UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Aplicación móvil para dispositivos iOS: “VidWRC”

Proyecto Fin de Carrera

Autor: Mahy Quintana Díaz.
Tutor: Cayetano Guerra Artal.
Ingeniería en Informática.
23/06/2014

Proyecto fin de carrera de la Escuela de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria presentado por el alumno:

Mahy Quintana Díaz

Título del proyecto: Aplicación móvil para dispositivos iOS: “VidWRC”

Tutor:

Cayetano Guerra Artal

Dedicatoria

A mis padres,
a mi hermana,
a mi novia y
a mis abuelas.

Agradecimientos

A mi tutor Cayetano, por interesarse desde el primer momento en la idea, por guiarme y por ayudarme con las pequeñas complicaciones que surgieron. Muchas gracias.

A todos mis compañeros y amigos de clase, pero en especial a los que siempre han estado ahí: Dani, Diego, Marcos e Imanol. Muchas gracias por echarme una mano cuando hacia falta y por hacer más divertido este largo camino. Espero que esta amistad dure para siempre.

A mis amigos de toda la vida, a todos, siempre han sido un gran apoyo para mi.

A mis padres, sin ellos esto no habría sido posible, a mi hermana y a mi novia, por su apoyo en todo momento, a mi familia en general y en especial a mis dos abuelas, a mi tío José Manuel , a mi tía Inma y a mi madrina Rosa, gracias por todo el apoyo y ánimos que me han dado siempre. ¡Muchas gracias a todos!

Índice

Índice de figuras

Índice de imágenes

1.- Introducción

Este apartado introductorio iniciará al lector los objetivos que se pretenden conseguir con este proyecto y se dará una descripción de la estructura de este documento, aportándose una breve descripción de cada apartado.

1.1.- Presentación.

Según el "V Estudio Anual de Mobile Marketing" el 78% de los usuarios, de teléfonos inteligentes, dedica mas de una hora al día a acceder a internet, siendo la media concretamente de **dos horas y media.**

Por otra parte, este estudio indica también que las aplicaciones se consolidan como vía de acceso a Internet. El navegador del móvil ha dejado de ser el punto de referencia, para ceder el protagonismo a las aplicaciones: si en 2012 sólo el 41% de los usuarios de smartphones utilizaba aplicaciones para consultar Internet, en 2013 el porcentaje se ha incrementado al 71%. ¿Por qué se prefieren las aplicaciones? Porque proporcionan un acceso más fácil, inmediato, rápido y personalizable.

En otro estudio realizado por Ooyala sobre los hábitos de consumo de vídeo la conclusión es que el vídeo móvil crece tan rápido que podría constituir la mitad del consumo de todo el vídeo en línea para 2016.

Este informe mide hábitos recopilados de forma anónima respecto de la visualización de casi 200 millones de espectadores en más de 130 países cada mes, y procesa miles de millones de eventos de análisis de vídeo cada día, lo que lo convierte en un fiel reflejo del vídeo en línea global.

Ese crecimiento récord puede verse reflejado en que el tiempo invertido en la visualización de vídeos en teléfonos inteligentes y tabletas ha aumentado un **719%** desde el cuarto trimestre del 2011 y un 160% anual desde el cuarto trimestre de 2012.

Motivado por los datos presentados anteriormente y mi afición al mundo del motor es por lo que surge la temática de este proyecto.

Este proyecto aborda la creación de una aplicación móvil, enfocada a los aficionados del campeonato del mundo de rallyes, en la que el usuario pueda grabar, compartir, comentar y valorar sus propios vídeos, así como los vídeos de los demás usuarios. Además, los usuarios también podrán seguirse entre ellos, lo que les permitirá mantenerse informados tanto de nuevas publicaciones como de actividades realizadas por sus amigos. En resumidas cuentas, podría decirse que se trata de una red social donde todo gira en torno a los vídeos del campeonato del mundo de rallyes. Esta aplicación será desarrollada para dispositivos móviles con sistema operativo iOS.

¿Qué supone esta aplicación?
Esta aplicación supone comodidad y disfrute. Comodidad por el simple hecho de

evitar tener que estar haciendo búsquedas por la red para buscar imágenes del campeonato del mundo de rallyes, recordemos que las búsquedas en los dispositivos móviles no suelen ser muy cómodas por sus pantallas reducidas. Y disfrute, porque será un medio a través del cual se verán imágenes inéditas, captadas por los usuarios y que proporcionan un punto de vista diferente al que pueda ofrecer la televisión.

1.2.- Objetivos del proyecto.

El objetivo principal de este proyecto es aprender una nueva tecnología y adquirir cierto dominio en el desarrollo de aplicaciones móviles. Para ello se ha planteado diseñar e implementar desde cero la aplicación anteriormente mencionada.

Como objetivos secundarios pero no menos importantes, se proponen los siguientes:

- Consecución de las destrezas básicas y medias en el diseño de interfaces.
- Consecución de las destrezas básicas y medias en la implementación y conexión de un servidor de datos.
- Creación de una aplicación útil y con expectativas de futuro.
- Publicación de la aplicación en la AppStore.

1.3.- Esquema de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

En el primer apartado: "*Introducción*", se presenta una breve descripción del proyecto, así como los objetivos que se desean alcanzar con la realización del proyecto.

En el segundo apartado: "*Estado del arte*" se realiza un estudio previo de los servicios y aplicaciones existentes similares al objetivo del proyecto.

En el tercer apartado: *"Metodología y recursos utilizados"* se expondrá la metodología usada en este proyecto así como su justificación. Además se presentará una serie de recursos que sin ellos no habría sido posible realizar el proyecto.

En el cuarto apartado: *"Análisis"*, se realiza un estudio detallado de la tecnología a usar como de los requisitos del proyecto, punto clave para el correcto desarrollo del proyecto.

En el quinto apartado: *"Diseño"*, se abordarán cuestiones como el diseño de la base de datos o el desarrollo de prototipos que sirvan como guía para la interfaz de usuario.

En el sexto apartado: "*Desarrollo e implementación*", se explica como ha sido el proceso de creación de la aplicación, se muestran las diferentes librerías utilizadas y se presentan las soluciones adoptadas para solventar algunos problemas.

En el séptimo apartado, con título "*Resultados y conclusiones*" se exponen las vivencias relacionadas con la elaboración de este proyecto así como las conclusiones finales a las que se ha llegado después de la realización del mismo.

En el octavo apartado con título "Trabajo futuro", se presentan una serie de propuestas de mejora que podrían aplicarse en versiones futuras a la aplicación. Por otra parte también se presenta el desarrollo de nuevos clientes similares al desarrollado en este proyecto pero para otras plataformas.

En el apartado: "*Bibliografía*" se indican las páginas y los documentos que se han consultado para poder llevar a cabo este proyecto.

Y por último, se presenta un anexo llamado "*Manual de usuario*" donde se explica cómo se hace uso de la aplicación con imágenes descriptivas de la interfaz y secuencias de pasos para realizar un uso típico de la aplicación.

2.- Estado del arte

Es este apartado se hará un breve recorrido por las aplicaciones y servicios existentes relacionados con la temática de este proyecto, en cuestión abordaremos dos tipos de aplicaciones y servicios, los relacionados con la compartición de vídeos y los relacionados con el mundial de rallyes.

2.1.- Aplicaciones y servicios de compartición de vídeo.

A continuación se van a presentar una serie de servicios y aplicaciones diferenciando dos categorías, compartición de vídeos relacionados con el deporte y compartición de vídeos de propósito general:

-Servicios y aplicaciones de compartición de vídeo de propósito general:

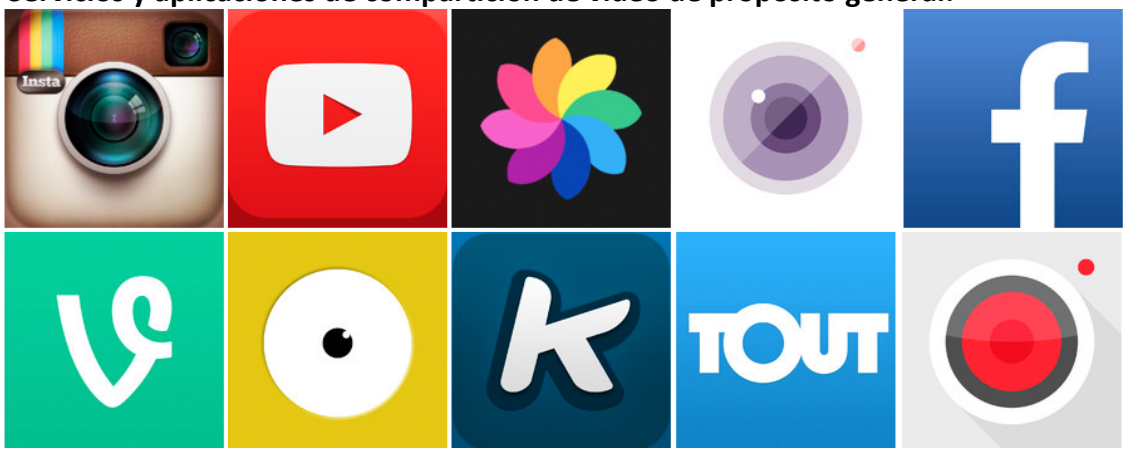


Imagen 2.1: Servicios y aplicaciones de compartición de vídeo.

-**Vine**: Vine es una aplicación desarrollada por Twitter que permite crear y publicar vídeos cortos (6 segundos). Fue lanzado el 24 de enero de 2013, inicialmente disponible para el sistema operativo iOS pero a partir del mes de junio del mismo año también se encuentra disponible para Android y posteriormente para Windows Phone. Actualmente esta red social cuenta más de 40 millones de usuarios registrados.

Podemos calificarla como aplicación de vídeos de propósito general aunque cuenta con categorías donde se contempla el deporte:

Imagen 2.2: Vine

-**Instagram**: Instagram es una aplicación para compartir fotos con la que los usuarios pueden aplicar efectos fotográficos como filtros, marcos, colores retro y vintage, luego pueden compartir las fotografías en diferentes redes sociales como Facebook, Tumblr, Flickr y Twitter. Instagram fue lanzada en octubre de 2010 y rápidamente ganó popularidad, con más de 100 millones de usuarios activos en abril de 2012. Originariamente, esta aplicación fue diseñada para iPhone pero a partir del año 2012 se encuentra disponible también para Android.

A partir de la versión 4.0 la aplicación permite al usuario la toma de vídeos con una duración máxima de 15 segundos. Esta nueva herramienta incluye estabilización de

imagen con la cual el usuario puede grabar buenas tomas incluso si está en movimiento.

-**Facebook**: Facebook es una red social creada por Mark Zuckerberg y fundado junto a Eduardo Saverin, Chris Hughes y Dustin Moskovitz. Originalmente era un sitio para estudiantes de la Universidad de Harvard, pero se abrió a cualquier persona con una cuenta de correo electrónico.

Facebook cuenta con más de 900 millones de miembros, y traducciones a 70 idiomas. En enero de 2013, Facebook llegó a los 1230 millones de usuarios,10 de los cuáles hay más de 600 millones de usuarios móviles.

A través de esta red social se pueden compartir infinidad de contenido entre los que se encuentran los vídeos.

-**YouTube**: es un sitio web en el cual los usuarios pueden subir y compartir vídeos. Fue creado por tres antiguos empleados de PayPal en febrero de 2005. En octubre de 2006, fue adquirido por Google Inc. a cambio de 1650 millones de dólares y ahora opera como una de sus filiales. Actualmente es el sitio web de su tipo más utilizado en internet.

YouTube usa un reproductor en línea basado en Adobe Flash y HTML5 para servir su contenido. Es muy popular gracias a la posibilidad de alojar vídeos personales de manera sencilla. Aloja una variedad de clips de películas, programas de televisión y vídeos musicales. Los enlaces a vídeos de YouTube pueden ser también insertados en blogs y sitios electrónicos personales usando su API o incrustando cierto código HTML.

-**Vimeo**: Vimeo es una red social basada en vídeos, lanzada en noviembre de 2004 por la compañía InterActiveCorp. El sitio permite compartir y almacenar vídeos digitales para que los usuarios comenten en la página de cada uno de ellos. Los usuarios deben estar registrados para subir vídeos, crear su perfil, cargar avatares, comentar y armar listas de favoritos.

Vimeo no admite anuncios de televisión, demostraciones de videojuegos o cualquier contenido que no haya sido creado por el usuario. El sitio, además, ha ganado reputación como «proveedor de imágenes» para diversos artistas, debido a la alta tasa de bits y resolución de sus vídeos.

En lo referente a la compartición de vídeos de propósito general, los servicios y aplicaciones mencionados anteriormente son los más importantes, aunque también existen muchos más como: Viddy, SocialCam, Cinemagram, Keek, Wopp, Tout, etc.

-Servicios y aplicaciones de compartición de vídeo relacionados con el deporte:

-**Coach's Eye**: Coach's Eye permite a los deportistas aprender a través del análisis de vídeos. Esta aplicación proporciona una manera fácil y eficaz de aprender o entrenar las técnicas de los diferentes deportes. El swing del golf, el lanzamiento de bola rápida del baseball, los pases del fútbol, el saque de volleyball y tenis, o la carrera en el atletismo, son algunas de las mejoras que se pueden lograr gracias a la "mirada del entrenador" electrónico.

La aplicación tiene cinco funcionalidades principales. La primera es grabar al deportista en acción. De forma instantánea, el vídeo puede ser analizado por el software de Coach's Eye.

En segundo lugar, la persona puede dibujar líneas, flechas y formas para indicar determinadas posturas o movimientos en el vídeo. Otra de las herramientas para agregar comentarios es mediante un mensaje de voz que acompaña la grabación.

Otra útil funcionalidad de Coach's Eye es la posibilidad de ver el vídeo frame por frame, es decir, en cámara súper lenta. Este mecanismo permite agudizar la visión del entrenador digital.

Por último, se puede compartir el vídeo con comentarios con tan solo tocar un botón. La aplicación permite enviarlo por mensaje de texto o correo electrónico, así como subirlo a YouTube, Evernote o Dropbox.

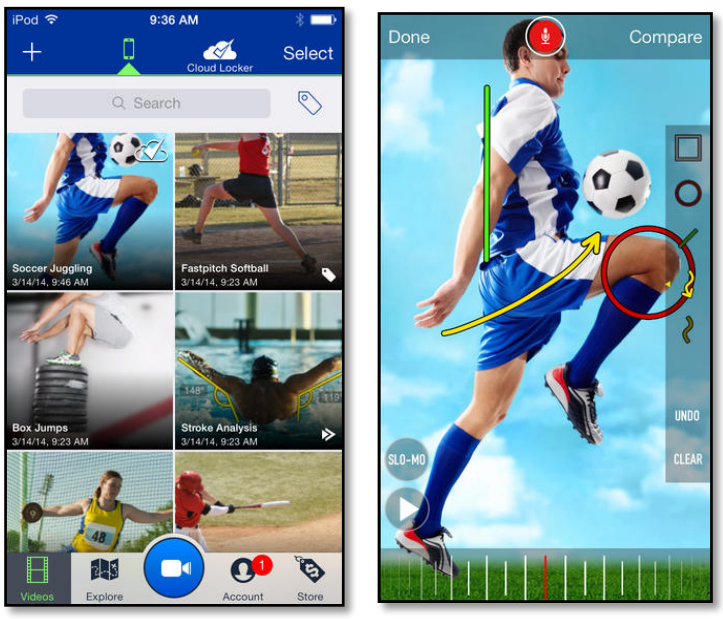


Imagen 2.3: Coach's Eye

-**Ubersense Coach: Slow Motion Vídeo Analysis:** esta aplicación es muy similar a la anterior, ya que permite a los deportistas grabar, comparar y analizar vídeos para mejorar sus técnicas y habilidades. A diferencia de Coach's Eye, Ubersense tiene un

enfoque más social ya que además de compartir el vídeo puedes seguir a otros usuarios.

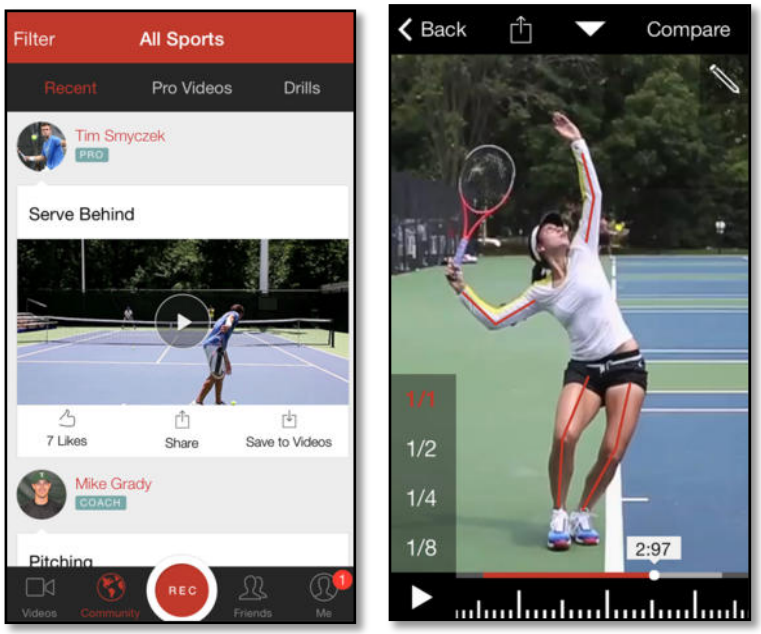


Imagen 2.4: Ubersense Coach

2.2.- Aplicaciones y servicios relacionadas con el mundial de rallyes.

Hay una gran cantidad de páginas webs relacionada con el mundo de los rallyes en general, así como una creciente cantidad de aplicaciones móviles. La mayoría de estos servicios son de carácter informativo, con un grado de interactividad bajo, aunque en ligero aumento motivado por la integración en las principales redes sociales.

- **Promotor oficial del WRC**: a continuación se mostrará la web oficial del Campeonato del Mundo de Rallyes (www.wrc.com), así como su equivalente en Facebook y su perfil de Twitter:

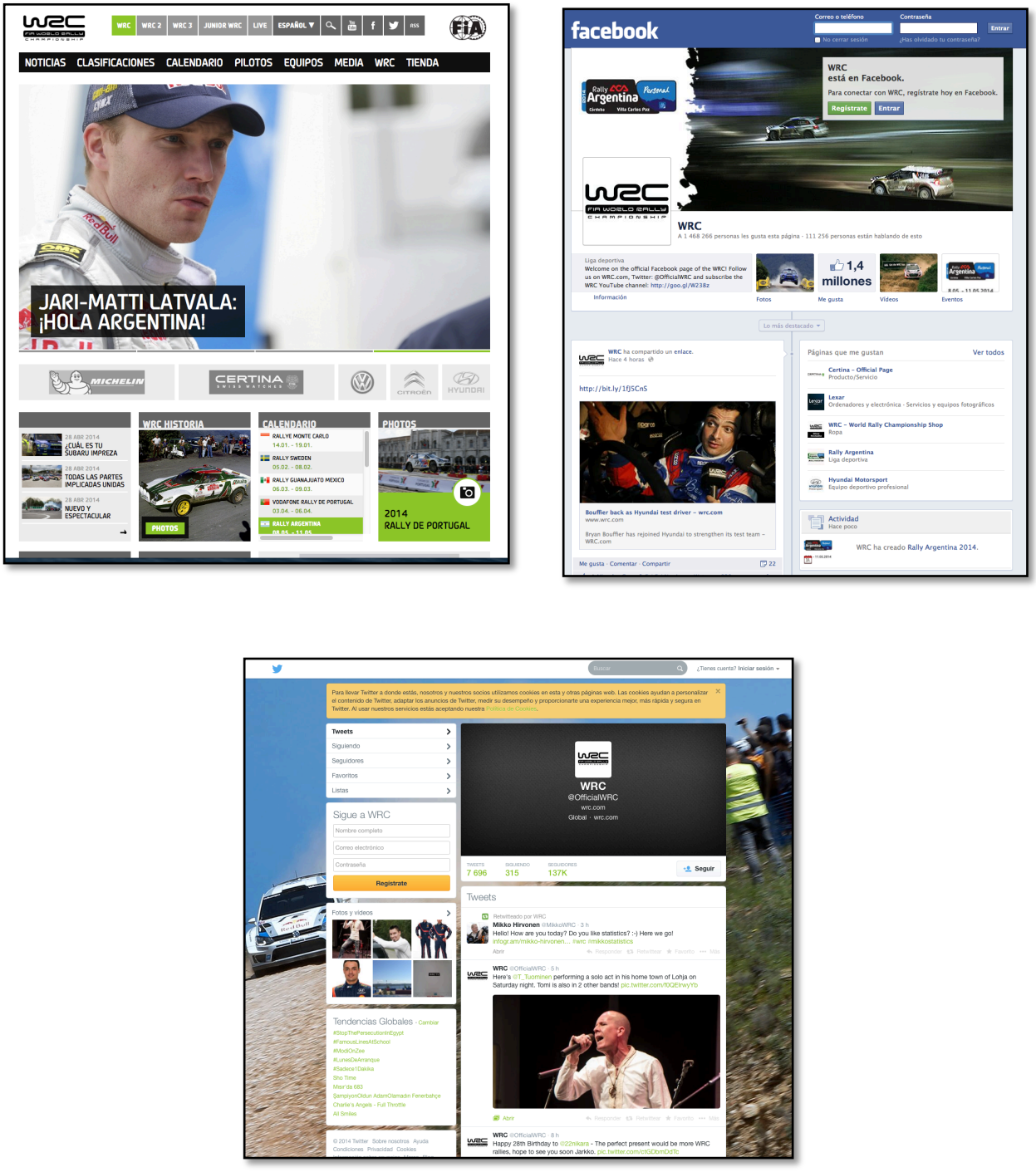


Imagen 2.5: Promotor oficial del WRC

Además de esto, el WRC también cuenta con una aplicación móvil donde podemos encontrar todo lo que se encuentra en su web oficial pero para formato móvil:

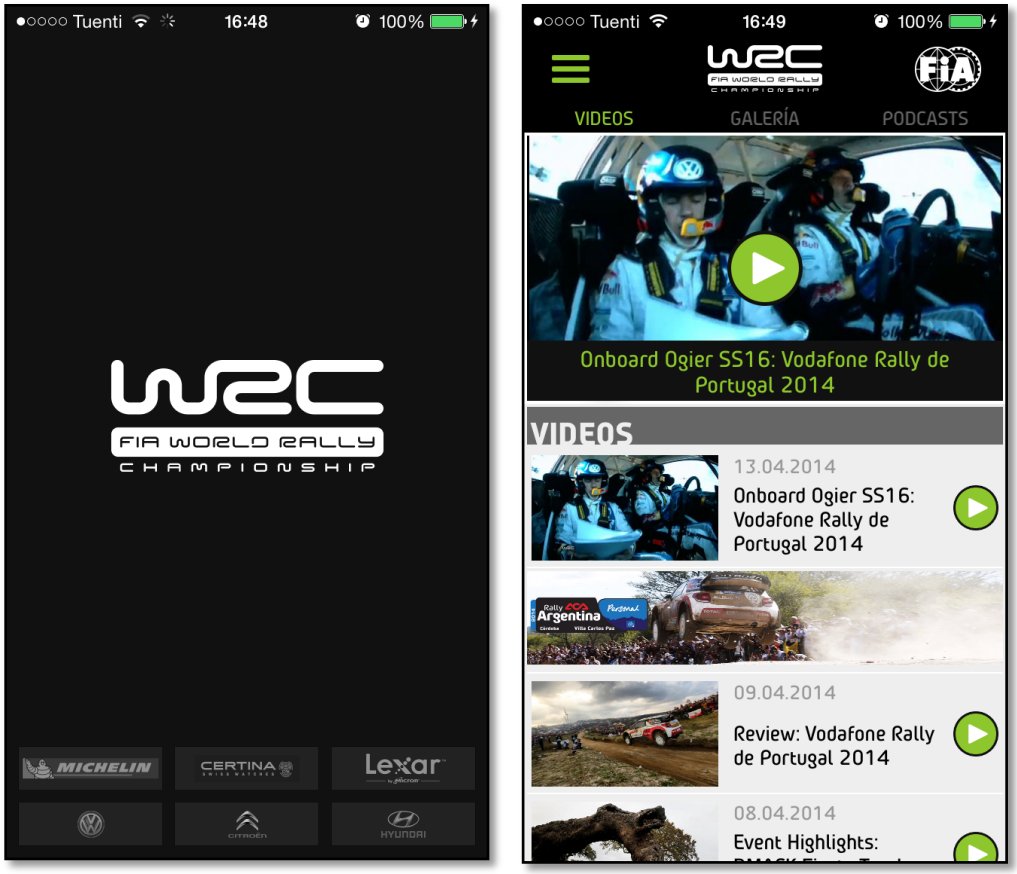


Imagen 2.6: Aplicación para iOS del promotor oficial del WRC

- **Páginas webs de las pruebas que componen en campeonato**:
  - http://www.rallyracc.com
  - http://www.adac-rallye-deutschland.de
  - http://www.rallyargentina.com
  - http://www.rallysweden.com
  - etc.

