Quedadas

Nombre Estudiante: Juan José Franco Peñaranda
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a:
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

06 de Junio de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Quedadas*
Nombre del autor:	*Juan José Franco Peñaranda*
Nombre del consultor/a:	*Eduard Martín Lineros*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*Castellano*
Palabras clave	*Asistencia a eventos, redes sociales, mensajería instantánea*

Resumen del Trabajo (máximo 250 palabras):

El objetivo de este proyecto es facilitar tanto la gestión de asistentes a pequeños eventos privados, como la de inscribirse a dichos eventos. Además de hacer uso de los grupos de Whatsapp/Telegram/Hangout, listas de correo electrónico y/o redes sociales de las que ya disponga el creador del evento para difundirlo. La solución facilitará a los usuarios que reciban la invitación a un evento confirmar su asistencia y ser notificados de cualquier modificación en el evento.

Para ello se va a usar una metodología en cascada, que en este caso creo que es la más apropiada ya que disponemos de los requisitos e hitos cerrados de antemano y no tenemos de cliente que nos indique los requisitos, ni equipo que necesitemos gestionar.

El resultado será una aplicación móvil con dos funciones principales, crear eventos y difundirlos en grupos de Whatsapp/Telegram/Hangout, listas de correo electrónico y/o en redes sociales y por otro lado, los usuarios que reciban dichas invitaciones podrán confirmar su asistencia.

Además vamos a enfatizar la importancia de que esta aplicación sea evaluada por usuarios reales cuyo feedback a su vez vuelva a alimentar la aplicación y con ello la experiencia de usuario (ux).

Abstract (in English, 250 words or less):

The goal of this project is to make easier the management of attendees at small private events, as well as to register for such events. In addition to making use of WhatsApp/Telegram/Hangout groups, email lists and/or social networks that the creator

of the event already has to spread it. This solution will make easier for users who receive the invitation to an event to confirm their attendance and be notified of any changes to the event.

For this, a waterfall methodology will be used, which in this case I think is the most appropriate since we already have the requirements and milestones closed in advance, and we do not have a client asking the requirements or a team to manage.

The result will be a mobile application with two main functions, create events and spread them in WhatsApp/Telegram/Hangout groups, email lists and/or social networks, and on the other hand, users who receive such invitations can confirm their attendance.

We will also emphasize the importance of this application being evaluated by real users whose feedback in turn feeds the application and, with it, the user experience (UX).

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

La popularización de las redes sociales como Facebook y Twitter y aplicaciones de mensajería instantánea (IMS) como Whatsapp, Telegram, Hangout o Skype entre otras han cambiado la forma en la que nos comunicamos. Pocos años atrás la forma principal de comunicarnos se basaba en comunicaciones uno a uno vía llamadas de voz o SMS, y aunque ya existían plataformas como el correo electrónico y plataformas de IMS su uso no era tan frecuente como lo es ahora y estaba restringido a grupos de usuarios avanzados y profesionales del sector TIC.

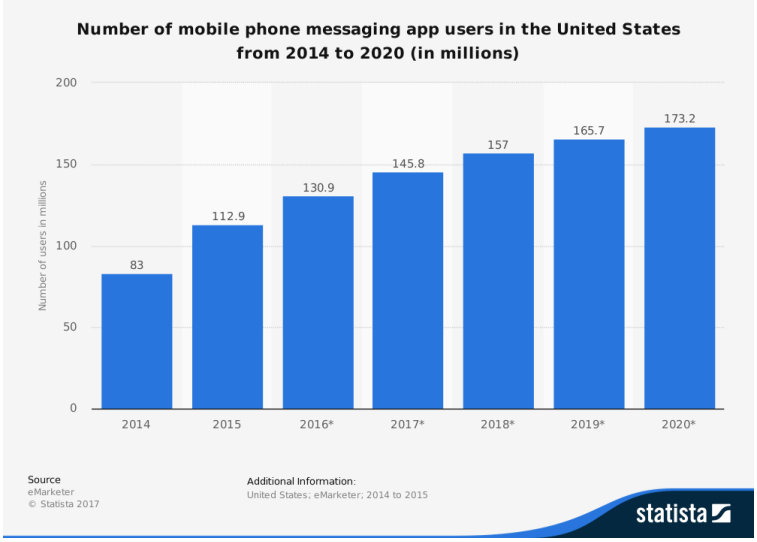


Ilustración 1 - Número de aplicaciones de mensajería móvil en EEUU

Uno de los cambios más notables es la comunicación en multidifusión, que básicamente consiste en suscribir a un grupo de personas a un mismo canal, ya sea un grupo de tu plataforma favorita de IMS, una lista de correo electrónico o mediante una red social.

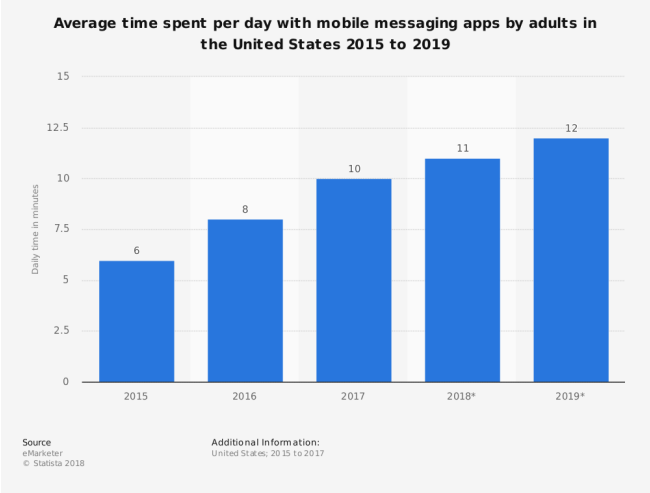


Ilustración 2 - Tiempo diario empleado en aplicaciones de mensajería móvil en EEUU

La forma de comunicación en multidifusión puede verse de forma especial en los grupos de Whatsapp, que han proliferado de manera exponencial en los últimos años, y aunque ofrecen una solución sencilla, rápida y fácil de usar para transmitir un mensaje a un grupo concreto de personas, también han provocado el efecto "ruido" en el canal, donde el exceso de mensajes provocan que este pierda parte de su utilidad, concretamente la de tener informados a sus participantes.

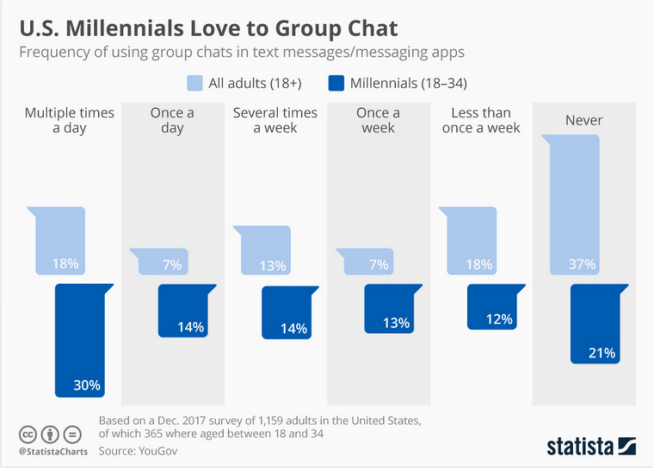


Ilustración 3 - Uso de grupos en apps de mensajería móvil en EEUU

Un uso habitual de la comunicación en multidifusión es la de crear un grupo relacionado con un evento que se está organizando, como por ejemplo una fiesta, una excursión o una reunión y usar este grupo como canal de difusión e información del evento, la forma de confirmar la asistencia suele ser pedir a los participantes del grupo que confirmen o no su asistencia al evento, pero esta información queda rápidamente oculta entre mensajes de "ruido" generados por los propios participantes además de los propios mensajes de cancelación de usuarios que previamente habían confirmado su asistencia y las solicitudes de información sobre el evento. En poco tiempo la tarea de controlar la asistencia al evento por parte del administrador del grupo se vuelve ardua y compleja.

Con este proyecto, al que he denominado Quedadas**,** quiero facilitar la tarea de controlar la asistencia a un evento que se está difundiendo mediante grupos en plataformas de IMS, redes sociales o listas de correo, para ello se desarrollará una aplicación que permita crear eventos, publicarlos en listas de correo, redes sociales y plataformas de mensajería instantánea y a su vez facilite a los usuarios de estos canales acceder a la información del evento y confirmar o no su asistencia de forma rápida y sencilla. Además proveerá al administrador del evento la información relativa a la asistencia y le mantendrá informado de los cambios en esta mediante un sistema de notificaciones. Con esta aplicación el creador del evento podrá fijar el mensaje del evento en el canal que esté utilizando (Twitter, grupo de Whatsapp o Telegram, etc..) para que siempre esté visible a los usuarios y estos podrán acceder desde dicho mensaje a la información del evento y gestionar su asistencia.

Un caso de uso de la aplicación sería poder publicar la fiesta de cumpleaños de tu hijo o hija en el grupo de Whatsapp de su clase. Hoy en día son famosos los grupos de Whatsapp que agrupan a los padres y madres de los compañeros de clase de tu hijo o hija, estos grupos tienen tendencia a ser muy ruidosos, con mensajes cruzados entre padres y madres intercambiando información sobre sus hijos, los profesores, los deberes y otros menesteres. Con Quedadas podría publicarse el cumpleaños como un mensaje de Whatsapp, fijar dicho mensaje durante los días previos al cumpleaños y desde dicho mensaje los padres que tuviesen la aplicación podrían acceder a la información del cumpleaños y confirmar la asistencia de su hijo o hija.

Hay otros muchos casos de uso donde Quedadas puede facilitar la gestión de asistentes a un evento, como en los foros para concertar una actividad deportiva como pádel, futbol, etc.., publicar una charla en Twitter y tener una idea aproximada de los asistentes a la charla. También sería útil para gestionar las plazas disponibles en un vehículo para un trayecto que se realiza con regularidad o para un viaje esporádico publicando el viaje en el canal apropiado.

Actualmente existen aplicaciones y servicios que ofrecen una solución a ciertas de las problemáticas descritas anteriormente, y en algunos casos de forma similar a la que propongo con Quedadas, o no resuelven por completo la problemática que resuelve Quedadas o lo hacen de una forma más difícil para el usuario o a costa de mantener otras funcionalidades que no se van a usar. Entre ellas destacan:

Facebook Events:

Ilustración 4 - App de Facebook Events

Web: https://events.fb.com/#why-facebook-events

Como nos indican en [Tech crunch 2018] esta herramienta de facebook está enfocada a la creación de eventos y su difusión, haciendo los eventos públicos y buscando maximizar su difusión, pero no provee un mecanismo sencillo para gestionar la asistencia. Además requiere bastante trabajo para crear eventos y mantenerlos, muchas descripciones, imágenes y opciones. Quedadas por otro lado está pensado para controlar la asistencia a eventos de particulares, facilitando el proceso de crear el evento y de administrar la asistencia a dicho evento al usuario, para que a este le sea muy rápido crear un evento y compartirlo con sus amigos. Otra diferencia importante con Facebook Events, es que los listados de eventos no son públicos, sólo puedes ver aquellos a los que has tenido acceso a su invitación. Una de las funcionalidades que potencian Quedadas y que no tiene Facebook Events es el sistema de notificaciones cada vez que un usuario cambia su asistencia a un evento, que es vital para que el creador del evento esté al tanto en tiempo real de la asistencia al evento. Por otro lado en el servicio de Facebook los eventos están limitados a usuarios de Facebook, que aunque quizás no sea un gran problema, en Quedadas no quiero estar sujeto a una red social concreta, quero que se pueda acceder a la información sin necesidad de tener una cuenta en Facebook.

MeetUp:

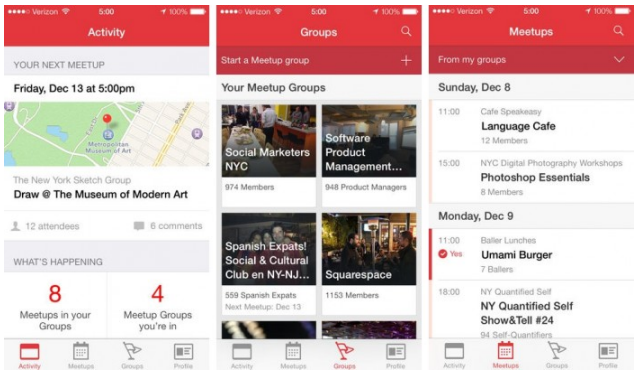


Ilustración 5 - App Meetup

Web: https://secure.meetup.com/es-ES

El servicio de MeetUp resuelve la gestión de asistencias a un evento de forma parecida a como quiero hacerlo en Quedadas, permitiendo saber al administrador los asistentes de dicho evento. En Top Apps podemos encontrar un resumen bastante completo sobre esta aplicación [Top Apps 2018]. Pero la idea de Quedadas difiere en el enfoque que se hace de los eventos. Por el lado de MeetUp el grupo se crea alrededor de la temática de los eventos, es decir, se crea un grupo de una temática concreta, como por ejemplo aprender un idioma, se describe en qué consistirá la actividad de dicho grupo y luego se van proponiendo eventos dentro del grupo creado. En Quedadas tengo un enfoque distinto, ya que persigo un objetivo distinto, quiero partir de los grupos que las personas ya se han creado en Whatsapp, Telegram, etc.… y de las comunidades a las que pertenecen en las distintas redes sociales y que puedan usar estas para proponer eventos, tenga o no que ver con la temática del grupo o comunidad. La finalidad de Quedadas no es encontrar desconocidos para crear un evento, sino crear eventos dentro de un grupo de gente conocida. Este planteamiento no deja a gente desconocida para el creador del evento fuera de dicho evento, ya que un miembro del grupo puede compartir el evento sus propios grupos. Lo que busca Quedadas en este aspecto y lo diferencia de MeetUp es que las personas que acceden al evento tengan un vínculo, directo o indirecto con el creador del evento, para evitar la asistencia de personas de las cuales no se tenga ninguna referencia. Por ejemplo, en lugar de crear un Meetup para organizar un partido de Padel, en Quedadas se crea un evento para jugar al Pádel y se comparte por los grupos o redes sociales en los que el creador del evento sabe que puede haber interés en asistir al evento, como por ejemplo grupos de familia, amigos o de trabajo. Otra diferencia importante con MeetUp, es que los listados de eventos no son públicos, sólo puedes ver aquellos a los que has tenido acceso a su invitación.

Google Calendar:

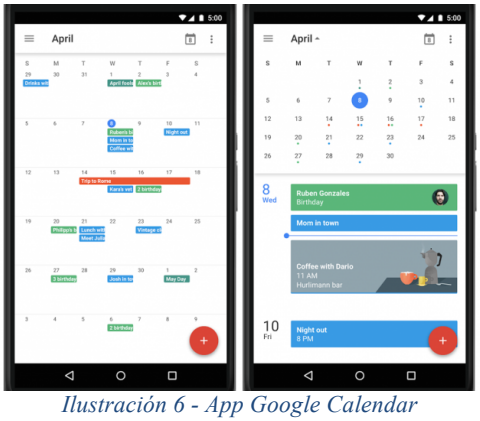


Ilustración 6 - App Google Calendar

Web: https://calendar.google.com/calendar/

Con Google Calendar podríamos dar respuesta a las mismas necesidades que damos con Quedadas, además de otras muchas funcionalidades, podemos crear eventos, publicarlos en nuestros grupos y redes sociales, y controlar la asistencia de a ellos. Pero es complejo de usar, requiere que los usuarios tengan una cuenta de Gmail y las notificaciones se realizan mediante email. Podemos encontrar un articulo sobre esta aplicación en The Next Web [The Next Web 2018]. En Quedadas voy mejorar todos estos aspectos, creando una aplicación especializada en gestionar la asistencia a eventos, con una interfaz de usuario intuitiva y fácil de usar, que permita acceder con cuentas de Gmail,

Facebook y Twitter y las notificaciones se reciban en el centro de notificaciones del dispositivo móvil.

Eventbrite:

Ilustración 7 - App Eventbrite

Web: https://www.eventbrite.es/

El servicio de Eventbrite es el mismo caso que Facebook Events, pero más focalizado en la venta de entradas para los eventos, y por tanto deja sin cubrir los mismos casos que este. Aunque da la opción de crear eventos gratuitos y privados, el enfoque de Quedadas centrado en resolver la gestión de asistencia a eventos privados proporciona una mejor experiencia de usuario debido a necesitar un trabajo de mantenimiento de los eventos muy inferior al de Eventbrite. Un artículo sobre Eventbrite con mas detalles sobre su funcionamiento se puede consultar en la web de Yalantis [Yalantis 2018].

Mobilize:

Ilustración 8 - App Mobilize

Web: https://mobilize.io/

Este servicio que resuelve de forma interesante la gestión de asistencia a un evento queda dentro del mismo tipo de soluciones que MeetUp, más centrado en crear una comunidad alrededor de una temática o temáticas relacionadas, y no ofrecen el enfoque de Quedadas de basarse en los grupos o comunidades a las que ya pertenece el usuario creador del evento.

Los casos mostrados anteriormente son una recopilación de las aplicaciones disponibles para solventar el problema propuesto en el proyecto, aunque hay muchos mas servicios

de los documentados, creo que estos son los principales y además en mi opinión son una muestra representativa del tipo de funcionalidades ofrecidas por este tipo de servicios y donde Quedadas puede tener su razón de ser.

1.2 Objetivos del Trabajo

El objetivo de este proyecto es crear un servicio compuesto por una aplicación móvil que permita:

a) Crear eventos, que contienen información como la fecha de inicio del evento, su duración, su ubicación, el precio si lo tuviese y el máximo de participantes si procede.
b) Duplicar un evento existente, introduciendo simplemente su nueva fecha, para facilitar la gestión de eventos periódicos.
c) Permite publicar un evento en cualquier red social, plataforma de mensajería instantánea o lista de correo y abrir la aplicación con la información del evento desde dicha publicación.
d) Permite al creador del evento modificar la fecha o ubicación del evento y este cambio será notificado a todos los usuarios que hayan confirmado en el evento para que puedan cambiar su asistencia si procede.
e) La aplicación notifica al creador del evento cada vez que un usuario de la aplicación confirma su asistencia o la cancela.
f) Permite al creador del evento enviar una notificación de recordatorio a los usuarios confirmados en el evento.
g) Permite al creador cancelar un evento y todos los participantes serán notificados.
h) Los eventos solo serán accesibles para aquellos usuarios que tengan acceso al mensaje del evento.
i) Permite a un usuario de la aplicación ver la información del evento, calcular la ruta al evento y confirmar o cancelar su asistencia al evento en cualquier momento.
j) Permite a un usuario de la aplicación ver los asistentes al evento.

Dicho servicio también contará con un backend, que básicamente consistirá en un CRUD (Create, Read, Update, Delete) para gestionar los eventos creados. Las funciones del backend serán:

a) Almacenar los eventos creados desde el dispositivo móvil
b) Recuperar los eventos
c) Modificar los eventos
d) Eliminar los eventos

Para comunicar la aplicación móvil con el backend se dotará al servicio de un API Restful que será capaz de realizar las siguientes funciones:

a) Petición para recuperar los eventos de un usuario
b) Petición para recuperar los eventos en los que participa un usuario
c) Petición para modificar un evento
d) Petición para eliminar un evento
e) Petición para modificar la asistencia de un usuario a un evento

1.3 Enfoque y método seguido

Como he dicho en el punto anterior he decidido desarrollar un producto nuevo, frente a la opción de adaptar un producto existente porque los productos disponibles actualmente en le mercado se centran en dos ámbitos distintos al que quiero abordar con mi proyecto.

Del conjunto de productos analizados, una parte de ellos se centran en la difusión del evento y su finalidad es la de conseguir el máximo de audiencia posible, dejando como funcionalidad secundaria la de gestionar la asistencia al evento. El resto de los productos analizados centran su estrategia en crear una comunidad alrededor del evento, siendo muy complicado o imposible crear eventos privados sin tener una temática asociada al evento.

Para el desarrollo del proyecto se va a usar una metodología en cascada, que en este caso creo que es la más apropiada ya que disponemos de los requisitos e hitos cerrados de antemano y no tenemos un cliente que nos indique los requisitos, ni equipo que necesitemos gestionar. Distribuiremos las fases acordes con los hitos marcados por cada PEC y cada uno de estos hitos tendrá asociado un entregable que servirá como validador del progreso del proyecto y en el caso del último hito el entregable serán los documentos y aplicaciones acordados en el proyecto.

Siguiendo el enfoque en cascada, el proyecto contendrá una fase de análisis y diseño de la cual obtendremos los requisitos de la solución, para ello me valdré de la metodología de diseño centrado en el usuario (DCU) haciendo uso de escenarios de uso, encuestas y una evaluación heurística. Tras la fase de diseño habrá una fase de implementación, en la que desarrollaremos el prototipo final, por último, en la fase de pruebas realizaremos los casos de test diseñados para validar el prototipo final, así como una evaluación con usuarios finales. Tras una última fase donde aplicaremos los cambios y/o arreglos necesarios sobre el prototipo final este se convertirá en el producto final que entregaremos a la finalización del proyecto.

En el proceso de implementación del producto nos vamos a preocupar también por la gestión de la configuración del software y para ello se usarán técnicas de integración continua, habilitando un repositorio en Bitbucket para el versionado del código, este repositorio se enlazará a un servicio de Jenkins que se encargará de la generación de un nuevo APK tras cada actualización del repositorio. Como soy el único desarrollador del proyecto no aplicaré ningún flujo especifico de git, simplemente trabajaré sobre una rama de desarrollo y haré las publicaciones en la rama master, que será la encargada de notificar a Jenkins que genere un nuevo APK. Cada nuevo APK será publicado por Jenkins de forma automática en el Play Store bajo la modalidad de aplicación BETA cerrada.

Además, vamos a incorporar, como innovación metodológica, valorar la importancia de la evaluación realizada con usuarios reales, a través de evaluación heurística con expertos de usabilidad y realización de cuestionarios.

1.4 Planificación del Trabajo

Esta sección especifica los recursos necesarios para la consecución del proyecto y la planificación de este, dicha planificación se ha elaborado usando una tabla de hitos y tareas para dar una visión cuantitativa de las tareas a realizar, los hitos que deben lograrse y los entregables asociados a cada hito. Por último se ha elaborado un diagrama de Gantt para poder llevar a cabo un seguimiento temporal del estado del proyecto.

1.4.1 Recursos

Esta tabla contiene los recursos y la disponibilidad de estos.

Recursos		
Cantidad	Descripción	Disponibilidad
1	Programador	L,M,X,J:1.5h/V,S,D: 3h
1	Analista	L,M,X,J:1.5h/V,S,D: 3h
1	Ordenador MAC con OSX 10 o posterior	Total
1	Licencia para publicar en Android	De por vida
1	Servidor virtual en la nube para alojar el backend	1 año
1	Dispositivo móvil Android	Total
1	Conexión a Internet	Total

Ilustración 9 - Tabla de recursos

1.4.2 Tabla de días festivos

Esta tabla representa los días festivos que están dentro de la planificación del proyecto y los días que por otros motivos tampoco se llevará a cabo ninguna tarea planificada. La última columna indica las horas no trabajadas del proyecto, que ya han sido restadas de la planificación.

Días Festivos		
19/3/2018	San José (Nacional)	1.5 horas
29/3/2018	Jueves Santo (Nacional)	1.5 horas
30/3/2018	Viernes Santo (Nacional)	3 horas
3/4/2018	Bando de la huerta (Regional)	1.5 horas
1/5/2018	Día del trabajador (Nacional)	1.5 horas
10/6/2018	Día de la Región de Murcia (Regional)	0 horas
Días		
05/05	Compromiso familiar	3 horas

Ilustración 10 - Tabla días festivos

1.4.3 Tabla de hitos y tareas

Estas tablas contienen las tareas de cada hito, su estimación en horas y el entregable asociado al hito correspondiente.

1.4.3.1 Hito PEC1

Tabla del primer hito del proyecto, que consiste en la justificación y planificación del proyecto.

Nombre	Duración	Inicio	Final
PEC1- Plan de trabajo	21 d / 45 h	22/Febrero	14/Marzo
Tareas			Duración (h)
Seleccionar TFM			12
Analizar soluciones existentes			3
Planificar TFM			19.5
Redactar Memoria			10.5
Entregable	Capítulo 1 TFM		

Ilustración 11 - Tabla Hito PEC1

1.4.3.3 Hito PEC2

Tabla del segundo hito, que recopila las tareas asociadas al hito de análisis y diseño del proyecto.

