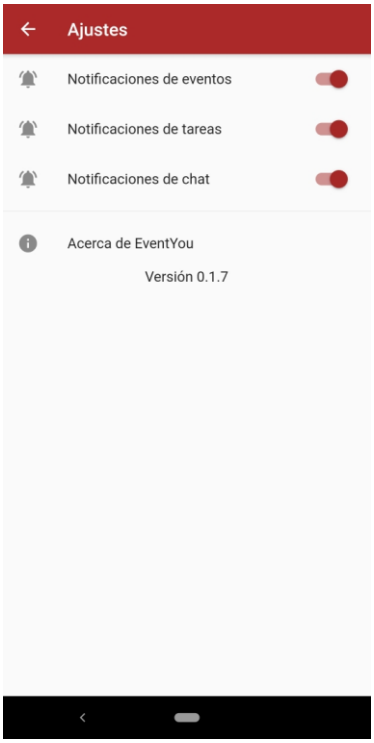

Ilustración 51: Pantalla de la sección de ajustes

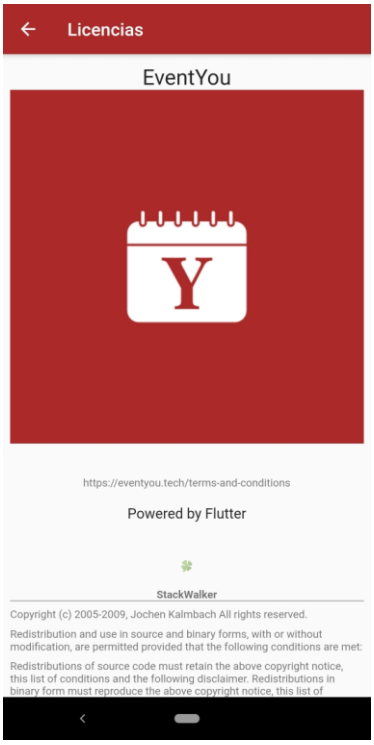

Ilustración 52: Pantalla de licencias

Pruebas

Durante el desarrollo de la aplicación, así como una vez terminado, se han utilizado mecanismos de pruebas con el fin de comprobar el correcto funcionamiento de todos los elementos tanto visuales como de lógica. Además de cubrir la necesidad de verificar los flujos de interacción que se espera que tengan los usuarios para encontrar fallos. Para este fin se hacía imprescindible contar con usuarios reales que tuvieran el rol de *tester,* tanto en Android como en iOS.

Los mecanismos de distribución de las versiones que se iban liberando han sido dos diferentes. Durante casi todo el desarrollo se utilizó el producto de Firebase denominado *App Distribution* y en la fase final se utilizaron los canales oficiales de Google y Apple para dicho propósito.

Para el uso del primer mecanismo es necesario que el *tester* tenga cuenta de Google mientras que, en el segundo, en el caso del canal oficial de Google también y en el caso de Apple es necesario tener un id de Apple.

El producto *App Distribution* permite gestionar la distribución, a *testers* individuales o grupos, de la aplicación de forma manual mediante enlaces de invitación o de forma automática. El sistema automático se encarga de enviar correos electrónicos con las instrucciones para participar en las pruebas y de notificar a los participantes cuando se libera una nueva versión. Además, gracias a la integración con los otros productos de Firebase de analíticas y *crashlytics* es posible detectar errores en versiones concretas, lo que redunda en un seguimiento muy detallado de cómo se están comportando las versiones.

En Android se puede instalar la aplicación, habilitando el permiso para orígenes desconocidos, de forma manual o más cómodamente mediante la aplicación auxiliar "App Tester" de Firebase, que es distribuida *ad hoc* mediante el correo con las instrucciones. Mientras que en iOS hay que utilizar dicha aplicación auxiliar de forma obligatoria.