- **Otras webs y revistas**:
  - http://www.revistascratch.com/wrc
  - http://www.motorpasionf1.com/categoria/mundial-rallies
  - http://www.autohebdosport.es
  - etc.
  -

- **Aplicaciones móviles**:

Estas aplicaciones son en su mayoría de carácter informativo. En ellas podremos encontrar las clasificaciones de los rallyes ya disputados, información sobre cual será la siguiente prueba dentro del calendario, entrevistas a pilotos, fotos, etc.

- iRally

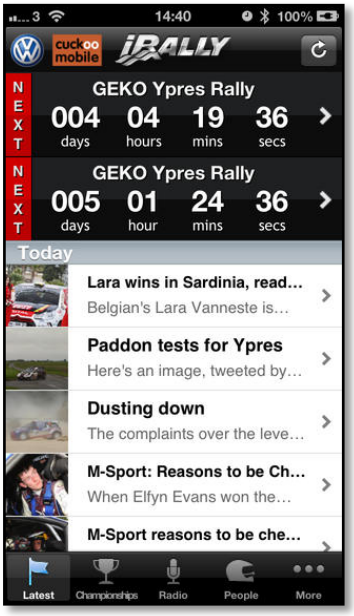


Imagen 2.7: Aplicación iRally

- WRC News.

Imagen 2.8: Aplicación WRC News

- WRC Fans App 2014.

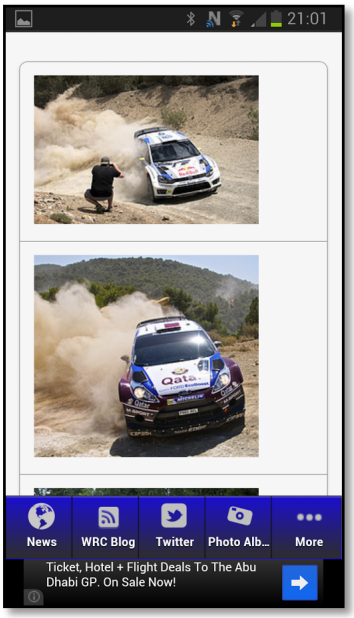


Imagen 2.8: WRC Fans App 2014.

3.-Metodología y recursos utilizados

3.1.- Metodología de desarrollo

Como modelo de ciclo de vida se usará aproximación incremental, se creará primero una aplicación base funcional y luego se irán desarrollando mejoras y nuevas funciones para ir cubriendo los requisitos restantes. Este tipo de ciclo de vida permite ir testeando y validando la aplicación. Otro motivo por el que se ha usado este tipo de ciclo de vida es porque desde un principio quedan bien definidos los requisitos, lo cual permite ir desarrollando la aplicación en función de estos.

1. Análisis de requisitos de usuario.
   Definición de lo que el usuario espera del sistema.
2. Análisis de requisitos software.
   En este apartado se trata de obtener una definición del software que se va a construir. Se construye un modelo lógico del software.
3. Diseño arquitectónico.
   Se define la estructura del software, equivale a la estructura más su control. Se especifica cada módulo y se diseñan las interrelaciones entre ellos.
4. Diseño detallado y producción.
   Se detalla cada componente de la arquitectura definido anteriormente. A continuación, se produce en los lenguajes de programación elegidos el sistema software para su posterior integración y testeo.
5. Transferencia.
   Se implanta el sistema y se configura en un entorno operativo para dejarlo ejecutable para el usuario.
6. Operación y Mantenimiento
   Una vez que el sistema se encuentra operativo se debe monitorear para comprobar su correcto funcionamiento. Si se detectase cualquier error o perdida de rendimiento, estos se subsanarían como parte del mantenimiento.

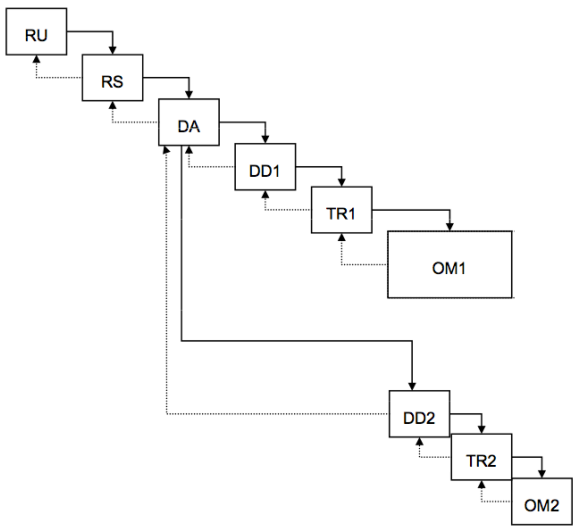


Figura 3.1: Modelo de ciclo de vida

3.2.- Recursos utilizados

3.2.1.- Recursos hardware

Equipo informático:

- Procesador: Intel core i5 – 3.1GHz
- Memoria RAM: 4GB
- Disco duro: 1TB
- Tarjeta gráfica: AMD Radeon HD 6970M 1024MB
- Pantalla: 27”
- Teclado y ratón.

Dispositivos móviles:

- iPod touch 4G
  - Pantalla: 3,5”
  - Procesador: Apple A4 - 1GHz
  - Memoria RAM: 256MB
  - Memoria flash: 8GB
  - Sistema operativo: iOS6
- iPhone 4s
  - Pantalla: 3,5”
  - Procesador: Apple A5 - 1GHz (doble núcleo)
  - Memoria RAM: 512MB
  - Memoria flash: 16GB
  - Sistema operativo: iOS7
- iPhone 5
  - Pantalla: 4”
  - Procesador: Apple A6 – 1.3GHz (doble núcleo)
  - Memoria RAM: 1GB
  - Memoria flash: 16GB
  - Sistema operativo: iOS7

3.2.2.- Recursos software

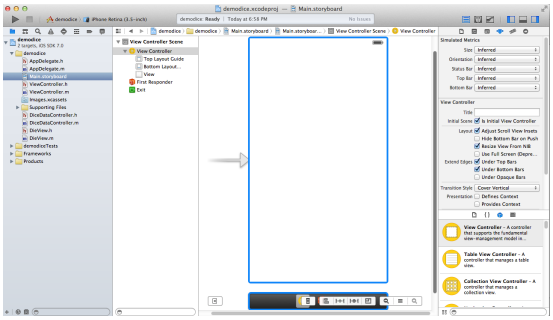
- **SO – Mac OS X Mavericks (versión 10.9)**

OS X Mavericks es la décima versión principal de OS X para ordenadores, portátiles y servidores Mac. Este es un sistema operativo basado en Unix que ha sido desarrollado y comercializado por Apple Inc. Como características principales se puede destacar el aumento de la duración de baterías en portátiles, mayor rendimiento y un uso más eficiente de la memoria del sistema. En lo que respecta a su interfaz, esta versión ha cambiado considerablemente al implementar un sistema de notificaciones similar al disponible en iOS, además se han incluido

aplicaciones propias de iOS tales como: “Notas”, “Mapas”, etc.

- **Xcode 5.1**

Xcode es el entorno de desarrollo que ofrece las librerías, herramientas de desarrollo, compilación y testeo que se necesitan para desarrollar aplicaciones para iOS.



- **iOS**

iOS es un sistema operativo móvil de la empresa Apple Inc. Originalmente fue desarrollado para el iPhone (iPhone OS), siendo después usado en dispositivos como el iPod Touch, iPad y el Apple TV, la instalación de este sistema operativo no esta permitida en hardware de terceros.

3.2.3.- Tecnologías utilizadas

- **Objective-C**: Objective-C es un lenguaje de programación orientado a objetos creado como un superconjunto de C para que implementase un modelo de objetos parecido al de Smalltalk. Originalmente fue creado por Brad Cox y la corporación StepStone en 1980. En 1988 fue adoptado como lenguaje de programación de NEXTSTEP y en 1992 fue liberado bajo licencia GPL para el compilador GCC. Actualmente se usa como lenguaje principal de programación en Mac OS X, iOS y GNUstep.

- **Cocoa touch**: Cocoa touch es un framework que permite el desarrollo de aplicaciones nativas para iOS, este incluye reconocimiento gestual, animaciones y una librería distinta para la interfaz de usuario; el cual se usa en dispositivos Apple como el iPhone, el iPod Touch y el iPad.

  Principalmente, el lenguaje en el que se programa con esta biblioteca es Objective-C.

- **JavaScript:** JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

  Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor.

  JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes. Fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de Mocha, el cual fue renombrado posteriormente a LiveScript, para finalmente quedar como JavaScript.

4.- Análisis

4.1.- La aplicación

Como ya se ha mencionado, en apartados anteriores, la finalidad de este proyecto no es el desarrollo en sí de una aplicación, sino la adquisición de unas destrezas y unos conocimientos determinados en el desarrollo de aplicaciones para dispositivos móviles, concretamente dispositivos iOS. Así pues, el contenido exacto se fue definiendo mientras se avanzaba en el aprendizaje y en las primeras versiones de la aplicación.

Hay que mencionar que desde un principio se impusieron una serie de funcionalidades mínimas que debía cumplir la aplicación, así como otras funcionalidades "opcionales" que pudieran ser implementadas en el futuro. A continuación se presenta el análisis correspondiente a la aplicación final que se ha desarrollado.

4.1.1.- Especificación de requisitos de usuario

Este proyecto aborda la creación de una aplicación móvil, enfocada a los aficionados del campeonato del mundo de rallyes, en la que el usuario pueda grabar, compartir, comentar y valorar sus propios vídeos, así como los vídeos de los demás usuarios. Además, los usuarios también podrán seguirse entre ellos, lo que les permitirá mantenerse informados tanto de nuevas publicaciones como de actividades realizadas con sus amigos. En resumidas cuentas, podría decirse que se trata de una red social donde todo gira en torno a los vídeos del campeonato del mundo de rallyes. Esta aplicación será desarrollada para dispositivos móviles con sistema operativo iOS.

- Funcionales:
  - Cualquier persona podrá registrarse como usuario.

- Un usuario podrá ver vídeos publicados por otros usuarios.
- Un usuario podrá grabar un vídeo con la cámara, añadirle un título, indicar el rallye al que pertenece, el día del rallye en el que fue tomado, y publicar el vídeo en el sistema.
- Un usuario podrá compartir su vídeo en Facebook.
- Un usuario podrá eliminar sus vídeos.
- Un usuario podrá comentar los vídeos.
- Un usuario podrá ver los comentarios de un vídeo.
- Un usuario podrá ver a quien le gusta un determinado vídeo.
- Un usuario podrá indicar que le gusta un vídeo.
- Un usuario podrá indicar que un vídeo es inadecuado.
- Un usuario podrá buscar vídeos por su título.
- Un usuario podrá buscar a otros usuarios por su nombre.
- Un usuario podrá seguir a otro usuario.
- Un usuario podrá modificar su foto de perfil.
- Un usuario podrá gestionar sus notificaciones.
- Un usuario podrá eliminar su cuenta.
- Un usuario podrá salir del sistema en cualquier momento.
- Un usuario podrá ver quien le sigue.
- Un usuario podrá ver sus seguidores.
- Un usuario podrá ordenar los vídeos por fecha.
- Un usuario podrá ordenar vídeos por valoración.
- Un usuario podrá ver sus datos de perfil.
- Un usuario podrá ver su actividad reciente (a que usuarios les gustan sus vídeos, quien o quienes han comentado en sus vídeos, quien o quienes han comenzado a seguirle y quien o quienes creen que su vídeo es inapropiado)
- Un usuario podrá ver las publicaciones que les gustan a sus seguidos.
- Los datos de inicio de sesión podrán ser almacenados localmente en el dispositivo para evitar al usuario tener que introducirlos cada vez que quieran acceder.

- Rendimiento
  - La aplicación no puede bloquear el terminal.
  - El sistema debe adecuar la transferencia de datos de acuerdo a la velocidad de la conexión de red disponible en cada momento.
  - El servidor debe soportar un gran numero de peticiones por minuto.
- Seguridad
  - Únicamente las personas registradas podrán hacer uso del sistema.
- Implantación
  - La aplicación debe desarrollarse para dispositivos iOS.
  - La interfaz de la aplicación debe estar en inglés.
  - Se deberá informar al usuario de cualquier error o problema que evite el correcto funcionamiento de la aplicación.
- Disponibilidad del sistema
  - El sistema debe estar disponible 24 horas al día, 365 días al año.

4.1.2.-Especificación de requisitos software

4.1.2.1.- Requisitos funcionales

- Una persona podrá registrarse a través de un formulario donde se le solicitará:
  - una foto de perfil (el usuario deberá poder acceder tanto a la cámara como a la biblioteca del dispositivo),
  - un nombre de usuario (tiene que ser único, en caso contrario la aplicación no deberá aceptar este nombre como válido),
  - una contraseña (si no dispone de un mínimo de 6 caracteres, la aplicación no deberá aceptar la contraseña como válida),
  - una dirección de correo electrónico (este correo debe tener una estructura válida, en caso contrario, la aplicación no deberá aceptar el correo como válido).
- Una persona podrá iniciar sesión usando:
  - su cuenta de Facebook (para ello se debe hacer uso de la API de Facebook),
  - su cuenta de Twitter (para ello se debe hacer uso de la API de Twitter),
  - su cuenta del sistema creada previamente.
- Un usuario podrá indicar que le gusta un vídeo a través de un botón situado en la parte inferior del vídeo. Este evento deberá quedar registrado en el sistema de manera que se pueda acceder posteriormente.
- Un usuario podrá comentar un vídeo a través de una interfaz específica, este comentario debe guardarse en el sistema para que otros usuarios puedan verlo.

- Un usuario podrá indicar que un vídeo es inadecuado a través de un botón. Este botón ejecutará una acción que guardará esa acción en el sistema. Si un vídeo cuenta con más de diez indicaciones de que no es un vídeo adecuado, el sistema deberá censurar el vídeo, evitando que pueda ser reproducido.

- Un usuario podrá reproducir un vídeo pulsando en un icono de "play" situado siempre en la parte superior de la imagen del vídeo. Al pulsar este botón se solicitará al servidor de vídeos que comience el streaming.

- Un usuario podrá navegar entre las diferentes vistas de la aplicación a través de una barra inferior donde se muestren iconos representativos del contenido o actividad mostrada en esa sección.

- Un usuario podrá modificar su imagen de perfil tomando una nueva fotografía con la cámara o bien refrescando la imagen desde la red social con la que ha iniciado sesión.

- Un usuario podrá activar o desactivar las notificaciones a través de un switch.

- Si ocurre algún error o surge algún problema en la aplicación, esta debe mostrar un mensaje de alerta haciendo uso de un UIAlertView.

- Un usuario podrá ordenar las listas de vídeos en función de su antigüedad en el sistema o de su valoración.

- Si el servidor detecta que se ha publicado un vídeo nuevo, este debe ejecutar la acción de enviar una notificación global a todos los usuarios del sistema.

- Un usuario podrá seguir a otro usuario, esto debe quedar registrado en la base de datos para que quede constancia de ello.

- Un usuario podrá ver a quien o quienes les gusta un vídeo, para ello la aplicación debe hacer previamente una consulta a la base de datos donde se solicite las actividades de tipo "me gusta".

- Un usuario podrá ver los comentarios de un determinado vídeo, para ello la aplicación debe hacer previamente una consulta a la base de datos donde se soliciten los comentarios.

- Un usuario podrá ver la cantidad de "me gusta" que tiene un vídeo, para ello la aplicación debe hacer previamente una consulta a la base de datos donde se contabilicen las actividades de tipo "me gusta".

- Un usuario podrá ver la cantidad de comentarios que tiene un vídeo, para ello la aplicación debe hacer previamente una consulta a la base de datos donde se contabilice el número de comentarios relacionados con dicho vídeo.

- Un usuario puede publicar un vídeo que ya tenia previamente grabado o puede capturar el vídeo justo antes de publicarlo. Para grabar el vídeo la aplicación debe proporcionar acceso a la cámara del dispositivo, y para seleccionar un vídeo ya existente la aplicación debe proporcionar acceso a la biblioteca del dispositivo.

- El usuario podrá seleccionar cual será el "thumbnail" del vídeo antes de que este sea publicado, la aplicación proporcionará un slider con el que se puedan ir viendo los frames del vídeo y así poder seleccionar el deseado.

- Un usuario podrá compartir un vídeo en Facebook indicándolo a través de un botón, este botón iniciará la ejecución de un código el cual conectará con Facebook a través de su API y creará una entrada en el muro del usuario.

4.1.2.1.1.- Diagrama de casos de usos

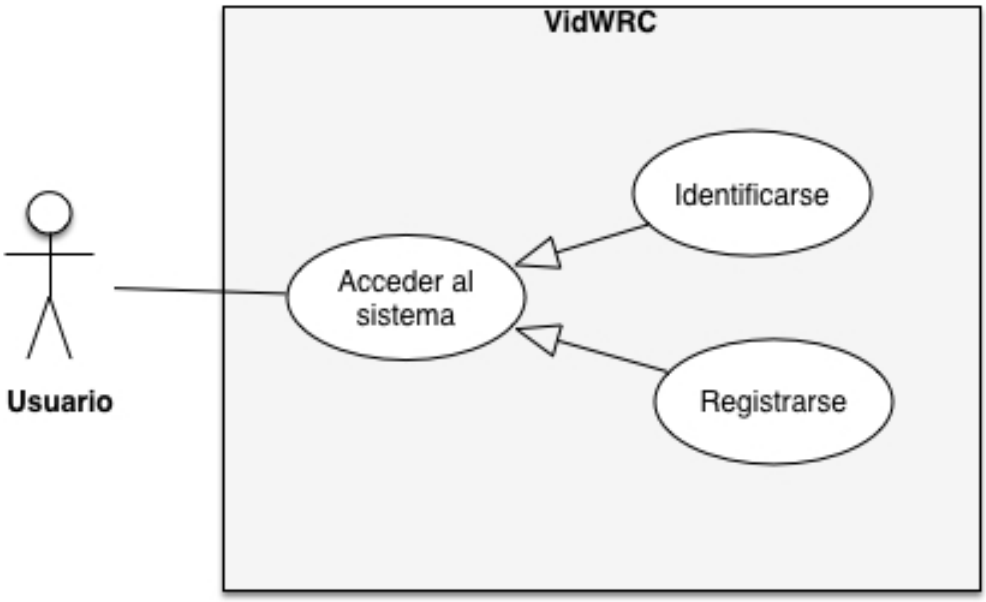


Figura 4.1: Diagrama de casos de uso

Figura 4.2: Diagrama de casos de uso

4.1.2.1.2.-Actores del software

Para este sistema solo existe un actor y es el usuario del sistema.

4.1.2.2.-Requisitos no funcionales

- La aplicación no puede bloquear el terminal, las acciones deben ser asíncronas.

- El sistema debe adecuar la transferencia de datos de acuerdo a la velocidad de la conexión de red disponible en cada momento.

- El servidor debe soportar un gran numero de peticiones por minuto.

- Únicamente las personas registradas podrán hacer uso del sistema.
- La aplicación debe desarrollarse para dispositivos iOS.
- La interfaz de la aplicación debe estar en inglés.
- El sistema debe estar disponible 24 horas al día, 365 días al año.

4.2.- La plataforma móvil.

¿Por qué iOS y no Android o Windows Phone?
En este apartado no se ha hecho un análisis comparando las diferentes plataformas y sus ventajas o inconvenientes, se ha decidido desarrollar para la plataforma iOS simplemente porque se disponían de los recursos necesarios para poder hacerlo.

iOS es un sistema operativo móvil de la empresa Apple Inc. Originalmente fue desarrollado para el iPhone (iPhone OS), siendo después usado en dispositivos como el iPod Touch, iPad y el Apple TV, la instalación de este sistema operativo no esta permitida en hardware de terceros.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten en deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal).

iOS se deriva del sistema operativo de equipos de escritorio Mac OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Unix.

Pantalla principal
La pantalla principal (llamada «SpringBoard») es donde se ubican los iconos de las aplicaciones y el Dock en la parte inferior de la pantalla donde se pueden anclar aplicaciones de uso frecuente, aparece al desbloquear el dispositivo o presionar el botón de inicio. La pantalla tiene una barra de estado en la parte superior para mostrar datos, tales como la hora, el nivel de batería, y la intensidad de la señal. El resto de la pantalla está dedicado a la aplicación actual.

Multitarea
Antes de iOS 4, la multitarea estaba reservada para aplicaciones por defecto del sistema. A Apple le preocupaba los problemas de batería y rendimiento si se permitiese correr varias aplicaciones de terceros al mismo tiempo. A partir de iOS 4, dispositivos de tercera generación y posteriores permiten el uso de 7 APIs para multitarea, específicamente:

- Audio en segundo plano
- Voz IP
- Localización en segundo plano
- Notificaciones push
- Notificaciones locales
- Completado de tareas
- Cambio rápido de aplicaciones

Tecnologías no admitidas
iOS no permite Adobe Flash ni Java. Steve Jobs escribió una carta abierta donde critica a Flash por ser inseguro, con errores, consumir mucha batería, ser incompatible con interfaces multitouch e interferir con el servicio App Store. En cambio iOS usa HTML5 como una alternativa a Flash. Esta ha sido una característica muy criticada tanto en su momento como la actualidad.

Aplicaciones
Las aplicaciones deben ser escritas y compiladas específicamente para la arquitectura ARM, por lo que las desarrolladas para Mac OS X no pueden ser usadas en iOS. Aplicaciones nativas de terceros están disponibles por medio de la App Store. Para poder publicar una aplicación en la AppStore y que cualquier usuario pueda usarla, previamente hay que pagar la cuota del iOS Developer Program.

Los desarrolladores pueden poner un precio por encima del mínimo ($0.99 dólares) a sus aplicaciones para distribuirlas en el App Store, de donde recibirán el 70% del dinero que produzca la aplicación. En alternativa, el desarrollador puede optar por lanzar la aplicación gratis, y de esta forma no pagar ningún costo por distribuir la aplicación. (excepto por la cuota del iOS Developer Program).

Arquitectura del sistema operativo:
La arquitectura del sistema operativo iOS tiene 4 capas definidas:

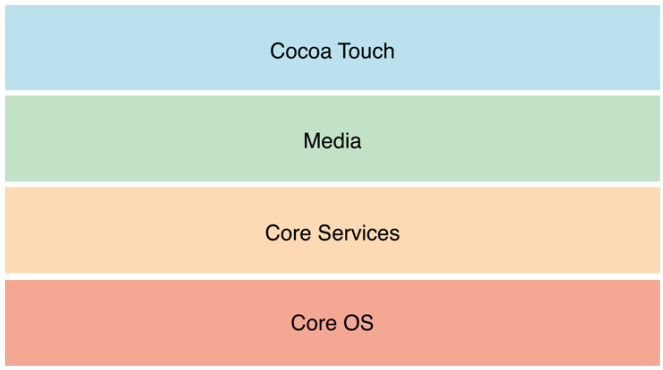


Figura 4.3: Arquitectura del SO iOS

Core OS: esta es la capa justo encima del hardware y proporciona una variedad de servicios que incluyen el acceso a accesorios externos, el acceso a red de bajo nivel así como también ofrece servicios fundamentales del sistema operativo como el manejo de memoria, manejo del sistema de ficheros, gestión de hilos, etc.

Core Services: esta capa ofrece ofrece acceso a los servicios fundamentales del sistema operativo, dentro de los cuales están SQLite Library, XML Libraries, CFNetwork Framework, Core Foundation Framework y Security Framework, entre otros.

Media: esta capa maneja los archivos de audio, vídeo, fotos, la cámara, el reproductor de música, etc. También maneja las bibliotecas necesarias para dibujar 2D y 3D. Incluye lo necesario para poder programar en C y Objective-C, con soporte de tecnologías como openGL-ES, Quartz y Core Audio. También contiene una tecnología llamada Core Animation, un motor de animación basado en Objective-C.

Cocoa Touch: Provee las clases primarias para implementar un evento gráfico; cada aplicación en el iPhone usa un framework para implementar interfaces de usuario; también incluye otros frameworks que permiten el acceso a características del dispositivo, entre los cuales están UIKit, Address Book, Core Location, Message y Game Kit, etc.

4.3.- El servidor.