Nombre	Duración	Inicio	Final
PEC2 - Diseño	17 d / 29 h	15/Marzo	4/Abril
Tareas			Duración (h)
Análisis etnográfico de los escenarios con usuarios			3
Crear modelo conceptual			2
Primer prototipo de la aplicación móvil			4
Evaluación prototipo con usuarios			3
Prototipo final			3
Análisis y diseño de la arquitectura del servicio			2
Análisis y diseño del backend			2
Análisis y diseño del API			2
Análisis y diseño de la aplicación móvil			2
Integrar modelo conceptual, prototipo, análisis y diseños en la memoria del TFM			6
Entregable	DCU, Escenarios de uso y Arquitectura (integrados en el TFM)		

Ilustración 12 - Tabla hito PEC2

1.4.3.4 Hito PEC3

Esta tabla representa el hito de implementación de la solución propuesta en el proyecto.

Nombre	Duración	Inicio	Final
PEC3 - Implementación	40 d. / 85.5 h.	5/Abril	16/Mayo
Tareas			Duración (h)
[INVESTIGACIÓN] Plugins sociales ionic 3			8
[INVESTIGACIÓN] Deep Link en ionic 3			8
[BACKEND] Implementar Landing page del backend			1.5
[BACKEND] Implementar CRUD			9
[BACKEND] Documentar CRUD			3
[API] Implementar crear evento			1
[API] Implementar listar eventos			0.75
[API] Implementar cancelar evento			0.75
[API] Implementar duplicar evento			0.75
[API] documentar gestión de eventos			5
[APP] Implementar login app móvil			3.5
[APP] Implementar crear evento			4
[APP] Implementar listado de quedadas			5
[APP] Implementar detalle de un evento			4
[APP] Implementar cancelar evento			1.5
[APP] Implementar duplicar evento			3
[APP] Implementar compartir evento mediante uri			5
[APP] Documentar gestión eventos			3
[API] Implementar confirmar/cancelar asistencia			0.75
[API] documentar cambio de asistencia			1
[APP] Implementar abrir evento desde una uri			2.5
[APP] Implementar cambiar asistencia a evento			1.5
[APP] Documentar gestión eventos			3
[TEST] Integración y e2e			7
[TEST] Pruebas de usabilidad			3
Entregable	Prototipo, Pruebas y documentación integrada en la memoria del TFM		

Ilustración 13 - Tabla hito PEC3

1.4.3.5 Hito PEC4

Esta tabla contiene la estimación del último hito con tareas del proyecto y están dedicadas a la corrección de errores y mejoras en la implementación, preparación de la memoria final y la creación de la presentación.

Nombre	Duración	Inicio	Final
PEC4- Entrega final	21 d / 45 h	17/Mayo	6/Junio
Tareas			Duración (h)
Tareas pendientes, feedback de usuarios y corrección de errores del hito anterior			15
Terminar memoria y manual de usuario			18
Hacer la presentación			12
Entregable	Producto, Memoria, Manual de usuario y Presentación		

Ilustración 14 - Tabla hito PEC4

1.4.3.6 Hito Tribunal

Este hito no contiene tareas y no se puede estimar en horas, refleja la disponibilidad requerida por el tribunal para la resolución de las preguntas que crean oportunas.

Nombre	Duración	Inicio	Final
Tribunal	5 días	18/Junio	22/Junio
Tareas			Duración (d)
Responder preguntas del tribunal			5
Entregable	Respuestas al tribunal		

Ilustración 15 - Tabla hito Tribunal

1.4.4 Diagrama de Gantt

Para obtener una visión en el tiempo de la planificación del proyecto he realizado un diagrama de Gantt. Al ser el único involucrado en el proyecto todas las tareas se realizarán en serie, salvo las de pruebas de usabilidad, donde participarán usuarios de la aplicación que harán dichas pruebas de usabilidad.

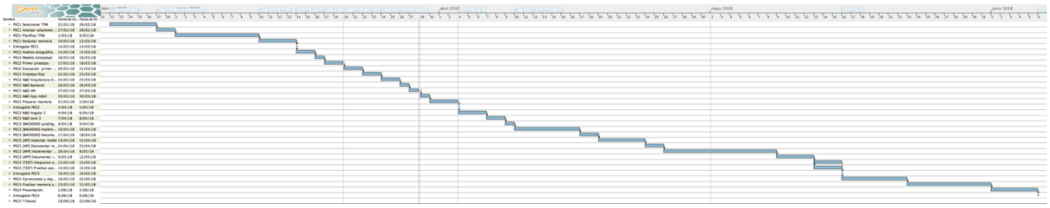
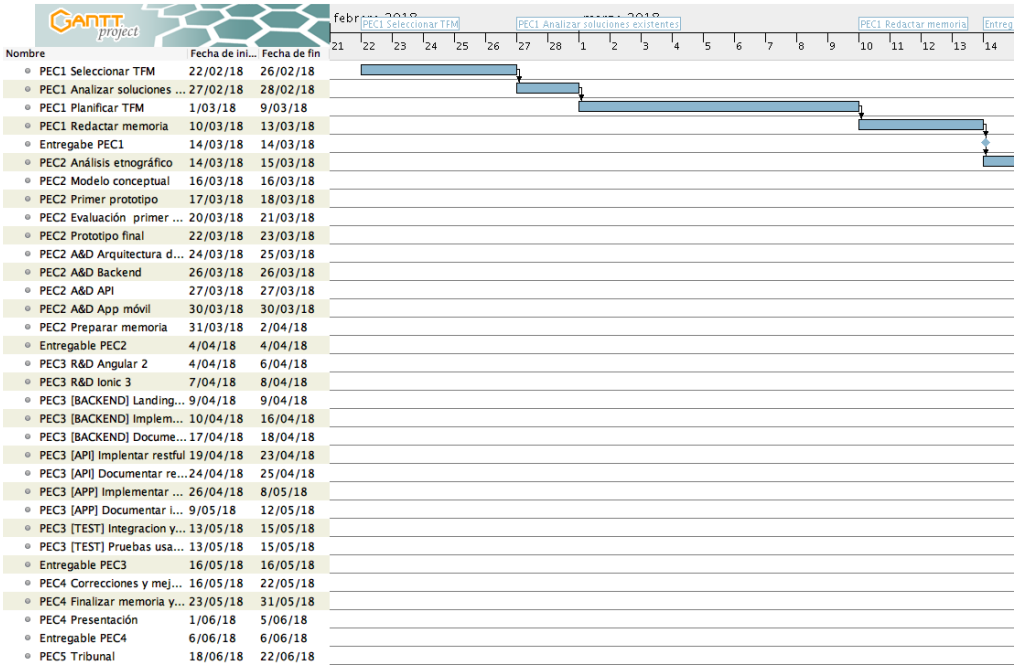
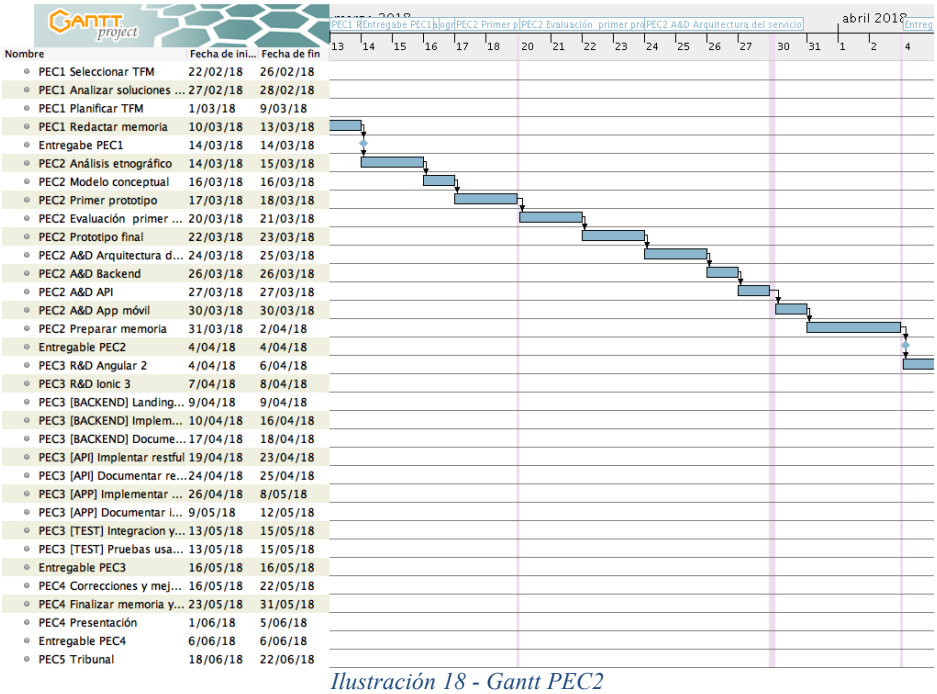


Ilustración 16 - Gantt completo



Nombre	Fecha de ini...	Fecha de fin
PEC1 Seleccionar TFM	22/02/18	26/02/18
PEC1 Analizar soluciones ...	27/02/18	28/02/18
PEC1 Planificar TFM	1/03/18	9/03/18
PEC1 Redactar memoria	10/03/18	13/03/18
Entregabe PEC1	14/03/18	14/03/18
PEC2 Análisis etnográfico	14/03/18	15/03/18
PEC2 Modelo conceptual	16/03/18	16/03/18
PEC2 Primer prototipo	17/03/18	18/03/18
PEC2 Evaluación primer ...	20/03/18	21/03/18
PEC2 Prototipo final	22/03/18	23/03/18
PEC2 A&D Arquitectura d...	24/03/18	25/03/18
PEC2 A&D Backend	26/03/18	26/03/18
PEC2 A&D API	27/03/18	27/03/18
PEC2 A&D App móvil	30/03/18	30/03/18
PEC2 Preparar memoria	31/03/18	2/04/18
Entregable PEC2	4/04/18	4/04/18
PEC3 R&D Angular 2	4/04/18	6/04/18
PEC3 R&D Ionic 3	7/04/18	8/04/18
PEC3 [BACKEND] Landing...	9/04/18	9/04/18
PEC3 [BACKEND] Implem...	10/04/18	16/04/18
PEC3 [BACKEND] Docume...	17/04/18	18/04/18
PEC3 [API] Implentar restful	19/04/18	23/04/18
PEC3 [API] Documentar re...	24/04/18	25/04/18
PEC3 [APP] Implementar ...	26/04/18	8/05/18
PEC3 [APP] Documentar i...	9/05/18	12/05/18
PEC3 [TEST] Integracion y...	13/05/18	15/05/18
PEC3 [TEST] Pruebas usa...	13/05/18	15/05/18
Entregable PEC3	16/05/18	16/05/18
PEC4 Correcciones y mej...	16/05/18	22/05/18
PEC4 Finalizar memoria y...	23/05/18	31/05/18
PEC4 Presentación	1/06/18	5/06/18
Entregable PEC4	6/06/18	6/06/18
PEC5 Tribunal	18/06/18	22/06/18

Ilustración 17 - Gantt PEC1



Nombre	Fecha de ini...	Fecha de fin
PEC1 Seleccionar TFM	22/02/18	26/02/18
PEC1 Analizar soluciones ...	27/02/18	28/02/18
PEC1 Planificar TFM	1/03/18	9/03/18
PEC1 Redactar memoria	10/03/18	13/03/18
Entregabe PEC1	14/03/18	14/03/18
PEC2 Análisis etnográfico	14/03/18	15/03/18
PEC2 Modelo conceptual	16/03/18	16/03/18
PEC2 Primer prototipo	17/03/18	18/03/18
PEC2 Evaluación primer ...	20/03/18	21/03/18
PEC2 Prototipo final	22/03/18	23/03/18
PEC2 A&D Arquitectura d...	24/03/18	25/03/18
PEC2 A&D Backend	26/03/18	26/03/18
PEC2 A&D API	27/03/18	27/03/18
PEC2 A&D App móvil	30/03/18	30/03/18
PEC2 Preparar memoria	31/03/18	2/04/18
Entregable PEC2	4/04/18	4/04/18
PEC3 R&D Angular 2	4/04/18	6/04/18
PEC3 R&D Ionic 3	7/04/18	8/04/18
PEC3 [BACKEND] Landing...	9/04/18	9/04/18
PEC3 [BACKEND] Implem...	10/04/18	16/04/18
PEC3 [BACKEND] Docume...	17/04/18	18/04/18
PEC3 [API] Implentar restful	19/04/18	23/04/18
PEC3 [API] Documentar re...	24/04/18	25/04/18
PEC3 [APP] Implementar ...	26/04/18	8/05/18
PEC3 [APP] Documentar i...	9/05/18	12/05/18
PEC3 [TEST] Integracion y...	13/05/18	15/05/18
PEC3 [TEST] Pruebas usa...	13/05/18	15/05/18
Entregable PEC3	16/05/18	16/05/18
PEC4 Correcciones y mej...	16/05/18	22/05/18
PEC4 Finalizar memoria y...	23/05/18	31/05/18
PEC4 Presentación	1/06/18	5/06/18
Entregable PEC4	6/06/18	6/06/18
PEC5 Tribunal	18/06/18	22/06/18

Ilustración 18 - Gantt PEC2

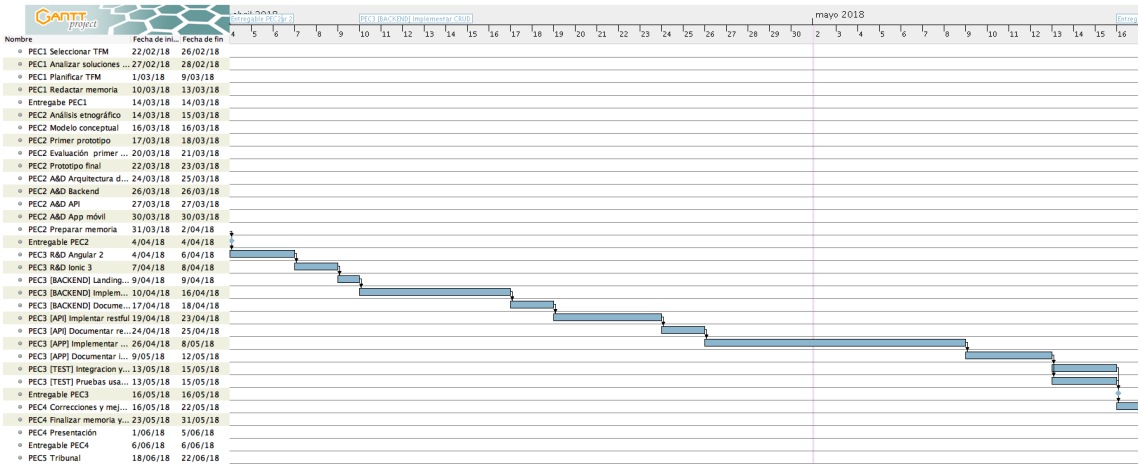


Ilustración 19 - Gantt PEC3

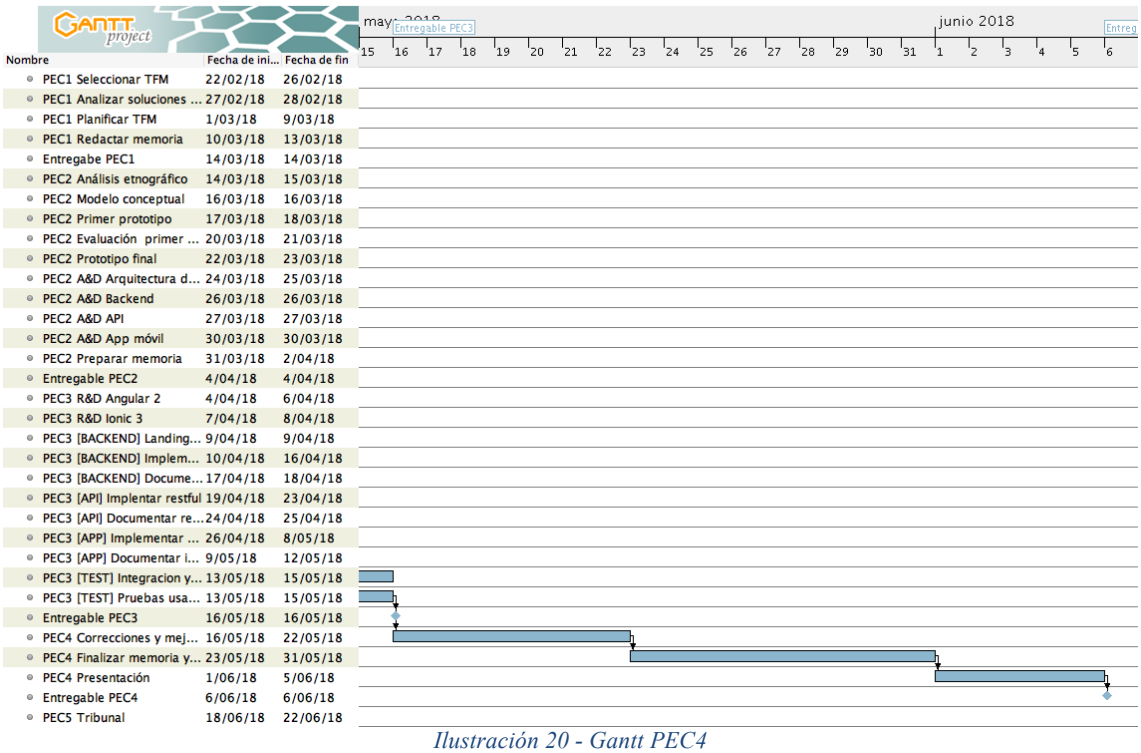


Ilustración 20 - Gantt PEC4

1.5 Breve sumario de productos obtenidos

Producto
Aplicación para dispositivos Android
Descripción

Aplicación para dispositivos Android que nos permite publicar eventos en servicios de mensajería instantánea, redes sociales y/o listas de correo y gestionar la asistencia a estos.

Ilustración 21 - Tabla producto app móvil

Producto
Backend
Descripción
Backend para almacenar los eventos generados por los usuarios de la aplicación móvil. Mantendrá el estado de cada evento y permitirá su modificación y eliminación.

Ilustración 22 - Tabla producto backend

Producto
API Restful
Descripción
Interfaz de comunicación entre la aplicación móvil y el backend, nos permitirá gestionar el backend desde la aplicación móvil o desde cualquier otro cliente que la implemente.

Ilustración 23 - Tabla producto API

1.6 Breve descripción de los otros capítulos de la memoria

Capitulo 2: Fase de Diseño. En este apartado se desarrollarán los usuarios y escenarios donde se usará la aplicación. Se creará un modelo conceptual en forma de prototipo de baja definición, este prototipo será revisado por un grupo de usuarios para obtener sus impresiones. Con las impresiones obtenidas por los usuarios se obtendrá un prototipo en Justinmind que será evaluado por un experto en usabilidad para obtener su evaluación heurística. Este apartado estará compuesto por los puntos:

2.1 Analizar usuarios y escenarios de uso
2.2 Modelo conceptual
2.3 Evaluación con usuarios
2.4 Prototipo no funcional

Capitulo 3: Arquitectura de la aplicación. En este apartado se hará un análisis de la arquitectura técnica de la solución y su posterior diseño. También se analizarán y diseñaran las arquitecturas de cada uno de los productos que componen la solución usando diagramas de componentes. Este apartado estará compuesto por los puntos:

3.1 Análisis y diseño de la arquitectura del servicio
3.2 Análisis y diseño del backend
3.3 Análisis y diseño del API
3.4 Análisis y diseño de la aplicación móvil

Capitulo 4: Implementación. Es este apartado se justificará la elección de las herramientas y tecnología utilizadas en la implementación de la solución. Se

comentarán los principales patrones de diseño utilizados en cada producto, así como el código mas representativo de cada uno. Por último, se mostrará el prototipo funcional obtenido. Este apartado estará compuesto por los puntos:

4.1 Herramientas y tecnología
4.2 Backend y API
4.3 Aplicación móvil

Capitulo 5: Pruebas. Este apartado contendrá los casos de test diseñados para la validación de la solución y su resultado. También contendrá la encuesta realizada a un grupo de usuarios para su posterior evaluación.

Capitulo 6: Conclusiones. Este apartado desarrollará las conclusiones del proyecto y abordará los posibles trabajos futuros.

Capitulo 7: Glosario. Glosario de términos usados a lo largo de la memoria del proyecto.

Capitulo 8: Bibliografía. Bibliografía consultada para la realización del proyecto.

2. Análisis y diseño

2.1 Análisis de escenarios de uso

Para obtener las funcionalidades que debe proporcionar la aplicación para ser una herramienta realmente útil he realizado una serie de entrevistas con potenciales usuarios de la aplicación. Durante cada entrevista he obtenido un escenario de cada usuario en el que la aplicación podría haber facilitado la tarea de planificar y controlar la asistencia a un evento.

2.1.1 Escenario 1

Nombre	Juan José Franco
Edad	37 años
Profesión	Desarrollo de software
Descripción del usuario	
Juanjo está casado y tiene dos hijos. Se dedica al desarrollo de aplicaciones móviles en la empresa Centic, que no trabaja los viernes por la tarde. Es un enamorado del desarrollo y le encanta probar tecnologías nuevas. Sus deportes favoritos son montar en bicicleta de montaña y hacer escalada. En su tiempo libre trata de ir al cine y a charlas sobre tecnología con sus amigos.	
Descripción del escenario	
Miércoles a las 19.00h, Juanjo acaba de salir del trabajo, recibe una mensaje de su mujer en el que le dice que su hija tiene un cumpleaños el viernes por la tarde, que pase a comprar un regalo para la amiga de su hija. Cuando llega a casa planifica con su mujer la tarde del próximo viernes. Su hijo pequeño va a estar con su madrina en el zoo, su hija en un cumpleaños y su mujer quiere aprovechar para ir a la peluquería. Cuando han terminado de organizarse, Juanjo decide que el viernes puede aprovechar para hacer una salida en bicicleta de montaña, abre su grupo de WhatsApp de "MTB Fighters" y deja un mensaje indicando que va a salir el viernes a las 16.00h desde la casa forestal del coto "Los Cuadros", a los pocos minutos recibe un mensaje de Miguel que se apunta a la salida. Por la noche recuerda que su cuñado le había dicho que le gustaría salir algún día con él, así que vuelve a escribir el mismo mensaje en el Telegram de su cuñado, este le responde que irá, pero que no sabe donde está la casa forestal. Juanjo envía una ubicación de la posición donde han quedado a su cuñado por Telegram. El jueves por la mañana Juanjo tiene 18 mensajes del grupo "MTB Fighters", precios de los nuevos cambios de shimano, algunas fotos de rutas MTB de la zona de los pirineos y un mensaje de Rafa diciendo que el también se apunta. Al salir el jueves del trabajo, Juanjo ve que tiene 30 mensajes en el grupo de "MTB Fighters", Ángel y Pedro también se han apuntado a la salida del viernes y Rafa propone subir a las antenas por la ruta del calvario, pero Ángel dice que después de las últimas lluvias el sendero está en muy mal estado. El viernes por la mañana el grupo de "MTB Fighters" tiene 150 mensajes, ayer salieron	

los nuevos modelos de suspensiones de aire/aceite de Rox y ha habido bastante discrepancia entre su precio y su calidad en el grupo. El viernes a las 16.00h Juanjo está en el punto acordado, Tras esperar cinco minutos aparecen Ángel y Pedro, a los pocos minutos aparece su cuñado. Tras esperar quince minutos decide llamar a Rafa, que no ha aparecido todavía, Rafa dice que dijo en el grupo que no podría venir, pero con tanto mensaje Juanjo no se dio cuenta. Tras la llamada deciden comenzar la ruta. Cuando llevan 10 minutos de ruta Juanjo recibe una llamada de Alberto, diciendo que está en la casa forestal y que no hay nadie, Juanjo no había visto su mensaje de WhatsApp diciendo que se apuntaba pero que llegaría 20 minutos mas tarde, que por favor le esperasen. El grupo se da la vuelta para recoger a Alberto. Varias semanas después Juanjo vuelve a disponer de una tarde de viernes y decide volver a planificar la salida en bicicleta y vuelve a iniciar todo el proceso.