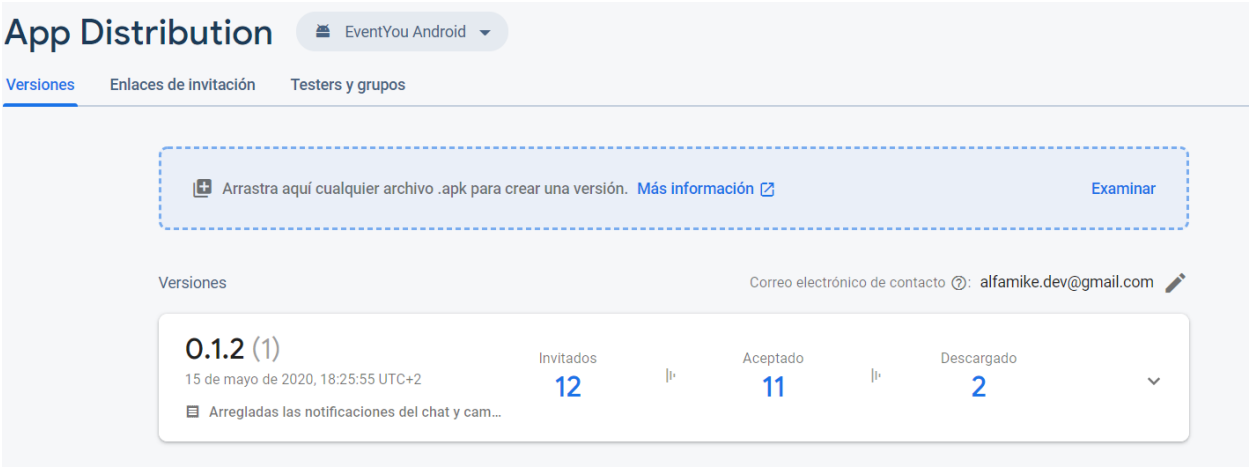

Ilustración 53: Panel de distribución de Firebase para Android

Ilustración 54: Panel de distribución de Firebase para iOS

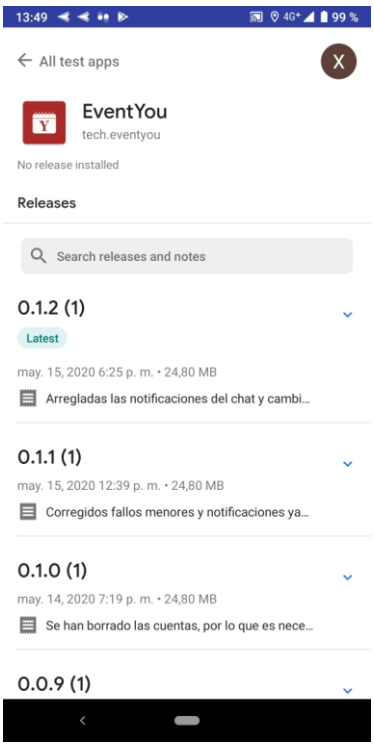

Ilustración 55: Aplicación "App Tester" de Firebase

El mecanismo comentado aporta agilidad en la distribución, así como información muy valiosa. No obstante, aunque la interfaz es intuitiva y no tiene complejidad, tiene un enfoque destinado a personas con cierto manejo del tema. Por este motivo, y dada la necesidad de hacer llegar la aplicación de la forma más sencilla y familiar posible a todo el mundo decidí cambiar a los mecanismos de los canales oficiales.

Los canales oficiales, Google Play Console y App Store Connect, ofrecen todas las funcionalidades de distribución con el añadido de pasar las revisiones oficiales de estas plataformas. Esto supone aumentar los plazos, pues hay que esperar a dichas revisiones, y rellenar toda la documentación oficial para cada plataforma.

Para Android se ha creado un segmento cerrado de pruebas denominado “TFG” que permite a las cuentas de Google indicadas acceder a la ficha y por lo tanto a la aplicación mediante Google Play Store. Una gran ventaja es la optimización de la aplicación para las diferentes arquitecturas móviles.

Ilustración 56: Panel de gestión de versiones en Google Play Console

Para iOS se ha utilizado la distribución mediante TestFlight. Se ha creado un grupo de *testers* denominado "TFG" que recibe las notificaciones de las nuevas compilaciones. La instalación de la aplicación se realiza al introducir un código en la aplicación TestFlight de Apple, disponible en la App Store.