Para llevar a cabo este proyecto se necesita un sistema servidor de información, para suplir esta necesidad existen dos opciones, crear el servidor o usar un servicio existente que haga las veces de servidor para nuestra aplicación. A continuación se hará un breve análisis, y se seleccionará la opción que mas se adecue a las necesidades y características de este proyecto.

Servidor de datos

Si se eligiese la opción de crear el servidor, este debería hacerse usando tecnologías como PHP, NodeJS, MySQL, PostgreSQL, etc.
Esto implicaría tener que aprender esas tecnologías así como aprender a trabajar con frameworks que nos faciliten la labor, lo cual representaría un gasto importante de tiempo y la desviación del objetivo principal del proyecto, que es aprender sobre el desarrollo de aplicaciones móviles para dispositivos iOS.

En cambio si se opta por usar algún servicio existente, se evitaría ese gasto de tiempo. Por ello se ha investigado acerca de posibles alternativas, las cuales consisten en servicios en la nube que proporcionan un backend (BaaS).

¿Qué es BaaS?, **backend as a service** es un modelo para proporcionar a los desarrolladores web y de aplicaciones móviles una forma de vincular estas aplicaciones al almacenamiento en la nube , servicios analíticos y otras características tales como la gestión de usuarios, la posibilidad de enviar notificaciones push y la integración con servicios de redes sociales. Estos

servicios se prestan a través de la utilización de kits personalizados de desarrollo de software (SDK) y las interfaces de programación de aplicaciones (API). BaaS es un modelo relativamente reciente en la computación en la nube, donde la mayoría de empresas datan del 2011 o posterior.

El uso de este tipo de sistemas permiten prescindir de un sistema gestor de base de datos y del desarrollo del servicio web que comuniquen la aplicación con dicho sistema. Configurar y desarrollar este tipo de modelo de persistencia de datos suele resultar costoso.

¿Qué ventajas ofrece el uso de backend en la nube? La ventaja más evidente es el ahorro de tiempo, ya que este sistema elimina la necesidad de construir un backend propio. Otro aspecto importante es la escalabilidad, este aspecto es transparente al desarrollador y solo repercute en el precio del servicio. Aparte de lo mencionado anteriormente, estos servicios cuentan con muchas funcionalidades integradas que facilitan mucho las labores más comunes en las aplicaciones, como puede ser la gestión de usuarios, integración con redes sociales, etc. Por último también hay que destacar que tienen un alto grado de personalización y que cuentan con servicios analíticos muy útiles a la hora de seguir la evolución de las aplicaciones.

Un aspecto muy importante es el precio de estos servicios. La mayoría de BaaS siguen un modelo freemium, por lo que solo se paga por determinadas características o si se superan ciertos umbrales de uso.

Existen varios BaaS en el mercado pero los más importantes son:

- Parse
- Appcelerator

¿Cuál escoger? Para dar respuesta a esta cuestión se ha investigado acerca de las características de cada uno así como de las funcionalidades que ofrecen, todo esto centrándose en el uso de una cuenta gratuita.

Parse:
Almacenamiento de archivos: 20GB
Tamaño máximo de la base de datos: 20GB
Transferencia de datos: 2TB
Tareas concurrente en la nube: 1
Límite de peticiones por segundo al servidor: 30/s

Appcelerator:
Almacenamiento: 20GB
Límite mensual de peticiones al servidor: 5.000.000
Límite diario de peticiones : 250.000

Atendiendo a las características que ofrecen, Parse parece ser la mejor

opción, además de esto, Parse cuenta con una comunidad de desarrolladores importante y sus SDKs para las diferentes plataformas están muy bien documentados. Por todos estos motivos el Baas que se utilizara en el desarrollo de este proyecto será **Parse**.

Servidor de vídeo

Para el almacenamiento de vídeos se podría utilizar un backend en la nube como los vistos anteriormente, el problema repercute en el coste que esto supondría. Las tarifas de estos servicios van acorde a los requisitos requeridos, por lo tanto, a mayor demanda de espacio, mayor será el importe.

Aparte del precio del almacenamiento hay que considerar otros factores al trabajar con vídeos, un archivo de vídeo en FHD de un minuto de duración suele ocupar entre 170MB y 190MB, una cantidad muy grande de megabytes para ser transferidos de una sola vez y más teniendo en cuenta las capacidades de las redes móviles actuales (3G → 0.4 Mbps , 3G+ → 7.2Mbps, 4G → 150Mbps).

La solución es utilizar un servidor de vídeo streaming adaptativo, el streaming adaptativo es una técnica que permite codificar el vídeo original en varias calidades, y "servirle" al cliente final aquella que mejor se ajuste a su velocidad de conexión a internet. El servidor de streaming se conecta con el reproductor de vídeo, lee de forma dinámica (durante todo el tiempo de visionado) las condiciones de velocidad del cliente y adecua automáticamente la calidad para optimizar la experiencia de visualización y para asegurarse de que incluso aquellos usuarios con conexión muy baja (por ejemplo equipos móviles con 3G en zonas de poca cobertura) puedan reproducir correctamente la señal.

Al igual que en el caso del servidor de datos, implementar un servidor de vídeo propio requiere mucho tiempo y conocimiento técnicos que no se poseen por lo que se ha optado por buscar una solución alternativa, esta solución es, utilizar YouTube como servicio de almacenamiento de vídeo. Se ha escogido YouTube y no otros sistemas porque YouTube cuenta con todas las características necesarias y además es un servicio muy usado entre los aficionados a los rallyes, esto supone una ligera ventaja ya que el usuario conocerá de antemano los servicios que usa y estará familiarizado con su funcionamiento.

A continuación se muestran unas ilustraciones y una tabla comparativa de las opciones contempladas para el servidor de vídeo:

Opción A

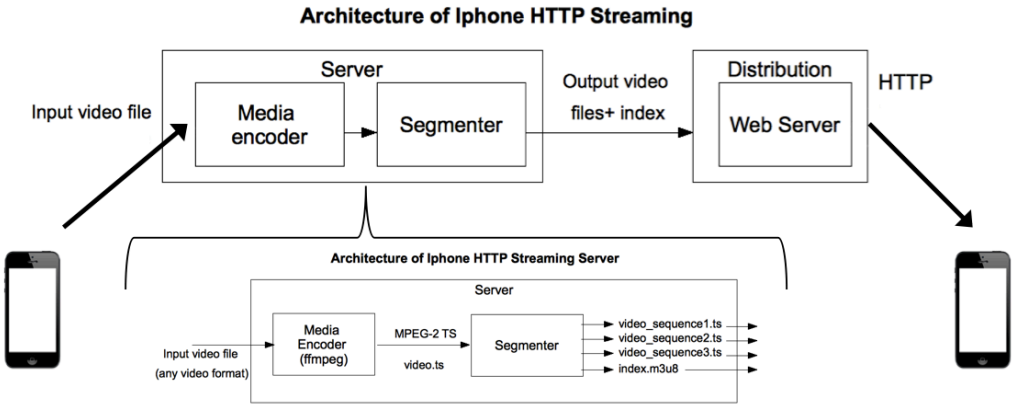


Figura 4.4: Servidor de vídeo (primera opción)

Opción B

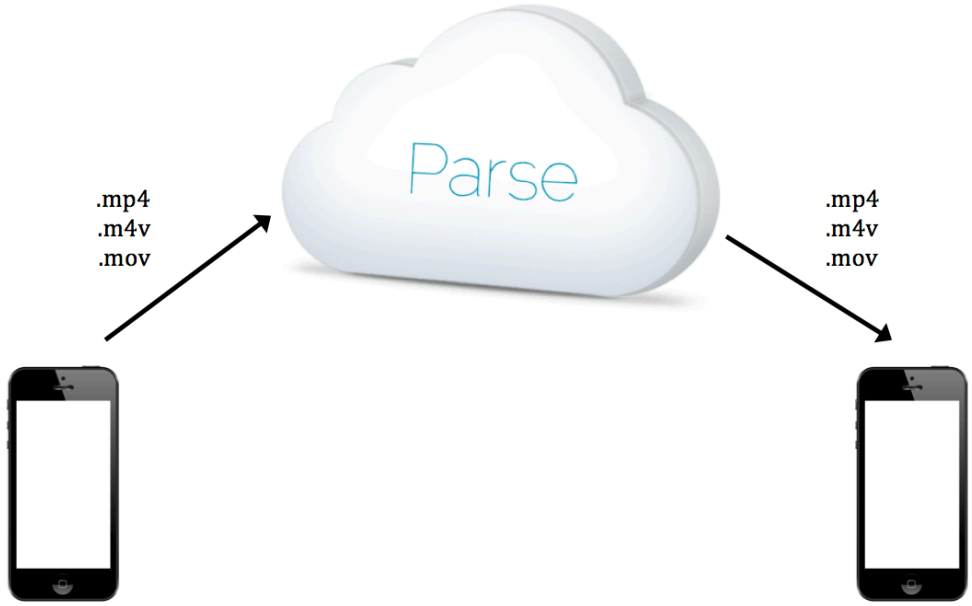


Figura 4.5: Servidor de vídeo (segunda opción)

Opción C

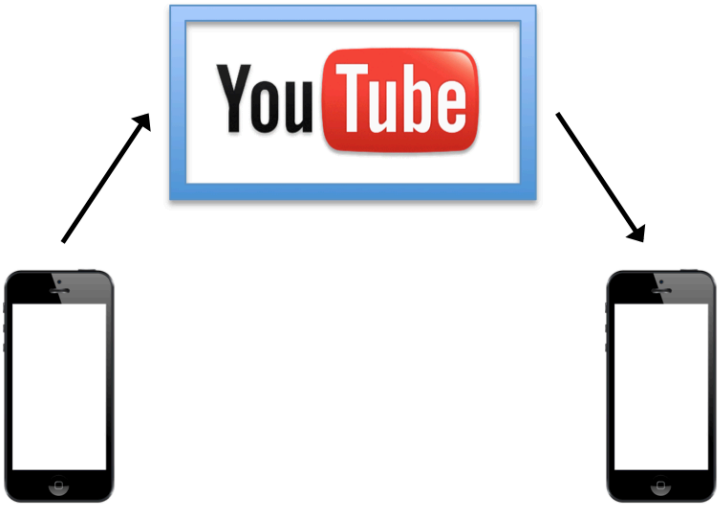


Figura 4.6: Servidor de vídeo (tercera opción)

	Encoder/segmenter Opción A	Parse/ Opción B	YouTube/ Opción C
Comienzo de la reproducción	Rápida	Lenta	Rápida
Consumo de datos	Adecuado. En relación al visionado del vídeo.	Excesivo (todo el vídeo).	Adecuado. En relación al visionado del vídeo y a la velocidad de red disponible.
Complejidad de integración con ios	Alta	Baja.	Media.
Complejidad de la Implementación del sistema	Alta*	Nula	Nula

*Existen sistemas que ya lo hacen como Amazon Elastic Transcoder, Zencoder, etc.

5.- Diseño

5.1.- Diagrama de clases del sistema.

A continuación se presenta el diagrama de clases donde se modela el sistema de información que se utilizará para el desarrollo de este proyecto, en el se podrán ver las relaciones que existen entres los diferentes objetos.

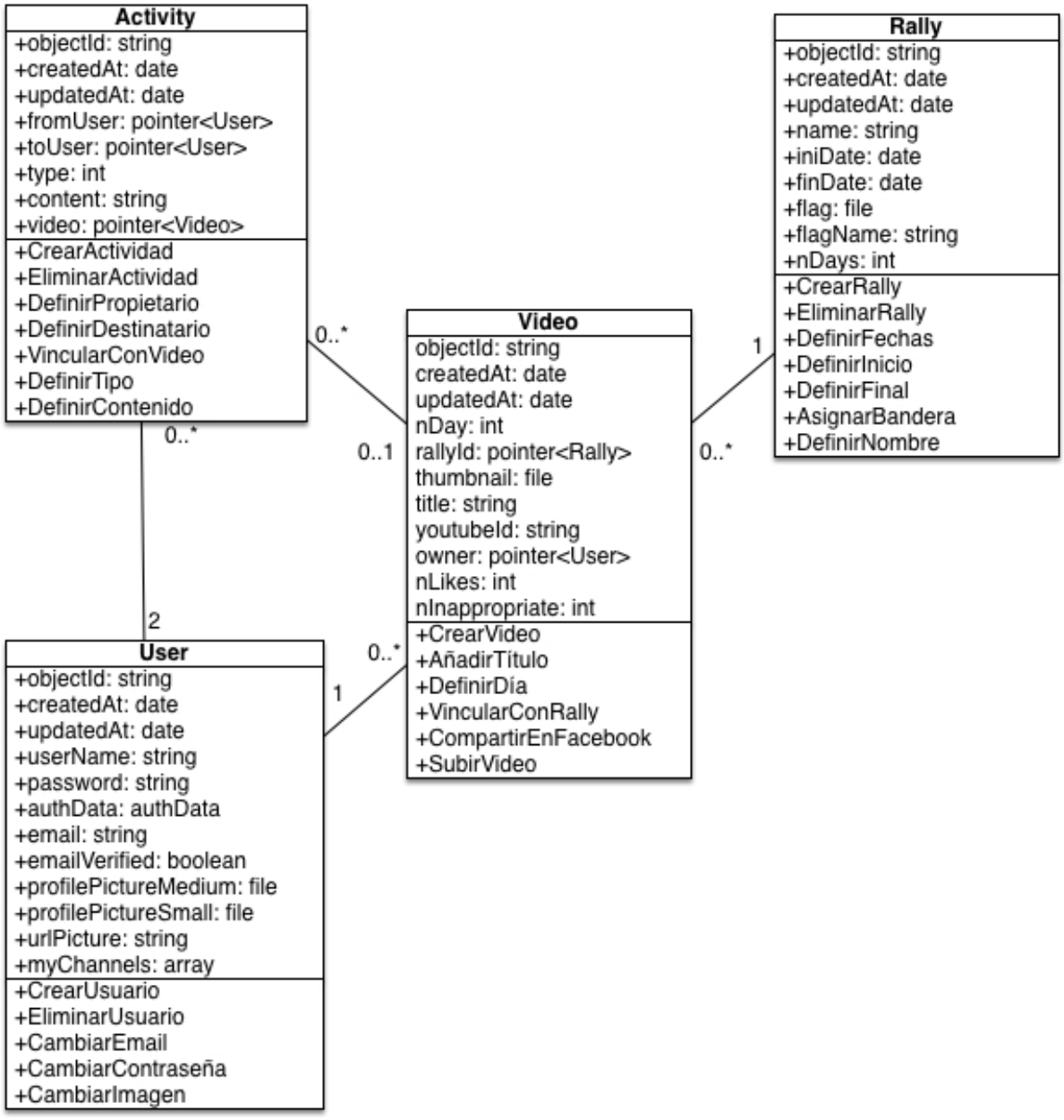


Figura 5.1: Diagrama de clases

5.2.- Diagramas de secuencia.

Los diagramas de secuencia están elaborados en lenguaje UML y representan de forma esquemática las acciones que debe ir realizando un rol en el sistema a lo largo de un periodo de tiempo para completar una acción. La elaboración de estos diagramas se apoya en el desarrollo previo de los diagramas de clases y los casos de uso que son elementos típicos en la ingeniería del software.

Iniciar sesión en el sistema

Los siguientes diagramas de secuencia representan las distintas acciones que el usuario debe realizar para iniciar sesión en el sistema:

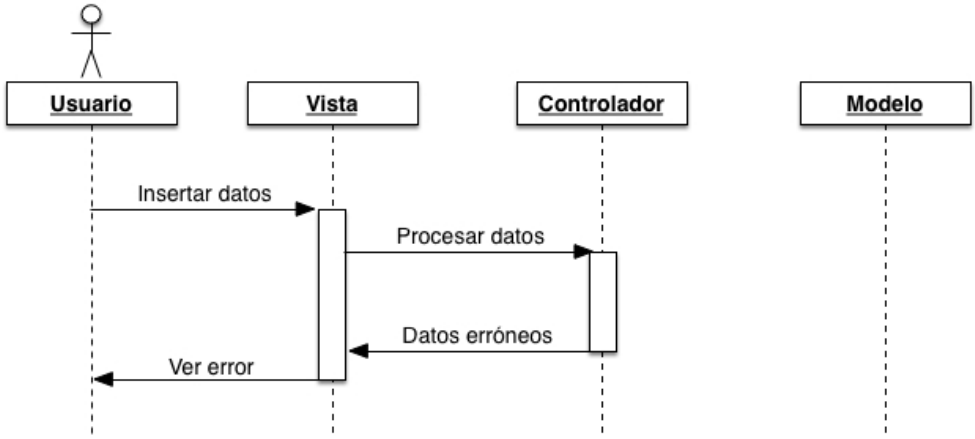


Figura 5.2: Diagrama secuencia (inicio de sesión 1)

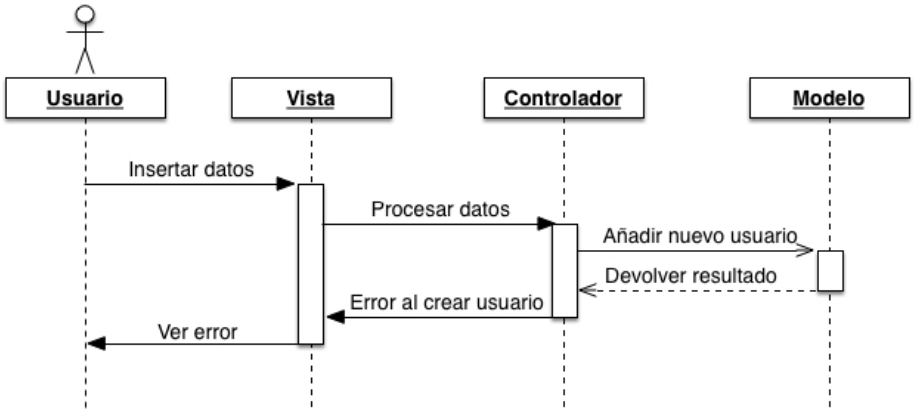


Figura 5.3: Diagrama secuencia (inicio de sesión 2)

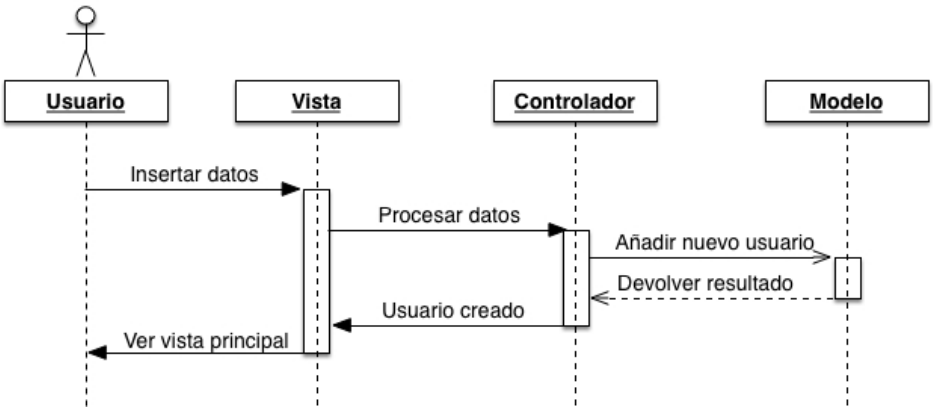


Figura 5.4: Diagrama secuencia (inicio de sesión 3)

El actor de este diagrama es un usuario que no se encuentra registrado en el sistema, este rellenará un formulario donde se le solicita una imagen, un nombre de usuario, una contraseña y un email. Una vez proporcionados los datos anteriores, el usuario intentará acceder pulsando sobre un botón, esta acción hará que el controlador ejecute una serie de comprobaciones, estas, se encargarán de comprobar que el email tiene una estructura correcta y que la contraseña cuenta con más de 6 caracteres, si se cumplen estos requisitos, el controlador envía la información de registro al modelo, con el objetivo de crear un nuevo usuario, en caso contrario, se informa al usuario de que alguno de los datos introducidos es erróneo. Una vez ejecutada la acción de "crear nuevo usuario" en el modelo, el controlador recibe un aviso y ejecuta un código donde se comprobará si ha habido algún error, en caso afirmativo, se muestra un mensaje de error al usuario, por el contrario, si todo ha ido bien, se muestra la vista principal de la aplicación.

Publicar vídeo

A continuación se muestra la secuencia de acciones que un usuario debe seguir para publicar un vídeo en el sistema:

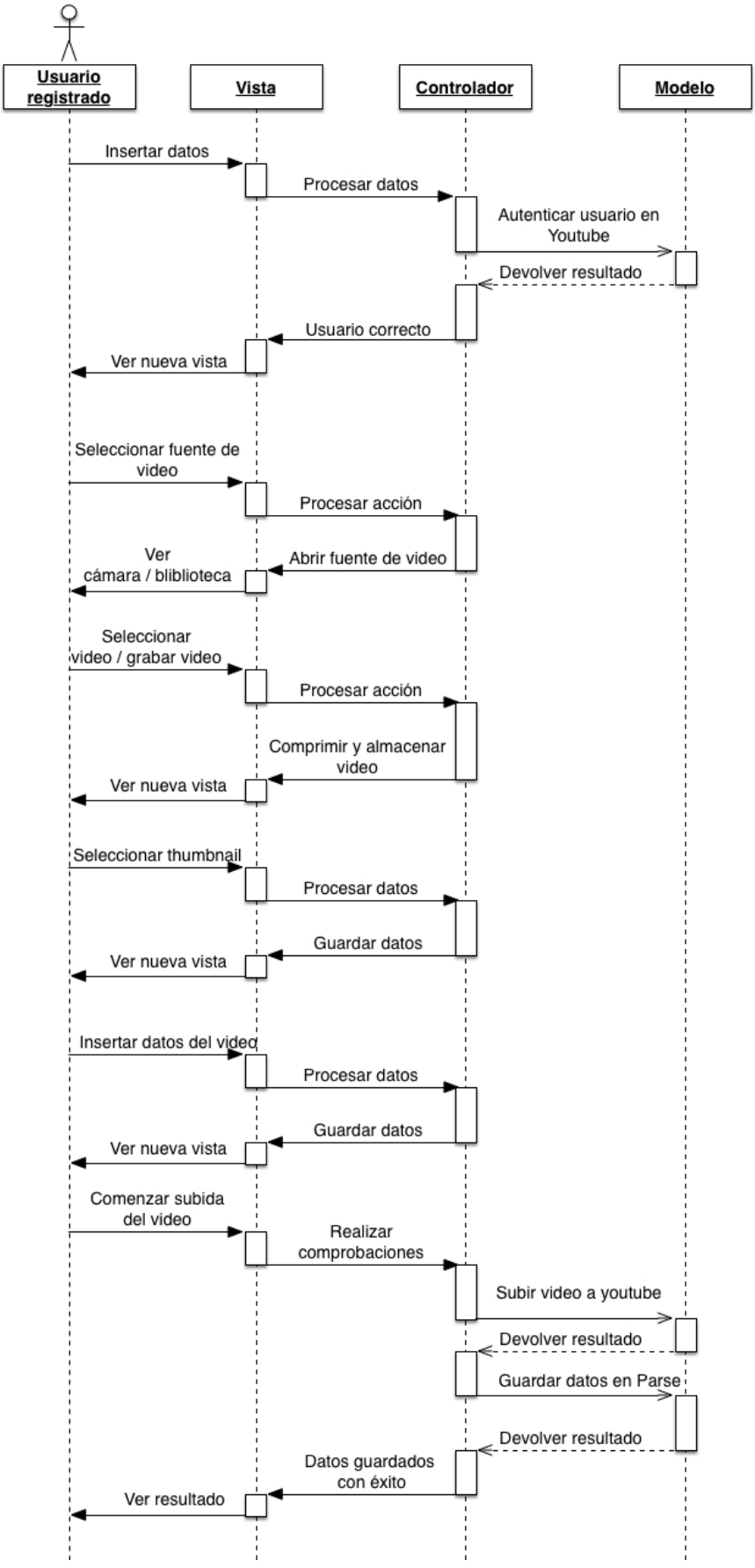


Figura 5.5: Diagrama secuencia (publicar vídeo)

El actor en este diagrama se trata de un usuario ya registrado en el sistema. Este usuario desea publicar un nuevo vídeo por lo que deberá subirlo al sistema. Lo primero que se le solicita al usuario son los datos de acceso a su cuenta de YouTube, una vez introducidos el controlador se encarga de establecer conexión con los servidores de YouTube e iniciar una sesión válida. Si todo va bien el controlador procesará la información recibida y se mostrará al usuario la siguiente vista del proceso. En esta nueva vista se da la opción de elegir la fuente de vídeo (cámara o biblioteca), cuando el usuario elige una de las opciones el controlador se encarga de realizar las gestiones necesarias para mostrar al usuario el medio seleccionado. En este punto el usuario se encarga de seleccionar/ crear un vídeo, una vez hecho esto el controlador se encarga de comprimir el vídeo y se le muestra la siguiente vista del proceso al usuario. En esta nueva vista, el usuario seleccionará el "thumbnail" que se mostrará como vista representativa del vídeo, al seleccionarla, el controlador guarda la imagen localmente y se presenta la siguiente vista al usuario. En esta nueva vista el usuario introducirá la información relativa al vídeo, a continuación el controlador la procesará y la almacenará. Luego, se muestra la ultima vista del proceso de publicación, en ella el usuario confirma que quiere subir el vídeo, llegados a este punto, el controlador comprueba que se dispone de toda la información necesaria y comienza la subida del vídeo a los servidores de YouTube, una vez finalizada, el controlador recibe una aviso y ejecuta un código de comprobación. Si el controlado comprueba que no ha habido errores en el proceso de subida del vídeo, este comienza con el procesado de la información. Una vez que se dispone de toda la información referente al vídeo, el controlado envía dicha información al sistema para que sea almacenada. Si todo el proceso ha ido bien el controlador recibirá un aviso de que la información se han guardado con éxito y se le mostrará al usuario un aviso indicándole que el vídeo se ha publicado correctamente.