Ilustración 24 - Tabla escenario 1

2.1.2 Escenario 2

Nombre	Francesco Rivola
Edad	34 años
Profesión	Desarrollo de Software
Descripción del usuario	
Francesco está casado y tiene dos hijas. Se dedica al desarrollo de aplicaciones web y la coordinación de equipos en la empresa Xepient Solutions. En su tiempo libre le gusta probar nuevas tecnologías, pasar tiempo con su familia y viajar.	
Descripción del escenario	
Pronto es el cumpleaños de su hija mayor y Francesco se encarga de organizar su fiesta de cumpleaños. Una vez que ha decidido el sitio y el día con su mujer, empieza a contactar con todos los invitados. Para ello envía un mensaje de WhatsApp con la información del evento al grupo de los padres de la clase de colegio de su hija, al grupo de la familia (abuelos, tíos y primos) y finalmente a un último grupo de amigos de la guardería. En breve empieza a recibir confirmaciones de varios de los grupos, algunos irán, otros agradecen la invitación, pero no van a poder asistir y otros no saben todavía si podrán ir. Pasada una semana Francesco tiene que llamar al sitio del cumpleaños para dar el número aproximado de personas que irán a la fiesta. Con una libreta en la mano, empieza por el grupo de familia donde rápidamente cuenta la gente que irá. No es tan fácil en los demás grupos ya que hay mucha más gente y durante la semana han habido muchos más mensajes sobre el colegio, no relacionados con la fiesta de cumpleaños. Poco a poco Francesco va leyendo el historial de mensajes y va apuntando quien irá. Finalmente, para los indecisos decide enviar un nuevo mensaje a cada grupo. Al final del día vuelve a revisar los grupos y finalmente en la libreta tiene la lista casi definitiva de la gente que irá y con ella llama al sitio de celebraciones. Unos días antes del gran día, Francesco decide enviar otro mensaje a cada grupo de recordatorio de la fiesta y para volver a dejar la dirección y el horario de la fiesta.	

El día de la fiesta Francesco recibe algún mensaje privado de algunos padres volviendo a preguntar por el lugar del sitio ya que no vieron el recordatorio debido al gran numero de mensajes del grupo del colegio.

Ilustración 25 - Tabla escenario 2

2.1.3 Escenario 3

Nombre	Joaquín Lasheras
Edad	39 años
Profesión	Profesor de Universidad en informática
Descripción del usuario	
Joaquín está casado y con una hija, se dedica a la informática y además imparte como profesor asociado asignaturas en la Universidad	
Descripción del escenario	
Febrero de 2018, el próximo 10 de marzo me entero de que tengo la boda de un buen amigo y me ofrezco a organizar su despedida de soltero. El primer paso es preguntar al novio por la lista de gente que quiere que le de la información acerca de su despedida. El novio me pasa una lista inicial (y me avisa que más adelante me pasa el resto). Lo primero que hago es crear un grupo de WhatsApp con todos en común donde preguntar por las fechas que a cada uno le viene mejor y pedir sugerencias. Finalmente se decide hacer dos despedidas, una en Granada y otra en Murcia a la semana siguiente (el novio es de Murcia y la mayoría de los invitados también, que además suelen ser padres por lo que no pueden viajar). Echo esto, necesito crear otro grupo de WhatsApp solo para los de granada. En el de granada resulta que hay algunos que todavía están dudosos, pero yo ya debo reservar espacio, por lo que decido reservar apartamento para 6 (incluido novio). Todo va encajando, 6 van a Granada y unos 15 a Murcia. De repente el novio me pasa el número de dos personas más que no estaban avisadas, tengo que meterlo primero al grupo de Murcia y volver a repetirle toda la información y luego se resulta que se apuntan a granada, con lo cual al final somos 7 en Granada y ya tenemos problemas de alojamiento. Me pongo a buscar alternativas, y justo unos días antes de irme resulta que entre todos los mensajes chorras que mandábamos al grupo, resulta que el 7 en cuestión me dijo que era duda y a falta de 2 días me dice que no viene, con lo cual he perdido el tiempo buscando alternativa de alojamiento. A la vuelta de Granada, resulta que todos los que vienen de Granada menos yo deciden que con una despedida sobra, pero eso no me lo dicen también hasta última hora, con lo cual me obligan a cambiar la actividad que teníamos preparada (paintball) a algo donde podamos ir menos gente y además me obliga a cambiar la cena de 15 a 10 personas. Y a todo esto debo de interpretar los mensajes de cada uno a través de grupos de WhatsApp llenos de fotos y comentarios que me hacen perderme. PD: al final las despedidas salieron bien, pero gracias a montón de tiempo que me costó a mi estar llamando a cada una de las personas a pesar de que teníamos el grupo de WhatsApp.	

Ilustración 26 - Tabla escenario 3

2.1.4 Escenario 4

Nombre	Julián Serrano
Edad	38 años
Profesión	Desarrollo de Software

Descripción del usuario
Julián pertenece al dpto. de Sistemas de Información de una empresa de servicios. Compagina su actividad profesional con la participación en el circuito profesional de tenis a nivel de clubs; además de jugar de forma muy asidua al Pádel.
Descripción del escenario
Semana del 19 al 25 de marzo. Comienza una nueva semana y han de planificarse todos los entrenamientos del equipo absoluto de tenis del Olimpic club de Murcia. Julián como subcapitan participa muy activamente en esta tarea. El equipo está formado por 8 miembros, cada uno con unas particularidades muy dispares a la hora de planificar y afrontar los entrenamientos de la semana (lesiones, preparación para torneo individual...) además se da la circunstancia muy habitual de invitar jugadores externos para entrenar en las instalaciones del club. El modus operandi es disponer de unas pistas en unos horarios concretos valorando; posibles inclemencias del tiempo, si hay eliminatoria próxima, respetar dejar pistas libres para el uso por parte de abonados del club etc. Esta tarea está encargada de hacerla el capitán. Una vez disponemos de los horarios y pistas disponibles de cada día de la semana; el capitán y subcapitan se encargarán de asignar a los jugadores a cada pista respetando la disponibilidad y necesidades de cada uno. Esto acaba convirtiéndose en un autentico caos; ya que el capitán y subcapitan mandan a un grupo de mensajería instantánea el cuadro, automáticamente reciben una respuesta por el mismo medio de un jugador queriendo cambiar su turno, otro de los jugadores hace una llamada por teléfono al subcapitan pero no al capitán para indicar que le vendría mejor el horario del Jueves pero puede adaptarse al del Viernes; de repente, esta semana va a entrenar con nosotros un miembro del Murcia Club de Tenis, pero no se le mete en el grupo porque es algo puntal, pero esto implica que no esté al corriente de las modificaciones en los cuadros y las observaciones de los miembros del equipo... la realidad es que el flujo de información desorganizado, la falta de la centralización de un punto de información, unido al poco tiempo de los responsables para gestionar todo esto, hace que sea bastante caótico llevar esta tarea a cabo y acaba por ocasionar que se van rellenando las pistas día a día. Al final el capitán y subcapitan hablan por teléfono y en el grupo de mensajería de forma diaria y varias veces para ir cerrando pistas un día para otro. De forma paralela entre los socios y miembros del equipo plantean partidos de Pádel a modo de diversión, normalmente esto se plantea de nuevo en un grupo de mensajería en el que Miguel lanza la cuestión, se necesitan 3 palas (3 jugadores) para el martes a las 8. Casi inmediatamente Juan contesta, yo me apunto, ya sólo quedan 2. Dos horas después Miguel dice que no quedaban pistas a las 8 y la ha cogido a las 9, ya no sabe si Juan que a las 8 podía a las 9 va a poder mantener su candidatura. La única forma de saberlo es llamarlo y preguntarle; Juan dice que no porque tiene que recoger a su hijo que sale de Karate a las 10 y no le da tiempo ya. Miguel vuelve a poner en el grupo 3 palas para Martes a las 9, José dice que el si va, vuelven a ser 2; Fede, también puede, y pregunta cuantos faltan, a lo que Miguel responde que 1; pero Pedro se

confunde con la extensa conversación en la app y que lee varias horas después de la contestación de Miguel a Fede de que quedaba aun una plaza y lo que recuerda es que Juan decía que jugaba por lo que no se apunta pensando que no hay plazas. Víctor que sí lo había leído dice que se apunta. Esto no lo lee Fede que quedando poco tiempo para el partido pregunta si falta alguien, pero Miguel que es la persona que realmente debe saber cuantos van está en una reunión y no contesta; Fede por su parte se pone a buscar en sus grupos una persona para jugar ya que supone que aun falta 1 jugador. Al final aparecen 5 personas en la pista de pádel y uno de ellos se queda sin jugar.

Ilustración 27 - Tabla escenario 4

2.1.5 Escenario 5

Nombre	Laura Franco
Edad	32 años
Profesión	iOS developer
Descripción del usuario	
Laura se dedica al desarrollo de aplicaciones móviles para iPhone, iPad y Apple Watch. En su tiempo libre le gusta ir al cine y pasear si el día está soleado.	
Descripción del escenario	
Marzo 2018. Están a punto de estrenar uno de esos blockbusters que tanto gustan a Laura. Abre el grupo de WhatsApp para tal fin y comunica fecha y hora del estreno, así como una breve sinopsis de la película y la valoración que tiene en varias páginas de crítica de cine. El grupo comienza a intercambiar mensajes sobre la película y en poco tiempo se acumula una gran cantidad de mensajes entre los que se esconden los mensajes de confirmación de asistencia. Laura se encarga de comprar las entradas para que todos puedan sentarse juntos en el cine y para ello comienza a revisar todos los mensajes del grupo para saber cuantas entradas tiene que comprar.	

Ilustración 28 - Tabla escenario 5

2.1.6 Escenario 6

Nombre	Rocío Morales
Edad	36 años
Profesión	Gestora de contenidos y marketing digital
Descripción del usuario	
Se dedica a la redacción de contenidos digitales en Bilnea donde trabaja de lunes a viernes. Vive con su pareja, pero le gusta mantener el contacto con sus amigos y suele hacer planes con ellos los fines de semana cuando tiene tiempo libre ya que los horarios no coinciden con los de su pareja. Le encanta viajar y la decoración; disfruta paseando por el monte y con una buena conversación	
Descripción del escenario	
Rocío tiene días de vacaciones previstos para ir a FITUR como cada año y quiere visitar la feria con sus amigos. Ha comprado el billete con mucha antelación y cuando quedan pocos días plantea a su grupo de amigos dedicados al Turismo ir juntos.	

El lunes a las 12:00h Rocío envía un mensaje de WhatsApp al grupo de 'Turistas' para recordarles que el jueves a las 10:00h estará en la puerta Sur de IFEMA lista para realizar el recorrido de la feria dirección Norte con la intención de conocer primero las novedades Internacionales y acabar en la zona española hacia la hora de comer. A la 13:00h tiene 25 mensajes, Iván y Antonio aceptan, estarán allí a esa hora, suben juntos, ya tienen reservado el apartamento y los pases para los tres. A las 14:00h Rocío tiene 55 mensajes, Manu también irá, vive fuera, pero ha encontrado un buen vuelo para unirse al grupo a la hora propuesta.
A las 15:45h Rocío tiene en el grupo 95 mensajes. Antonio e Iván viajarán en coche porque no ya quedan trenes, conversan sobre los fallos del año pasado y que este no se les podía hacer muy tarde para llegar a los stands de España, importante coger cita para las degustaciones en La Rioja.
A las 16:30h Rocío tiene 24 nuevos mensajes, Daza comenta que también irá pero que ella se aloja con su familia.
El martes a las 9:00h Rocío recuerda que su amiga Isabel se había interesado por la feria meses antes, por lo que decide mencionarle el plan por Hangouts.
A las 10:00h recibe 5 mensajes, confirma que irá pero que no sabe llegar hasta IFEMA.
A las 18:10h Rocío le envía un email a Isabel con la ubicación y las indicaciones exactas explicándole que por correo las tendrá a mano ya que en Hangouts la conversación del chat se borra.
El miércoles a las 21:00h Rocío tiene 103 mensajes nuevos, comentan que la cadena Hotusa realiza una cena de gala el jueves noche para sus clientes y no pueden faltar. Antonio explica que ya ha gestionado las invitaciones y añade la ubicación del alojamiento, está mejor situado que el del año pasado, pero es más viejo y caro, hay opiniones de todos.
El jueves a las 10:00h Rocío está en el punto acordado. Tras esperar varios minutos Rocío envía un mensaje a Manu para preguntarle, este le recuerda que avisó en el grupo 'Turistas' de que su avión se retrasaba una hora y se encontraría con el grupo dentro a las 11h, se vería con todos en Japón, donde quería recoger información para su próximo viaje. A los pocos minutos llegan Iván y Antonio.
Daza llama a Rocío preguntándole qué le queda, resulta que está esperando a todos en la puerta Norte, escribió en el grupo de WhatsApp que por favor cambiaran el sentido de la visita, ella va a la feria por trabajo y tiene reuniones concertadas, así es más productiva su jornada. Rocío recuerda que se debatió sobre el cambio de puerta en el grupo, pero no se llegó a confirmar finalmente.
Ahora los amigos, 20 minutos más tarde se desplazan desde el punto de salida para encontrarse con Daza y empezar la ruta a la inversa. Rocío no sabe nada de Isabel, suponía que no sería puntual y no la espera en el origen, le envía un Telegram cuando llegan a la puerta Norte para citarla directamente en Japón a las 11:30h donde se encontrarán todos con Manu, este stand se ha convertido en el punto real de partida de la ruta juntos. Isabel confirma a Rocío su llegada a las 11:30 por Telegram.
El próximo año volverán a planificar el recorrido juntos a la feria.

Ilustración 29 - Tabla escenario 6

2.1.7 Funcionalidades

De los escenarios analizados en los puntos anteriores he extraído los requisitos funcionales que debe cumplir la aplicación móvil para cubrir dichos escenarios.

	Historias de usuario	Escenarios
1	Como usuario de la aplicación móvil quiero poder crear eventos que contengan la fecha de inicio del evento, su duración, su ubicación y el máximo de participantes si procede.	1,2,3,4,5,6
2	Como usuario de la aplicación móvil quiero poder duplicar un evento existente introduciendo simplemente una nueva fecha	1,3,4,6
3	Como usuario de la aplicación móvil quiero publicar un evento en mis grupos de WhatsApp y Telegram y poder abrir la aplicación con la información del evento desde dicha publicación.	1,2,3,4,5,6
4	Como usuario de la aplicación móvil quiero ver la información del evento calcular la ruta al evento y confirmar o cancelar mi asistencia al evento accediendo desde la invitación.	1,2,3,4,6
5	Como usuario de la aplicación móvil quiero poder ver los asistentes al evento.	1,2,3,4,5,6
6	Como usuario de la aplicación móvil y creador del evento quiero poder modificar la fecha o ubicación del evento y que este cambio sea notificado a todos los usuarios que hayan confirmado su asistencia al evento para que puedan cambiarla si procede.	3,4
7	Como usuario de la aplicación móvil y creador del evento quiero ser notificado cada vez que un usuario de la aplicación confirme o cancele su asistencia a un evento.	1,2,3,4,5,6
8	Como usuario de la aplicación móvil y creador del evento quiero poder cancelar un evento y que todos los participantes sean notificados.	3,4
9	Como usuario de la aplicación móvil y creador del evento quiero que los eventos solo sean accesibles para aquellos usuarios que tengan acceso a la invitación del evento.	1,2,3,4,5,6

Ilustración 30 - Tabla funcionalidades

2.1.8 Casos de uso

Usando como base las funcionalidades que he detectado en el apartado anterior he diseñado los diagramas de caso de uso que engloban dichas funcionalidades.

2.1.8.1 Rol de usuarios

Un usuario que tiene la aplicación en su móvil es un usuario no registrado, al acceder a la aplicación pasa a ser un usuario registrado, cuando se une a un evento pasa a ser un usuario invitado de dicho evento y cuando crea un evento pasa a ser usuario invitado del evento creado y organizador.

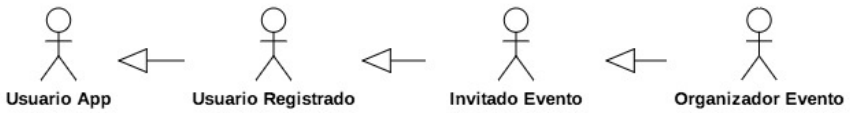


Ilustración 31 - Rol usuarios

2.1.8.2 Registrarse en la aplicación

Cuando un usuario con la aplicación descargada usa su cuenta de Facebook, Twitter o Google para acceder a la aplicación se convierte en un usuario registrado.

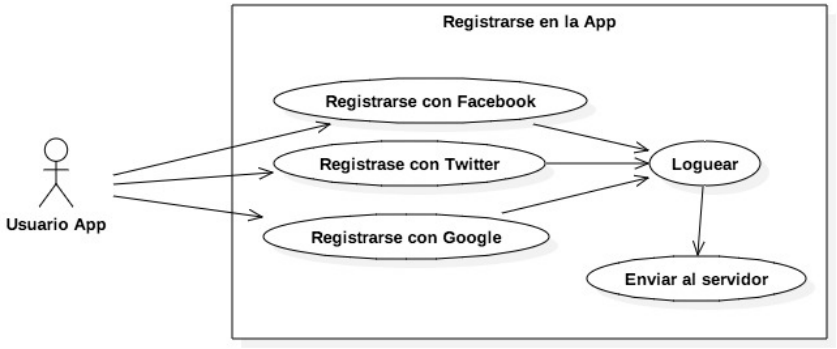


Ilustración 32 - Registro

2.1.8.3 Gestionar eventos

Un usuario que ha creado un evento puede modificar los datos de dicho evento, duplicarlo, cancelarlo y eliminarlo.

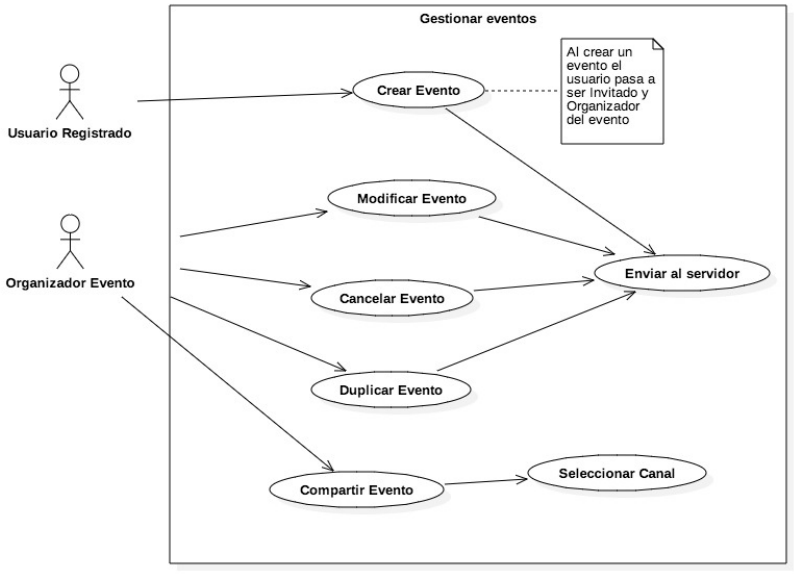


Ilustración 33 - Gestionar eventos

2.1.8.4 Unirse a un evento

Un usuario registrado puede unirse a un evento desde una invitación que le hayan compartido mediante un grupo de WhatsApp, Telegram o cualquier red social.

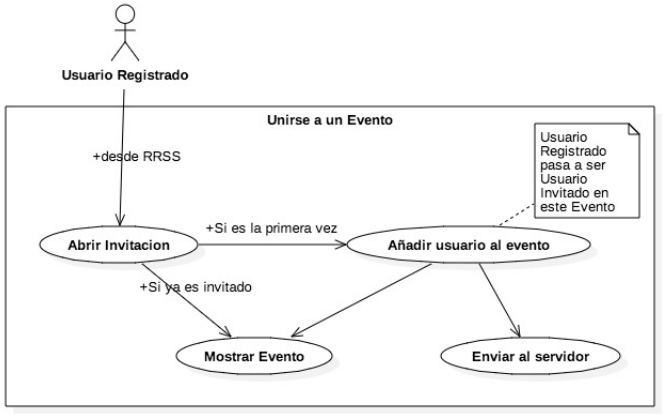


Ilustración 34 - Unirse a eventos

2.1.8.5 Consumir eventos

Un usuario invitado a un evento puede modificar su asistencia y ver los datos del evento.

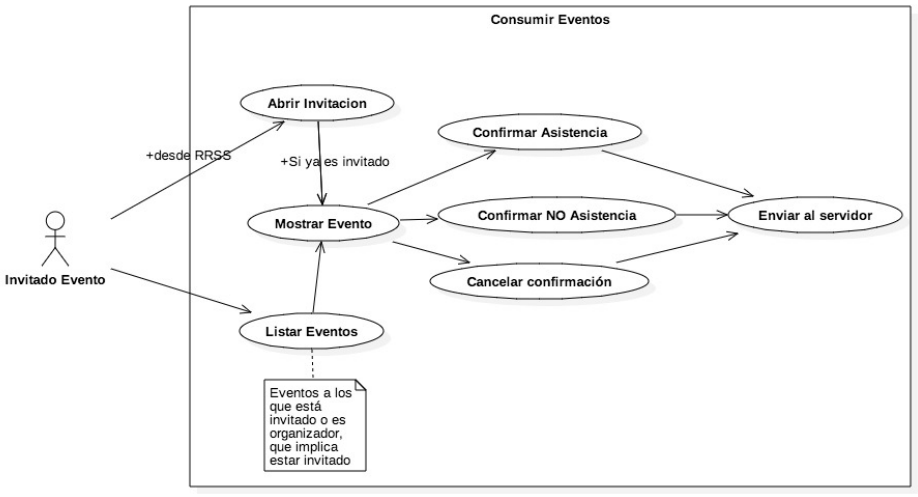


Ilustración 35 - Consumir eventos

2.2 Modelo conceptual

Con las funcionalidades extraídas del apartado anterior he construido un mapa de navegación y un modelo conceptual que me ayuden a transmitir al usuario el concepto de aplicación que voy a diseñar para solventar las necesidades planteadas. Con es el modelo conceptual me vuelvo a entrevistar con los usuarios para validar dicho modelo conceptual y construir un prototipo de alta definición.

2.2.1 Mapa de navegación

Partiendo de los escenarios y las historias de usuario extraídas de dichos escenarios he realizado la siguiente propuesta de mapa de navegación y modelo conceptual que implementará la aplicación móvil.

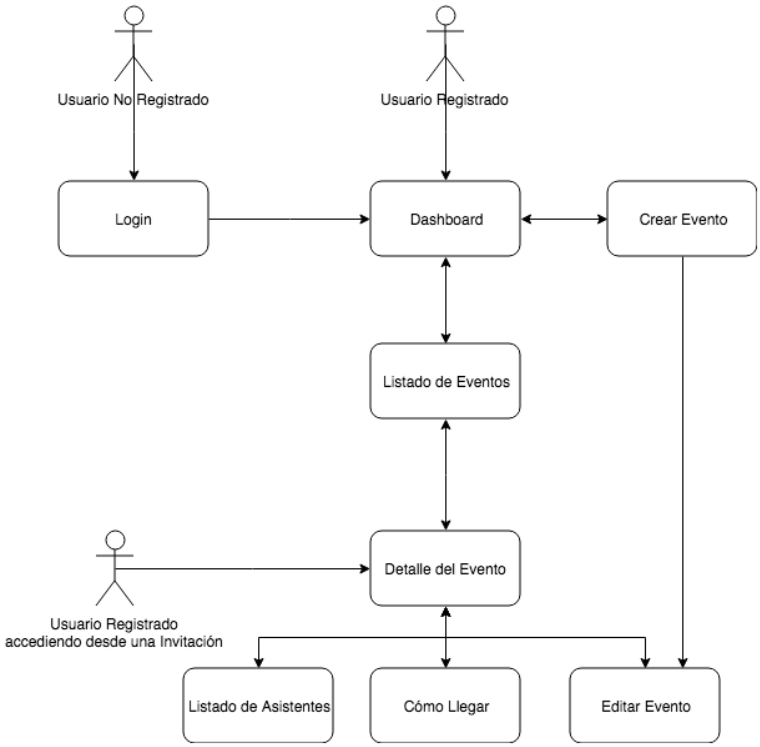


Ilustración 36 - Mapa de navegación

La aplicación tiene tres puntos de entrada, el primero es mediante la pantalla de Login, por esta pantalla accederán los usuarios que aún no se han registrado en la aplicación. Una vez que el usuario se registre en la aplicación y aquellos usuarios que ya estén registrados accederán a la pantalla de Dashboard, desde la pantalla de Dashboard podrán acceder a la pantalla de Crear Evento o a la de Listado de Eventos. Desde el listado de eventos se puede acceder a la pantalla de Detalles de un evento, a esta pantalla también se puede acceder directamente desde un enlace que el usuario haya recibido en su aplicación de WhatsApp o Telegram. Desde la pantalla de detalles se puede acceder a la pantalla de Listado de asistentes, Cómo Llegar y Editar Evento, a esta última pantalla también se puede llegar desde la creación de un nuevo evento.