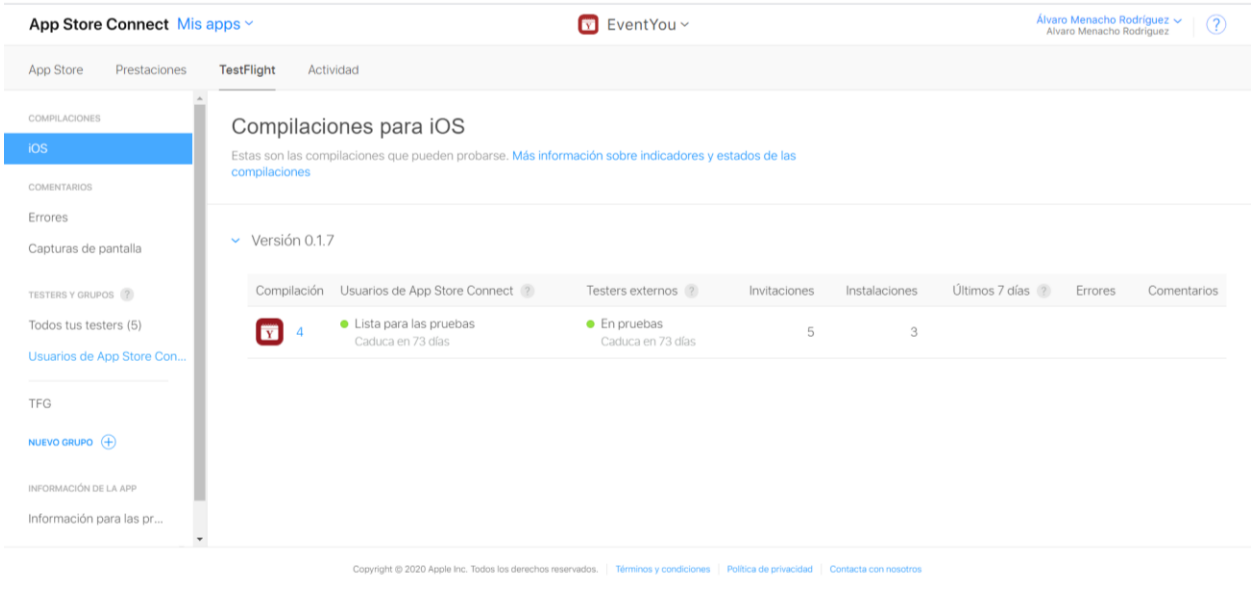

Ilustración 57: Panel de gestión de versiones en TestFlight

Ilustración 58: Aplicación TestFlight

5 CONCLUSIONES

El diseño y desarrollo de una aplicación móvil es todo un reto técnico que suele llevar meses incluso para equipos de desarrollo profesionales. Poner en funcionamiento cada una de las partes, como son el frontend, el backend o el modelo de datos suele llevar un trabajo intenso y complicado, ya que todo debe interactuar e ir de la mano.

Plasmar en una aplicación las ideas que se tienen en mente y conseguir que funcione es una tarea árdua. De hecho, con frecuencia es necesario adaptarse a las circunstancias y a los inconvenientes técnicos. Por ello se hace más crítico si cabe elegir adecuadamente el framework y el lenguaje utilizado; para que la transición a la realidad sea lo más indolora posible.

A nivel técnico ha sido todo un reto desarrollar esta aplicación desde cero en cuatro meses utilizando un framework nuevo y desconocido para mí. No obstante, estoy muy satisfecho con el resultado obtenido y con haber alcanzado todos los objetivos que me propuse. He realizado un trabajo muy motivante y que era entretenido, además de aportarme mucho técnicamente.

Estoy muy contento con la aplicación desarrollada. Creo que es intuitiva y tiene cabida en el amplio mercado de los eventos sociales. Por lo que el desarrollo realizado supone una base perfecta para el lanzamiento en los próximos meses.

En cuanto al framework utilizado, Flutter ha demostrado ser un entorno de desarrollo rápido y eficiente. Permite desarrollar la interfaz gráfica en tiempo récord y con elementos renderizados al nivel de código nativo que consiguen un rendimiento increíble en todas las plataformas. La versatilidad para construir tus propios widgets es inmensa y el lenguaje Dart está perfectamente diseñado para acometer esta tarea. La cada vez más amplia comunidad de desarrolladores de Flutter está aportando cada mes nuevos paquetes que complementan la más escasa parte de backend y lógica de las plataformas; y ampliando la infinidad de widgets personalizables.