Buscar objetos en el sistema

La siguiente secuencia representa las acciones que el usuario de be realizar para buscar usuarios o vídeos en el sistema:

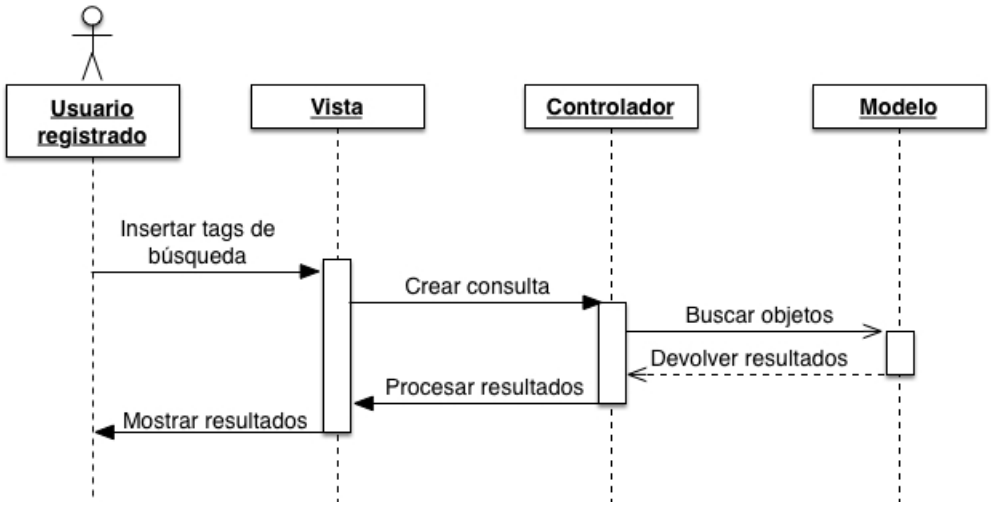


Figura 5.6: Diagrama secuencia (buscar)

Al igual que en el diagrama anterior, el actor es un usuario registrado. Lo primero que debe hacer para buscar "algo" (usuarios o vídeos), es insertar unos tags de búsqueda. Una vez insertados, el controlador genera una consulta y la envía al sistema. La base de datos buscará todos los objetos que se encuentren dentro del rango de búsqueda generado por el controlador y devolverá una lista. El controlador procesará dicha lista para generar una vista para el usuario.

5.3.- Diseño de prototipos para las interfaces de usuario.

Los prototipos de las interfaces de usuario son unos elementos muy importantes en la ingeniería del software y ayudan a representar el aspecto final que debe tener la aplicación. A continuación se muestran unos diseños por los que se guiará la implementación de la aplicación:

Vista inicial de la aplicación:

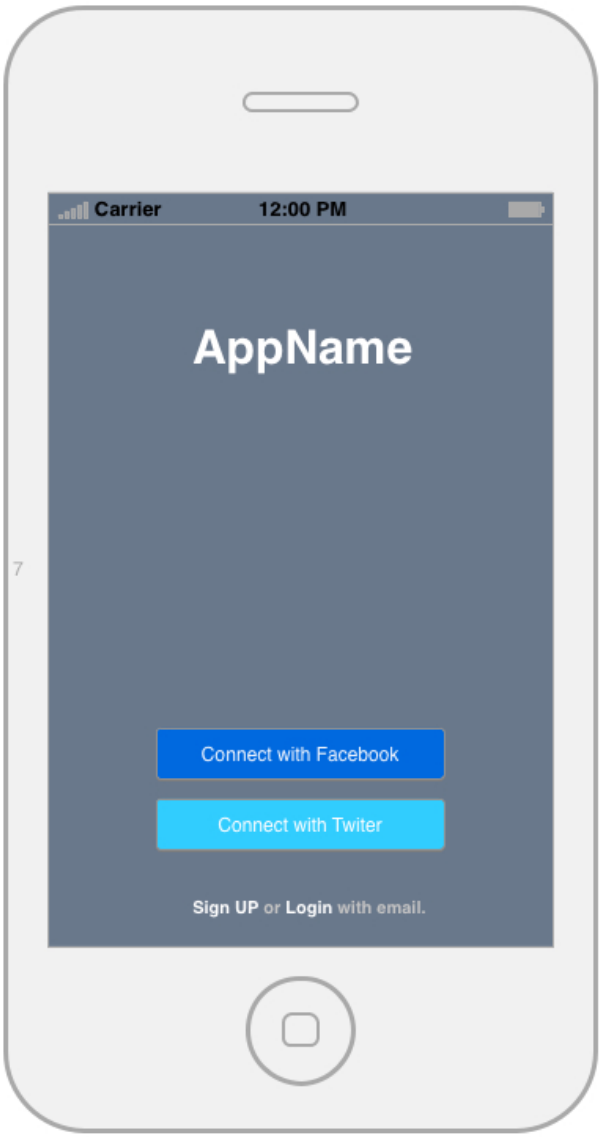


Figura 5.7: Prototipo de la vista inicial

Vista de registro:

Figura 5.8: Prototipo de la vista de registro

Vista de autentificación de usuario:

Figura 5.9: Prototipo de la vista de autentificación

Vista principal de la aplicación:

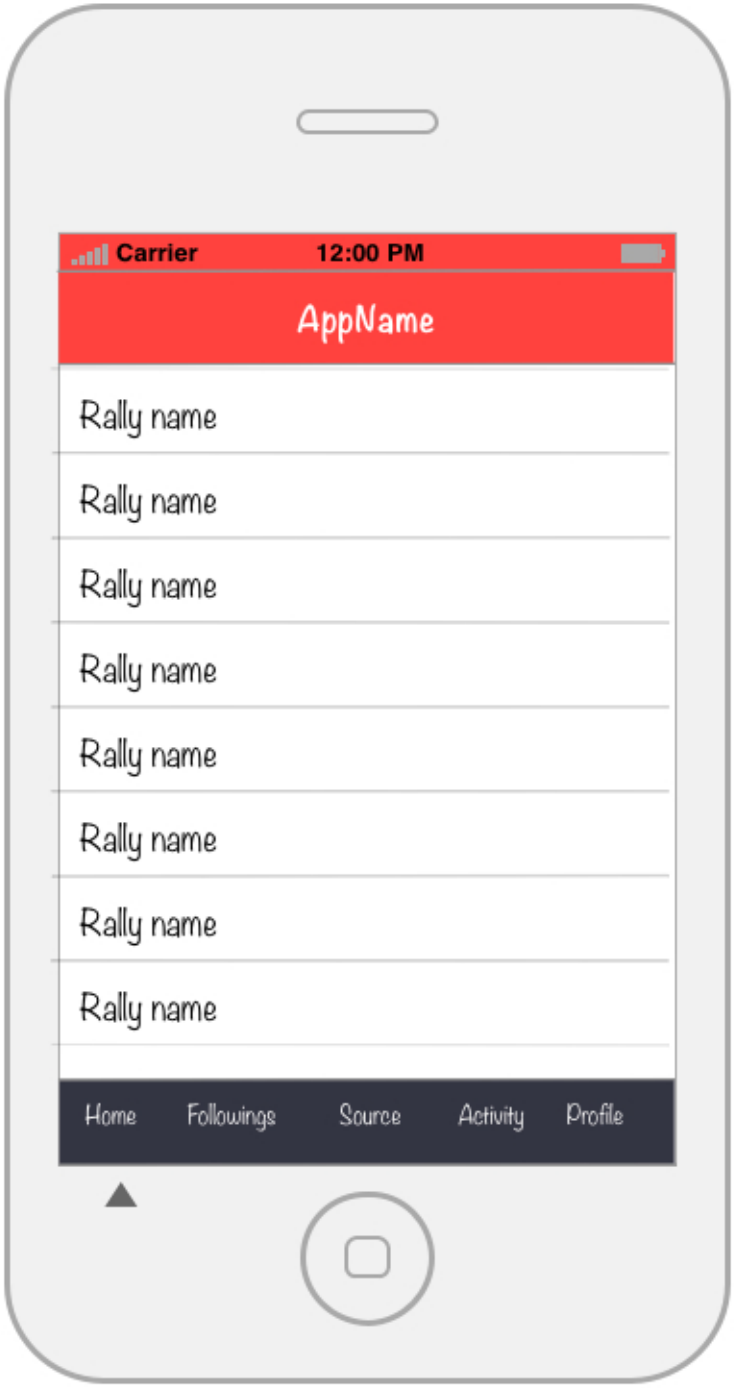


Figura 5.10: Prototipo de la vista principal

Vista de comentarios:

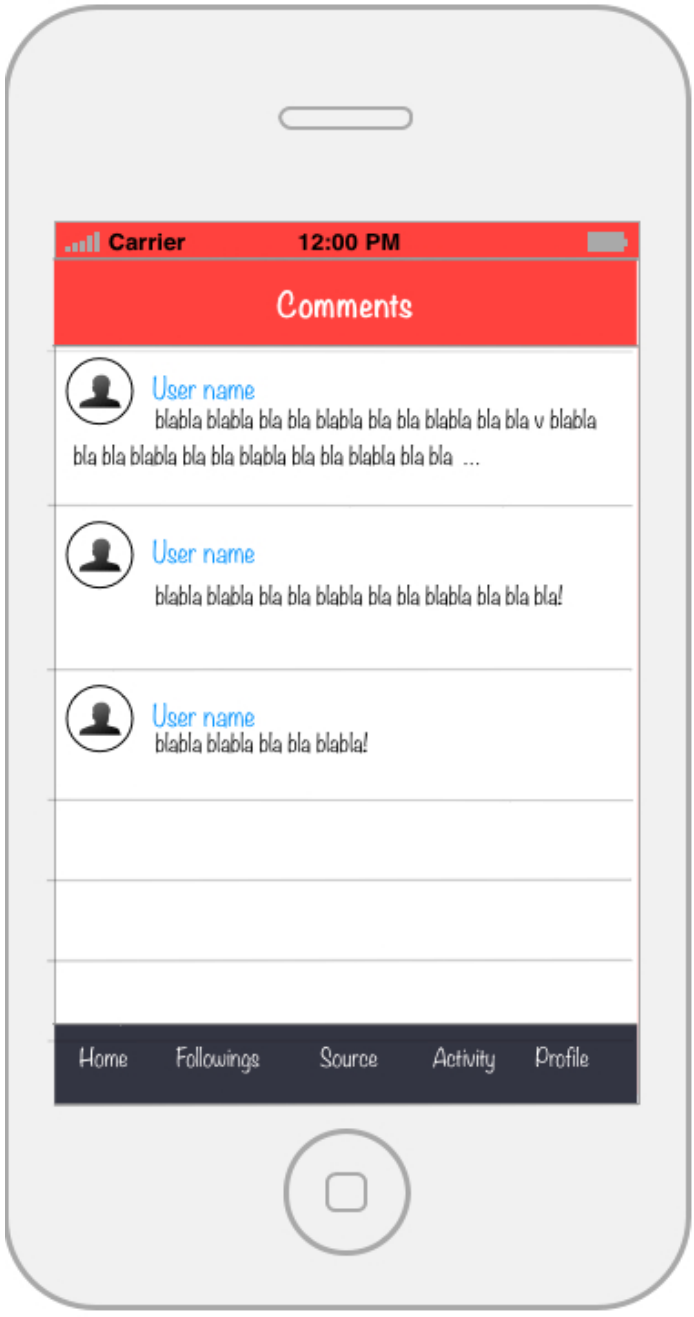


Figura 5.11: Prototipo de la vista de comentarios

Vista del perfil de usuario:

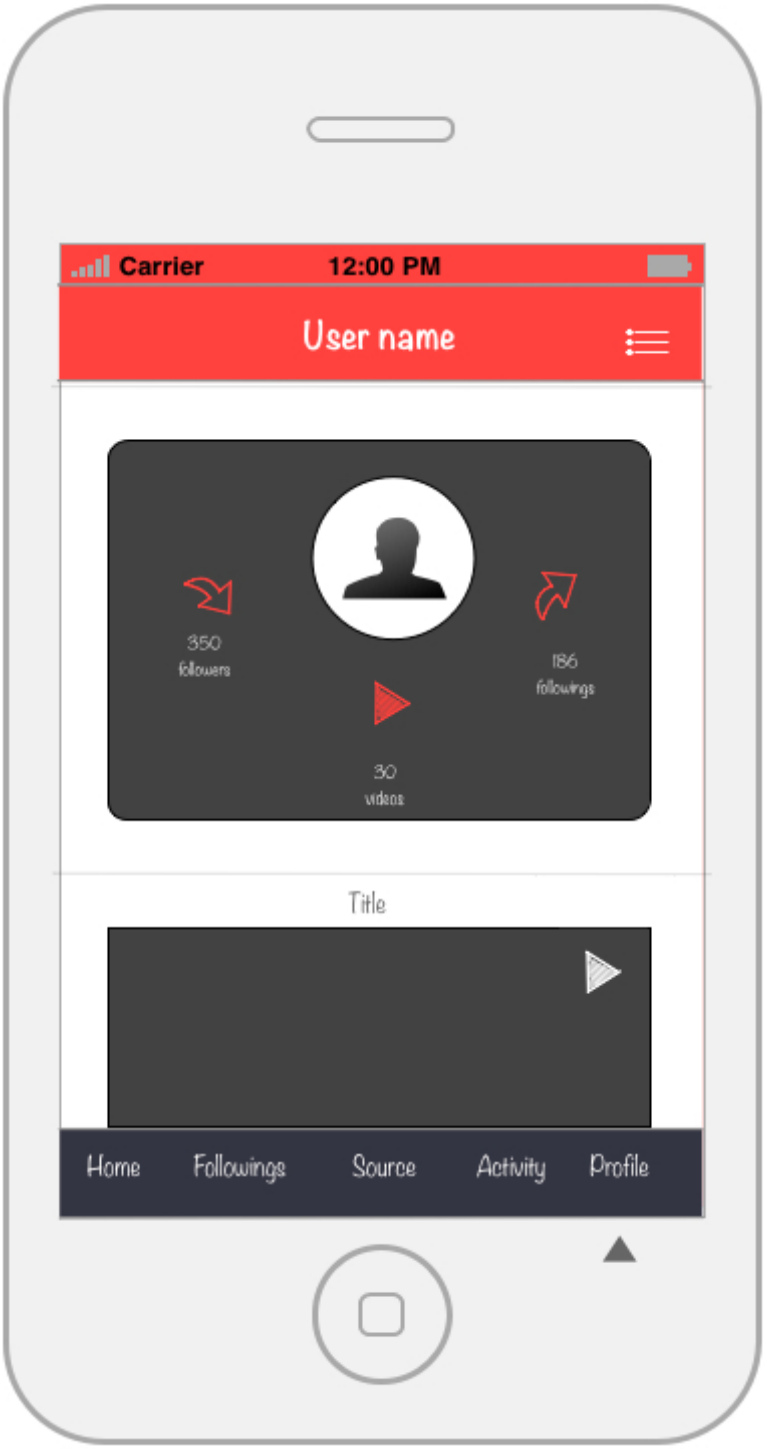


Figura 5.12: Prototipo de la vista de perfil

Vista detallada de un rallye:

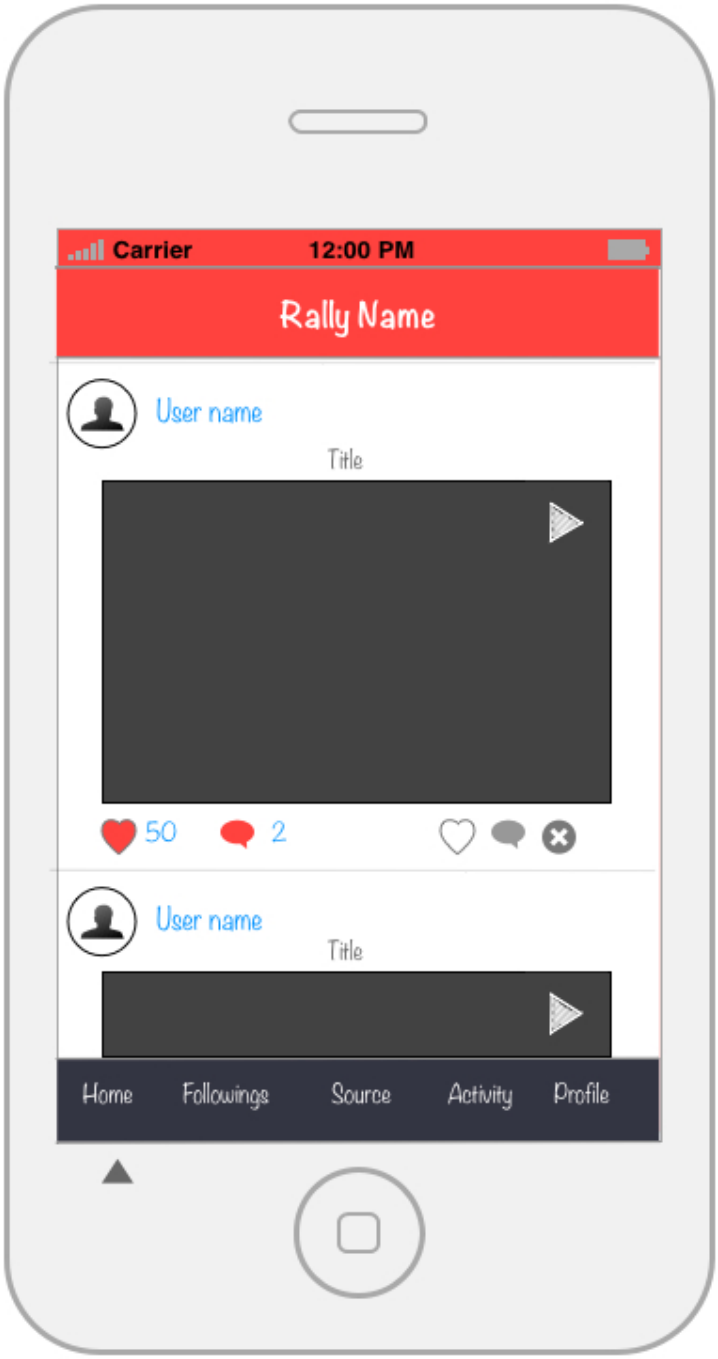


Figura 5.13: Prototipo de la vista detallada de un rallye

5.4.- Diagrama entidad relación.

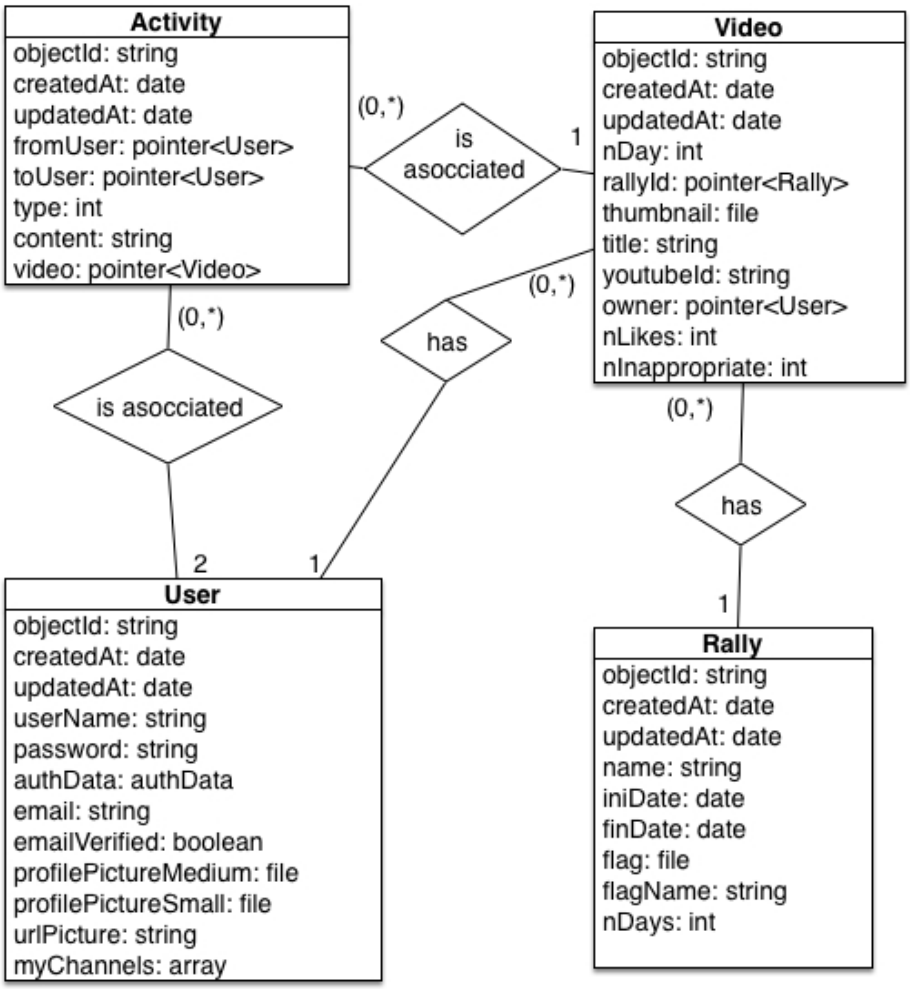


Figura 5.14: Diagrama entidad relación

5.5.- Diseño arquitectónico.

Arquitectura de N-Capas donde las N-Capas se encuentran encapsuladas en los servicios en la nube Parse + Youtube. Esta arquitectura de N-capas presenta muchas ventajas frente a otras como la arquitectura cliente-servidor, algunas de estas ventajas son una alta tolerancia a fallos, una alta disponibilidad, una alta escalabilidad, etc. Desde nuestro punto de vista, se podría decir que la arquitectura es una especie de cliente servidor, ya que la arquitectura de N-Capas es transparente para nosotros (usuarios de estos servicios), pero en el fondo somos conocedores las ventajas que supone.

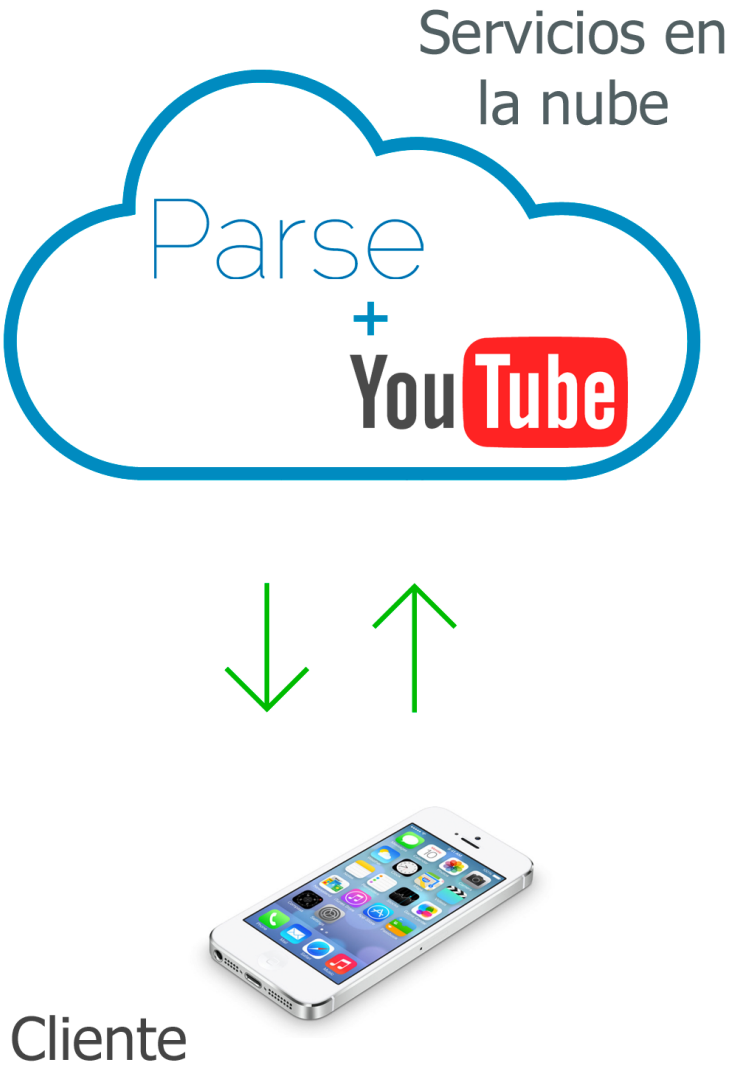


Figura 5.15: Arquitectura del sistema

5.5.1.- Parse

La arquitectura de un servicio de esta características es bastante compleja, Parse utiliza la infraestructura de Amazon Web Services para albergar su sistema.