2.2.2 Prototipo de baja resolución

Conociendo los escenarios donde debe funcionar la aplicación móvil y una vez extraídas las historias de usuario que debe cumplir y en base al mapa de navegación propuesto en el punto anterior he realizado el primer prototipo no funcional y de baja resolución. Este prototipo servirá para validar con los usuario de cada escenario que la aplicación propuesta por el prototipo puede satisfacer sus necesidades y en caso contrario acordar los cambios necesarios para que se adapte a dichas necesidades.

2.2.2.1 Pantalla de Login

Ilustración 37 - Modelo conceptual, LOGIN

Esta pantalla permite que un usuario se registre en la aplicación usando su cuenta de Facebook, Twitter o Google.

2.2.2.2 Pantalla Dashboard

Ilustración 38 - Modelo conceptual, Dashboard

Esta pantalla muestra los datos acumulados de los eventos, todos los eventos, sus eventos, los eventos a los que asistirá el usuario, a los que no asistirá y en los que no ha confirmado su asistencia. También muestra una tarjeta con información sobre el próximo evento y si presiona en ella accederá a la pantalla de detalles de dicho evento. La tarjeta también tiene el botón de editar el evento, que solo se habilitará si el usuario es el creador del evento y le llevará a la pantalla de edición del evento. Otro botón que contiene la tarjeta es el de compartir la invitación del evento, este botón abrirá el dialogo de Android para compartir contenido y el usuario podrá compartir la invitación al evento en forma de enlace web en cualquier aplicación que tenga instalada y permita la funcionalidad de compartir. Desde la tarjeta también podemos acceder a la pantalla de cómo llegar desde el botón que tiene un mapa dibujado. Además de la tarjeta, el dashboard muestra el número de usuarios con los que estás conectado a través de algún evento y tiene el botón (+) que permite crear un nuevo evento.

2.2.2.3 Pantalla Listado de Eventos

Ilustración 39 - Modelo conceptual, Eventos

Esta pantalla muestra todos los eventos en los que participa el usuario en forma de tarjeta, esta tarjeta tiene la misma forma y funcionalidad que la tarjeta del dashboard, si el usuario presiona en ella accederá a la pantalla de detalles de dicho evento. La tarjeta también tiene el botón de editar el evento, que solo se habilitará si el usuario es el creador del evento y le llevará a la pantalla de edición del evento. Otro botón que contiene la tarjeta es el de compartir la invitación del evento, este botón abrirá el dialogo de Android para compartir contenido y el usuario podrá compartir la invitación al evento en forma de enlace web en cualquier aplicación que tenga instalada y permita la funcionalidad de compartir. Desde la tarjeta también podemos acceder a la pantalla de cómo llegar desde el botón que tiene un mapa dibujado.

2.2.2.4 Pantalla Detalles del Evento

Ilustración 40 - Modelo conceptual, Detalles

En esta pantalla se muestran los datos del evento, contiene la información sobre el creador del evento y la fecha y duración de este. A mitad de pantalla aparece el titulo y la descripción del evento y debajo de esta aparecen dos etiquetas, la primera muestra el numero de asistentes confirmados al evento, esta etiqueta se puede pulsar y llevará al usuario a la pantalla de asistentes al evento, la otra etiqueta es simplemente informativa, indicando el número de plazas disponibles. La parte inferior de esta pantalla contiene

tres botones que indican si el usuario ha confirmado o no su asistencia al evento y desde los que puede cambiar su asistencia.

2.2.2.5 Pantalla Crear Evento y Editar Evento

Ilustración 41 - Modelo conceptual, Crear y Editar

Esta pantalla tiene dos funcionalidades, dependiendo si vamos a crear un evento o editar uno ya existente. La pantalla de crear/editar un evento se compone de un formulario que permite al usuario rellenar o modificar los datos de un evento, estos datos son, el nombre del evento, el título, los detalles del evento, el número de participantes o indicar que no hay restricción de asistencia, la duración del evento o indicar que no tiene una duración determinada, la ubicación del evento, que podremos afinarla haciendo uso de un mapa y por último la fecha y hora del evento.

2.2.2.6 Pantalla Listado de Asistentes

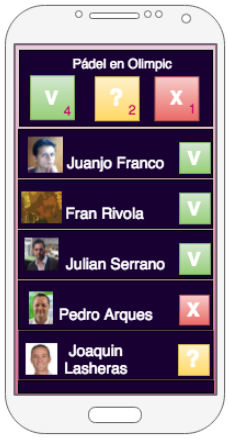


Ilustración 42 - Modelo conceptual, Asistentes

En esta pantalla podemos visualizar los datos de acumulado de asistencias al evento y ver en detalle en que estado se encuentra la asistencia de cada usuario invitado al evento.

2.2.2.7 Pantalla Cómo Llegar

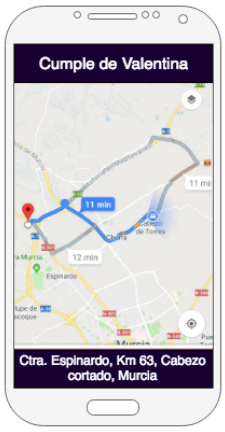


Ilustración 43 - Modelo conceptual, Cómo llegar

En esta pantalla se mostrará un mapa con la ruta desde la posición actual del usuario hasta la ubicación del evento.

2.3 Evaluación con usuarios

He realizado una segunda fase de entrevistas con los mismos usuarios que ayudaron a confeccionar los escenarios para la captación de las historias de usuario, en esta nueva fase hemos analizado el modelo conceptual de los puntos anteriores y hemos consensuado los siguientes puntos:

1. La pantalla de dashboard no resulta útil, la información relevante para los usuarios empieza en el listado de eventos que están organizando o a los que han sido invitados. Para satisfacer esta sugerencia la pantalla de dashboard se elimina del diseño y la pantalla de listado de eventos pasa a ser la pantalla principal.

2. En la pantalla de listado de eventos es necesario poder diferenciar los eventos que organizas de los que estás invitado, es necesario poder buscar un evento por nombre y que el orden de los eventos sea en orden cronológico del más próximo al más lejano en el tiempo. En esta pantalla sería conveniente tener un mecanismo de filtrado que permitiese mostrar los eventos bajo los criterios de si el usuario asiste, no asiste, no se ha pronunciado al respecto, por fecha y eventos que ya han pasado. Para satisfacer estas sugerencias la pantalla de listado de eventos dispondrá de un buscador por nombre y se separará en dos listados, el listado de eventos que organiza el usuario y los eventos a los que es invitado. Las sugerencias de filtrado son altamente interesantes, pero por cuestiones de tiempo y prioridad serán relegadas a una versión posterior de la aplicación.

3. En la pantalla de listado de eventos cada evento debe potenciar la información que interesa al usuario dependiendo de si es organizador o invitado, siendo de vital importancia para el organizador poder invitar a más usuarios a un evento y

ver el estado actual de la asistencia. Desde el punto de vista del usuario invitado debe potenciarse la información del evento y poder cambiar el estado de su asistencia. Para satisfacer estas sugerencias las tarjetas de cada evento serán ligeramente distintas dependiendo de si el usuario es organizador o invitado, en las de organizador prevalecerá un botón para invitar a más usuarios y la información de la asistencia. Para las tarjetas en las que el usuario es un invitado se resaltará el mecanismo para indicar si se asistirá o no y los datos del evento como la fecha y dirección del evento.

4. En la creación de un nuevo evento debe poder introducirse la dirección a mano para indicar información que no aparece en los mapas, como nombre de un restaurante o referencias locales como el parking de un centro comercial. Para satisfacer esta sugerencia se permitirá introducir la dirección de forma manual y disponer de un botón que buscará esa dirección en un mapa de Google y permitirá afinar la dirección, manteniendo como texto la dirección introducida a mano y haciendo está más precisa mediante las coordenadas obtenidas del mapa.

5. La pantalla de creación de un nuevo evento debe ser más guiada, debe especificar de forma más clara que datos deben introducirse en cada apartado. Para satisfacer esta sugerencia la pantalla de edición se va a separar en tres bloques, el bloque del evento que pedirá el nombre del evento, en qué consistirá y cómo se llevará a cabo, el bloque que solicitará cuando se hará el evento y el bloque de donde se hará el evento.

6. La pantalla de detalles debe resaltar la información del evento para los usuarios invitados y la información de asistencias y el listado de asistentes para el usuario organizador. Para satisfacer esta sugerencia se va a separar la pantalla de detalles en dos secciones, la de información del evento y la del listado de asistentes.

2.4 Prototipo no funcional

Partiendo de las sugerencias de los usuarios sobre el modelo conceptual obtenidas de la segunda entrevista he realizado un prototipo no funcional. Este prototipo es una aproximación bastante fidedigna del resultado final esperado por los usuarios. Su versión web puede ser consultada en la dirección http://prototype.quedadas.ovh/.

Ilustración 44 - Prototipo y mapa de navegación

2.4.1 Pantalla de Login

Ilustración 45 - Prototipo, LOGIN

En la pantalla de Login se mantiene el mismo concepto, acceso mediante las tres principales redes sociales y el logo de la aplicación para adornar dicha pantalla. Esta pantalla solo será necesaria la primera vez que acceda el usuario, una vez que este haya iniciado sesión las siguientes veces que abra la aplicación aparecerá directamente en la pantalla de Listado de Eventos, a menos que la abra desde una invitación a un evento, en cuyo caso accederá directamente a la pantalla de Detalles del Evento.

2.4.2 Pantalla de Listado de Eventos

La pantalla de Listado de Eventos ha sido remodelada en base a la información obtenida de los futuros usuarios de la aplicación, esta pantalla se ha dividido en dos secciones mediante pestañas.

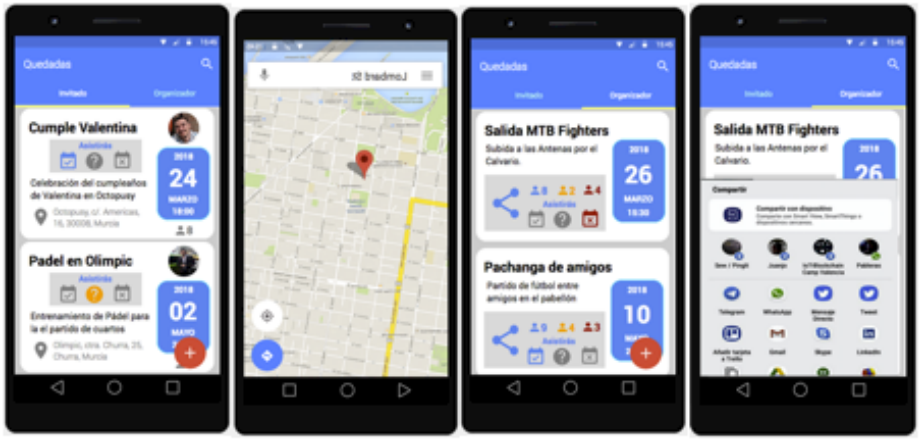


Ilustración 46 - Prototipo EVENTOS

La pestaña de “Invitados” muestra el listado de eventos en los que el usuario de la aplicación ha sido invitado, en la tarjeta de cada evento se resalta al organizador del evento, la fecha y la dirección. Si el usuario pulsa sobre la dirección se abre la pantalla de Google Maps con la ruta para llegar desde la ubicación actual del usuario hasta la ubicación del evento. En la tarjeta también tiene una botonera que sirva para que el usuario pueda ver su asistencia y cambiarla sin necesidad de acceder al detalle del evento.

En la pestaña de “Organizador” se muestran los eventos que están organizados por el usuario, en la tarjeta de cada evento se resalta la fecha y la botonera para gestionar la asistencia, pero en el caso de organizador se le ha añadido el acumulado de asistencias, para que el organizador pueda tener una idea de intención de asistencia y el botón para compartir el evento. Con estos cambios el organizador puede tener una idea del estado de todos los eventos y continuar difundiéndolos por sus grupos sin necesidad de acceder al detalle.

Si pulsamos sobre una tarjeta la aplicación nos llevará a la pantalla de detalles del evento asociado a la tarjeta.

Si pulsamos sobre el botón de añadir un evento la aplicación nos llevará a la pantalla de crear un nuevo evento.

2.4.3 Pantalla de Crear y Editar Evento

La pantalla de Crear y Editar se ha diseñado bajo la premisa de guiar al usuario en la inserción de datos, para ello se han sustituido los textos típicos de un formulario por preguntas que sirven como guía a los usuarios a la hora de introducir los datos del evento.

El primer bloque de datos tienen que ver con información sobre el tipo de evento.
El segundo bloque es para insertar la información de cuando se celebra el evento, estos datos se insertan mediante los componentes de fecha y hora proporcionados por le sistema.

Ilustración 47 - Prototipo CREAR y EDITAR

El tercer bloque es para indicar la ubicación del evento, esta información se puede introducir escribiendo la dirección con el teclado y además se puede afinar dicha ubicación usando un mapa de Google. Con este mecanismo el usuario puede personalizar el texto de la dirección a la misma vez que proporciona unas coordenadas precisas de la misma.
El último bloque hace referencia a las restricciones de asistentes y tiempo del evento, que son opcionales.

Si pulsamos sobre el botón guardar y estamos creando un nuevo evento, se creará el nuevo evento y la aplicación nos llevará a la pantalla de detalles del evento, en le caso de estar editando un evento existente, se actualizarán sus nuevos datos y volveremos a la pantalla de detalles igualmente.

2.4.4 Pantalla de Detalles del Evento

Ilustración 48 - Prototipo DETALLES

La pantalla de detalles del evento se ha dividido en dos secciones mediante el uso de pestañas para una mejor organización de la información del evento.

La pestaña de “Información” contiene los datos que más interesan a un usuario que es invitado a un evento, como quién es el organizador, el resumen de asistentes, la fecha, los detalles del evento, la duración y la dirección.

Desde el icono de dirección se accede al mapa de Google con las indicaciones de cómo llegar desde la ubicación actual hasta la del evento. En la parte inferior de esta sección se encuentra la botonera desde la que el usuario puede conocer el estado de su asistencia al evento y cambiarlo si procede.

La pestaña “Invitados” contiene el resumen de asistentes, así como el listado de cada asistente y el estado de su asistencia al evento.

En el caso de que el usuario sea el organizador del evento aparecerán los botones de editar y compartir el evento en la barra de navegación de la pantalla. Desde el botón de compartir podrá difundir el evento entre sus contactos y con el botón editar podrá acceder a la pantalla de editar el evento para cambiar su información.

3. Arquitectura de la aplicación

Para poder cubrir todas las necesidades de la aplicación es necesario crear una infraestructura que permita compartir la información de los eventos entre los usuarios de la aplicación y que sea capaz de notificarlos cuando se producen modificaciones sobre un evento.

3.1 Análisis y diseño de la arquitectura del servicio

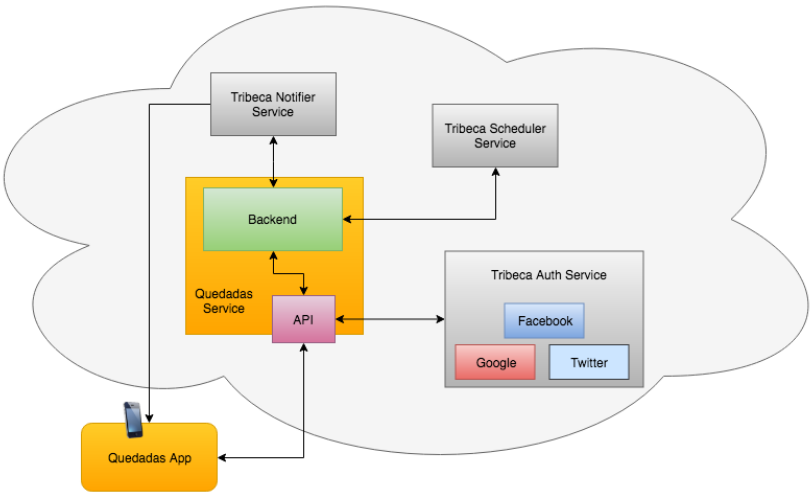


Ilustración 49 - Arquitectura del proyecto

La arquitectura se compone de dos módulos como podemos ver en el diagrama. El modulo "Quedadas App" y el módulo "Quedadas Service".

El módulo "Quedadas App" es la aplicación móvil que he descrito en los capítulos anteriores, es la interfaz que tendrá el usuario para interactuar con la solución diseñada en este proyecto. La aplicación móvil se comunicará con el módulo "Quedadas Service" mediante el protocolo HTTP, consumiendo un servicio Restful que expone dicho modulo. Por otro lado, la aplicación móvil recibirá notificaciones mediante un servicio PUSH que le informará de los cambios realizados en los eventos en los que el usuario participa, ya sea como organizador o como invitado.

El módulo "Quedadas Service" está compuesto por la aplicación "API" y la aplicación "Backend" como podemos ver en la ilustración superior. La aplicación API es una pasarela o gateway encargada de exponer el servicio restful para comunicarse con la aplicación móvil. El API se encarga de proveer la funcionalidad de autenticación a la aplicación móvil mediante las redes sociales de Facebook, Google y Twitter, para ello delega esta gestión en el Servicio "Tribeca Auth Service", que se encargará de comprobar si las credenciales de usuario aportadas por la aplicación móvil son válidas y comunicarse con los servicios de Facebook, Twitter y Google. En caso de ser una petición válida este servicio devolverá un JSON Web Token (JWT) a la aplicación API para que pueda continuar con la gestión de credenciales, esta a su vez entregará el JWT

generado a la aplicación móvil para que esta pueda realizar el resto de las peticiones. La aplicación API también se encargará de recoger todas las peticiones de la aplicación móvil, comprobar que tienen el JWT adecuado y entregárselas a la aplicación Backend para que esta las procese y así responder a la aplicación móvil con las respuestas facilitadas por la aplicación Backend.

La aplicación de Backend se encarga de realizar las funciones almacenar los eventos creados por los usuarios desde la aplicación móvil y permitir que estos puedan modificarlos. Además de las labores de persistencia, la aplicación de Backend se encarga de notificar a los usuarios los cambios realizados en eventos a los que pertenecen, para ello se integra con el servicio "Tribeca Notifier Service" y así poder enviar notificaciones PUSH a los usuarios cuando lo estime oportuno. Otra función que ofrece la aplicación Backend es la de enviar recordatorios de los eventos a los usuarios, para ello se integra con el servicio "Tribeca Scheduler Service" y así poder registrar notificaciones programadas para un momento concreto. Ni "Tribeca Scheduler Service" ni "Tribeca Notifier Service" son parte de la solución aportada por este proyecto, son dos servicios independientes que permiten extender las funcionalidades de la solución de una forma rápida y desacoplada.

3.2 Análisis y diseño del Backend

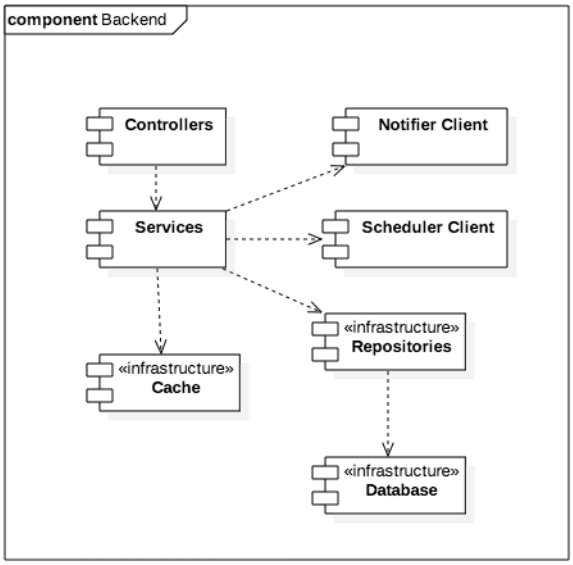


Ilustración 50 - Componentes del backend

Backend es un microservicio que se encarga de recibir y procesar todas las peticiones enviadas por API, su punto de entrada es el componente "Controllers", este componente es el responsable de procesar cada petición de API e invocar a la funcionalidad del componente "Services" adecuada. El componente "Services" encapsula toda la lógica de negocio del Backend, proporcionando una interfaz con todos los servicios disponibles al componente "Controllers". Esta lógica consta de creación, actualización, borrado y lectura de los eventos almacenados en la base de datos. "Services hace uso del componente "Cache" para recuperar los datos solicitados por "Controllers" desde la base de datos en memoria, si "Cache" dispone de los datos y son datos actualizados los

devolverá a “Controllers” en caso contrario, “Services” hará uso del componente “Repositories” para que este acceda a la base de datos y recupere la última versión de los datos, en este caso también actualizara los datos de “Cache”. Parte de la lógica de “Service” es enviar notificaciones a los usuarios o planificar recordatorios, esta lógica la hace mediante los clientes “Notifier client” y “Scheduler client”, que son la interfaz de comunicación con los servicios externos “TribecaNotifier Service” y “Tribeca Scheduler Service”.

En los siguientes diagramas de secuencia vemos como interactúan los distintos componentes que forman el módulo Backend para proporcionar las distintas funcionalidades de este.

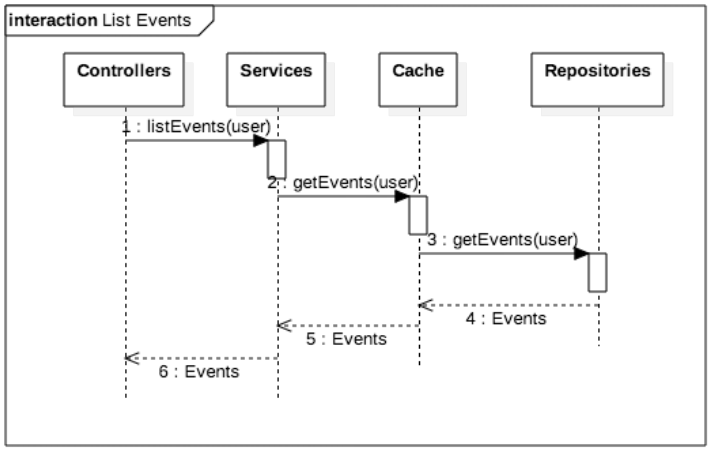


Ilustración 51- Backend, listado de eventos

El componente “Controllers” recibe la petición de obtener el listado de eventos, se lo notifica al componente “Services”, que a su vez se lo solita al componente “Cache”, este lo solicita al componente “Repositories” que obtiene el listado de la base de datos y lo devuelve a través de cada componente hasta llegar a “Controllers”.

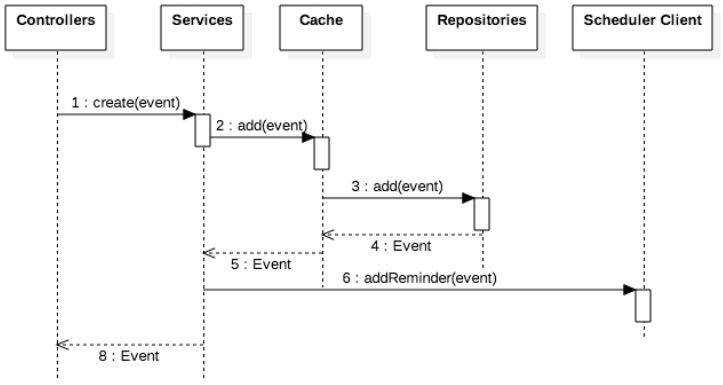


Ilustración 52 - Backend, crear evento

Para el caso de crear un nuevo evento la secuencia es similar, salvo que en este caso se solicitará el almacenamiento del evento enviado por “Controllers”, este caso además hará uso del componente “Scheduler client” para registrar un recordatorio que será enviado a los usuarios del evento cuando se aproxime la fecha de este.