Desde mi punto de vista la adopción de este framework puede ser exponencial con el lanzamiento del sistema operativo Fuchsia de Google y sobretodo por tener detrás a este gigante tecnológico. Han conseguido crear un ecosistema que puede romper el mercado al desplegar aplicaciones para todas las plataformas a partir de un único código base y con el rendimiento comentado anteriormente.

Por otro lado, Firebase y en general Google Cloud han demostrado ser un backend versátil y completo. Todos los productos son fáciles de utilizar, son escalables y permiten implementar de forma rápida elementos esenciales para cualquier aplicación, ahorrando al desarrollador meses de desarrollo.

Como líneas de actuación futuras se podría desplegar la aplicación como una aplicación web progresiva que complementara las aplicaciones móviles e incluso probar las versiones de escritorio. Además, la aplicación permite incorporar nuevas funcionalidades en todas sus partes, por lo que las posibilidades son casi infinitas.

REFERENCIAS

[1] «Stadista,» 2020. [En línea]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Último acceso: Mayo 2020].

[2] Comisión Nacional de los Mercados y la Competencia, «Panel de Hogares CNMC,» CNMCData, 2019.

[3] Google Developers, «Youtube,» 2015. [En línea]. Available: https://www.youtube.com/watch?v=PnIWl33YMwA. [Último acceso: Mayo 2020].

[4] Google Developers, «Google Developers,» 2018. [En línea]. Available: https://developers.google.com/events/flutter-live. [Último acceso: Mayo 2020].

[5] «Linkedin Learning,» 2019. [En línea]. Available: https://learning.linkedin.com/blog/tech-tips/the-fastest-growing-skills-among-software-engineers--and-how-to-. [Último acceso: Mayo 2020].

[6] «flutter.dev,» 2020. [En línea]. Available: https://flutter.dev/docs/resources/technical-overview. [Último acceso: Mayo 2020].

[7] Alibaba Tech, «Medium,» 2018. [En línea]. Available: https://medium.com/hackernoon/making-the-most-of-flutter-from-basics-to-customization-433171581d01. [Último acceso: Junio 2020].

[8] «dart.dev,» 2020. [En línea]. Available: https://dart.dev/guides/language/specifications/DartLangSpec-v2.2.pdf. [Último acceso: Mayo 2020].

[9] «Firebase Open Source,» 2020. [En línea]. Available: https://firebaseopensource.com/. [Último acceso: Mayo 2020].

[10] «FlutterFire,» 2020. [En línea]. Available: https://github.com/FirebaseExtended/flutterfire. [Último acceso: Mayo 2020].

[11] «Plugin Auth,» 2020. [En línea]. Available: https://pub.dev/packages/firebase_auth. [Último acceso: Mayo 2020].

[12] «Plugin Firestore,» 2020. [En línea]. Available: https://pub.dev/packages/cloud_firestore. [Último acceso: Mayo 2020].

[13] «Plugin Storage,» 2020. [En línea]. Available: https://pub.dev/packages/firebase_storage. [Último acceso: Mayo 2020].

[14] «Cloud Functions,» 2020. [En línea]. Available: https://firebase.google.com/products/functions?hl=es-419. [Último acceso: Mayo 2020].

[15] «Plugin Analytics,» 2020. [En línea]. Available: https://pub.dev/packages/firebase_analytics. [Último acceso: Mayo 2020].

[16] «Plugin Crashlytics,» 2020. [En línea]. Available: https://pub.dev/packages/firebase_crashlytics. [Último acceso: Mayo 2020].

[17] «Plugin Dynamic Links,» 2020. [En línea]. Available: https://pub.dev/packages/firebase_dynamic_links. [Último acceso: Mayo 2020].

[18] «Firebase Cloud Messaging,» 2020. [En línea]. Available: https://firebase.google.com/products/cloud-messaging?hl=es-419. [Último acceso: Mayo 2020].

[19] «Plugin Messaging,» 2020. [En línea]. Available: https://pub.dev/packages/firebase_messaging. [Último acceso: Mayo 2020].

[20] «Firebase InApp Messaging,» 2020. [En línea]. Available: https://firebase.google.com/products/in-app-messaging?hl=es-419. [Último acceso: Mayo 2020].