-Como sistemas de gestión de bases de datos, Parse utiliza Cassandra, MySQL y MongoDB.

-Para permitir ejecutar código en la nube, Parse utiliza la máquina virtual V8, la cual permite ejecutar código javascript. De esta manera, se permite a los usuarios ejecutar su propio código e interactuar con los objetos almacenados en sus bases de datos.

A continuación se muestra un diagrama donde se puede observar la complejidad del sistema que se encuentra detrás de este backend:

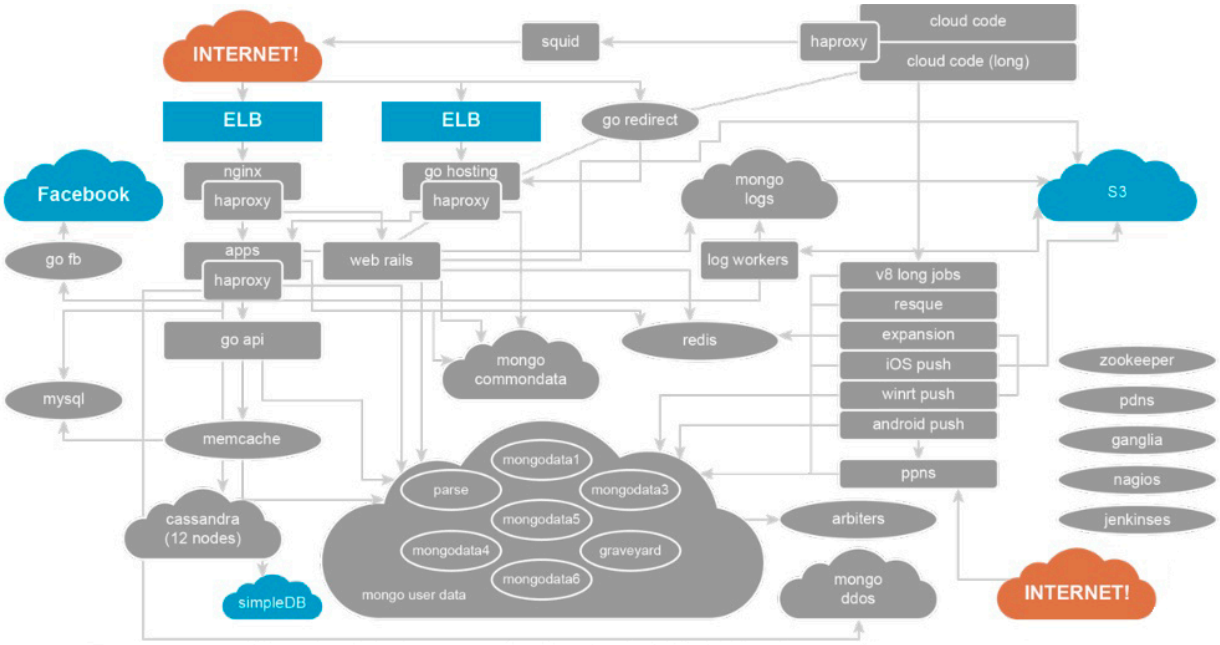


Figura 5.15: Arquitectura de Parse

6.- Desarrollo e implementación.

En los siguientes apartados se abordará todo el proceso de desarrollo de nuestro sistema y se explicará el por qué de algunas decisiones tomadas. Además se expondrán diversos problemas que han surgido así como las soluciones adoptadas.

6.1.- Configuración e implementación del backend.

En este apartado se tratará todo lo relacionado con el backend, veremos su configuración así como parte del código que se ejecuta en él.

Lo primero de todo fue crear una cuenta en www.parse.com y una vez dentro del sistema, crear una "App":

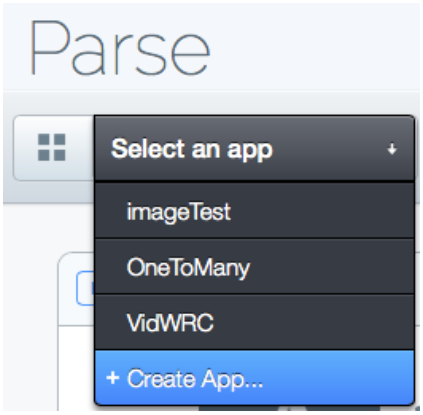


Imagen 6.1: Creación de una nueva "App" en Parse

Una vez creada la “App” ya tenemos acceso al entorno de configuración y podremos comenzar con la configuración el nuestro sistema.

6.1.1.- La base de datos.

Parse utiliza mongoDB como sistema de bases de datos y una de sus características más importantes es que se trata de un sistema de base de datos NoSQL. Dentro de las bases de dato NoSQL se distinguen varios tipos dependiendo de la forma de almacenar la información y mongoDB se engloba dentro de las bases de datos documentales ya que almacena la información como un documento. MongoDB utiliza para ello una estructura simple como es JSON y utiliza una clave única para cada registro.

Una vez conocido el tipo de base de datos con el que se trabajará pasamos a ver como ha sido el proceso de creación y configuración:

1. Lo primero de todo es crear las clases (“tablas”) de nuestra aplicación, para ello solo hay que seguir una serie de pasos muy sencillos. Parse proporciona clases con sus estructuras ya predefinidas, entre ellas la clase “User”, la cual hemos usado.

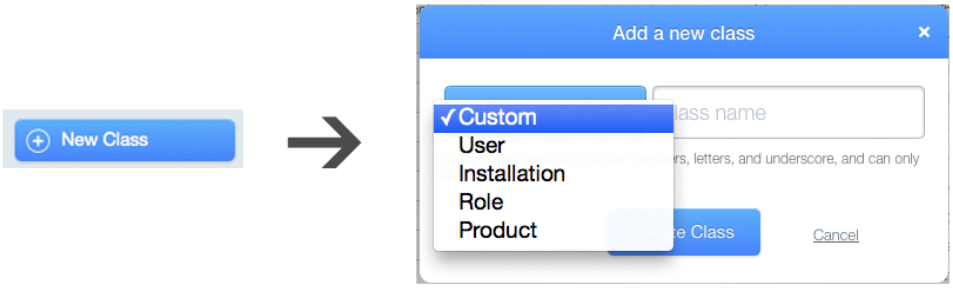


Imagen 6.2: Creación de una nueva clase en Parse

2. Una vez creadas las clases lo siguiente será definir la estructura (“columnas”) de cada clase:

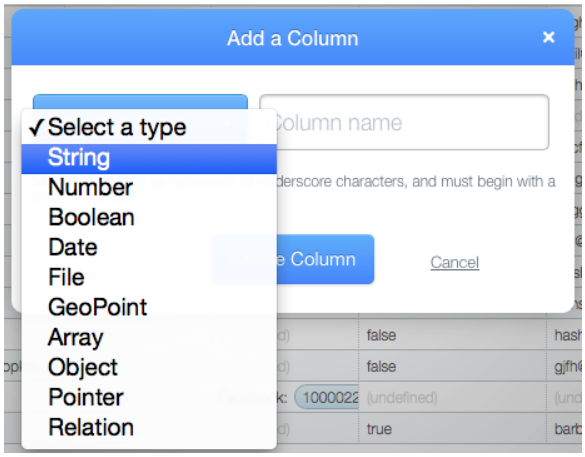


Imagen 6.3: Creación de una nueva columna en Parse

Una vez hecho esto ya tenemos nuestro modelo de datos reflejado en el sistema de bases de datos. A continuación se muestra la clase “User”:

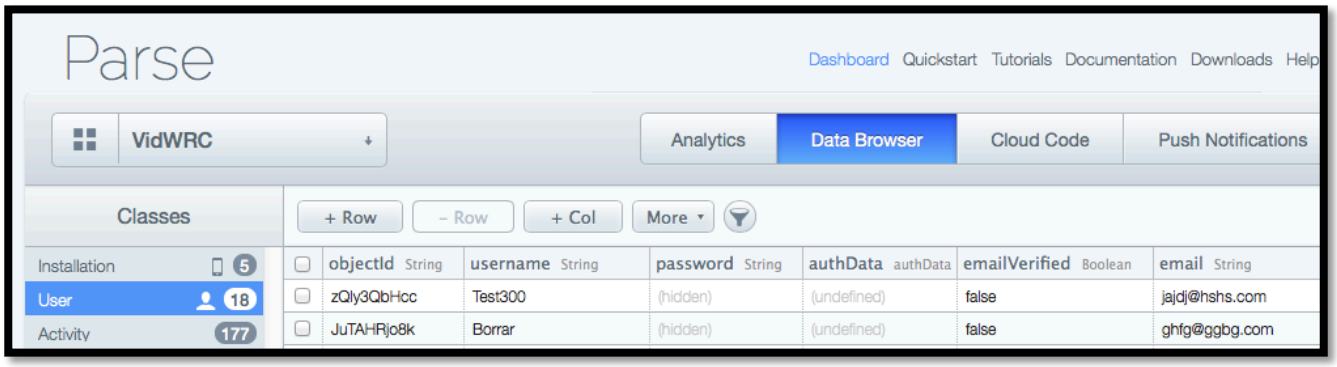


Imagen 6.4: Navegador de datos de Parse

```
{
        "objectId":"string"
        "username":"string"
        "bcryptPassword:"string"
        "email":"string"
        "emailVerified":"boolean"
        "profilePictureMedium":"file"
        "profilePictureSmall":"file"
        "sessionToken":"string"
        "createdAt":"date"
        "updatedAt":"date"
}
```

¿Cómo se relacionan los objetos en nuestra base de datos?

Al especificar la estructura de una clase, se nos da la opción de indicar que un elemento sea de tipo “Pointer”, de tipo “Relation”, o de tipo “Array”.

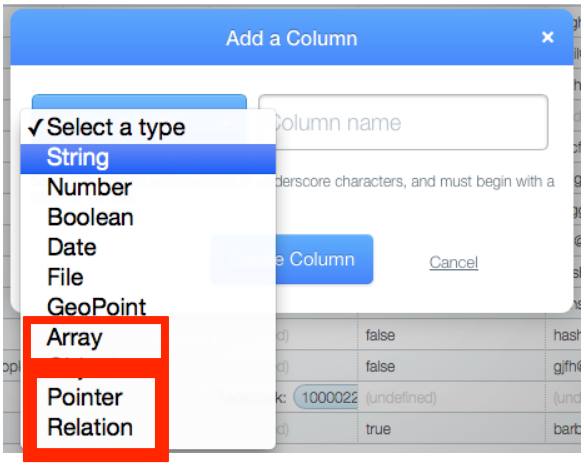


Imagen 6.5: Tipos de relaciones

- Pointer (adecuado para definir relaciones uno-a-uno y uno-a-muchos).
- Arrays (adecuado para definir relaciones uno-a-muchos y muchos-a-muchos).
- Relation (adecuado para definir relaciones muchos-a-muchos).

Dependiendo del tipo de relación que queramos modelar se usará un tipo u otro, en nuestro caso hemos usado únicamente el tipo “Pointer”. Un objeto de tipo “pointer” no es más que una referencia a un objeto, pudiendo ser este objeto de cualquier clase. A través de esa referencia (“pointer”) podremos navegar y obtener información del objeto referenciado (esto se verá de manera más clara en el código).

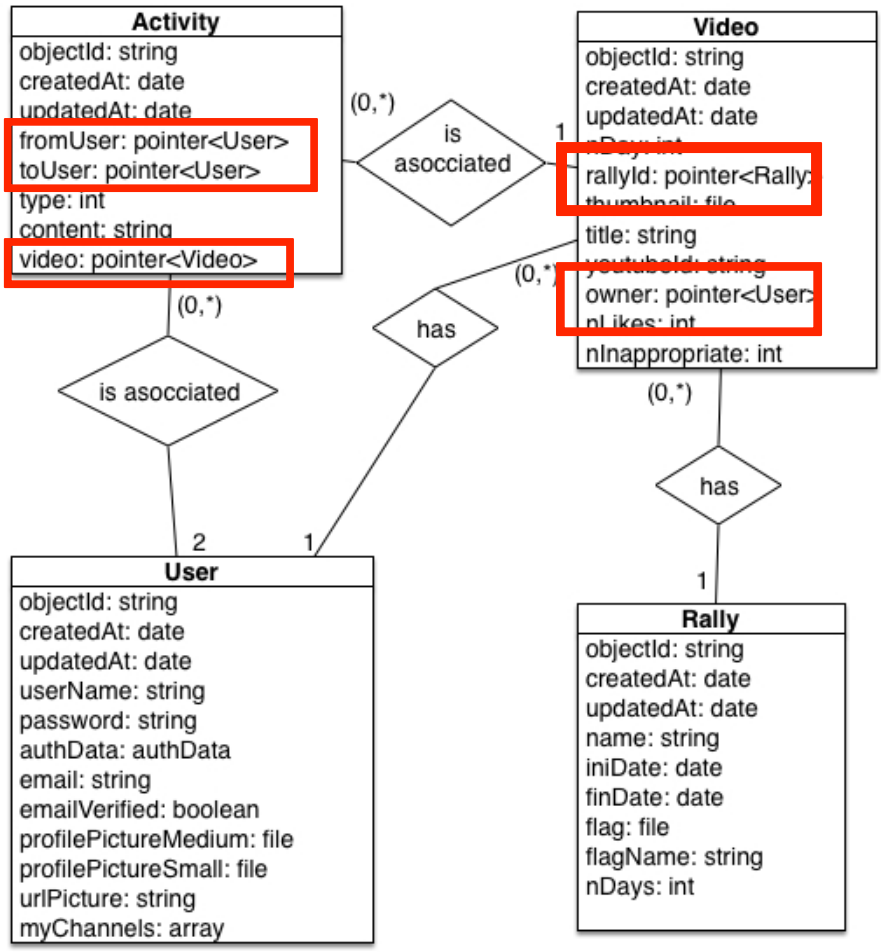


Figura 6.1: Objetos de tipo puntero utilizados

6.1.2.- Cloud code.

El concepto de **Cloud Code** podría resumirse básicamente como código de nuestra aplicación que se ejecuta en la nube.

Este código en la nube puede ejecutarse o bien desde el código de nuestra app, mediante una sencilla llamada, o bien por la persistencia de un nuevo registro en una tabla de nuestro modelo de datos.

En nuestro caso solo usamos el código en la nube para ejecutar acciones ante cambios en nuestra base de datos, a continuación se explicarán los diferentes métodos que se ejecutan y ante que circunstancias se ejecutan:

Eliminar actividades relacionadas con un vídeo:
Este método se ejecuta cuando se elimina un vídeo de la base de datos, lo que hace es eliminar todas las actividades relacionadas con el vídeo ya que carecen de sentido si el vídeo ya no existe.

```
Parse.Cloud.afterDelete("Vídeo", function(request) {
  query = new Parse.Query("Activity");
  query.equalTo("video", request.object);
  query.find({
    success: function(Activity) {
      Parse.Object.destroyAll(Activity, {
        success: function(ok) {
          console.log("Actividades eliminadas correctamente");
        },
        error: function(error) {
          console.error("Error al eliminar actividades:");
        }
      });
    },
    error: function(error) {
      console.error("Error al buscar actividades");
    }
  });
});
```

Eliminar todo lo relacionado con un usuario:

Este método se ejecuta cuando se elimina una cuenta de usuario. Este método lo que hace es llamar a dos funciones, una encargada de eliminar los vídeos relacionados con el usuario y otra encargada de eliminar las actividades relacionadas con este usuario.

```
/*Eliminar todo lo relacionado con un usuario*/
Parse.Cloud.afterDelete("_User", function(request) {
deleteUserActivityRelated(request.object, {
    success: function(returnValue) {
      response.success();
    },
    error: function(error) {
      response.error(error);
    }
  });

deleteUserVideoRelated(request.object, {
    success: function(returnValue) {
      response.success();
    },
    error: function(error) {
      response.error(error);
    }
  });

});
```

```
/*Eliminar actividades generadas o dirigidas hacia un usuario*/
function deleteUserActivityRelated(user,callback){
query = new Parse.Query("Activity");
  query.equalTo("fromUser", user);
  query.find({
    success: function(Activity) {
      Parse.Object.destroyAll(Activity, {
        success: function(ok) {
          console.log("Actividades eliminadas correctamente (2)");
        },
        error: function(error) {
          console.error("Error al eliminar actividades (2)”);
        }
      });
    },
    error: function(error) {
      console.error("Error buscando actividades para eliminar (2)”);
    }
  });

  query = new Parse.Query("Activity");
  query.equalTo("toUser", user);
  query.find({
    success: function(Activity) {
      Parse.Object.destroyAll(Activity, {
        success: function(ok) {
          console.log("Actividades eliminadas correctamente (3)");
        },
        error: function(error) {
          console.error("Error al eliminar actividades (3)");
        }
      });
    },
    error: function(error) {
      console.error("Error buscando actividades para eliminar (3);
    }
  });
}

/*Eliminar vídeos publicados por un usuario*/
function deleteUserVideoRelated(user,callback){
query = new Parse.Query("Vídeo");
  query.equalTo("owner", user);
  query.find({
    success: function(vídeos) {
      Parse.Object.destroyAll(vídeos, {
        success: function(ok) {
          console.log("Vídeos eliminados correctamente");
        },
        error: function(error) {
          console.error("Error al eliminar vídeos");
        }
```

```
      });
    },
    error: function(error) {
      console.error("Error al buscar vídeos para eliminar);
    }
  });
}
```

Sistema de censura automática de vídeos
Este sistema consta de tres métodos:

El primero de ellos se ejecuta cuando se añade una nueva actividad de tipo “vídeo inapropiado”, este método se encarga de incrementar el contador de reportes de cada vídeo; si un contador llega al límite establecido de 10 reportes por “vídeo inadecuado”, se ejecuta una sección de código donde se copia el vídeo reportado a una tabla no accesible por los usuarios (Censored_Vídeos).

El segundo de los métodos se ejecuta cuando se elimina una actividad de tipo “vídeo inapropiado” y se encarga de decrementar el contador de reportes.

Y el tercero de los métodos, que se ejecuta justo después de copiar un vídeo a la tabla de censura (tabla inaccesible por los usuarios), elimina el vídeo de la tabla “Vídeo” que si es accesible por los usuarios. De esta forma es como se censuran los vídeos (cambiándolos de tabla). Los vídeos almacenados en la tabla “Censored_Vídeos” podrán ser examinados posteriormente por el administrador del sistema, y vueltos a publicar si se consideran adecuados o, en caso contrario, eliminados por completo del sistema.

```
/*Ejecutar acción al añadir una actividad de tipo “inappropriate” -->
incrementar nInap de Vídeo*/
Parse.Cloud.afterSave("Activity", function(request) {
  if(request.object.get("type") == 3){
      actividad = request.object;
      query = new Parse.Query("Vídeo");
      query.get(request.object.get("vídeo").id, {
      success: function(vídeo) {
        vídeo.increment("nInap");
        vídeo.save();
        if (vídeo.get('nInap')>0) {
            var CensoredVideo = Parse.Object.extend("Censored_Vídeos");
            var censoredVideo = new CensoredVideo();
            censoredVideo.set("nDay",vídeo.get("nDay"));
            censoredVideo.set("title",vídeo.get("title"));
            censoredVideo.set("thumbnail",vídeo.get("thumbnail"));
            censoredVideo.set("rallyId",vídeo.get("rallyId"));
            censoredVideo.set("youtubeID",vídeo.get("youtubeID"));
            censoredVideo.set("oldObjectId",vídeo.id);
            censoredVideo.set("nLikes",0);
            censoredVideo.set("nInap",0);
            censoredVideo.save();
        }
      },
      error: function(error) {
      request.object.destroy({
          success:function() {
            console.log("Actividad Inappropriate eliminada ya que no existe
vídeo asociado");
          },
            error:function(error) {
           response.error('No se pudo eliminar actividad Inappropriate sin
vídeo asociado');
          }
      });
    }
  });
  }
});

/*Ejecutar acción al eliminar una actividad de tipo “inappropriate” -->
decrementar nInap de Vídeo*/
Parse.Cloud.afterDelete("Activity", function(request) {
  if (request.object.get("type") == 3){
      query = new Parse.Query("Vídeo");
      query.get(request.object.get("vídeo").id, {
      success: function(vídeo) {
        vídeo.increment("nInap",-1);
        vídeo.save();
      },
      error: function(error) {
```

```
        console.error("Error al decrementar nInap en vídeo: ");
    }
  });
  }
});

/*Eliminar vídeo de la tabla Vídeo para que no pueda ser visualizado*/
Parse.Cloud.afterSave("Censored_Vídeos", function(request) {
  query = new Parse.Query("Vídeo");
  query.equalTo("objectId", request.object.get("oldObjectId"));
  query.find({
    success: function(Vídeo) {
      Parse.Object.destroyAll(Vídeo, {
        success: function(ok) {
          console.log("Vídeo censurado correctamente");
        },
        error: function(error) {
          console.error("Error al eliminar vídeo para censurarlo");
        }
      });
    },
    error: function(error) {
      console.error("Error al buscar vídeo para censurar");
    }
  });
});
```

Sistema automático de envió de notificaciones

Este sistema consta de un único método que se ejecuta justo después de la publicación de un vídeo en el sistema. Este método es el encargado de enviar una notificación a los usuarios del sistema con información del nuevo vídeo.

```
Parse.Cloud.afterSave("Vídeo", function(request) {
  if (!request.object.existed()){
    var rallyDay = request.object.get("nDay")
    query = new Parse.Query("Rallyes");
    query.get(request.object.get("rallyId").id, {
    success: function(rallye) {
      var rallyName = rallye.get("name")
      var rallyId = rallye.id;
      var pushQuery = new Parse.Query(Parse.Installation);
      pushQuery.equalTo('channels','NewVideos');

      Parse.Push.send({
        where: pushQuery,
        data:{
        alert: "New vídeo added to: " + rallyName,
        badge: "Increment",
        type:"0",
        r:rallyId,
```

```
        d:rallyDay
      }
      }, {
      success: function() {
        console.log("Notificación enviada correctamente");
      },
      error: function(error) {
        console.log("Error al enviar notificación después de añadir vídeo”);
      }
      });
    },
    error: function(error) {
      console.error("Error al buscar rallye para enviar notificación");
    }
    });
  }
});
```

6.2.- Desarrollo de la aplicación iOS.

6.2.1.- Arquitectura de una aplicación iOS

Para poder entender mejor los siguientes apartados es importante conocer algunos aspectos básicos de las aplicaciones en iOS.

Todas las aplicaciones tienen un objeto UIAplication desde donde se controlan, esta es una clase singleton. Esta clase, UIAplication, usa un delegado para ajustar el comportamiento de la aplicación ante determinados eventos, este delegado es una subclase de UIApplicationDelegate y es donde el programador puede modificar código para adaptar el comportamiento a sus necesidades. Esta clase delegada, que a partir de ahora la llamaremos AppDelegate, será nuestro punto de partida a la hora de crear una nueva aplicación y nos será de gran utilidad a hora de gestionar eventos producidos por el sistema.