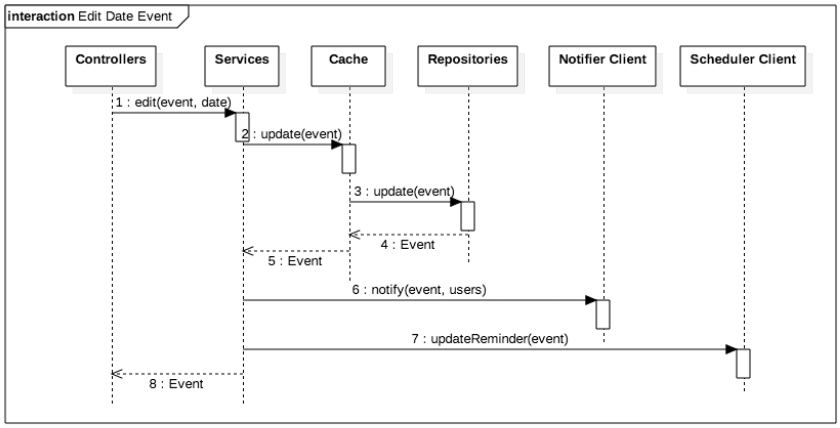


Ilustración 53 - Backend, cambiar fecha a un evento

La secuencia de mensajes para editar un evento es similar a las anteriores, pero en este caso también participa el componente "Notifier client" que se encargará de notificar vía PUSH a todos los participantes de un evento del cambio que se ha realizado en este. En el siguiente diagrama podemos ver

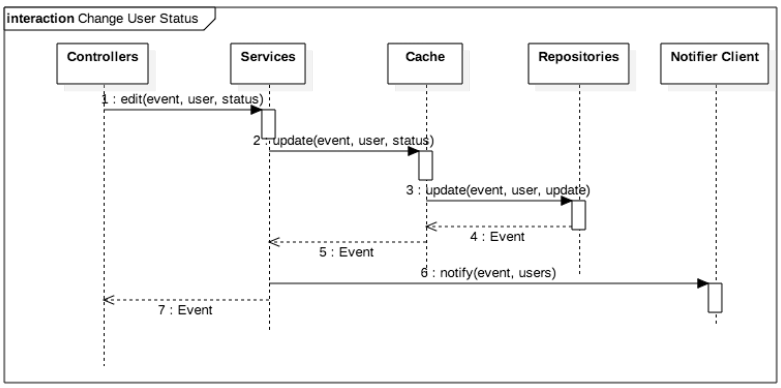


Ilustración 54 - Backend, cambiar asistencia a un evento

La secuencia para cambiar el estado de la asistencia de un usuario a un evento es un caso particular de la secuencia de edición, es este caso se envían los mismos mensajes entre los distintos componentes, pero con distintos parámetros. El resto de las secuencias que dan lugar a las funcionalidades del Backend son una pequeña variación del diagrama de edición, pero con los parámetros involucrados en la actualización.

3.3 Análisis y diseño del API

El API está formado por cuatro componentes, el "Request Handler" que es el encargado de recibir todas las peticiones de los clientes móviles, verificar la autenticidad de la petición y entregarla al componente "Routes Service".

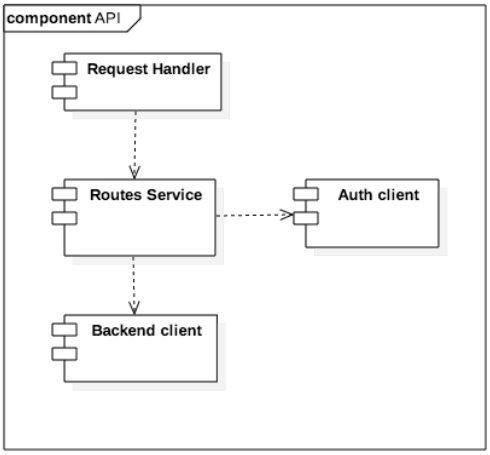


Ilustración 55 - Componentes del API

El componente "Routes Services" es el encargado de llamar a la función adecuada del componente "Backend Service" dependiendo de la ruta solicitada por la aplicación móvil o de obtener un token de acceso (JWT) del servicio "Tribeca Auth Service" haciendo uso del componente "Auth client". El componente "Backend Service" encapsula la comunicación entre el API y el Backend y el componente "Auth client" encapsula la funcionalidad para comunicar el API con el servicio de autenticación externo. El API realiza dos tipos de operaciones, una operación de registrar y validar usuarios y otra de validar y encaminar el resto de las peticiones al Backend.

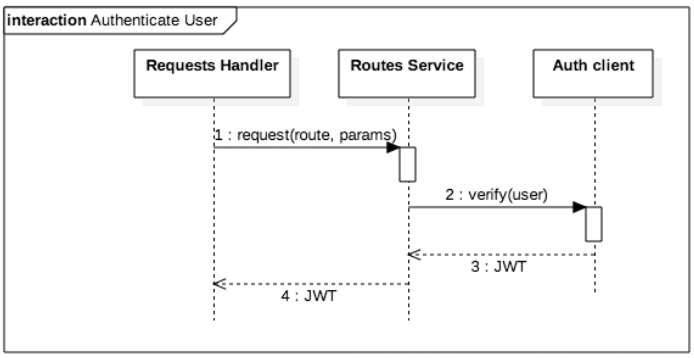


Ilustración 56 - API, Autenticar a un usuario

Del primer tipo de peticiones sólo existe una, la encargada de registrar o validar usuarios, esta petición se recibe en el componente "Request Handler", que la redirige directamente al componente "Routes Service" para que este haciendo uso del componente "Auth client" autentique al usuario mediante el servicio externo, que en caso de validar al usuario le entregará un JSON Web Token (JWT) que servirá para validar el resto de las peticiones de dicho usuario. Este JWT será devuelto a la aplicación móvil para que pueda usarlo en el resto de las peticiones. Además de devolverlo a la aplicación móvil, el componente "Request Handler" lo almacenará para poder verificar el resto de las peticiones que haga este usuario.

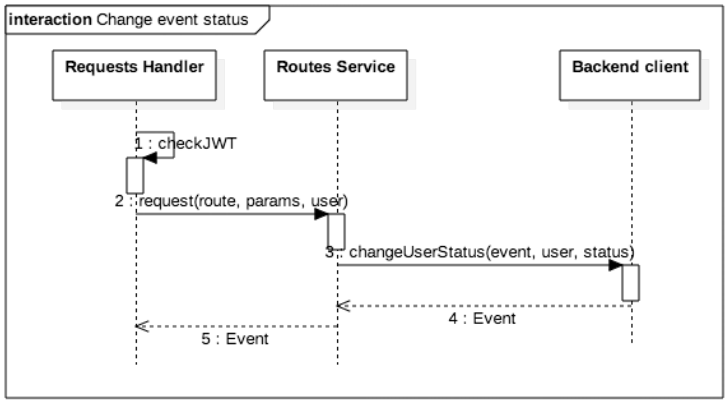


Ilustración 57 - API, cambiar asistencia de un usuario

El resto de las secuencias del API son muy similares, el componente recibe una petición de la aplicación móvil mediante el componente "Request Handler", este comprueba la autenticidad del usuario mediante su JWT que previamente a almacenado, en caso de ser una validación positiva, envía la petición al componente "Routes Services" con los parámetros y la ruta de la petición para que este haciendo uso del componente "Backend client". En el diagrama anterior se muestra la secuencia que seguirá una petición para cambiar el estado de asistencia de un usuario en un evento.

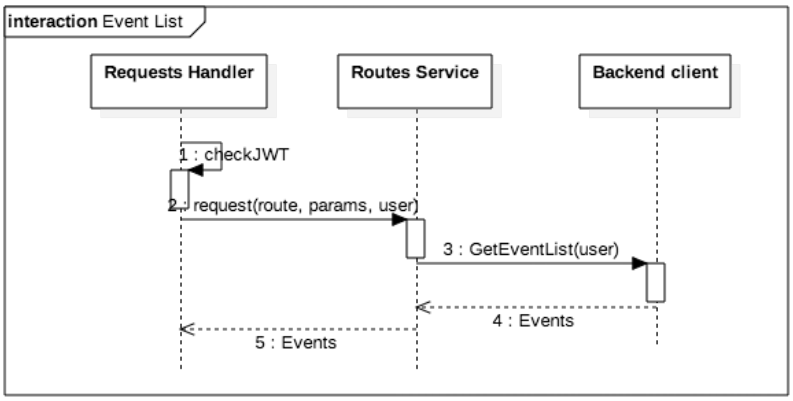


Ilustración 58 - API, recuperar el listado de eventos

El resto de las secuencias que ocurren en el API son del segundo tipo expuesto en los párrafos anteriores, con la salvedad que el componente "Routes Services" hara uso de unas funciones u otras del componente "Backend client" según la ruta y los parámetros recibidos en la petición.

3.4 Análisis y diseño de la aplicación móvil

La aplicación móvil contiene el componente "Screen Manager" que es el encargado de orquestar la visualización de los componentes de tipo "Screen". Los componentes de tipo "Screen" como "Login Screen" son los componentes que representan cada pantalla que se muestra en la aplicación móvil, cada uno tendrá su interfaz gráfica y capturará los eventos necesarios para dotar de la funcionalidad necesaria a la pantalla que representan.

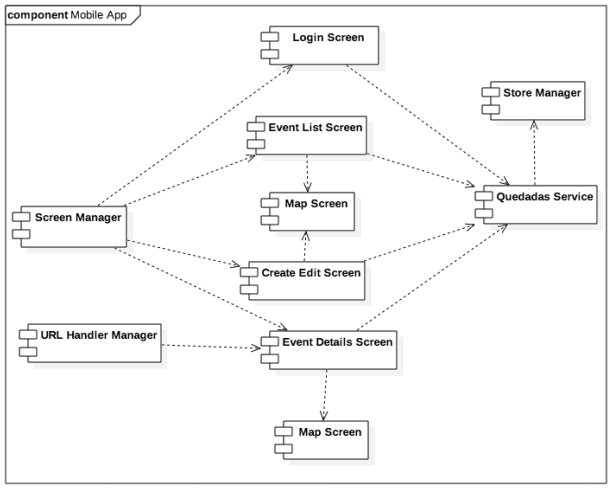


Ilustración 59 - Componentes de la App móvil

Además, estos componentes harán uso del componente "Quedadas Service" que es el encargado de realizar la comunicación con el API para intercambiar los eventos de los usuarios y realizar las operaciones oportunas sobre estos. El componente "Quedadas Service" hace uso del componente "Storage Manager" para mantener una copia de los eventos usados en la memoria del dispositivo móvil y así poder visualizar los eventos, aunque este no disponga de conexión con el API. En esta primera versión el componente "Store Manager" tendrá una implementación basada en una caché persistente en memoria del tipo LRU, (Last Recent Used), que permitirá acceder a la información mas reciente obtenida del API, aunque no se disponga de conexión con el API, pero no. Permitirá realizar ninguna operación de modificación sobre los datos de la cache, para ello será necesario disponer de conexión con el API.

Las secuencias de la aplicación móvil son muy similares, consisten en mostrar las pantallas que solicita el usuario mediante la navegación, de la cual se encarga el componente "Screen Manager" y realizar peticiones al API para obtener la información que se muestra en dicha pantalla, para ello cada pantalla hace uso del componente "Quedadas Service"

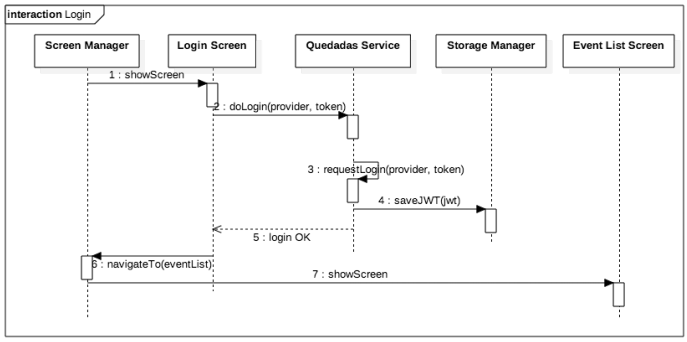


Ilustración 60 - App, iniciar sesión en la aplicación

En el diagrama anterior vemos la secuencia usada para que el usuario pueda iniciar sesión en la aplicación, que es la primera secuencia que ocurre cuando un usuario usa la aplicación por primera vez. En esta secuencia el componente "Screen Manager" se encarga de mostrar la pantalla de inicio de sesión, desde esta pantalla el usaurio puede elegir la red social con la que quiere autenticar y obtiene el token de autenticación de

dicho provider, este token se envía al componente “Quedadas Service” que lo intercambia con API por un JWT, cuando este componente obtiene el JWT lo almacena en el componente “Storage Manager”. Una vez acabado el proceso, el componente “Screen Manager” muestra la pantalla de listado de eventos.

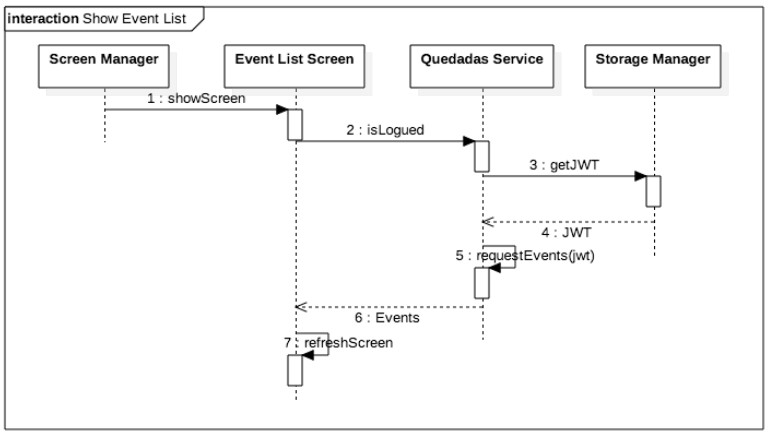


Ilustración 61 - App, Mostrar listado de eventos

El resto de secuencias siguen un mismo patron, el usuario navega de una pantalla a otra mediante el componente “Screen Manager”, cada pantalla usa el componente “Quedadas Service” para recuperar la información que tiene que mostrar. El componente “Quedadas Service” hace uso del componente “Storage Manager” para recuperar el JWT antes de hacer cada petición. Cuando el componente de tipo Screen, como el “Event List Screen” en el caso del diagrama anterior, recupera los datos se actualiza para mostralos.

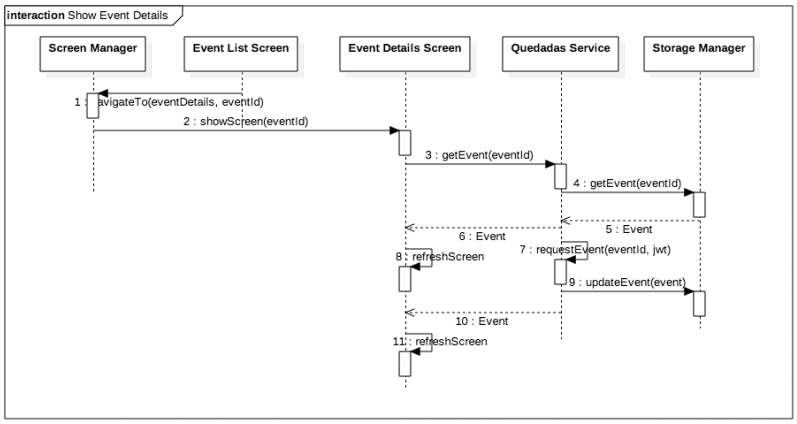


Ilustración 62 - App, Mostrar la información de un evento

En la secuencia de recuperar y mostrar información por pantalla puede haber una pequeña variación en el caso de que dicha información ya haya sido consultada con anterioridad, en este caso, el componente “Quedadas Service” recupera primero los datos cacheados del componente “Storage Manager”, los devuelve al componente de tipo pantalla que inicio la solicitud, y después realiza la petición al API para obtener la versión actualizada de estos, cuando el API devuelve la nueva versión de los datos el componente “Quedadas Service” se los entrega tanto al componente “Storage Manager” para que los almacene, como al componente de tipo Screen para que actualice su estado.

4. Implementación

Es este apartado explico la elección de las herramientas y tecnología utilizadas en la implementación de la solución.

- Visual Studio Code
- Jenkins
- Nginx
- PM2
- MongoDB
- ASP Core .Net
- ExpressJS
- IONIC 3

También comento los principales patrones de diseño utilizados en cada producto, así como el código mas representativo de cada uno. Por último, muestro el prototipo funcional obtenido.

4.1 Herramientas y tecnología

Para agilizar el proceso de implementación de la solución he diseñado un flujo de trabajo basado en el proceso de integración continua [Microsoft CI 2018].

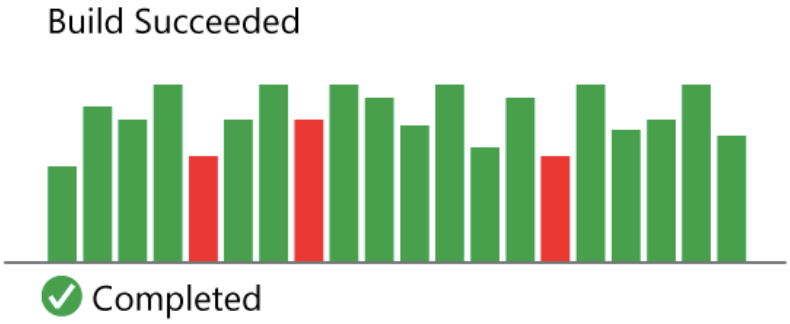


Ilustración 63 – Builds de Intgración Continua

Este flujo comprende desde la escritura del código fuente en el entorno de desarrollo hasta la puesta en producción de la aplicación móvil en el "Play Store" de Android o el despliegue del Backend o API en el vps. Además, he utilizado herramientas para la monitorización del servicio web y la aplicación móvil.

Tanto el Backend como el API hacen uso del mismo flujo, este comienza por escribir el código fuente de la aplicación usando el editor Visual Code, este código se mantiene versionado usando la herramienta git en Bitbucket, cuando se ha terminado una funcionalidad del API o del Backend se empujan todos los commits generados a la rama master de Bitbucket. Cuando Bitbucket recibe los cambios en su rama master avisa a Jenkins para que recupere estos cambios y realice el despliegue del código. Jenkins se encarga de recuperar la última versión de la rama master, pasar los test unitarios y de integración al código, si estos test son superados, Jenkins crea una nueva versión del API o del Backend y lo pone en producción en el VPS.

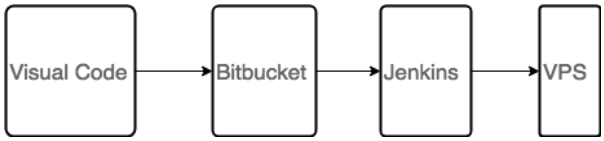


Ilustración 64 - CI backend y API

El flujo de la aplicación móvil es básicamnte el mismo, con la salvedad que en caso de superar los test Jenkins crea un nuevo apk y lo publica en el Play Store en versión beta.

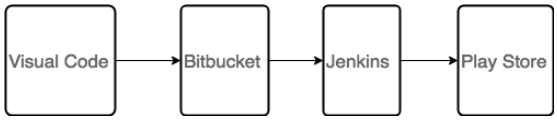


Ilustración 65 - CI App móvil

Gracias a este flujo de trabajo puedo publicar cada avance realizado y tener una versión estable de la solución siempre disponible para enseñar a los clientes con las últimas características desarrolladas.

4.1.1 Visual Studio Code

Visual Studio Code [Visual Studio Code 2018] es el editor elegido para el desarrollo del código, esta elección se ha basado en la facilidad de uso, la cantidad de plugins de los que dispone el editor para integrarse con nodejs, ionic, git y asp .net core. Por ser esta una solución implementada en varios lenguajes, entre ellos javascript, typescript y c#, he tenido que buscar un IDE con plugins para los distintos lenguajes, que además consuma pocos recursos, pero sin tener que renunciar a características como el autocompletado y sintaxis coloreada para cada lenguaje. Otra razón por la que he seleccionado VSC es por disponer de un entorno de depuración que se integra tanto con NodeJS para depurar el API, como con ASP .Net Core para el Backend y con Ionic para la aplicación móvil. Este editor también dispone de un plugin para la gestión de las aplicaciones Ionic, permitiendo lanzar a la aplicación en el emulador, el navegador o un dispositivo móvil. Por último, pero no menos importante, este editor es el que uso en mi entorno de trabajo y ya estoy familiarizado y puedo realizar las tareas de implementación de forma cómoda y productiva. Otras opciones como JetBrains o Visual Studio las he descartado por consumir muchos recursos o Sublime por no disponer de un depurador tan avanzado.

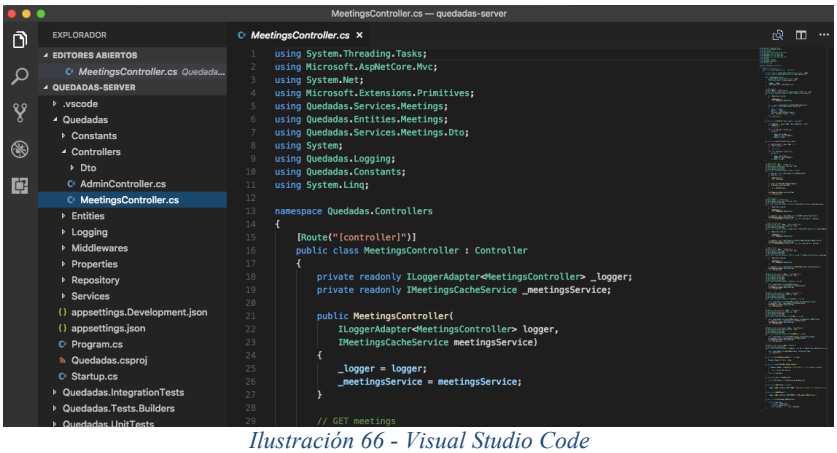


Ilustración 66 - Visual Studio Code

4.1.2 Jenkins

"La integración continua (continuous integration en inglés) es un modelo informático propuesto inicialmente por Martin Fowler que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes. Entendemos por integración la compilación y ejecución de pruebas de todo un proyecto"

Wikipedia[Wikipedia CI 2018]

Desde que Martin Fowler propuso el flujo de integración continua en 2006 [Martin Fowler CI 2018] ha habido muchos servidores y servicios para automatizar dicho flujo. Entre los mas conocidos destacan Jenkins [Jenkins CI 2018], CircleCI [Jenkins vs CCI 2018], Bamboo, FastLane y Visual Studio Team Service entre otros. En este caso he optado por una instalación en un servidor propio porque las soluciones SaaS son muy caras o muy limitadas y no todas permiten la instalación del software necesario para algunos de los despliegues, como por ejemplo el sdk de Android, clientes de Ionic o el sdk de asp .net core.

Además, tener el servidor alojado en un servidor privado permite la creación de bash script para apoyar la automatización de algunos procesos de despliegue y la personalización de la configuración del servidor para instalar herramientas de compilación cruzada para distintas plataformas. Entre los servidores de CI disponibles he optado por Jenkins por su gran flexibilidad, y por disponer de muchos plugins para crear distintos tipos de artefactos, para notificar por distintos medios como email, sms, notificaciones al móvil, etc. y por ser open source, lo que permite que la comunidad haga mejoras y añada nuevas funcionalidades y las publique para que otros usuarios podamos disfrutar de ellas.

En el caso concreto del servidor Jenkins usado en el proyecto está configurado para recibir una notificación de Bitbucket cada vez que se integra un cambio en la rama de master, cuando Jenkins recibe esta notificación, clona el repositorio que ha recibido la integración, ejecuta los test unitarios y de integración que tiene el proyecto, en caso de pasar los test, genera un nuevo artefacto, que puede ser una nueva versión del API, del Backend o un nuevo apk de la aplicación móvil, cuando ha creado el artefacto, envía un email para notificar de la correcta construcción de este y a continuación lo publica en el destino apropiado, ya sea el vps que aloja el API y el Backend o en el Play Store. En caso de fallar el proceso de test o de generación del artefacto se envía un email indicando cual ha sido el problema que ha detenido la nueva publicación.