[21] «Plugin InApp Messaging,» 2020. [En línea]. Available: https://pub.dev/packages/firebase_in_app_messaging. [Último acceso: Mayo 2020].

[22] «Firebase Extensions,» 2020. [En línea]. Available: https://firebase.google.com/products/extensions?hl=es-419. [Último acceso: Mayo 2020].

GLOSARIO

CNMC: Comisión Nacional de Mercado y Competencia

AOT Compilation: Ahead-of-time compilation

ECMA: European Manufacturers Association

SDK: Software Development Kit

Anexo A: Modelo de datos

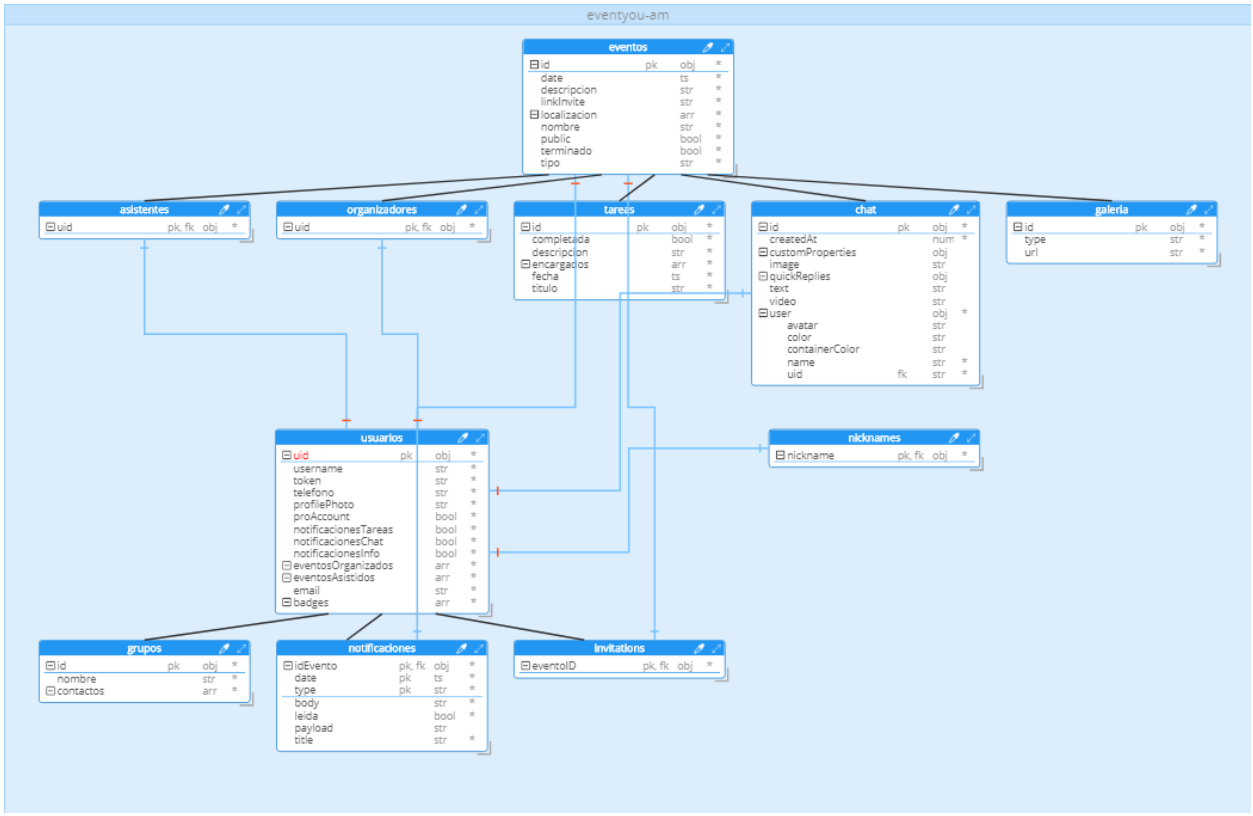

Ilustración 59: Modelo de datos de EventYou

Anexo B: Funciones en Firebase Cloud Functions

```
 const functions = require('firebase-functions');
 const admin = require('firebase-admin');

 admin.initializeApp();

 exports.notificationTrigger = functions.firestore.document('usuarios/
{usuario}/notificaciones/{notificacion}')
   .onCreate((snapshot, context) => {
     var notificacion = snapshot.data();
     var token = '';
     var payload = {
       "notification": {
         "title": notificacion.title,
         "body": notificacion.body,
         "sound": "default"
       },
       "data": {
         "idEvento": notificacion.idEvento,
       }
     }

     snapshot.ref.parent.parent.get().then((docUsuario) => {
       var usuario = docUsuario.data();

       switch (notificacion.type) {
         case 'notificationType.info': {
           if (usuario.notificacionesInfo) {
             token = usuario.token;
           }
           break;
         }
         case 'notificationType.chat': {
           if (usuario.notificacionesChat) {
             token = usuario.token;
           }
           break;
         }
         case 'notificationType.tareas': {
           if (usuario.notificacionesTareas) {
             token = usuario.token;
           }
           break;
         }
         default:
           break;
       }

       if (token === '') {
         return console.log('No se envía notificación al no estar acti
vada');
       } else {
         return admin.messaging().sendToDevice(String(token), payload)
;
       }
```

```
    }).catch((err) => {
      console.log(err);
    })
    console.log('Función completada');
    return 0;
  })
```

Código 1: Función para enviar notificaciones

```
 exports.deleteUserFromGroups =
functions.firestore.document('usuarios/{usuario}').onDelete(
  (snapshot, context) => {
    var usuario = snapshot.data();

    let queryGrupos = admin.firestore().collectionGroup('grupos');
    queryGrupos = queryGrupos.where('contactos', 'array-contains',
usuario.uid);
    return queryGrupos.get().then(querySnapshot => {