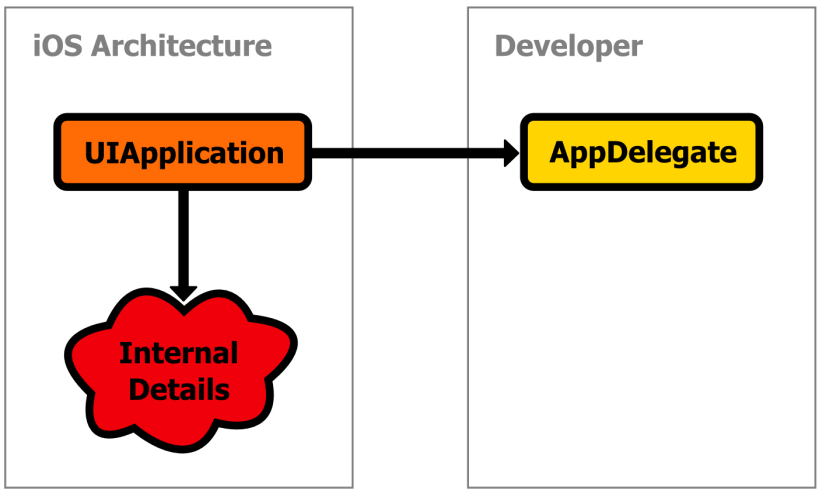


Figura 6.2: Arquitectura de una aplicación iOS

La clase AppDelegate cuenta con una propiedad de tipo UIWindow, esta propiedad contiene la ventana usada para presentar el contenido visual en la pantalla del dispositivo. Por lo tanto esta clase será la base sobre la que se mostrarán las nuevas vistas.

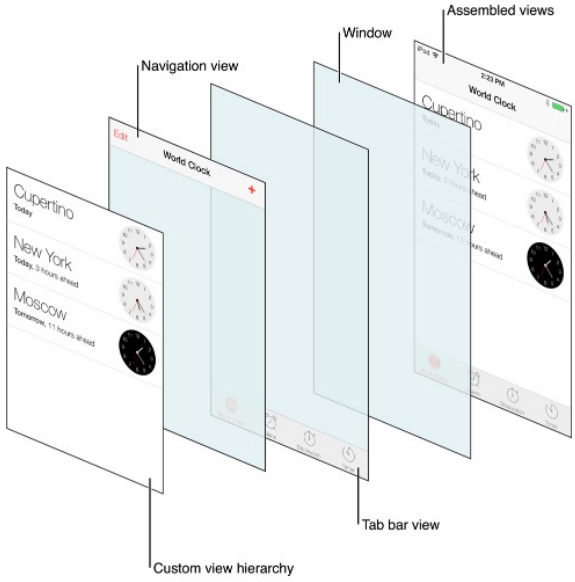


Figura 6.3: UIWindow

Una vez conocidos los ficheros básicos desde donde "arranca" una aplicación, vamos a ver como se estructura:

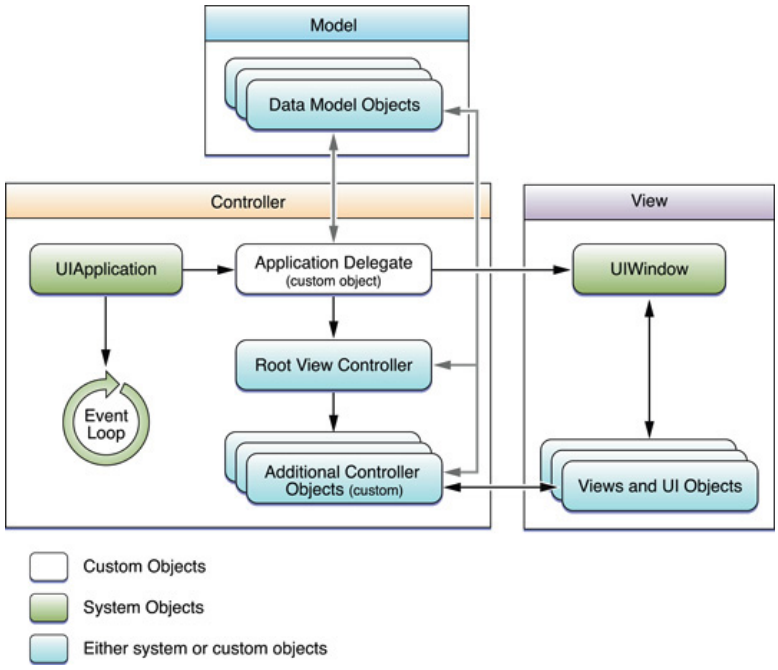


Figura 6.4: iOS - MVC

Como vemos en la figura anterior, el patrón seguido es el Modelo Vista Controllador, este

patrón presenta un diseño claro y fácil, a la vez que separa de manera clara las responsabilidades de cada parte.

- El **modelo** son los datos, independientes de su forma de representación.
- La **vista** son los objetos que se encargan de la representación gráfica del modelo y de los controles con los que el usuario puede interactuar.
- El **controlador** es el encargado de mediar entre la vista y el modelo, y es donde se coordina todo el trabajo. Este se encarga de acceder a los datos del modelo y de mostrarlos en la vista, también escucha eventos y manipula los datos como sea necesario.

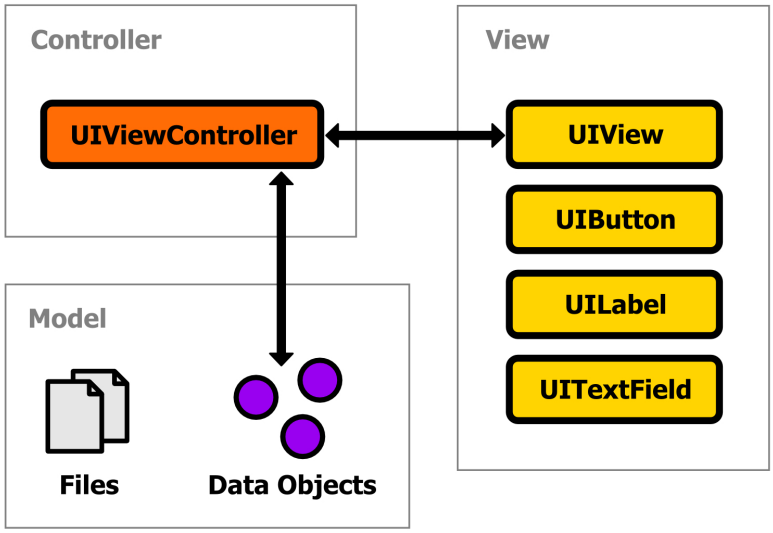


Figura 6.5: iOS - MVC

6.2.2.- Controles y librerías externas.

UIImage-Categories

Esta es una librería para el tratamiento y procesado de imágenes, fue creada en 2009 por Trevor Harmon. Dispone de varias funciones y métodos de los cuales se utilizarán:

Para redondear las esquinas de una imagen:

```
-(UIImage*)roundedCornerImage:(NSInteger)cornerSize
                   borderSize:(NSInteger)borderSize;
```

Para redimensionar una imagen y ajustar su calidad:

```
-(UIImage*)resizedImage:(CGSize)newSize
interpolationQuality:(CGInterpolationQuality)quality;
```

Imagen 6.6: Resultado de usar la función "`roundedCornerImage`"

Reachability

Esta es una librería para la monitorización del estado de la red, haciendo uso de ella la aplicación puede controlar en todo momento cuando dispone o no de conexión a internet.

FormatterKit

Es una librería para dar formato a diferentes tipos de datos y convertirlos en ristras legibles y en lenguaje natural. De entre la gran cantidad de funciones que tiene, nosotros únicamente utilizaremos una función, esta nos permitirá expresar el tiempo transcurrido entre dos fechas en lenguaje natural.

```
- (NSString *)stringForTimeIntervalFromDate:(NSDate *)startingDate
                                     toDate:(NSDate *)endingDate
```

Imagen 6.7: " stringForTimeIntervalFromDate:"

JSQFlatButton

Esta librería permite la creación de botones de manera fácil y rápida, además permite una mayor personalización que la que ofrecen la clase UIButton (clase por defecto que proporciona Apple).

Imagen 6.8: Ejemplo de un botón creado con la librería JSQFlatButton

RMMultipleViewsController

Esta es una clase que nos permite crear un control capaz de gestionar varias vistas y alternar entre ellas a través de un pequeño segmento situado en la barra de navegación. Es muy útil a la hora de agrupar vistas similares.

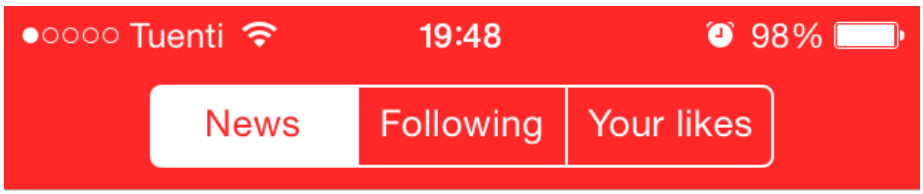


Imagen 6.9: Ejemplo del control "RMMultipleViewsController"

RMPickerViewController

Esta es una clase que nos permite crear un control fusionando dos elemento gráficos como son el UIPickerView y el UIActionSheet.

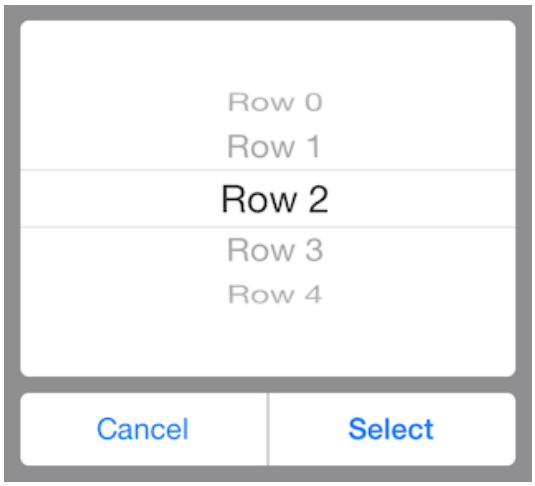


Imagen 6.10: Ejemplo del control "RMPickerViewController"

RMStepsController

Esta es una clase que nos permite crear una barra de navegación por pasos, donde cada paso es una nueva vista. Este control es ideal para guiar a los usuarios a través de un proceso.

Imagen 6.11: Ejemplo del control "RMStepsController"

GTScrollNavigationBar

Esta es una clase que nos permite crear una barra de navegación replegable en función de los movimientos de scroll. Es muy útil cuando se trabaja con pantallas pequeñas ya que permite un mejor aprovechamiento de la pantalla.

MBProgressHUD

Esta librería nos permite crear indicadores de actividad personalizados, es una alternativa a los indicadores de actividad que proporciona Apple.

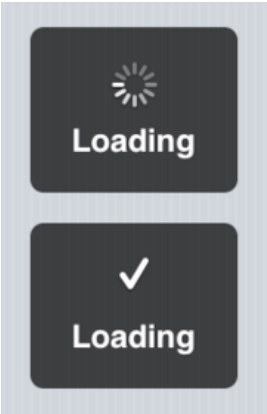


Imagen 6.12: Ejemplo del control "MBProgressHUD"

XCDYouTubeVideoPlayerViewController

Esta librería proporciona un reproductor de vídeos de YouTube. Es muy fácil de usar y hace que el proceso de visualización sea muy transparente al usuario, es decir, el usuario no notará que se esta visualizando un vídeo procedente de YouTube.

Facebook SDK para iOS

Este kit de desarrollo cuenta con todas la herramientas necesarias para interactuar con Facebook desde una aplicación iOS. Haciendo uso de sus métodos se puede hacer infinidad de cosas, desde publicar una foto en el muro hasta conocer toda su lista de amigos.

Youtube API

La API de datos de YouTube permite incorporar la funcionalidad de YouTube en tu propia aplicación. Puedes utilizar la API para obtener resultados de búsqueda y recuperar, insertar, actualizar y eliminar recursos como vídeos o listas de reproducción.

Parse SDK para iOS

Este kit proporciona todo lo necesario para conectarse e interactuar con los servicios que ofrece Parse. Además abstrae al programador de todo lo relacionado con las conexiones y permite centrarse en los datos.

6.2.3.- Conexión con el backend e interacción con el mismo.

Para poder acceder a nuestro backend debemos hacerlo mediante la API que nos proporciona Parse, para ello debemos incluir el framework de Parse a nuestro proyecto, así como otros framworks y librerías de los que este depende.

Cuando creamos una "App" en Parse, se nos proporciona un código de identificación y unos códigos de acceso. Estos códigos son los que nos permitirán conectarnos al backend.

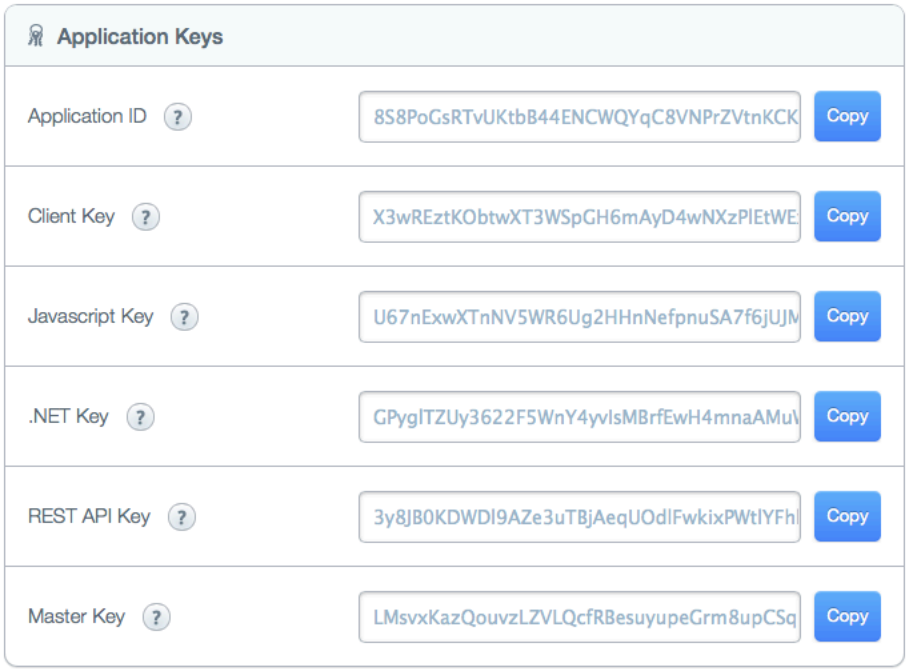


Imagen 6.13: Claves de acceso al backend

Para configurar el acceso debemos añadir la librería a nuestro AppDelegate e insertar la siguiente línea de código dentro de la función `application:didFinishLaunchingWithOptions:`. Con esta línea configuraremos el acceso al backend indicando cual es nuestro identificador y nuestra clave:

```
[Parse setApplicationId:@"8S8PoGsRTvUKtbB44ENCWQYqC8VNPrZVtnKCK94D"
              clientKey:@"X3wREztKObtwXT3WSpGH6mAyD4wNXzPlEtWExPQd"];
```

Una vez hecho esto ya podemos conectarnos a nuestro backend siempre que dispongamos de conexión.

A continuación vamos a mostrar unos segmentos de códigos extraídos de la aplicación para entender como funcionan las inserciones, consultas y demás:

Registro de un usuario

En el siguiente fragmento de código se muestra como se realiza la creación y registro

de un nuevo usuario en nuestro sistema. Lo primero que se hace es crear un objeto de tipo PFUser, luego, se cumplimentan sus propiedades a partir de los valores insertados por el usuario en los diferentes campos de texto del formulario y por último, se llama al método **signUpInBackgroundWithBlock:**, el cual creará el nuevo usuario en la base de datos.

```
PFUser *user  = [PFUser user]; //Se crea un objeto de tipo User
user.username = _nameTextField.text;
user.password = _passTextField.text;
user.email    = _emailTextField.text;

//Registro del usuario
[user signUpInBackgroundWithBlock:^(BOOL succeeded, NSError *error) {
    if (!error) {
         ...
    }else {
         ...
    }
}];
```

Login de un usuario

El login de un usuario se realiza a través del método **logInWithUsernameInBackground: password: block:**, este se encarga de establecer conexión con la base de datos y comprobar que existe el usuario especificado , en caso contrario devolverá un error donde se indique por qué no se ha podido realizar login.

```
[PFUser logInWithUsernameInBackground:_nameTextField.text
password:_passwordTextField.text block:^(PFUser *user, NSError *error)
{
      if (!error && user) {
         ...
      } else {
         ...
      }
}];
```

Consulta de los rallyes que forman el campeonato

A continuación podemos ver como se efectúa una consulta, lo primero de todo es especificar sobre que clase queremos hacer la consulta y luego se pueden especificar parámetros opcionales como la manera de ordenar los resultados etc. Una vez hecho esto solo nos queda enviar la petición, para ello se utiliza el método **findObjectsInBackgroundWithBlock:,** el cual devuelve un objeto de tipo NSArray que contendrá tantos objetos como rallys existan en la base de datos.

```
PFQuery *query = [PFQuery queryWithClassName:@"Rallye"];
[query orderByAscending:@"iniDate"];
[query findObjectsInBackgroundWithBlock:^(NSArray *objects, NSError
*error) {
  if (!error) {
    ...
  } else {
    ...
  }
}];
```

Guardando un nuevo vídeo en la base de datos

Una vez que hemos creado el vídeo que vamos a guardar en la base de datos lo único que hay que hacer es llamar al método **`saveInBackgroundWithBlock:`**, el cual se encargará de guardar el vídeo e informarnos si todo ha ido bien o ha habido algún error.

```
PFObject *vídeo = [PFObject objectWithClassName:@"Vídeo"];
[vídeo setObject:_vídeo.ID forKey:@"youtubeID"];
[vídeo setObject:_vídeo.nDay forKey:@"nDay"]
[vídeo setObject:_vídeo.title forKey:@"title"];
[vídeo setObject:@0 forKey:@"nLikes"];
vídeo[@"rallyId"] = [PFObject objectWithoutDataWithClassName:@"Rallye"
objectId:_vídeo.objectId]; //Puntero al rallye
vídeo[@"owner"] = [PFObject objectWithoutDataWithClassName:@"_User"
objectId: [PFUser currentUser].objectId ]; //Puntero al propietario

[vídeo saveInBackgroundWithBlock:^(BOOL succeeded, NSError *error) {
  if (!error) {
    ...
  } else {
    ...
  }

}];
```

Como vemos es muy sencillo interactuar con nuestro backend, todo se limita a usar los diferentes métodos que nos ofrece Parse.

En cuanto a la **seguridad** de las conexiones, estas se realizan a través de conexiones seguras **HTTPS** y las contraseñas se almacenan cifradas haciendo uso del algoritmo **Bcrypt**.

6.2.4.- Vistas e implementación.

En esta sección veremos como ha sido implementada la aplicación, partiremos de una visión global e iremos "desmenuzando" alguna de las vistas más importantes así como sus controladores.

Comenzaremos por la vista inicial de nuestra aplicación, es en esta vista donde el usuario puede elegir el método de inicio de sesión que prefiera. Para una mayor comodidad del usuario se han proporcionado tres posibles opciones de inicio de sesión, usando Facebook, usando Twitter o mediante el sistema tradicional de usuario y contraseña.

Para proporcionar acceso haciendo uso de las redes sociales se han usado las clases PFTwitterUtils y PFFacebookUtils, que proporciona Parse en su SDK. Lo primero de todo fue crear la aplicación VidWRC tanto en

Imagen 6.14: Vista inicial de la aplicación

Facebook como en Twitter, cuando creamos la aplicación se nos proporciona un identificador así como otros parámetro que necesitaremos para poder usar estos servicios desde nuestra aplicación.

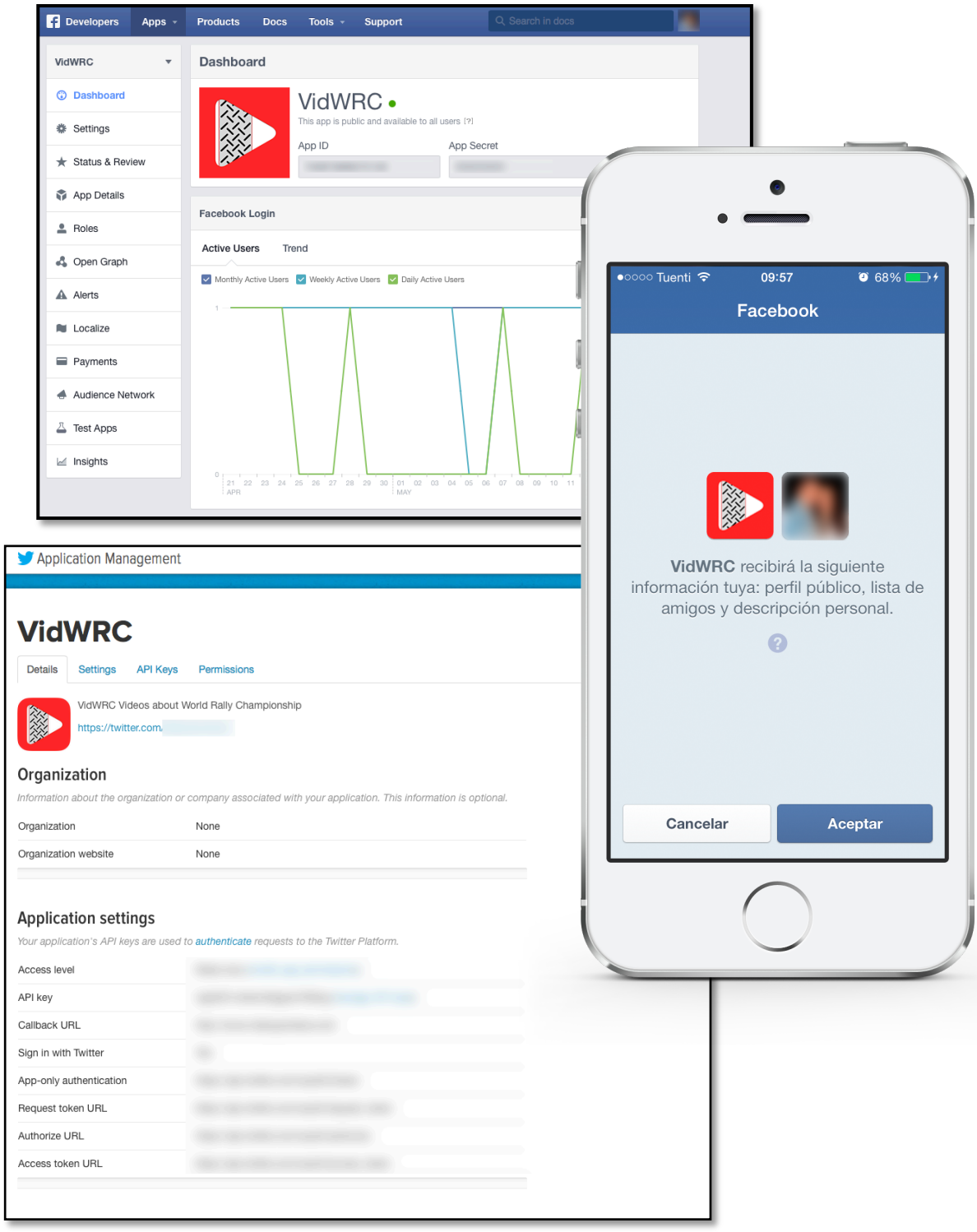


Imagen 6.15: Mosaico

Cuando dispongamos de todos los datos, solo hay que insertarlos en nuestra aplicación de la forma especificada por la documentación de Parse. Tras configurar varios parámetros, ya podremos hacer uso de los servicios, a continuación se muestra el fragmento de código que se utiliza en la aplicación para conectarse haciendo uso de Facebook:

```
- (void)connectWithFacebookButton{
    // Establecemos todos los permisos que necesitamos de la cuenta de Facebook del usuario
   NSArray *permissionsArray = @[@"user_about_me"];

   if (![[UIApplication sharedApplication].delegate performSelector:@selector(isParseReachable)]) {
     ...
   }else{
      [PFFacebookUtils logInWithPermissions:permissionsArray block:^(PFUser *user, NSError *error) {
        if (!user) {
              if (error) {
                NSLog(@"Se ha producido un error: %@", error);
                 UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Log In Error" message:[error
                  description] delegate:nil cancelButtonTitle:nil otherButtonTitles:@"Accept", nil];
                 [alert show];
              } else {
                 NSLog(@"El usuario ha cancelado el inicio de sesión.");
                 UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Log In Error" message:@"The user
                 cancelled the Facebook login." delegate:nil cancelButtonTitle:nil
                 otherButtonTitles:@"Accept", nil];
                 [alert show];
               }
        } else if (user.isNew) {
              NSLog(@"Usuario registrado y logeado con Facebook, operación finalizada con éxito");
              [self updateProfileWithFacebook]; //Actualizar perfil con la información de Facebook.
             //Cerrar modalView para acceder a la vista principal de la app.
              [self dismissViewControllerAnimated:YES completion:nil];
        } else {
              NSLog(@"Usuario logeado con Facebook, operación finalizada con éxito");
              [self updateProfileWithFacebook]; //Actualizar perfil con la información de Facebook.
              //Cerrar modalView para acceder a la vista principal de la app.
              [self dismissViewControllerAnimated:YES completion:nil];
        }
      }];
   }
}
```

Una vez que hayamos iniciado sesión, podremos ver las diferentes vistas que componen nuestra aplicación. Para explicar las diferentes partes de las que consta esta aplicación, se partirá de la explicación de los controladores que manejan la navegación y se continuará con las vistas que manejan estos controladores.