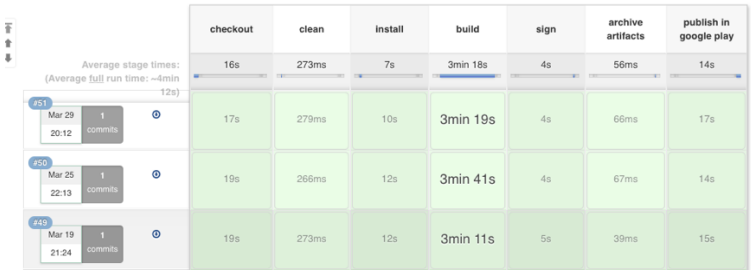


Ilustración 67 – Publicar nuevo APK

4.1.3 Nginx

Para que la aplicación móvil o cualquier otro cliente tenga acceso al API es necesario usar un servidor de estáticos, que se encargue de gestionar el acceso al API, para este trabajo se usan mayoritariamente dos herramientas, apache y nginx, ambos son servidores de estáticos y cumplen la función de publicar una aplicación al resto de internet [Digital Oceans 2018]. En mi caso he elegido nginx como servidor de estáticos por su facilidad de configuración y su flexibilidad a la hora de integrarse con otras tecnologías como nodejs, socketio, ssl entre otras, su enfoque de un fichero de configuración basado en directivas hace que la configuración de distintos recursos con distintas configuraciones sea más sencilla. Además, su rendimiento es superior al de apache a la hora de servir flujos de datos dinámicos.

4.1.4 PM2

La aplicación API se encarga de hacer de pasarela o puerta de enlace entre las aplicaciones móviles y el Backend, pero no deja de ser una aplicación escrita en nodejs usando el framework express. Por esta razón necesita de una herramienta que la mantenga en funcionamiento y detecte cuando ha sufrido una caída y vuelva a ponerla en funcionamiento.
Para esta tarea he elegido PM2 [PM2 2018], que es el gestor de procesos mas usado en el mundo Nodejs para monitorizar sus procesos y proveer una interfaz que permita iniciarlos, pararlos, eliminarlos y reiniciarlos además de aportar información del estado actual de cada proceso.

4.1.5 MongoDB

Para almacenar los eventos en el backend he decidido usar la base de datos MongoDB, que es una base de datos NoSQL. Las bases de datos NoSQL, a diferencia de las bases de datos relacionales, nos permiten tener un modelo de datos sin esquema, más flexible, donde el impacto de añadir nuevos elementos al modelo es mínimo o incluso nulo.
En el caso de este proyecto, MongoDB permite ir evolucionando el modelo de datos en función de las nuevas necesidades detectadas sin que esto suponga un gran impacto en el desarrollo de la aplicación. Por otro lado, el hecho de almacenar sus documentos en formato JSON, que es el mismo formato usado en el API restful, nos permite intercambiar datos entre el API y el backend sin necesidad de una lógica compleja de mapeo de datos.
Otro motivo para la elección de MongoDB es su soporte nativo para datos geoespaciales, esto nos permite realizar búsquedas basadas en la geoposición de los datos, que aunque no lo he planteado para la primera versión del proyecto, nos permitirá en versiones futuras realizar búsquedas de los eventos cercanos a la geoposición del usuario.
Para la elección de MongoDB también se ha tenido en cuenta que es una base de datos distribuida y fácilmente escalable, que es un aspecto fundamental para un proyecto como este, con tintes de red social, donde prima la rápida disponibilidad de los datos más que la consistencia entre dichos datos, pudiendo permitirnos cierto grado de inconsistencia temporal en los datos en sus fases de sincronización.

4.1.6 ASP .Net Core 2.0

El backend del proyecto es la parte que soportará la mayor carga de trabajo, ya que debe procesar las peticiones de todos los usuarios y aplicar la lógica de negocio de cada

petición y responder en el mínimo de tiempo posible. Para esta labor he decidido usar ASP .NET Core 2.0, porque es una de las plataformas mejor optimizadas para el trabajo multihilo y el aprovechamiento de los distintos procesadores del servidor. Otra razón para su elección es la implementación del patrón MVC dentro de sus distintas capas de middleware lo que permite un desarrollo rápido a la vez que seguro.

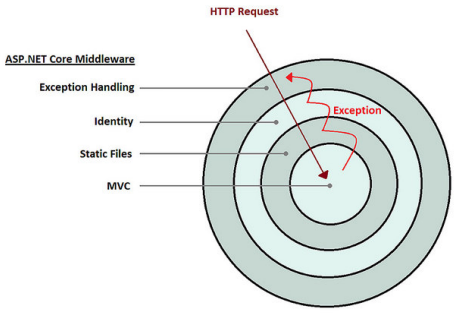


Ilustración 68 - Arquitectura ASP .NetCore

Además, aún siendo tecnología Microsoft, puede ejecutarse sobre sistemas Linux, lo que nos permite tener lo mejor de los dos mundos, el respaldo de una gran empresa como Microsoft y ejecutarse sobre el sistema operativo más seguro y optimizado para su uso en servidores.
Por otro lado el poder programar en C# que es un lenguaje fuertemente tipado proporciona la seguridad de poder depurar y refactorizar de forma segura el código de la aplicación. En el caso del backend, que es donde reside la mayor parte de la lógica de negocio del proyecto es importante poder depurar y refactorizar de forma segura y poder capturar la mayor parte de los errores en tiempo de compilación y no en ejecución.

4.1.7 ExpressJS

El proyecto lo he planteado como una arquitectura de microservicios, compuesta por Backend Service, Auth Service, Scheluder Service y Notification Service, pero es necesario tener un punto de entrada que centralice la comunicación entre la aplicación móvil y todos los microservicios. En este proyecto esta responsabilidad recae sobre el modulo del API, que se encarga de redirigir las peticiones de los clientes móviles y los distintos servicios. Para implementar este modulo he decidido usar ExpressJS, que se ejecuta sobre la plataforma NodeJS.

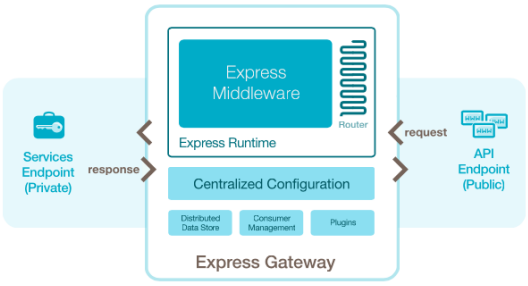
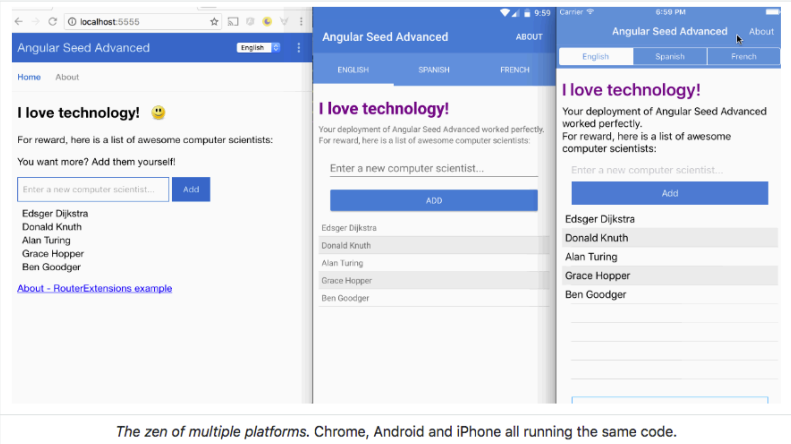


Ilustración 69 - ExpressJS usado como Gateway

ExpressJS es el framework ideal para implementar un gateway porque es muy ligero ya que corre sobre NodeJS y su huella de memoria es mínima. Se programa en JavaScript, que implementa JSON de forma nativa, lo que permite trabajar con los datos recibidos por los clientes móviles y enviarlos al servicio apropiado sin necesidad de estar envolviendo y desenvolviendo dichos datos. El patrón de gateway carece de lógica de negocio, ya que su trabajo consiste básicamente en distribuir peticiones entre los microservicios, y por tanto el usar un lenguaje no tipado reduce la cantidad de código necesario para realizar este trabajo y no supone un gran riesgo a la hora de depurar o refatorizar su código. ExpressJS hace uso del patron middleware, con lo que es facilita crear procesos comunes que se ejecutarán sobre todas las peticiones recibidas, agilizando el desarrollo de este módulo. El enfoque asincrono de ExpressJS permite de forma fácil programar la recepción de peticiones desde el cliente, redirigirlas al servicio adecuado, esperar la respuesta de este y contestar al cliente sin necesidad de complejas estructuras que nos proporcionen la asincronía necesaria para este proceso y sin bloquear el hilo principal de ejecución del programa.

4.1.8 Ionic 3

Para el desarrollo de la aplicación móvil he decidido usar IONIC 3, ya que me permite usar el mismo código fuente para crear la aplicación de Android, iOS y una Progressive Web Applications. Este framework hace uso de Angular 4+, por lo que dispone de mucha documentación para su aprendizaje y de una gran comunidad a parte de Google respaldando el proyecto. Además dispone de una serie de layout pensados para dispositivos móviles que permite adaptarse de forma ideal a cada tamaño de pantalla, facilitando en gran medida la creación de interfaces que deban adaptarse a distintos tamaños de pantalla. Otra de las razones por las que he decidido usar IONIC3 es por estar basado en estándares web, lo que permite aprovechar los conocimientos de dicha plataforma y aplicarlos en el desarrollo móvil. Entre las distintas opciones existentes para el desarrollo de aplicaciones multiplataforma, IONIC es la única que genera tanto la aplicación para Android, iOS, Web y PWA y este factor lo he considerado definitivo a la hora de decantarme por dicha plataforma.



The zen of multiple platforms. Chrome, Android and iPhone all running the same code.

Ilustración 70 - Aplicaciom multiplataforma

4.2 Backend y Gateway

4.2.1 Backend

El Backend de Quedadas lo he implementado usando una arquitectura en tres capas, capa de Presentación, capa de Negocio y capa de Persistencia.

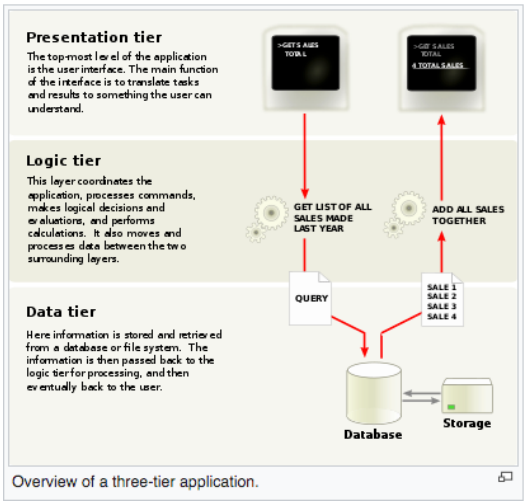


Ilustración 71 - Arquitectura en 3 capas

He usado el patrón **MVC** (Modelo Vista Controlador) para separar las tres capas de la arquitectura, la capa de persistencia contiene los datos del modelo, la vista contiene el JSON devuelto en cada petición y el controlador encapsula la lógica de negocio de cada petición.

```
namespace Quedadas.Controllers
{
    [Route("[controller]")]
    public class MeetingsController : Controller
    {
        private readonly ILoggerAdapter<MeetingsController> _logger;
        private readonly IMeetingsCacheService _meetingsService;

        public MeetingsController(
            ILoggerAdapter<MeetingsController> logger,
            IMeetingsCacheService meetingsService)
        {
            _logger = logger;
            _meetingsService = meetingsService;
        }
```

Ilustración 72 - Ejemplo de controlador

Los controladores reciben las peticiones del API, las procesan, guardando o recuperando los datos que sean necesarios desde el modelo y las envían a la vista para que esta las devuelva al API

```
[HttpPost("{id}/join", Name = "Join")]
[ProducesResponseType(typeof(Meeting), 200)]
[ProducesResponseType(404)]
[ProducesResponseType(400)]
public async Task<IActionResult> Join(string id)
{
    await _meetingsService.AttendToMeeting(id, AttendeeStatus.Maybe);
    var meeting = await _meetingsService.GetMeeting(id);
    SetETagResponseHeader(meeting.ETag);
    return Ok(meeting);
}
```

Ilustración 73 – Petición del Backend para añadir usuarios al evento

El framework asp.net core permite registrar middleware durante el flujo de la petición http recibida. Esto permite centralizar la gestión de errores no controlados, gestión de la respuesta frente excepciones de la capa de Negocio, y realizar las comprobaciones de autenticación de cada petición antes de pasarla al controlador y evitar así repetir estos flujos por cada controlador.

```
public static class AutenticationMiddlewareExtensions
{
    public static IApplicationBuilder UseCustomAuthentication(this IApplicationBuilder builder)
    {
        return builder.UseMiddleware<AuthenticationMiddleware>();
    }
}
```

Ilustración 74 - Ejemplo middleware

En la capa de Persistencia he implementado el patrón **Repository**, este facilita el acceso y persistencia de datos a la vez de abstraerse de la capa de Negocio.

```
namespace Quedadas.Repository
{
    public interface IRepository
    {
        Task Delete<T>(Expression<Func<T, bool>> expression) where T : Metadata;
        Task Delete<T>(T item) where T : Metadata;
        Task DeleteAll<T>() where T : Metadata;
        Task<T> Single<T>(Expression<Func<T, bool>> expression) where T : Metadata;
        Task<long> Count<T>(Expression<Func<T, bool>> expression) where T : Metadata;
        Task<IEnumerable<T>> Get<T>(Expression<Func<T, bool>> expression, int limit, int skip, Sort<T>[] sort = null) where T : Metadata;
        Task Add<T>(T item) where T : Metadata;
        Task Add<T>(IEnumerable<T> items) where T : Metadata;
        Task Upsert<T>(T item) where T : Metadata;
        Task<bool> Update<T>(T item, string eTag) where T : Metadata;
    }
}
```

Ilustración 75 - Patron Repository

4.2.2 Gateway

El API es el encargado de exponer todos los recursos de la plataforma a los clientes móviles. Esto lo hace mediante el patrón Gateway, que consiste en exponer un único servicio al cliente y dicho servicio se encargará de encaminar cada petición del cliente al servicio adecuado. También se encarga de unificar todas las respuestas enviadas al cliente de forma que estas queden homogeneizadas de cara al cliente. En el Anexo C pueden consultarse todas las peticiones gestionadas por el Gateway.

Todas las peticiones responden con el modelo de un evento de Quedadas actualizado tras la operación solicitada. Este modelo se entrega en formato JSON para que pueda ser procesado por los clientes móviles.

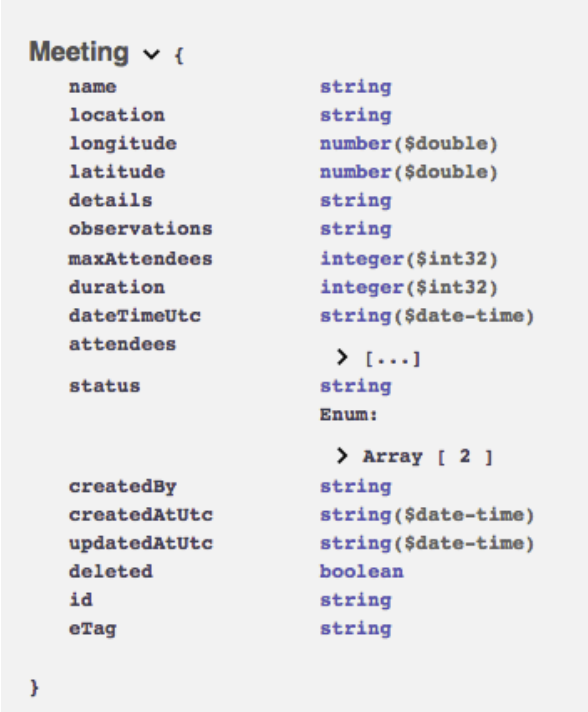


Ilustración 76 - Respuesta del API

- **name**: Título que hemos puesto al evento.
- **location**: Cadena libre para referenciar la ubicación del evento.
- **longitude**: Coordenada longitud de la ubicación del evento.
- **latitud**: Coordenada latitud de la ubicación del evento.
- **details**: Campo libre para indicar en qué consiste el evento.
- **observations**: Campo libre para introducir datos de interés para el asistente al evento.
- **maxAttendees**: Número máximo de participantes.
- **duration**: duración del evento en minutos.
- **dateTimeUtc**: Fecha y hora del evento en formato UTC.
- **attendees**: Listado de usuarios asociados al evento.
- **status**: Estado del evento, puede ser “Open”, “Full” o “Cancelled” según esté abierto, completado o haya sido cancelado.
- **createdBy**: identificador del usuario organizador del evento.
- **createdAtUtc**: Fecha UTC de la creación del evento.
- **updatedAtUtc**: Fecha de la última modificación del evento en formato UTC.
- **deleted**: Indica si el evento ha sido eliminado.
- **id**: identificador del evento.
- **eTag**: identificador para actualizar la caché del cliente móvil.

4.4 Aplicación móvil

La aplicación móvil la he desarrollado usando IONIC 3, que es un framework multiplataforma que hace uso de la tecnología web Angular 2 para crear aplicaciones móviles para Android e iOS. He decidido usar IONIC para poder aprovechar el desarrollo de la aplicación en Android y con el mínimo esfuerzo y tiempo obtener su versión en iOS. Otro motivo ha sido poder aprovechar la gran variedad de plugins existentes en la comunidad Cordova para hacer uso de servicios como notificaciones push, login con redes sociales, enlaces profundos, etc… de forma sencilla.

Para el desarrollo de la aplicación móvil he estructurado el proyecto por páginas, cada página representa una pantalla de la aplicación.

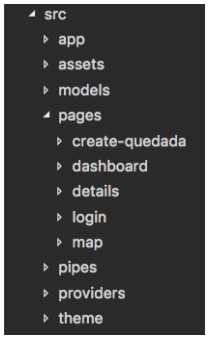


Ilustración 77 - Listado de pantallas

Cada componente de tipo página está compuesto por un fichero html que contiene la implementación de la vista de la pantalla a la que representa en html, un fichero typescript que contiene la lógica de dicha pantalla y un fichero css con los estilos necesarios para formatear la pantalla. También tienen un fichero con el nombre "xxxx.module.ts" donde se indican las dependencias necesarias para el funcionamiento de la pantalla.

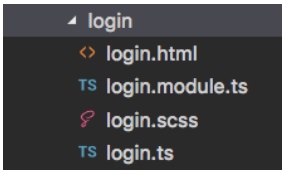


Ilustración 78 - Estructura pantalla

El módulo de entrada a la aplicación es el "app.component.ts" que es el encargado de controlar si el acceso se ha hecho desde un enlace profundo o desde el lanzador del móvil y lanzar la pantalla de dashboard si se ha iniciado desde el lanzador y o la de detalles si viene desde un enlace.
La pantalla de **Dashboard** se encarga de comprobar si el usuario aún no ha iniciado sesión y enviarlo a la pantalla de login en tal caso o mostrar el listado de eventos si ya tiene la sesión iniciada.

```
this.storage.get(this.KEY_TOKEN)
  .then((value) => {
    this.jwtToken = value;
    this.segment = 'ownerSegment';
    if(!this.jwtToken){

      this.navCtrl.push('LoginPage');
    }
    else{
     this.myId = this.jwtHelper.decodeToken(this.jwtToken)._id;
     console.log("Mi id es", this.myId);
     this.getQuedadasFromServer();
    }
  })
  .catch(() => {
    this.jwtToken = null;
    this.navCtrl.push('LoginPage');
  });
```

Ilustración 79 - Comprobar sesión

Esta pantalla se comunica con el API para cargar el modelo de la aplicación con los eventos del usuario y mostrarlos por pantalla, haciendo uso de un pipe para filtrar los eventos que organiza de los que es un invitado.

```
getQuedadasFromServer(){
  let load = this.loadingController.create();
  load.present();
  console.log('Me uqedo en quedadas el valor de token es ', this.jwtToken);
  this.quedadaApiProvider.getQuedadas(this.jwtToken)
    .then((data: any) =>{
      load.dismiss();
      this.quedadas = data.results;
      this.quedadasCopy = this.quedadas;
      console.log(data);
    })
    .catch(error => {
      load.dismiss();
      console.error(error);
    }
    );
}
```

Ilustración 80 - Obtener listado de eventos

Desde esta pantalla podemos navegar a la página de **Detalles** y a la de **Crear**, ambas páginas siguen el mismo patrón que la de **Dashboard**, se comunican con el API para obtener los datos de los eventos e interactuar con ellos y muestran los resultados de dicha comunicación en la vista de la pantalla. En el caso de la pantalla de Detalles, permite cambiar el estado de la asistencia del usuario, mostrar el mapa con la ruta al evento y si el usuario es el organizador también le permite compartir el evento y editarlo.

```
getQuedadaFromServer(){…
}

attendQuedadFromServer(){…
}

noAttendQuedadFromServer(){…
}

maybeAttendQuedadFromServer(){…
}

shareQuedada(quedada: any){…
}

goToEditQuedada(quedada){…
}
```

Ilustración 81 - Acciones sobre un evento

Otra responsabilidad que tiene la pantalla de Detalles es unir al usuario al evento que está visualizando, en caso de que aún no lo esté y no sea el organizador, para que el Backend pueda añadirlo como usuario invitado.

```
joinUserIntoQuedada(){
  let load = this.loadingController.create();
  load.present();
  this.quedadaApiProvider.joinQuedada(this.jwtToken, this.id)
    .then((data: any) =>{…
    })
    .catch(error => {…
    }
    );
}
```

Ilustración 82 - Añadir usuario a un evento

En el caso de la pantalla de **Crear**, nos permite rellenar los datos de nuestro evento y comunicarlo al API para que lo cree en el Backend.

```
createQuedada(){
  let newQuedada={
    "name":this.quedadaForm.controls['titulo'].value,
    "details": this.quedadaForm.controls['queHacer'].value,
    "observations": this.quedadaForm.controls['queSaber'].value,
    "location": this.quedadaForm.controls['direccion'].value,
    "latitude": this.latitudeMap,
    "longitude": this.longitudeMap,
    "maxAttendees": this.quedadaForm.controls['people'].value,
    "duration": this.quedadaForm.controls['duracion'].value,
    "DateTimeUtc":  new Date(
      this.quedadaForm.controls['dateQuedada'].value + 'T'+ this.quedadaForm.controls['dateQuedadaH'].value
    ).toISOString(),
  }

  this.quedadasApiProvider.createQuedada(newQuedada)
    .then(data =>{
      // Quedada created successfully
      this.navCtrl.pop();

    })
    .catch(error =>{
      // Error creating Quedada
      console.error(error);
    });
}
```

Ilustración 83 - Crear evento

En la aplicación hago uso del patrón ModelView-ViewModel (MVVM) para insertar la información que se muestra en cada pantalla a partir del modelo creado para la aplicación. Este patrón hace uso de la técnica de "double binding" para enlazar los datos entre la vista y el modelo en dos direcciones, los cambios que se realicen en la vista afectarán al modelo y los que se hagan en el modelo quedarán representados en la vista.

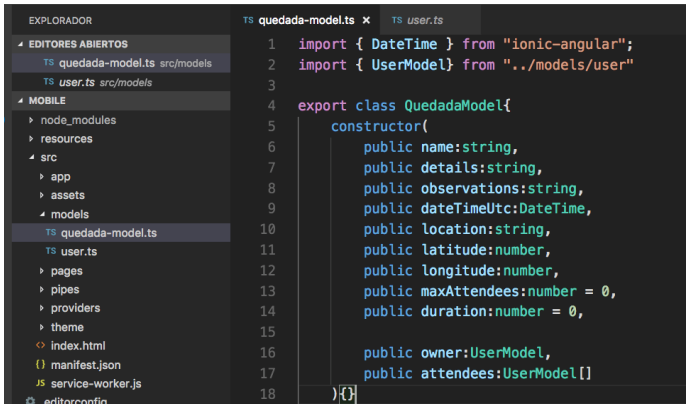


Ilustración 84 - Modelo del evento

En la aplicación he implementado un servicio de comunicaciones que, para comunicar la aplicación móvil con el API, este servicio encapsula todas las funcionalidades que provee el API y se inyecta como **provider** en los componentes página que la necesitan, como el **Login**, **Dashboard**, **Crear** o **Detalles**.

```
@Injectable()
export class QuedadasApiProvider {

  path: string = 'https://api.quedadas.ovh';
  private KEY_TOKEN = 'token_jwt';
  public jwtToken = null;
  jwt: string = '';

  constructor(public http: Http, public storage: Storage) {…
  }
  getQuedadas(token: string){…
  }
  getQuedada(token: string, id: string){…
  }
  joinQuedada(token: string, id: string){…
  }
  attendQuedada(token: string, id: string){…
  }
  doNotAttendQuedada(token: string, id: string){…
  }
  maybeAttendQuedada(token: string, id: string){…
  }
  createQuedada(quedada: any){…
  }
  private getToken() {…
  }
}
```

Ilustración 85 - Servicio restful

Todos los módulos y servicios son inyectados en cada componente de página usando el inyector de dependencias de Angular 2, que hace uso de la técnica “Lazy Load” la cual permite que se carguen las dependencias necesarias para cada componente sólo cuando se acceda a dicho componente, evitando la carga inicial de todas las dependencias al inicio de la aplicación.