      return querySnapshot.forEach(documentSnapshot => {
        var field = documentSnapshot.get('contactos');
        for (i = 0; i < field.length; i++) {
          if (field[i] === usuario.uid) {
            field.splice(i, 1);
            documentSnapshot.ref.update({ 'contactos': field });
          }
        }
        return console.log(`User removed from groups`);
      });
    });
  }
 )
```

Código 2: Función para eliminar al usuario de los grupos

```
 exports.deleteUserFromAssistants =
functions.firestore.document('usuarios/{usuario}').onDelete(
  (snapshot, context) => {
    var usuario = snapshot.data();

    let queryAsistentes =
admin.firestore().collectionGroup('asistentes');
    queryAsistentes = queryAsistentes.where('uid', '==',
usuario.uid);
    return queryAsistentes.get().then(querySnapshot => {
      return querySnapshot.forEach(documentSnapshot => {
        documentSnapshot.ref.delete();
        return console.log('User removed from assistants');
      })
    })
  }
 )
```

Código 3: Función para eliminar al usuario de los eventos a los que asiste

```
 exports.deleteUserFromOrganizers =
functions.firestore.document('usuarios/{usuario}').onDelete(
    (snapshot, context) => {
      var usuario = snapshot.data();

      let queryOrganizadores =
admin.firestore().collectionGroup('organizadores');
      queryOrganizadores = queryOrganizadores.where('uid', '==',
usuario.uid);
      return queryOrganizadores.get().then(querySnapshot => {
        return querySnapshot.forEach(documentSnapshot => {
          documentSnapshot.ref.delete();
          return console.log('User removed from organizers');
        })
      })
    }
 )
```

Código 4: Función para eliminar al usuario de los eventos organizados

```
 exports.deleteUserFromTasks =
functions.firestore.document('usuarios/{usuario}').onDelete(
    (snapshot, context) => {
      var usuario = snapshot.data();

      let queryTareas = admin.firestore().collectionGroup('tareas');
      queryTareas = queryTareas.where('encargados', 'array-contains',
usuario.uid);
      return queryTareas.get().then(querySnapshot => {
        return querySnapshot.forEach(documentSnapshot => {
          var field = documentSnapshot.get('encargados');
          for (i = 0; i < field.length; i++) {
            if (field[i] === usuario.uid) {
              field.splice(i, 1);
              documentSnapshot.ref.update({ 'encargados': field });
            }
          }
          return console.log(`User removed from tasks`);
        });
      });
    }
 )
```

Código 5: Función para eliminar al usuario de las tareas asignadas