Comenzando con los elementos de navegación, mostraremos en primer lugar la estructura básica de la aplicación, la cual está formada por un UITabBarController y cinco UINavigationController:

Imagen 6.16: Estructura de la aplicación

El **UITabBarController** permite la navegación entre controladores de vista a modo de pestañas. En una barra de navegación situada en la parte inferior de la pantalla, se añaden los iconos correspondientes a cada sección en las que se dividirá la aplicación. Cada sección tiene sus controladores de vista precargados. Cuando el usuario pulsa sobre alguno de los iconos de la barra de navegación, se muestra de forma inmediata, el controlador asociado con la vista que lo identifica.

Como se puede observar en la siguiente figura, nuestra aplicación cuenta con 5 secciones bien diferenciadas:

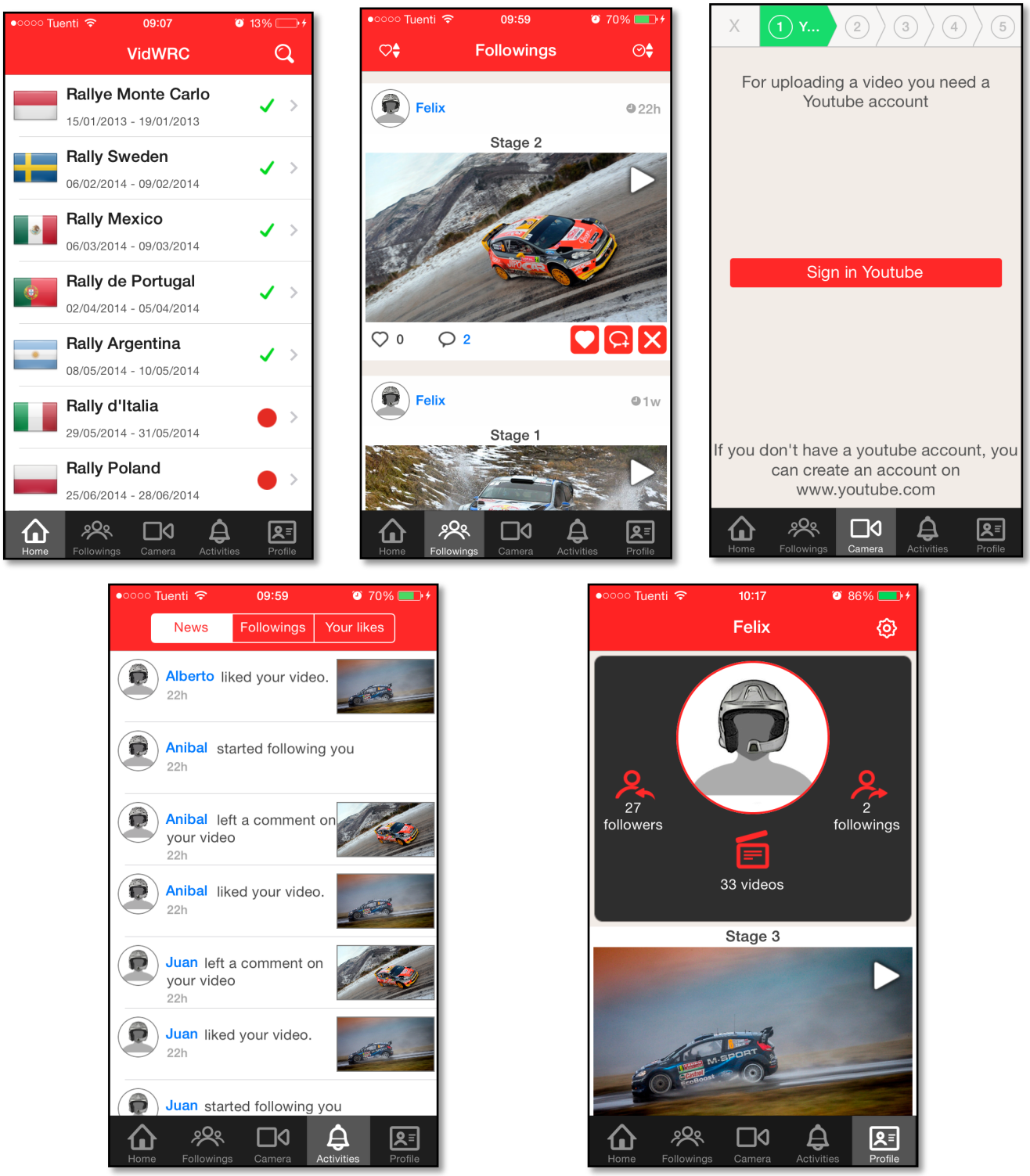


Imagen 6.17: Secciones de la aplicación

- En la primera sección, que será la vista inicial tras el inicio de sesión, se puede ver el calendario completo de rallyes que conforman el campeonato. Esta es la sección denominada “Home”.

- En la segunda sección, se recogen todos los vídeos que han sido publicados por los "siguientes" del usuario actual así como sus propios vídeo. A esta sección se le denomina "Followings".
- En la tercera sección es donde se llevará a cabo el proceso de subida y publicación de un nuevo vídeo por parte del usuario y se le denomina "Camera".
- En la cuarta sección es donde se recogen todas las actividades que se generan en el entorno del usuario, es decir, las actividades relacionadas con sus vídeos ("me gusta", comentarios), las actividades relacionas con su perfil (personas que comienzan a seguirle), las actividades de sus "siguientes" (vídeos que les gustan) y las actividades del propio usuarios. A esta sección se le denomina "Activities".
- Por último, la quinta sección de la aplicación, mostrará el perfil del usuario, donde se podrá ver el número de vídeos publicados, el número de seguidores y el número siguientes que tienes, así como la colección de vídeos publicados por el usuario. Esta es la sección denominada "Profile".

El **UINavigationController** permite la transición entre vistas (o controladores de vista) añadiendo de forma secuencial las nuevas vistas en una pila. Para hacer uso de este controlador, típicamente, se inserta una barra de navegación en la parte superior de la pantalla. Cuando se quiere visualizar una nueva vista, se coloca ésta sobre la vista actual con una pequeña animación de barrido, y aparece, en la parte izquierda de la barra de navegación, un botón con el nombre de la página anterior que acaba de ser ocultada. Pulsando sobre este botón, la vista actual desaparecerá, mostrando de nuevo la vista previa. Esta operación de paginación puede repetirse tantas veces como se desee, ocultando y mostrando vistas de la aplicación como si de una pila se tratara.

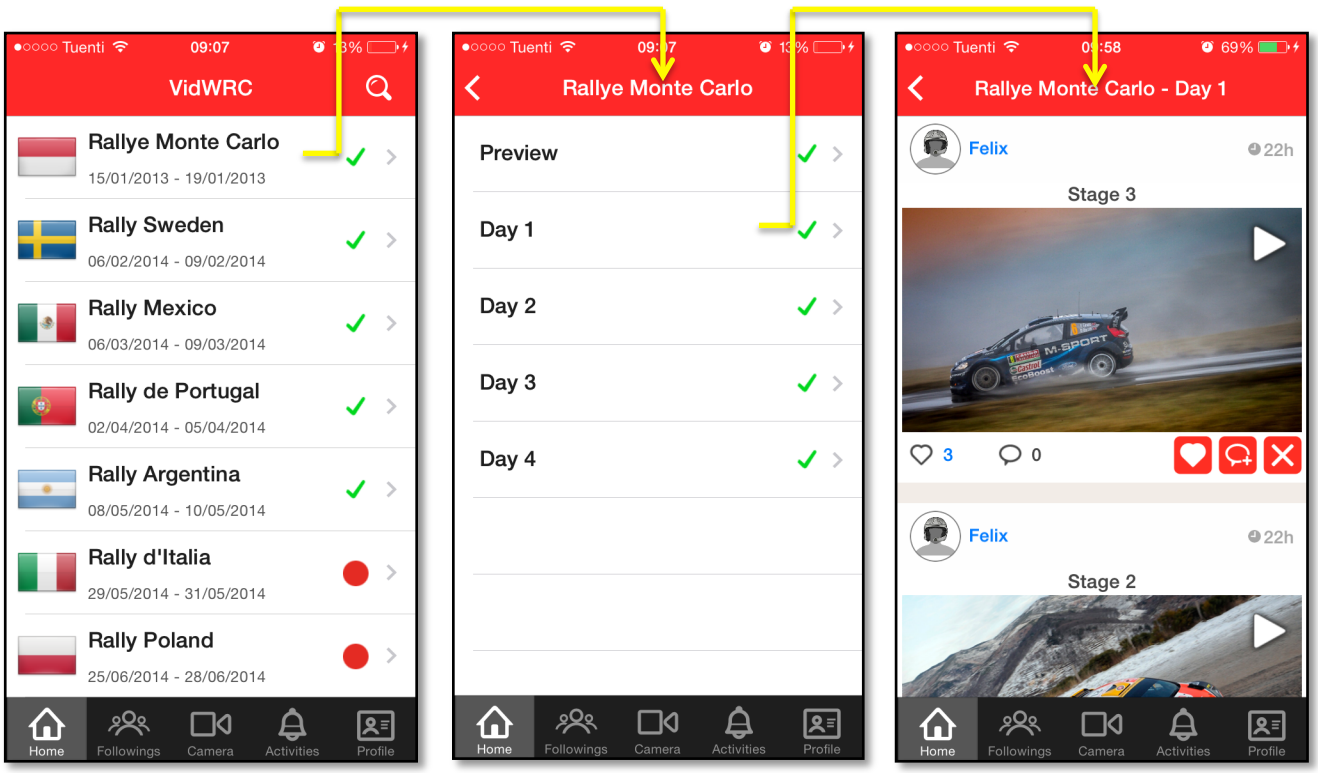


Imagen 6.18: Pila de vistas, UINavigationController

Partiendo de esta base, se ha continuado con la creación de vistas y sus respectivos controladores, los cuales complementarán esta estructura de navegación para dar forma a la aplicación final.

Comenzaremos por la **primera sección** de la aplicación, en este apartado es donde se podrán ver todos los vídeos existentes en el sistema, estos vídeos se presentarán ordenados en función del rallye al que pertenezcan. Sin un rallye todavía no se ha disputado, se presupone que no habrán vídeos relacionados con él, por lo que el acceso a este no estará disponible. Por otra parte, los rallyes normalmente se celebran a lo largo de varios días, por lo que se ha decidido dividir cada rallye en diferentes secciones que representan los días. Una vez el usuario haya seleccionado el rallye y el día ya podrá ver los vídeos disponibles (Imagen 6.18), a esta vista se le llamará vista detallada de un rallye.

A continuación se muestra la consulta a la base de datos donde se solicitan los vídeos para un rallye y un día concreto:

```
-(PFQuery*)queryForTable{
    //Clase a la que se va a realizar la consulta.
    PFQuery *videosQuery = [PFQuery queryWithClassName: @"Vídeo"];
    //Orden de los resultados.
    [videosQuery orderByDescending:@"createdAt"];
    //Vídeos que pertenezcan a al rallye: ...
    [videosQuery whereKey: @"rallyId" equalTo: [PFObject
    objectWithoutDataWithClassName:@"Rallye"
objectId:_rallye.objectId]];
    //Vídeos del día: ...
    [videosQuery whereKey:@"nDay" equalTo:_Day];
    //Adjuntar también la información a la que apunta el puntero owner.
    [videosQuery includeKey:@"owner"];

    return videosQuery;
}
```

Una vez que se obtienen los datos, se rellenan los campos de las celdas con esa información. Además, durante este proceso rellenado de celdas, se ejecutan nuevas consultas a la base de datos para cumplimentar información relacionada con cada vídeo(número de “me gusta”, comentarios y si el vídeo ha sido reportado como inadecuado). A continuación se muestra la celda prototipo que se utiliza para mostrar un vídeo y su información:

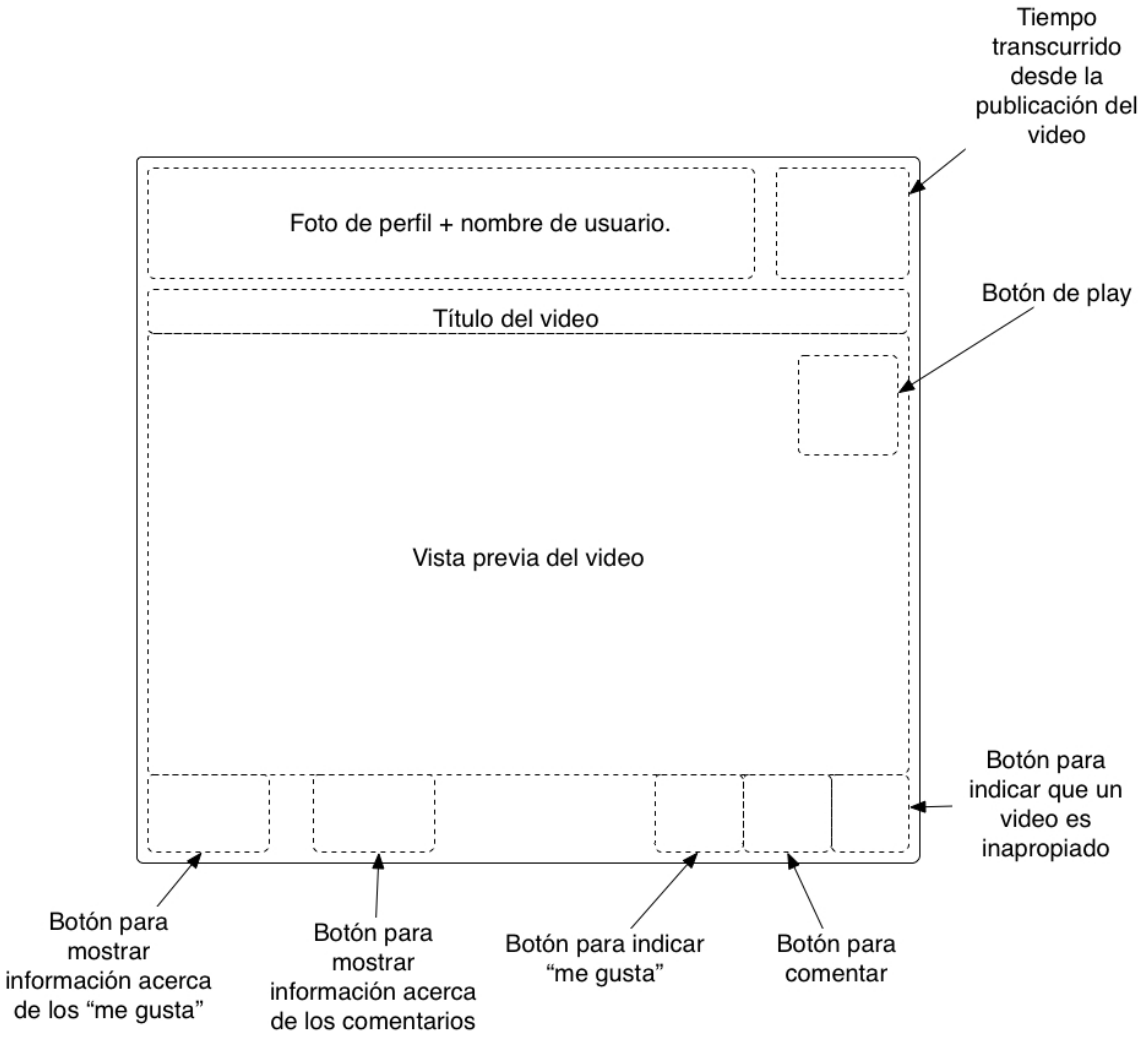


Figura 6.6 : Prototipo de celda para vídeos

Tras haber visto las diferentes pantallas de la primera sección, pasamos a ver la **segunda sección.** En esta segunda sección se recogen todos los vídeos que han sido publicados por los "siguientes" del usuario actual, así como sus propios vídeos.

Esta sección es exactamente igual que la vista detallada de un rallye, lo único que cambian son los datos mostrados, es decir la consulta a la base de datos cambia, a continuación se muestra esta consulta, que es una especie de JOIN si lo comparamos con una base de datos relacional como puede ser una base de datos MySQL:

```
-(PFQuery*)queryForTable{
     //Actividad --> Se obtienen los objetos de la clase "actividad" donde el
    usuario actual sigue a otros usuarios
    PFQuery *followingActivitiesQuery = [PFQuery queryWithClassName:@"Activity"];
    [followingActivitiesQuery whereKey:@"type" equalTo:@4];
    [followingActivitiesQuery whereKey:@"fromUser" equalTo:[PFUser currentUser]];
    followingActivitiesQuery.cachePolicy = kPFCachePolicyNetworkOnly;

    //Vídeo --> Se obtienen los objetos de la clase "vídeo" de las persona a las
    que sigue el usuario actual
    PFQuery *videosFromFollowedUsersQuery = [PFQuery queryWithClassName:
```

```
                                                    self.parseClassName];
    [videosFromFollowedUsersQuery whereKey:@"owner" matchesKey:@"toUser" inQuery:
    followingActivitiesQuery];

    //Vídeo --> Se obtienen los objetos "vídeo" del usuario actual.
    PFQuery *videosFromCurrentUserQuery = [PFQuery queryWithClassName:
    self.parseClassName];
    [videosFromCurrentUserQuery whereKey:@"owner" equalTo:[PFUser currentUser]];

    //Query compuesta
    PFQuery *videosQuery = [PFQuery orQueryWithSubqueries:[NSArray
    arrayWithObjects:videosFromFollowedUsersQuery,videosFromCurrentUserQuery,nil]];
    [videosQuery includeKey:@"owner"];

    return videosQuery;
}
```

Ahora pasamos a la **tercera sección** de nuestra aplicación, una de las más importantes, ya que es donde el usuario publica su vídeo al sistema. Como ya se comentó anteriormente, para almacenar los vídeos vamos a usar YouTube, esto implica que cada usuario deberá tener su propia cuenta para poder trabajar con este servicio. Por lo tanto, el primer paso en este proceso es permitir el inicio de sesión en YouTube.

Para comunicarnos con YouTube utilizamos la API que Google tiene disponible para los desarrolladores de iOS, con ella vamos a permitir al usuario acceder a su cuenta y de esta forma que pueda subir los vídeos. YouTube, al igual que Facebook y otros muchos servicios, utiliza un sistema centralizado de autenticación/autorización a través de terceros denominado OAuth2, con este sistema el usuario no revela sus credenciales a un tercero para que trabaje con su sesión, sino que este autoriza a un tercero para que realice nada más que unas actividades concretas. Para poder utilizar este sistema de autenticación/autorización debemos habilitarlo para nuestra aplicación. Esto se hace desde una cuenta de desarrollador de Google donde debemos registrar nuestra aplicación y configurar varios parámetros relacionados.

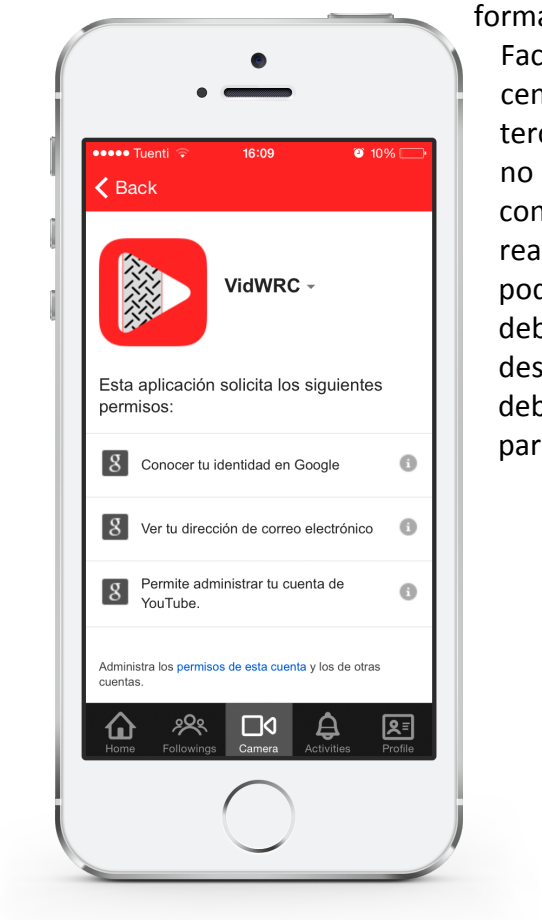


Imagen 6.19: Vista donde se solicitan los permisos para administrar cuenta de Youtube

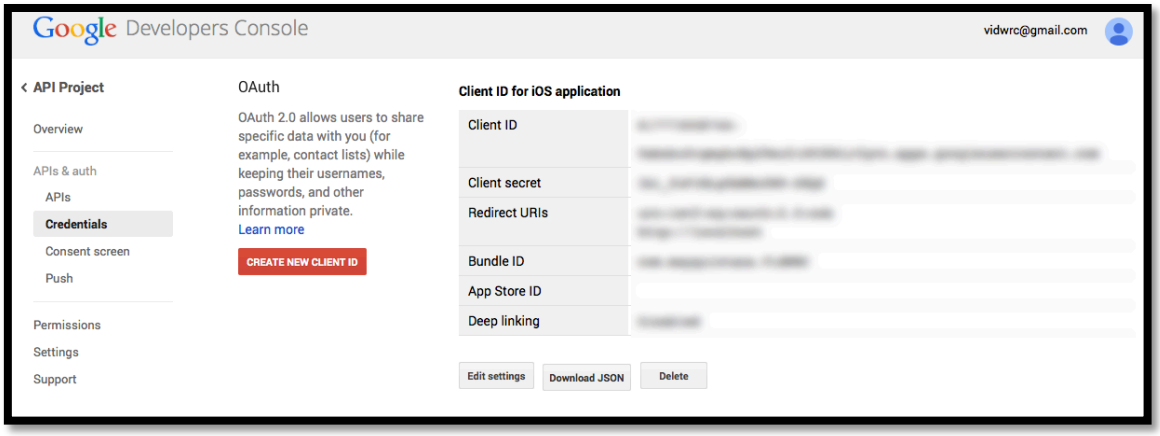


Imagen 6.20: Configuración de las claves de acceso para YouTube

Una vez que hayamos habilitado y configurado el servicio en Google, ya podemos hacer uso del sistema OAuth2 para conectarnos e iniciar una sesión en YouTube. La gestión de esta sesión se lleva a cabo haciendo uso de la API de YouTube. Para mayor comodidad del usuario, se almacenan los tokens de acceso a YouTube de tal manera que no necesite iniciar sesión cada vez que quiera subir un nuevo vídeo. A continuación podemos ver una captura del proceso de autenticación/ autorización en YouTube por parte del usuario:

Con la sesión ya iniciada, el siguiente paso será seleccionar el vídeo a subir, para ello se ofrecen dos opciones, usar un vídeo ya existente o utilizar la cámara para crear un nuevo vídeo. Para proporcionar acceso tanto a la cámara como a la biblioteca, usamos la clase **UIImagePickerController**, esta clase facilita el acceso a estos recursos y permite personalizar íntegramente la interfaz. Además es parametrizable y permite configurar diversos parámetros del vídeo como su duración, calidad, etc. Cuando se selecciona un vídeo de la biblioteca, esta clase se encarga automáticamente de comprimir el vídeo, pero cuando se captura un vídeo desde la cámara, este suele tener un gran tamaño debido a la calidad de grabación, lo cual supone un problema a la hora de subir el vídeo. Para solventar este problema se hace uso del framework **AVFoundation** en cual proporciona una gran cantidad de métodos para trabajar con contenido multimedia. Haciendo uso de algunas funciones de este framework, se reduce el bitrate del vídeo y podemos pasar de un vídeo de 46Mb a un vídeo de 14MB , el cual es mucho más manejable a la hora de subirlo a YouTube.

Después de seleccionar el vídeo, el siguiente paso será la elección del thumbnail o vista representativa del vídeo. Para ello se hace uso de un slider, que se deberá deslizar para avanzar a lo largo del vídeo y seleccionar esta imagen.