En el anexo D se encuentran los detalles de cada plugin usado y como configurarlo.

Como resultado de esta implementación he obtenido una aplicación móvil capaz de crear pequeños eventos, publicarlos en aplicaciones de mensajería instantánea como Whatsapp y gestionar sus asistentes. El enlace para descargar la aplicación desde el Play Store está disponible en la web https://quedadas.ovh.

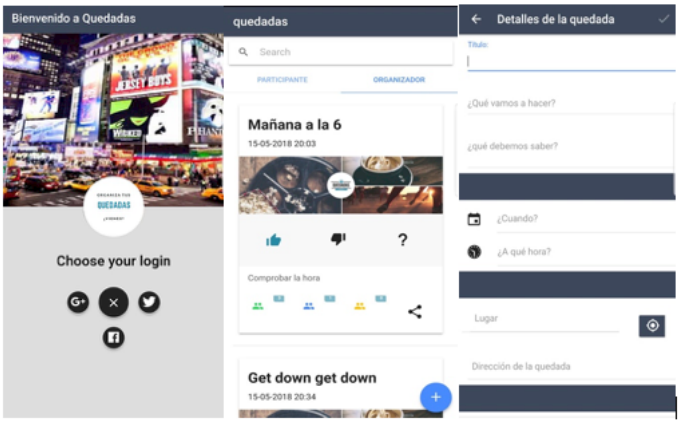


Ilustración 86 - Quedadas App

En el anexo A se encuentra el manual de usuario donde aparecen las pantallas principales de la aplicación.

5. Pruebas

Este apartado contiene los casos de pruebas funcionales diseñadas para la validación de la solución y su correcto funcionamiento. En el Anexo C he descrito cada uno de los escenarios de pruebas y el procedimiento seguidos para realizarlas.

Test	Descripción	Modelo	SO	Resultado
CT1	Login	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.0	OK
CT2	Login	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.1	OK
CT3	Login	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.2	Pendiente
CT4	Crear Eventos	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.3	OK
CT5	Crear Eventos	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.4	Pendiente
CT6	Publicar Eventos	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.5	OK
CT7	Consultar Evento	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.6	OK
CT8	Consultar Asistentes	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.7	OK
CT9	Modificar una Quedada	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.8	OK
CT10	Recibir notificación asistencia	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.9	OK
CT11	Cancelar una Quedada	SM-G955F. Samgung Galaxy S8+	Android 8.0.0. Samsung Experience 9.10	OK

6. Conclusiones

Desde el punto de vista académico este proyecto me ha servido para afianzar los conocimientos adquiridos durante el máster en materias como el diseño de aplicaciones, desarrollo para aplicaciones móviles y adquisición de requisitos centrados en el usuario, además de ampliar mi experiencia en la gestión integral de un proyecto.

El desarrollo de la aplicación se ha realizado usando la metodología de Diseño Centrado en el Usuario (DCU), lo que me ha permitido obtener información de los potenciales usuarios en una etapa temprana del proyecto y poder utilizar dicha información en el diseño de un modelo no funcional. El diseño no funcional me ha permitido volver a validar la propuesta con los usuarios de la aplicación antes de acometer el desarrollo final de esta, lo que me ha servido para poder centrar el esfuerzo en las características más demandadas por los potenciales usuarios.

Desde el punto de vista técnico decidí que la tecnología para desarrollar la aplicación sería Ionic, por ser un framework multiplataforma, lo que me permitiría a futuro, implementar la versión de iOS aprovechando la mayor parte del código de la aplicación de Android.

Además, no hace uso de técnicas de compilación y/o generación de bytecode a código nativo, como React Native o NativeScript. En su lugar crea la aplicación haciendo uso de un webview. Esto permite migrar la aplicación de forma rápida a una "Progressive Web App" (PWA) que es un enfoque deseable ya que permite evitar la necesidad de instalación de la aplicación, que es uno de los grandes problemas de la adquisición de usuarios además de un coste ya que las tiendas de aplicaciones se quedan con un 30% de los ingresos realizados mediante la aplicación. Esto que en un principio plateé como una ventaja también a resultado ser una experiencia de usuario menos atractiva y fluida que la que podríamos haber obtenido con otros frameworks multiplataforma como los nombrados anteriormente.

Al inicio del proyecto planteé crear una plataforma que permitiese crear eventos, distribuirlos de forma sencilla por sistemas de mensajería instantánea como Whatsapp, Telegram o Hangout y gestionar su asistencia. Aunque el objetivo general se ha cumplido, ya que he conseguido desplegar una plataforma funcional que permite crear pequeños eventos de carácter general, publicarlos y gestionar su asistencia, dos de las funcionalidades especificadas al inicio del proyecto no las he podido implementar, el login con Google y duplicar un evento.

El no haber podido desarrollar dos de las funcionalidades propuestas se ha debido a una desviación en la planificación inicial y por tanto la falta de tiempo para su implementación.

Esta desviación se ha debido a que durante la fase de investigación busqué todos los plugins que serían necesarios en el proyecto y valoré la dificultad de integrarlo en el proyecto, pero no tuve en cuenta los posibles conflictos que podrían surgir entre dichos plugins. Durante la fase de implementación me encontré con incompatibilidades entre los plugins de Login con Google, Google Maps y Notificaciones Push. Frente a esta problemática valoré que el plugin menos importante es el login con Google, ya que

disponía de esta misma funcionalidad con Twitter y Facebook, lo que podría ser suficiente para la funcionalidad de registro en la aplicación, así que decidí concentrar el esfuerzo en hacer funcionar juntos los plugins de notificaciones y mapas, que son necesarios para funcionalidades básicas de la aplicación, como notificar los cambios de estado de un asistente y mostrar el mapa para ubicar el evento. La resolución de estos conflictos consumió un tiempo que no había estimado dentro de la implementación, lo que me obligó a replantear las funcionalidades del proyecto y decidí dejar fuera la implementación de duplicar un evento, ya que bajo mi criterio no afecta de forma crítica al uso de la plataforma, sino que supone una característica para facilitar la creación de nuevos eventos a partir de eventos existentes.

6.1 Desarrollos futuros

Los desarrollos mas inmediatos que implementar en el proyecto serían acabar las dos funcionalidades que han quedado pendientes para completar la experiencia de usuarios que organizan eventos que se repiten periódicamente.

Otra funcionalidad que añadiría es permitir que el usuario pueda configurar la imagen y textos que se muestran cuando publica una invitación en WhatsApp o Telegram.

Para ampliar el número de potenciales usuarios de la aplicación añadiría internacionalización (i18n) para disponer de la aplicación también en al menos inglés.

La decisión de hacer el desarrollo sólo en Android se basó en la cuota de mercado de este sistema operativo frente a la de iOS y reducir tiempos para alcanzar el mercado lo antes posible y validar la idea con usuarios reales. Pero el hecho de haber usado un framework multiplataforma hace atractivo afrontar el desarrollo de la versión para iOS como uno de los siguientes pasos y así cubrir casi el 98% de la cuota de mercado de dispositivos móviles.

Transformar la aplicación en una Progressive Web Application para evitar la necesidad de necesitar el paso de instalación.

Otro desarrollo a estudiar para abordar en el futuro sería integrar plataformas de pago como Paypal, que ahora permiten transferir dinero entre particulares sin coste alguno, esta nueva funcionalidad permitiría a un usuario crear un evento que tuviese un coste asociado y recolectar el dinero desde la aplicación. Esta funcionalidad sería muy útil en eventos del tipo deportivos, donde existe un coste como el alquiler de las instalaciones asociado al evento, en este caso el organizador podría indicar como requisito para unirse al evento el pago de dicha cantidad.
Además esta última funcionalidad podría abrir una posible forma de monetizar el proyecto, centralizando los pagos en una cuenta de Paypal que sería la encargada de recoger el dinero y posteriormente entregárselo al organizador tras descontar una pequeña comisión.

7. Glosario

Android: Sistema operativo para dispositivos móviles creado por Google.
API: Application Programming Interfaces, Es la interfaz de programación de una aplicación.
Backend: Aplicación web que provee de cierta lógica de negocio a sus clientes mediante un API.
CRUD: Siglas de Create, Read, Update y Delete, que indican que se proveerá de los mecanismos necesarios para crear, leer, modificar y eliminar un dato de de un conjunto de datos.
DCU: Metodología para la recogida de requisito basada en el Diseño Centrado en el Usuario.
Endpoint: Punto de entrada del api Restful.
ExpressJS: Framework de NodeJS para servidores web.
Gateway: Pasarela para encaminar peticiones entre servicios.
Google Play Store: Tienda de aplicaciones de Google para los dispositivos móviles con su sistema operativo Android.
Facebook: Empresa americana basada en la red social del mismo nombre.
Hangout: Sistema de mensajería instantánea creado por Google.
HTTP: protocolo para visualización de documentos en internet.
IDE: Interface Development Environment, entorno de desarrollo utilizado.
JWT: JSON Web Token, credenciales de un usuario encriptadas en forma de cadena de texto en base 64.
Linux: Sistema operativo open source.
LRU: Last Recent Used, algoritmo de ordenación basado en el uso de los datos, ordena los datos en función de la última vez que se consultaron.
Microservicio: Paradigma por el que se separa una aplicación en aplicaciones mas pequeñas independientes entre ellas y con sus propias responsabilidades.
Microsoft: Empresa de software creadora del sistema operativo Windows.
Middleware pattern: Patron de diseño para añadir flujos comunes a distintos flujos de una aplicación.
MongoDB: Base de datos No SQL.
MVC: Patrón de diseño software que separa cada aplicación en tres capas, el Modelo, la Vista y el Controlador, cada una con sus responsabilidades.
Restful: servicio para el intercambio de datos en formato JSON usando el protocolo http.
Skype: Sistema de mensajería instantánea creado por Microsoft.
SMS: (Short Message Service) Servicio de mensajes de texto ofrecido por las operadoras telefónicas.
Telegram: Servicio de mensajería instantánea cuya fortaleza es la seguridad de las comunicaciones.
Twitter: Red social basada en mensajes cortos.
Upsert: Operación que realiza una inserción o actualización en base de datos de forma atómica.
UX: User eXperience, experiencia de usuario.
Whatsapp: Servicio de mensajería instantánea perteneciente a Facebook

8. Bibliografía

Web	Enlace	Fecha visita
Tech Crunch	https://techcrunch.com/2016/10/07/facebook-events-app/	6-Marzo-2018
Top Apps	http://www.topapps.net/apple-ios/meetup-gets-overhauled-for-ios-7.html/	6-Marzo-2018
The Next Web	https://thenextweb.com/google/2015/04/08/yaass-google-calendar-month-view-is-back/	7-Marzo-2018
Yalantis	https://yalantis.com/blog/event-planner-apps-technology-ever-wondered-eventbrite-others-kind-work/	7-Marzo-2018
Mobilize	https://mobilize.io/blog/mobilize-now-android/	7-Marzo-2018
Microsoft CI	https://www.visualstudio.com/es/learn/what-is-continuous-integration/?rr=https%3A%2F%2Fwww.google.es%2F	6-Abril-2018
Visual Studio Code	https://code.visualstudio.com/	6-Abril-2018
GIT SCM	https://git-scm.com/	8-Abril-2018
Bitbucket	https://bitbucket.org	8-Abril-2018
Wikipedia CI	https://es.wikipedia.org/wiki/Integraci%C3%B3n_continua	10-Abril-2018
Martin Fowler CI	https://www.martinfowler.com/articles/continuousIntegration.html	10-Abril-2018
Jenkins CI	https://jenkins.io/	10-Abril-2018
Jenkins vs CCI	https://circleci.com/migrate-jenkins-to-circleci/?utm_source=Google&utm_medium=SEM&utm_campaign=(Alpha)%20Search%20Activation%20Non%20Branded&utm_content=(Alpha)%20Search%20Activation%20Non%20Branded-Eng-NonBranded-Jenkins&utm_term=NewJenk2&gclid=Cj0KCQjwtZzWBRD2ARIsAIPenY3WFaT7SmL-a4n67osLJwSebDQflziiSSXtTIyP9pNOJAk2l_azv6kaAio5EALw_wcB	10-Abril-2018
StatusCake features	https://www.statuscake.com/features/	11-Abril-2018
Azure Application Insights	https://docs.microsoft.com/en-us/azure/application-insights/app-insights-overview	11-abril-2018
Digital Oceans	https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations	11-abril-2018
PM2	http://pm2.keymetrics.io/	11-Abril-2018
MongoDB	https://www.mongodb.com/compare/mongodb-mysql	12-Abril-2018
ASP .Net Core 2.0	https://docs.microsoft.com/es-es/aspnet/core/	18-Abril-2018
ASP .Net Middleware	https://github.com/MookieFumi/MyNetCoreWebApi	18-Abril-2018
ExpressJS Gateway	https://medium.com/@nodejs/q-a-with-express-gateway-team-a-microservice-api-gateway-built-on-express-ebb3b18b5bdb	19-Abril-2018
ExpressJS	https://expressjs.com/	19-Abril-2018
IONIC	https://ionicframework.com/docs/	20-Abril-2018
Nathan Walker	https://github.com/NathanWalker/angular-seed-advanced	20-Abril-2018
Three-tier architecture	https://en.wikipedia.org/wiki/Multitier_architecture#Three-tier_architecture	1-Mayo-2018

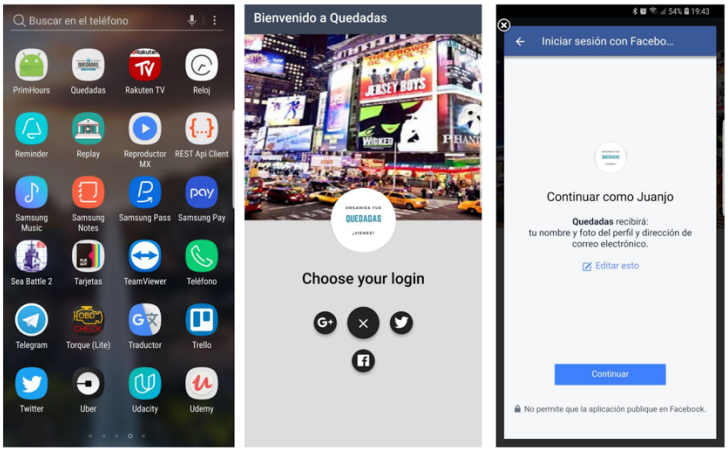
9. Anexos

Anexo A: Manual de usuario aplicación móvil

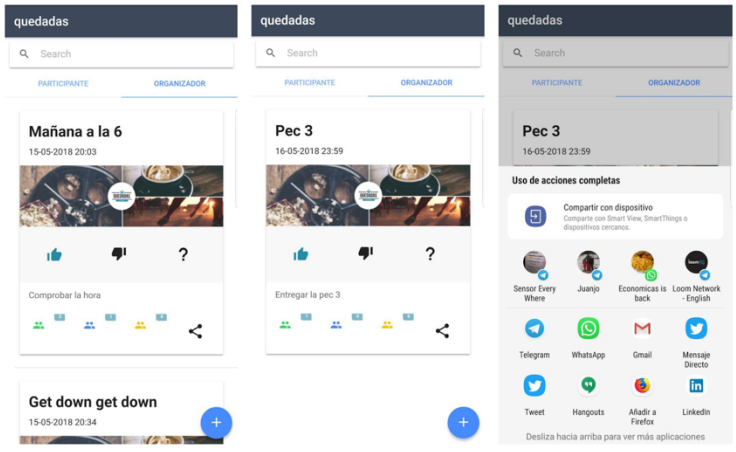
El enlace para descargar la aplicación desde el Play Store está disponible en la web https://quedadas.ovh. Para probar la aplicación puede usarse el usuario de prueba de Facebook siguiente:

Usuario: uoc_yjxffiq_test@tfbnw.net
Password: qwerty123456

Una vez que la aplicación ha sido instalada, desde el Play Store o directamente desde el APK, aparecerá su icono en el lanzador del móvil, si la abrimos nos aparecerá la pantalla de splash durante unos segundos y a continuación la pantalla para iniciar sesión, podremos elegir entre las redes sociales de Facebook, Twitter o Google para iniciar sesión en la aplicación.



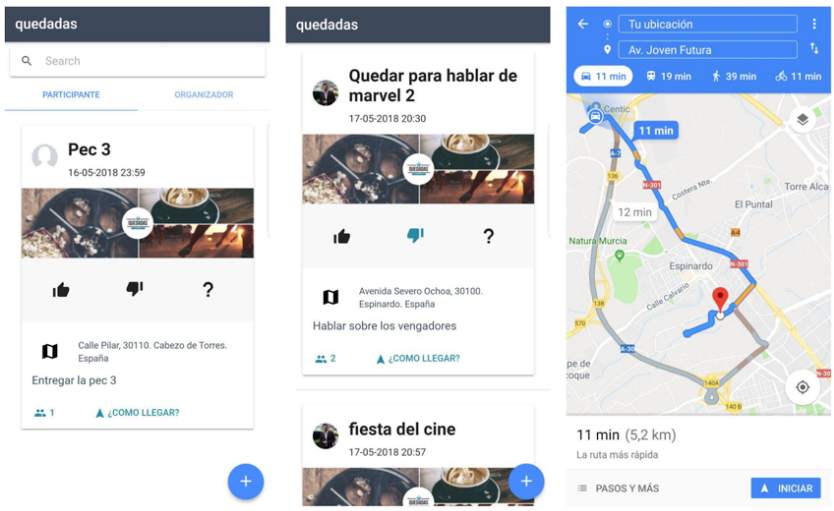
Una vez dentro de la aplicación nos mostrará el listado de Quedadas en las que ya estamos participando o las que estamos organizando y un buscador para filtrar por el nombre de la Quedada. Si aún no estamos en ninguna Quedada el listado estará vacío.



En la pestaña "**Organizador**" nos aparecerán las Quedadas que estamos organizando, sobre estas Quedadas podremos cambiar nuestra asistencia usando la botonera de

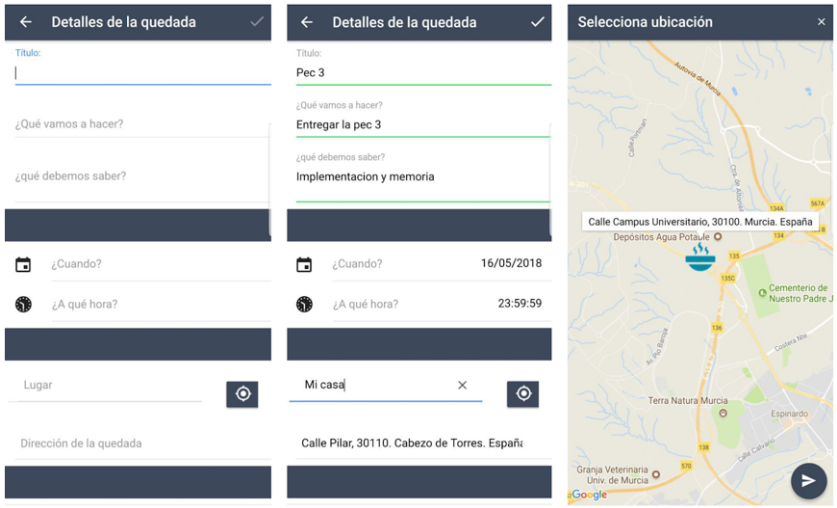
asistencia, publicar la Quedada mediante las aplicaciones que tengamos instaladas pulsando el botón de compartir o acceder a los detalles de la Quedada pulsando sobre el título de la Quedada.

Si seleccionamos la pestaña de "**Participante**" nos aparecerán las Quedadas en las que estamos participando.

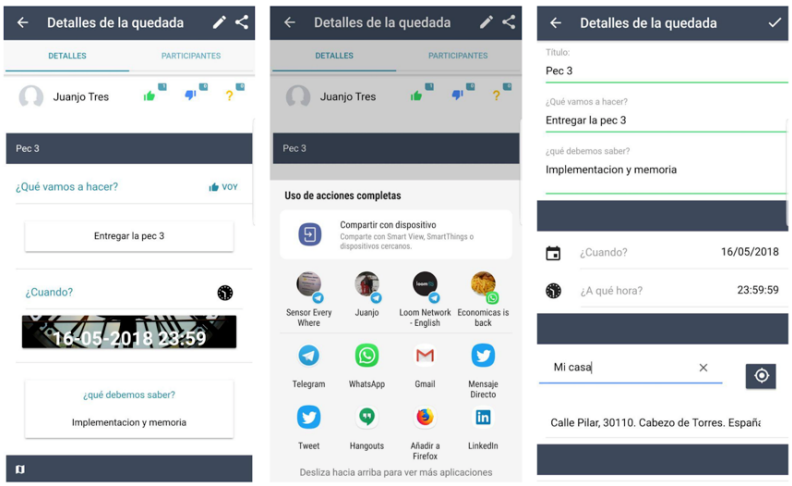


En estas quedadas podremos cambiar el estado de nuestra asistencia pulsando sobre la botonera de asistencia, calcular la ruta hasta la Quedada usando el botón de "¿Cómo llegar? o acceder a los detalles pulsando sobre el título de la Quedada.

Si pulsamos sobre el botón **(+)** accederemos a la pantalla de crear una nueva Quedada.

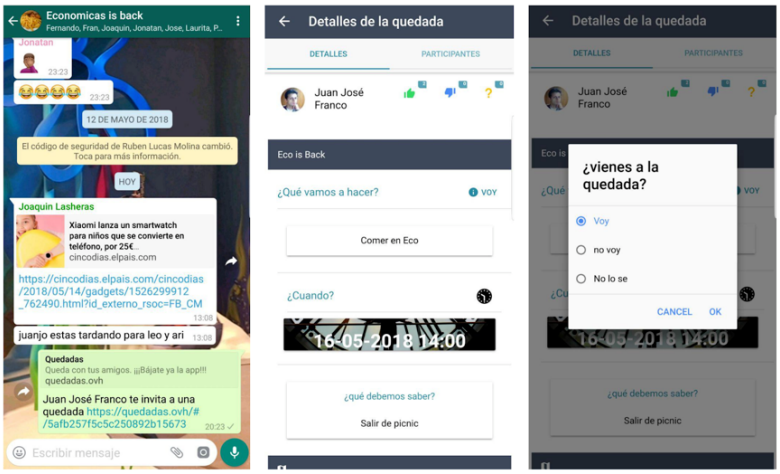


En esta pantalla tenemos el formulario introducir los datos de la Quedada y una pantalla de mapa para insertar la ubicación de la Quedada.
Si accedemos a los detalles de una Quedada de la que somos organizadores podremos compartir, editar y cambiar nuestra asistencia en la Quedada.

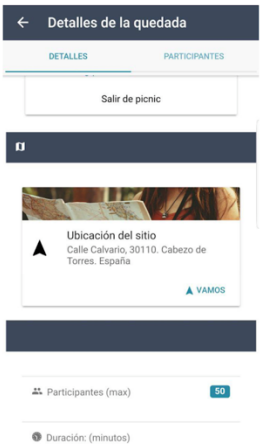


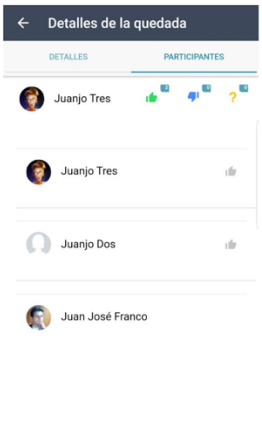
Si editamos los datos de una Quedada todos los participantes recibirán una notificación indicándoles que esta ha sido modificada.

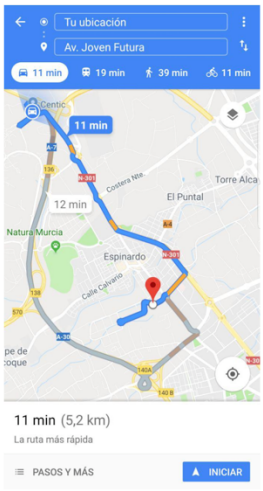
Al detalle de una Quedada en la que participamos podemos acceder desde el listado de Quedadas o desde una invitación que nos hayan compartido. En este caso no podremos ni editar ni compartir la Quedada.



Tanto si somos organizadores como si somos participantes podremos cambiar nuestra asistencia y cada vez que la cambiemos el organizador recibirá una notificación con nuestra nueva asistencia. Podremos ver el listado de participantes y el estado de su asistencia y ver la ruta hasta el evento.


Detalles de la quedada
DETALLES
PARTICIPANTES
Salir de picnic
Ubicación del sitio
Calle Calvario, 30110. Cabezo de Torres. España
VAMOS
Participantes (max)
50
Duración: (minutos)


Detalles de la quedada
DETALLES
PARTICIPANTES
Juanjo Tres
Juanjo Tres
Juanjo Dos
Juan José Franco


Tu ubicación
Av. Joven Futura
11 min
19 min
39 min
11 min
11 min
12 min
Torre Alca
El Puntal
Natura Murcia
Espinardo
Calle Calvario
11 min (5,2 km)
La ruta más rápida
PASOS Y MÁS
INICIAR

Anexo B: Manual de instalación

Para la instalación del Backend y el API es necesario disponer de un servidor Linux con nginx 1.10, Mongodb 3.4, NodeJS 8 y ASP Core .NET 2.0

Pasos:

- Instalar PM2 con el comando “npm i pm2 -g”
- Ir a la raíz de la carpeta “api-gateway”
- Ejecutar el comando “npm i”
- Ejecutar el comando “npm run build”
- Ejecutar el comando “pm2 start build/index.js –name=api”
- Ir a la raíz de la carpeta “backend”
- Ejecutar “dotnet restore”
- Ejecutar “dotnet build”
- Ejecutar “dotnet publish Quedadas/Quedadas.csproj -c Release”

Para poder comunicar el Gateway con la aplicación móvil este debe estar publicado en internet, en el anexo F está el fichero usado en nginx para exponer el api en internet.

Para compilar la aplicación móvil es necesario tener instalado NodeJS 8, Ionic 3.20 y Cordova 8.0.0.

Pasos:

- Ir a la raíz de la carpeta “mobile”
- Ejecutar el comando “ionic cordova platform add android”
- Ejecutar el comando “ionic cordova build android –prod –release”
- El APK se genera en la subcarpeta “platform/Android/app/build/output/apk/”

La forma mas cómoda de instalar el APK generado es enviarlo al móvil, ya sea por correo, whatsapp o similar e instalarlo desde el propio móvil.

Nota: La versión 2.1.2 del plugin FCM tiene un conflicto de dependencias con la librería Google Service, este conflicto estará resuelto en la próxima versión del plugin, no obstante, en el anexo D explico como solventarlo de forma manual.

Anexo C: Escenarios de pruebas

ID	CT1
FUNCIONALIDAD	Login
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Acceder a la aplicación usando una cuenta de Facebook
PRECONDICIONES	Tener la aplicación instalada
PROCEDIMIENTO	1. Abrir la aplicación 2. Pinchar en el menu de login 3. Pinchar en el icono de Facebook 4. Introducir los datos de la cuenta de Facebook 5. Pinchar en el botón de autorizar
RESULTADOS ESPERADOS	1. Se muestra la pantalla de listado de Quedadas
ESTADO	OK
ANOTACIONES	

ID	CT2
FUNCIONALIDAD	Login
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Acceder a la aplicación usando una cuenta de Twitter
PRECONDICIONES	Tener la aplicación instalada
PROCEDIMIENTO	1. Abrir la aplicación 2. Pinchar en el menu de login 3. Pinchar en el icono de Facebook 4. Introducir los datos de la cuenta de Facebook 5. Pinchar en el botón de autorizar
RESULTADOS ESPERADOS	1. Se muestra la pantalla de listado de Quedadas
ESTADO	OK
ANOTACIONES	

ID	CT3
FUNCIONALIDAD	Login
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Acceder a la aplicación usando una cuenta de Gmail
PRECONDICIONES	Tener la aplicación instalada
PROCEDIMIENTO	1. Abrir la aplicación 2. Pinchar en el menu de login 3. Pinchar en el icono de Gmail 4. Introducir los datos de la cuenta de Gmail

	5. Pinchar en el botón de autorizar
RESULTADOS ESPERADOS	1. Se muestra la pantalla de listado de Quedadas
ESTADO	Pendiente
ANOTACIONES	Incompatibilidad de plugins, por la librería google service, descartado para esta entrega

ID	CT4
FUNCIONALIDAD	Crear Eventos
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil quiero poder crear eventos que contengan la fecha de inicio del evento, su duración, su ubicación y el máximo de participantes si procede.
PRECONDICIONES	Usuario logueado
PROCEDIMIENTO	1. Pinchar el el icono (+) del listado de Quedadas 2. Insertar "Mi Evento" en el campo Título 3. Rellenar campo "¿Que vamos a hacer?" con el texto "Salir de paseo" 4. Rellenar campo "¿Que debemos saber?" con el texto "llevar zapatilals cómodas" 5. Pinchar sobre el campo "¿Cuando?" 6. Seleccionar la fecha 18-07-2018 7. Pinchar el botón done 8. Pinchar el campo "¿A que hora?" 9. Seleccionar la hora 20:00:00 10. Pinchar en el campo "Lugar" e introducir "Embalse Santomera" 11. Pinchar sobre el icono localización 12. Mantener pulsado el icono del mapa y arrastrar hasta la entrada de la presa 13. Pinchar el botón "flecha" 14. Deslizar el slider de "Participantes" hasta el valor 10 15. Pinchar en el botón "guardar"
RESULTADOS ESPERADOS	1. La aplicación vuelve al listado de Quedadas 2. La nueva Quedada aparece en el listado
ESTADO	OK
ANOTACIONES	

ID	CT5
FUNCIONALIDAD	Crear Eventos
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil quiero poder duplicar un evento existente introduciendo simplemente una nueva fecha
PRECONDICIONES	Usuario logueado
PROCEDIMIENTO	
RESULTADOS	1. La aplicación vuelve al listado de Quedadas

ESPERADOS	2. La nueva Quedada aparece en el listado
ESTADO	Pendiente
ANOTACIONES	Pendiente por falta de tiempo para su implementación

ID	CT6
FUNCIONALIDAD	Publicar Eventos
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil quiero publicar un evento en mis grupos de WhatsApp y Telegram y poder abrir la aplicación con la información del evento desde dicha publicación.
PRECONDICIONES	1. Usuario logueado 2. Queda "Mi Evento" creada" 3. Telegram instalado
PROCEDIMIENTO	1. Buscar Quedada "Mi Evento" en el listado de Quedadas 2. Pulsar sobre el botón compartir de la Quedada 3. Elegir compartir por Telegram 4. Pinchar un grupo de Telegram 5. Salid de la aplicación 7. Acceder al grupo de Telegram donde se ha publicado la Quedada 8. Pinchar sobre el enlace publicado en el grupo de telegram
RESULTADOS ESPERADOS	1. La aplicación vuelve al listado de Quedadas 2. El grupo de telegram tiene un mensaje con el enlace a la Quedada 3. La aplicación abre el detalle del evento al pinchar sobre el enlace de Telegram
ESTADO	OK
ANOTACIONES	

ID	CT7
FUNCIONALIDAD	Consultar Evento
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil quiero ver la información del evento calcular la ruta al evento y confirmar o cancelar mi asistencia al evento accediendo desde la invitación.
PRECONDICIONES	1. Usuario logueado 2. Queda "Mi Evento" creada" 3. Tener "Mi Evento" publicado en un grupo de Telegram
PROCEDIMIENTO	1. Buscar Quedada "Mi Evento" en el listado de Quedadas 2. Pinchar en el enlace de "Mi Evento" 3.
RESULTADOS ESPERADOS	1. La aplicación vuelve al listado de Quedadas 2. El grupo de telegram tiene un mensaje con el enlace a la Quedada
ESTADO	OK
ANOTACIONES	

ID	CT8
FUNCIONALIDAD	Consultar Asistentes
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil quiero poder ver los asistentes al evento.
PRECONDICIONES	1. Usuario logueado 2. Queda "Mi Evento" creada"
PROCEDIMIENTO	1. Buscar Quedada "Mi Evento" en el listado de Quedadas 2. Pinchar en el titulo de "Mi Evento" 3. En la pantalla de detalles del evento pinchar pestaña "Asistentes"
RESULTADOS ESPERADOS	1. La aplicación muestra la pantalla de listado de asistentes
ESTADO	OK
ANOTACIONES	

ID	CT9
FUNCIONALIDAD	Modificar una Quedada
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil y creador del evento quiero poder modificar la fecha o ubicación del evento y que este cambio sea notificado a todos los usuarios que hayan confirmado su asistencia al evento para que puedan cambiarla si procede.
PRECONDICIONES	1. Usuario logueado 2. Queda "Mi Evento" creada"
PROCEDIMIENTO	1. Buscar Quedada "Mi Evento" en el listado de Quedadas 2. Pinchar en el titulo de "Mi Evento" 3. En la pantalla de detalles pinchar en el botón "editar" 4. Cambiar la fecha en el campo "¿A que hora?" 5. Pinchar el botón "Guardar"
RESULTADOS ESPERADOS	1. La aplicación vuelve a la pantalla de detalles con la nueva hora del evento 2. Los usuarios unidos a la quedada reciben una notificación de cambio en la quedada
ESTADO	OK
ANOTACIONES	

ID	CT10
FUNCIONALIDAD	Recibir notificación asistencia
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil y creador del evento quiero ser notificado cada vez que un usuario de la aplicación confirme o cancele su asistencia a un evento.

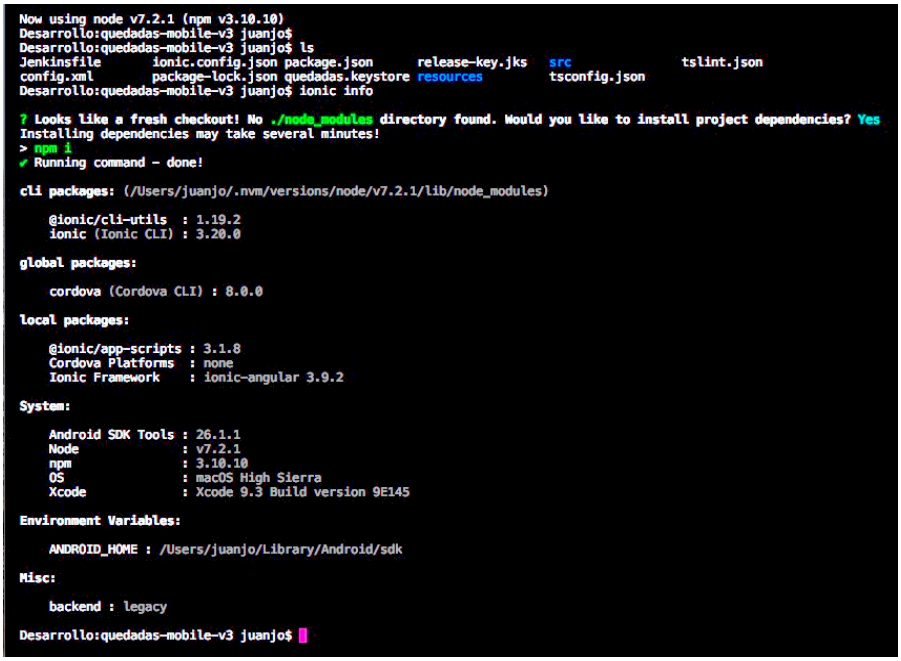
PRECONDICIONES	1. Usuario logueado 2. Estar unido a la quedada "Mi Evento"
PROCEDIMIENTO	1. Buscar Quedada "Mi Evento" en el listado de Quedadas 2. Pinchar en el botón "No voy"
RESULTADOS ESPERADOS	1. El usuario organizador del evento recibe la notificación
ESTADO	OK
ANOTACIONES	

ID	CT11
FUNCIONALIDAD	Cancelar una Quedada
DISPOSITIVO	SM-G955F. Samgung Galaxy S8+
SO	Android 8.0.0. Samsung Experience 9.0
DESCRIPCIÓN	Como usuario de la aplicación móvil y creador del evento quiero poder cancelar un evento y que todos los participantes sean notificados.
PRECONDICIONES	1. Usuario logueado 2. Ser organizador de "Mi Evento"
PROCEDIMIENTO	1. Buscar Quedada "Mi Evento" en el listado de Quedadas 2. Pinchar en el titulo de la Quedada 3. Pinchar en botón cancelar
RESULTADOS ESPERADOS	1. La Quedada desaparece del listado de Quedadas
ESTADO	OK
ANOTACIONES	

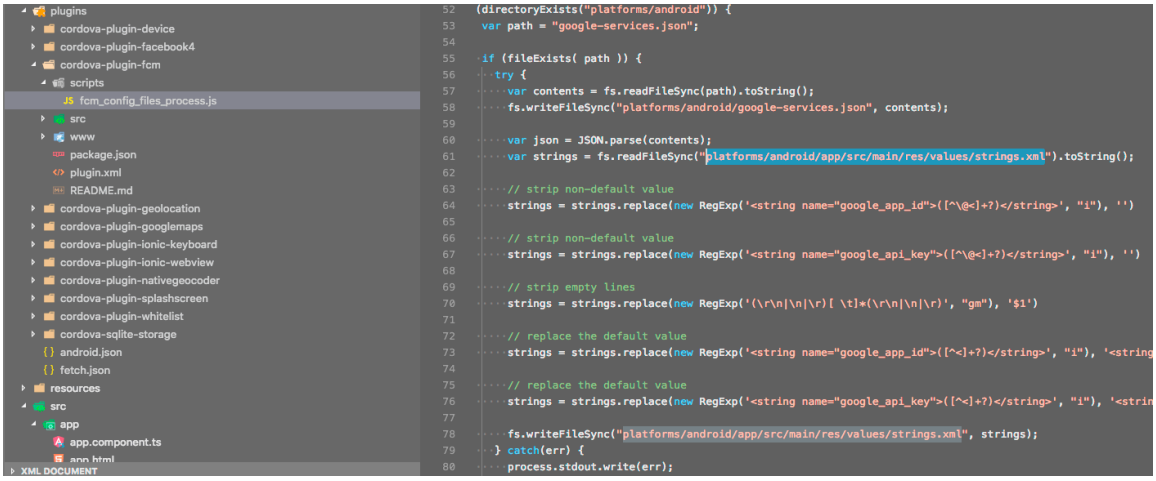
Anexo D: Plugins para IONIC

Plugin FCM

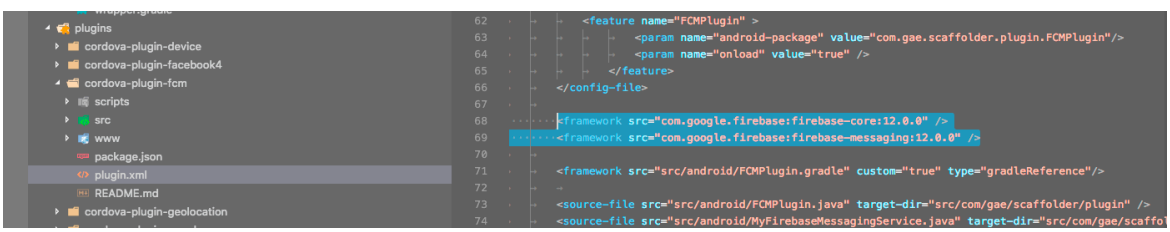
1 Actualizar ionic y cordova



2 Arreglar rutas del plugin



3 Arreglar versiones del plugin, subir a la 12.0.0

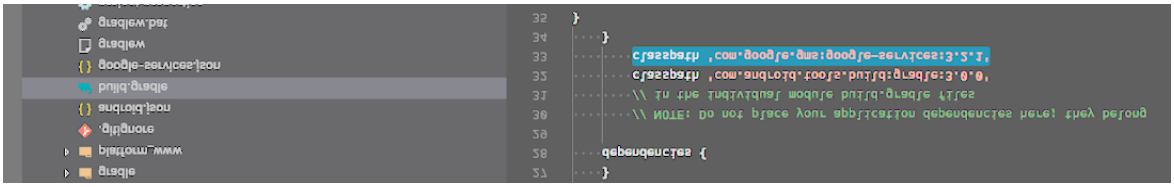


4 Reconstruir platform
ionic cordova platform rm android

ionic cordova platform add android

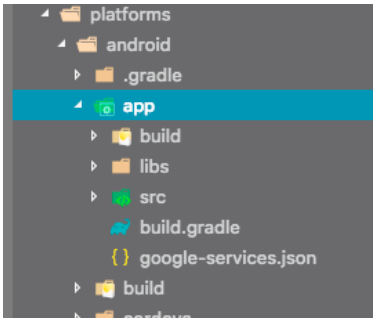
5 Modificar plugin de platform [platform>android>cordova-plugin-fcm>quedadas-FCMPlugin.gradle]

6 Añadir classpath build.gradle del proyecto [platform>android>build.gradle]



7 Añadir apply plugin al build.gradle de app [platform>android>app>build.gradle]

8 Añadir google-services.json [platform>android>app>google-services.json]



Con esto debería funcionar Mapa y FCM al mismo tiempo.

Plugin Deeplinks

1 Instalar plugin.

cordova plugin add ionic-plugin-deeplinks --variable URL_SCHEME=quedadas --variable DEEPLINK_SCHEME=https --variable DEEPLINK_HOST=quedadas.ovh

2 Instalar ts

```
npm install --save @ionic-native/deeplinks
```

3 Usar Deeplink

```
import { Deeplinks } from '@ionic-native/deeplinks';

constructor(private deeplinks: Deeplinks) { }

this.deeplinks.route({
     '/:quedadaId': QuedadaDetailsPage
   }).subscribe((match) => {
     // match.$route - the route we matched, which is the matched entry from
the arguments to route()
     // match.$args - the args passed in the link
     // match.$link - the full link data
     console.log('Successfully matched route', match);
   }, (nomatch) => {
     // nomatch.$link - the full link data
     console.error('Got a deeplink that didn\'t match', nomatch);
   });
```

https://ionicframework.com/docs/native/deeplinks/

Plugin SocialShare

https://ionicframework.com/docs/native/social-sharing/

```
ionic cordova plugin add cordova-plugin-x-socialsharing

npm install --save @ionic-native/social-sharing
```

Dashboard.module.ts

```
import { NgModule } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { DashboardPage } from './dashboard';
import { SocialSharing } from '@ionic-native/social-
sharing';

@NgModule({
 declarations: [
   DashboardPage,
 ],
 imports: [
```

```
    IonicPageModule.forChild(DashboardPage),
  ],
  providers: [SocialSharing]
})
export class DashboardPageModule {}
```

dashboard.ts

```
import { SocialSharing } from '@ionic-native/social-
sharing';


constructor(public navCtrl: NavController, public
navParams: NavParams, private localStorageProvider:
LocalStorageProvider, private socialSharing: SocialSharing)
{



 shareLink() {
   console.log('Sharing....');
   this.socialSharing.share(
     'Mensaje Quedada',
     'Subject quedada',
     null,
     'https://quedadas.ovh/#/5ad9ef668633bd06ba6dd032'
   ).then(
     () => console.log('Ok share'),
     () => console.log('Fails to share...')
   );
 }
```

Anexo E: Peticiones Gateway

POST Registro de dispositivos

https://api.quedadas.ovh/register/device

Recurso para registrar un dispositivo en Quedadas y que pueda recibir notificaciones mediante PUSH

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

BODY

```
{
    "deviceId":"nexus7-vrfmskf67eyr",
    "registerToken":"ABCDEF123456789",
    "model":"nexus 7",
    "platform":"android"
}
```

POST Login con Redes sociales

https://api.quedadas.ovh/auth/social

Recurso para acceder a la plataforma usando la autenticación de FAcebook, Twitter o Google

HEADERS

Content-Type	application/json
Accept	application/json

BODY

```
{
  "provider": "twitter",
  "access_token": "1234567890",
  "token_secret": "1234567890"
}
```

GET Listado de Quedadas

https://api.quedadas.ovh/meetings?pageSize=99

Recurso para obtener todas la squedadas de un usuario, tanto las que es organizador como participante

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

PARAMS

pageSize	99

POST Crear una Quedada

https://api.quedadas.ovh/meetings

Recurso para crear un Quedada y añadir a su creador como organizador

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

BODY


```
{
  "name":"Charla de JDD",
  "details": "Hablaremos sobre metodologías JDD",
  "observations": "Debeis traer vuestro ordenador con la batería a tope, hay que compart
  "location": "Centro comercial El Tiro",
  "latitude": -1.434256,
  "longitude": 37.765873,
  "maxAttendees": 18,
  "duration": 180,
  "DateTimeUtc": "2018-5-10T10:00:00Z"
```

GET Ver detalles de una Quedada

https://api.quedadas.ovh/meetings/5ad9efc78633bd06ba6dd034

Este recurso muestra los datos de la Quedada indicada mediante su id

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

PUT Modificar una Quedada

https://api.quedadas.ovh/meetings/5abf5f89ff381377cc48abc2

Este recurso permite modificar los datos de una Quedada, si alguno de estos datos son la fecha o el lugar se envía una notificación a los participantes

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

BODY

```
{
  "name":"Cocke And Meets 2",
  "location": "El Tiro, Murcia",
  "DateTimeUtc": "2018-5-12T10:00:00Z"
}
```

POST Duplicar una Quedada

https://api.quedadas.ovh/meetings/duplicate

Recurso para duplicar una Quedada usando como base una existente, mantendrá registrado a todos los participantes pero sin confirmar su asistencia y serán notificados de la nueva Quedada

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

BODY


```
{
  "duplicateFromMeetingId": "5ad9efc78633bd06ba6dd034",
  "name":"Charla de JDD",
  "details": "Hablaremos sobre metodologías JDD",
  "observations": "Debeis traer vuestro ordenador con la batería a tope, hay que compart
  "location": "Centro comercial El Tiro",
  "latitude": -1.434256,
  "longitude": 37.765873,
  "maxAttendees": 18,
  "duration": 180,
```


POST Cancelar una Quedada

https://api.quedadas.ovh/meetings/5ad9efc78633bd06ba6dd034/cancel

Recurso para cancelar una Quedada, envía una notificación a todos los asistentes indicando que la Quedada ha sido cancelada.

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

POST Confirmar asistencia

https://api.quedadas.ovh/meetings/5abf5f89ff381377cc48abc2/attend/yes

Este recurso cambia el estado de un usuario participante en una Quedada al estado que indica que asistirá a la Quedada

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

POST Confirmar que No asistirá

https://api.quedadas.ovh/meetings/5abf5f89ff381377cc48abc2/attend/no

Este recurso cambia el estado de un usuario participante en una Quedada al estado que indica que no asistirá a la Quedada

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

POST Cancelar confirmación de asistencia

https://api.quedadas.ovh/meetings/5abf5f89ff381377cc48abc2/attend/maybe

Este recurso cambia el estado de un usuario participante en una Quedada al estado que indica que aún no sabe si asistirá a la Quedada

HEADERS

Authorization	Bearer 123456
Content-Type	application/json

Anexo F: Fichero configuración Nginx

```
upstream quedadas_api {
  server 127.0.0.1:9100;
}
server {
    server_name api.quedadas.ovh;
    client_max_body_size 4G;
    access_log /home/tribeca/quedadas/api-gateway/logs/nginx-access.log;
    error_log /home/tribeca/quedadas/api-gateway/logs/nginx-error.log;
    location /static/ {
        alias /home/tribeca/quedadas/api-gateway/static/;
    }
    location / {
        proxy_pass http://quedadas_api;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
        proxy_redirect off;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_cache_bypass $http_upgrade;
        tcp_nodelay on;
    }
  # Error pages
  error_page 500 502 503 504 /500.html;
  location = /500.html {
      root /home/tribeca/quedadas/api-gateway/;
  }
    listen 443 ssl; # managed by Certbot
    ssl_certificate    /etc/letsencrypt/live/api.pingit.ovh/fullchain.pem;    #    managed    by
Certbot
    ssl_certificate_key   /etc/letsencrypt/live/api.pingit.ovh/privkey.pem;  #  managed  by
Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}
server {
    if ($host = api.quedadas.ovh) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
    listen 80;
    server_name api.quedadas.ovh;
    return 404; # managed by Certbot
```