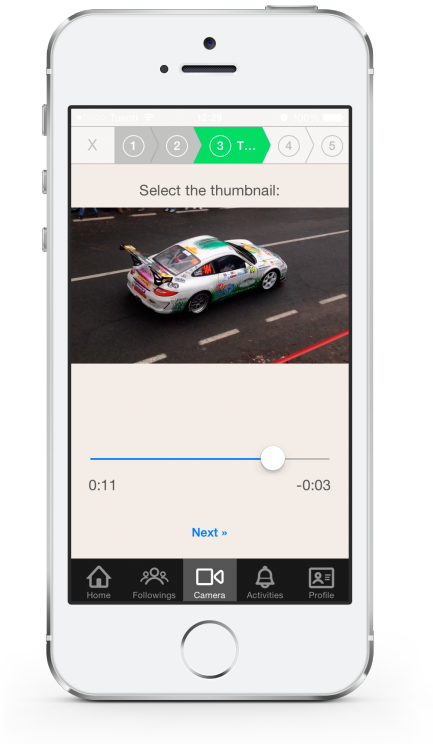


Imagen 6.21: Vista de selección del thumbnail

A continuación, el cuarto paso consiste en clasificar el vídeo, es aquí donde se establece el título, el rallye al que pertenece y el día; además, en esta vista también se le permite al usuario elegir si quiere que su vídeo sea compartido en Facebook (a través de una publicación en su muro) o no.

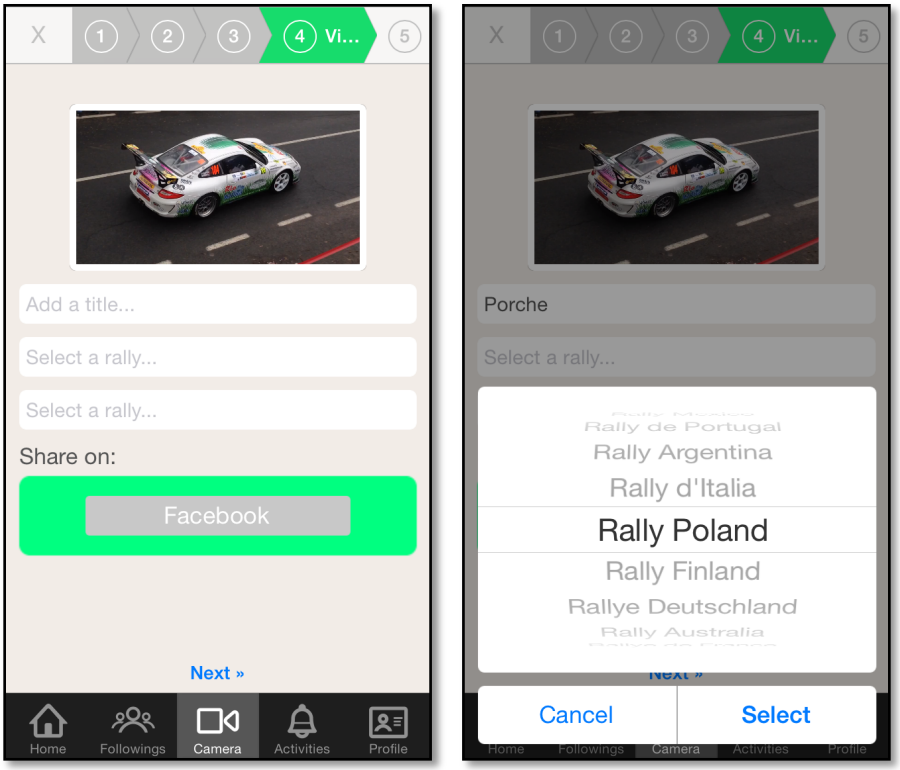


Imagen 6.22: Vistas para adjuntar información al vídeo

Por último, ya en el quinto paso de este proceso se deberá seleccionar si el vídeo que vamos a subir es público, es decir, que todos los usuario de YouTube puedan verlo, o si el vídeo es "oculto", es decir que nadie lo puede ver salvo que conozca su dirección. Una vez seleccionada la opción que el usuario considere adecuada, lo que queda es subir y publicar el vídeo. La publicación del vídeo la podemos dividir en tres etapas, la subida del vídeo a YouTube, la creación de un nuevo objeto en la base de datos, y la publicación de información.

- La subida del vídeo a YouTube es el primer paso de este proceso. Para ello lo primero que se hace es "recolectar" información acerca del vídeo como su título, su descripción, sus tags de búsqueda, etc. A continuación, ya con toda la información disponible, se comprueban otros parámetros y se comienza con la subida del vídeo. Una vez que el proceso ha finalizado se obtiene la información del vídeo en YouTube, la cual necesitaremos en el siguiente paso.

- El siguiente paso después de subir el vídeo, es crear un nuevo objeto en nuestra base de datos, este objeto deberá relacionarnos de alguna manera con ese vídeo subido a YouTube. Esa forma de relacionarnos es el ID del vídeo, este ID es el que nos proporciona YouTube al finalizar a subida del vídeo. Por lo tanto almacenando la información relacionada con el vídeo y su ID tenemos un objeto que representa el vídeo y través del cual podremos acceder en cualquier momento al vídeo en si.

- Una vez que se encuentra toda la información almacenada en los diferentes servidores que conforman nuestro backend, ya podemos notificar al resto de usuarios que existe un nuevo vídeo, para ello se envía una notificación al resto de usuario indicando que hay un nuevo vídeo disponible. Además es en este punto donde se publica el vídeo en el muro de Facebook del usuario, si así lo había decidido anteriormente.

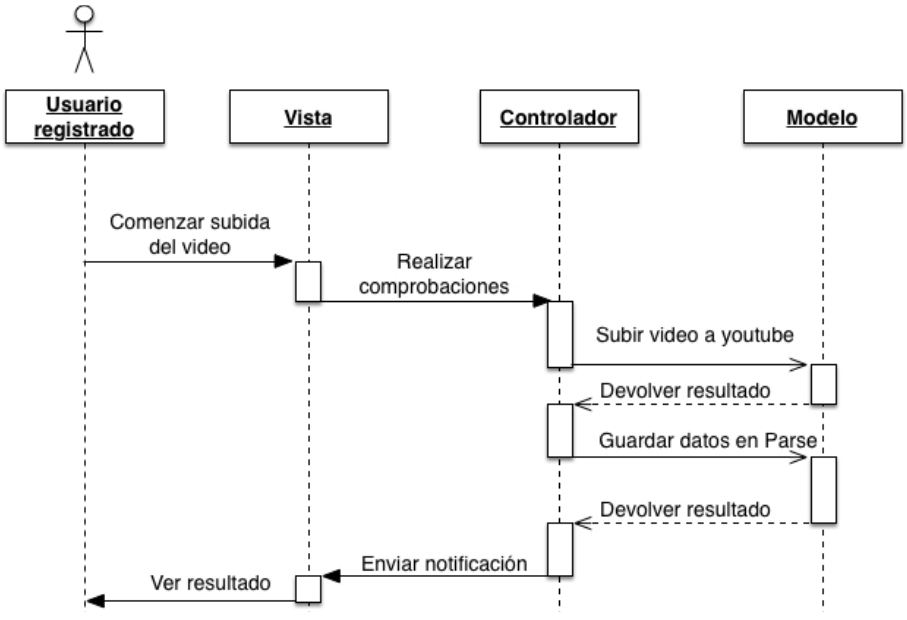


Figura 6.7: Diagrama de secuencia (almacenando vídeos + datos)

Ahora, pasamos a ver la sección denominada “Activities”, esta sección de subdivide en tres apartados distintos:

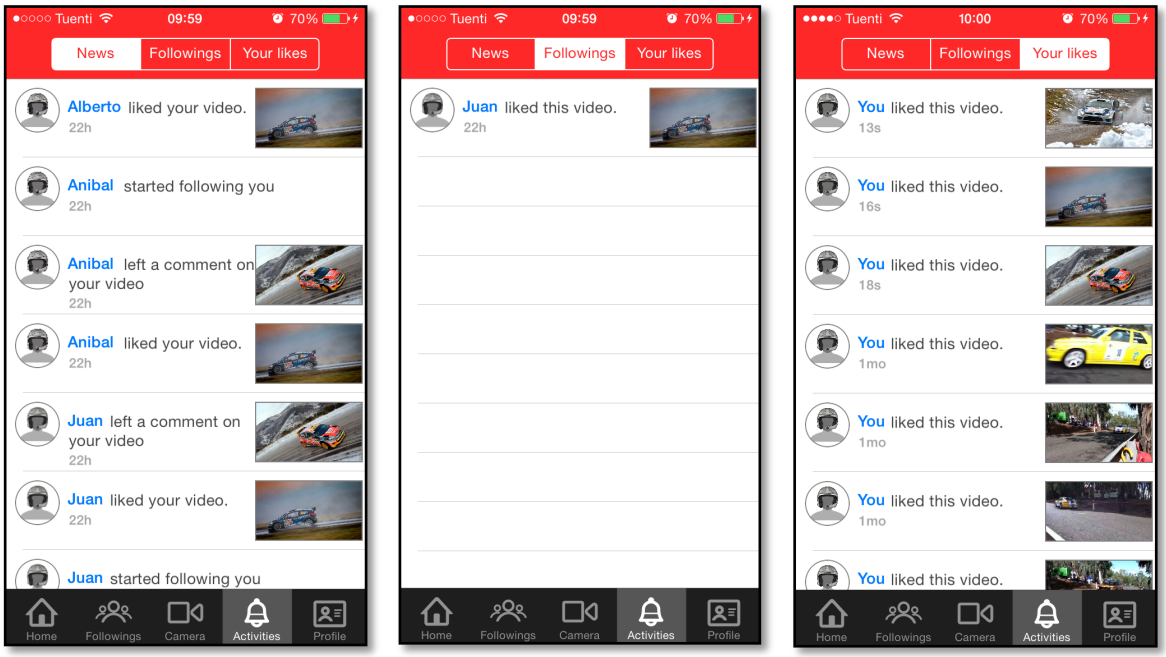


Imagen 6.23: Vistas de la sección Activities

- En el primer apartado, denominado “News”, es donde se pueden ver las actividades relacionadas con los vídeos del usuario, así como con su perfil. Existen tres actividades posibles:
  - Otro usuario comienza a seguir al usuario actual.
  - Alguien comenta un vídeo del usuario.
  - A alguien le gusta un vídeo del usuario.
- En el segundo apartado, denominado “Followings”, es donde se mostrarán los vídeos que les gusta a nuestros “siguientes”.
- Por último, en el apartado denominado “Your likes ”, se pueden ver todos los vídeos que han gustado al usuario actual.

El contenido de cada una de estas vistas podría ser muy extenso tras cierto tiempo de uso por lo que se muestran de forma paginada, es decir no se trae toda la información de una vez sino que se divide en fragmento y se van recuperando a medida que el usuario la va demandando. Esta técnica evita un consumo excesivo de datos así como la saturación de la memoria del sistema. Además de en esta sección, esta técnica también se usa en prácticamente todos los apartados de la aplicación donde la cantidad de datos a mostrar sea muy grande.

Como última sección de la aplicación tenemos la denominada “Profile”, en esta sección se mostrará información relacionada con el usuario y se le permitirá gestionar sus vídeos.

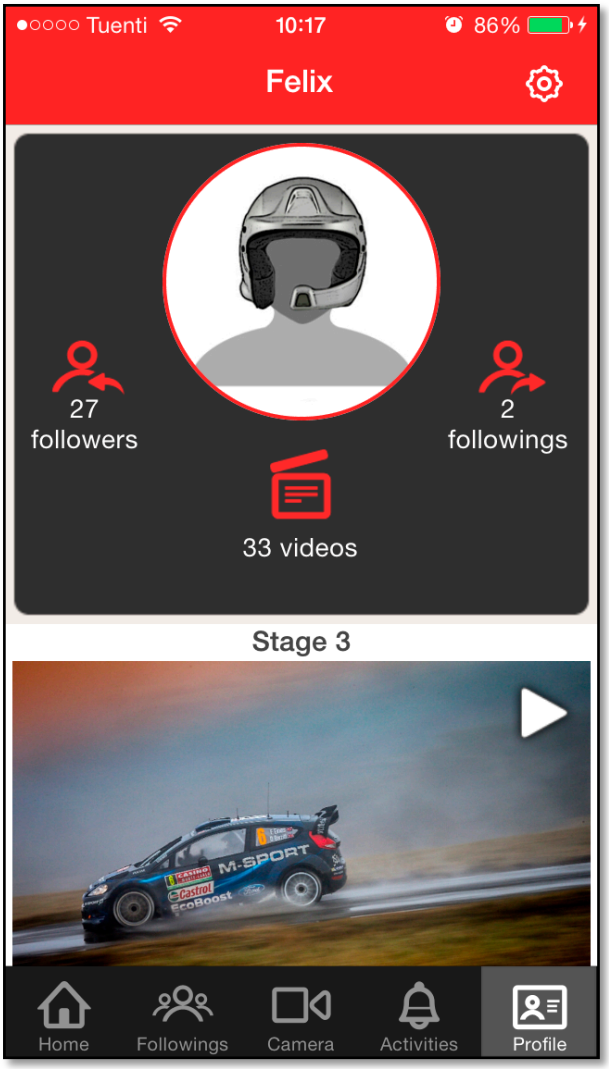


Imagen 6.24: Vista del perfil

Dentro de la sección “Profile” el usuario puede acceder a los ajustes, desde esta nueva vista, el usuario puede gestionar aspectos relacionados con su cuenta y modificar otros parámetros relacionados con la aplicación como pueden ser las notificaciones. Esta vista será diferente para unos u otros usuarios, todo depende de la forma en la que hallan iniciado sesión. A los usuario que usen las redes sociales para iniciar sesión solo se les permitirá cambiar su foto y eliminar su cuenta, en cambio, a los usuarios que inicien sesión haciendo uso de un nombre de usuario y una contraseña, podrán modificar también su dirección de correo y cambiar su contraseña.

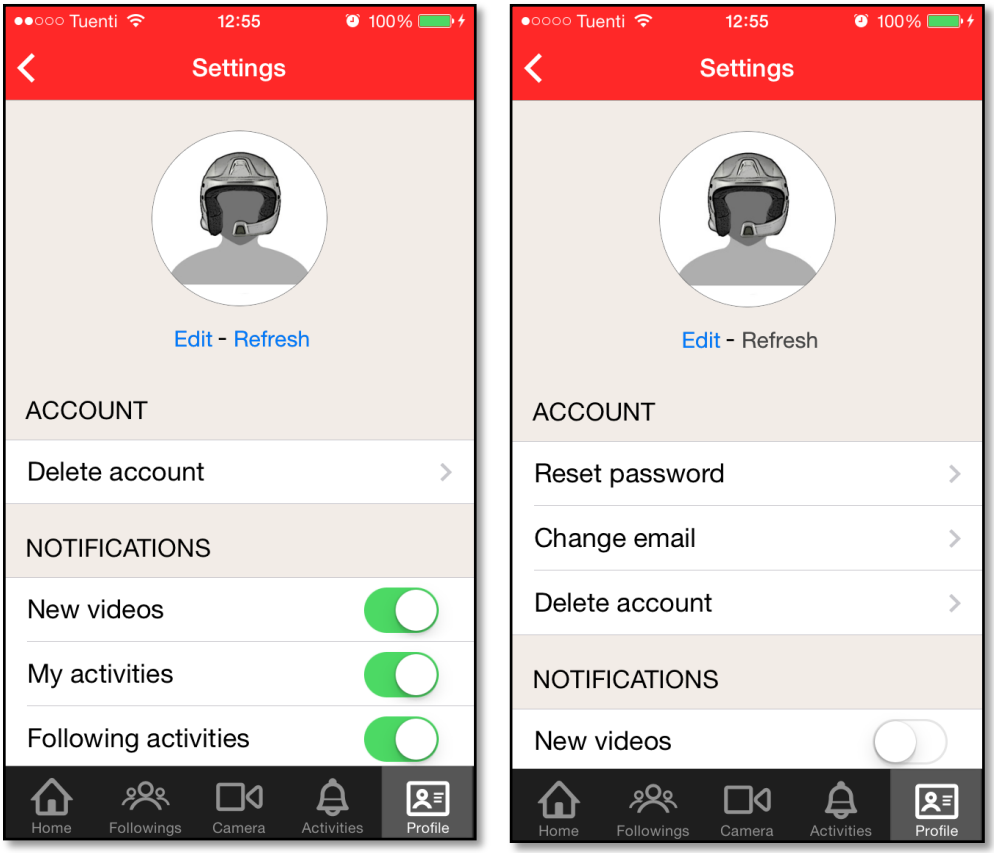


Imagen 6.25: Vistas de los ajustes

En lo referente a las notificaciones, desde esta vista podemos decidir que tipo de notificaciones recibir y cuales no. Este tema se tratará en mayor profundidad en el siguiente apartado de la memoria (6.2.5).

Vistas de uso común:
A continuación se expondrán una serie de vistas a las que se pueden acceder desde casi cualquier lugar de la aplicación:

Vista para la visualización de comentarios:

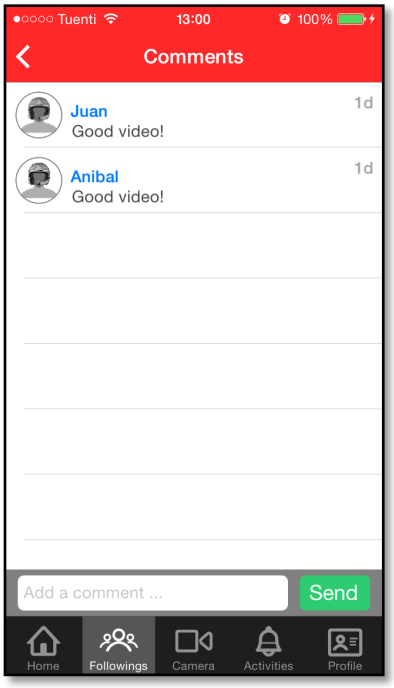


Imagen 6.26: Vista de comentarios

Esta es una vista estándar y se utiliza para ver los comentarios de cualquier vídeo.

Vista para listar usuarios (“me gusta”, followings, followers):

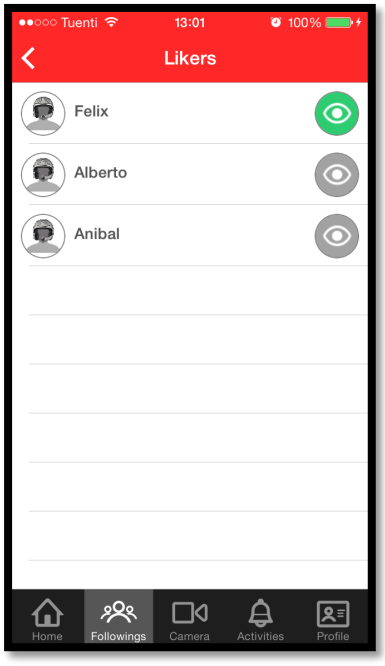


Imagen 6.27: Vista de usuarios

Al igual que la vista anterior esta es una vista estándar y se utiliza para mostrar listas de personas. Este tipo de vistas se utiliza cuando se quiere ver, por ejemplo, los usuarios que le han dado a “like” en algún vídeo, o cuando queremos ver quienes son los seguidores de algún usuario. Como se puede apreciar en la imagen superior , aparece un icono con un ojo a la derecha de cada usuario, este icono si es de color verde indica que se está siguiendo a ese usuario y en caso de ser de color gris indica que no se está siguiendo. Si se pulsa sobre este icono se puede comenzar a seguir a una persona o se puede dejar de seguir e esa persona.

Vista detallada de un vídeo:

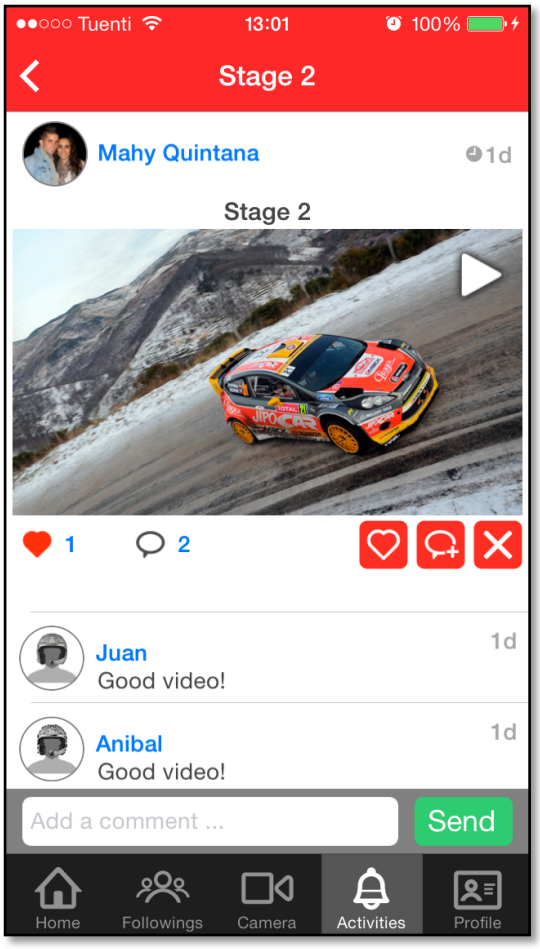


Imagen 6.28: Vista detallada de un vídeo

Esta vista solo es accesible desde la sección denominada “Activities” y se usa para mostrar un vídeo junto con sus comentarios. Además desde ella también se permite insertar nuevos comentarios e indicar que nos gusta ese vídeo.

Interfaz de búsqueda:

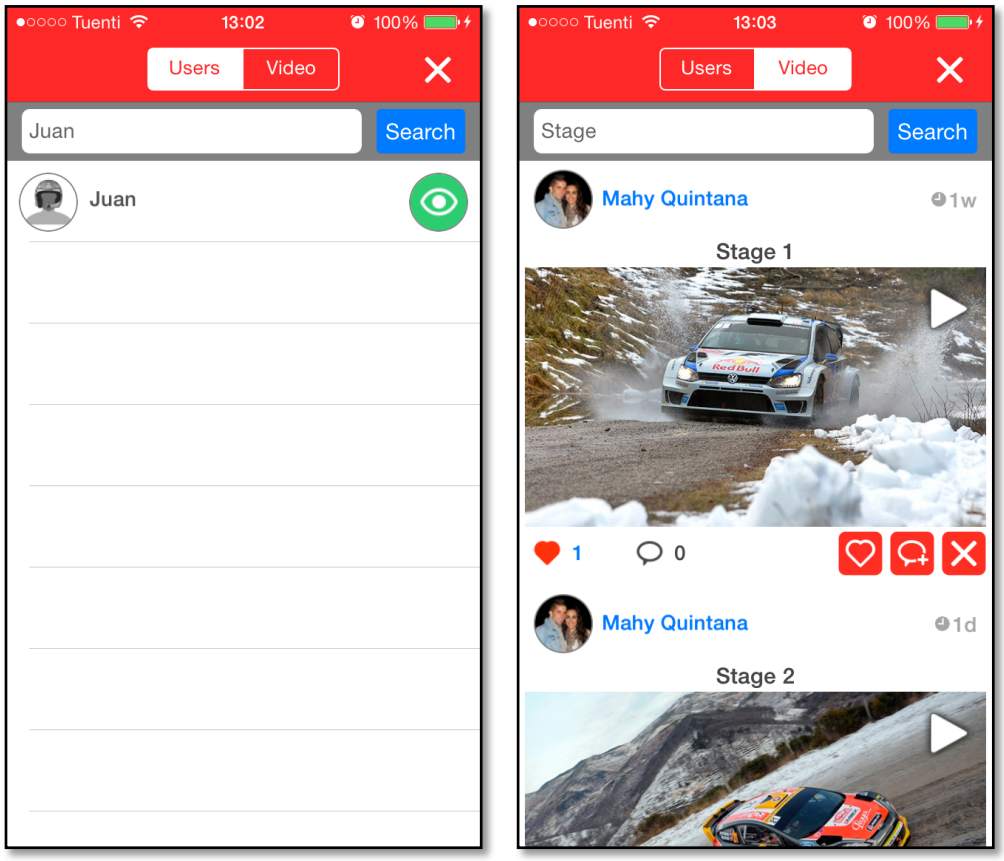


Imagen 6.29: Interfaz de búsqueda

Esta interfaz consta de dos vistas, una para la búsqueda de usuarios y otra para la búsqueda de vídeos.

¿Como se han implementados las actividades?
Una actividad es una objeto de la clase Activity el cual relaciona dos usuarios, esta relación puede ser de diferentes tipos y puede, o no, disponer de un contenido. A continuación se muestra una descripción detallada de lo que es cada campo de un objeto Activity y su significado.

Un objeto de la clase Activity cuenda con 5 campos relevantes:

- <pointer>fromUser: este campo es un puntero hacia un objeto de tipo User, en este caso, este campo apuntaría hacia el usuario que hace la acción.

- <pointer>toUser: este campo es exactamente igual que el anterior salvo que en vez de apuntar hacia quien hace la acción, apunta hacia quien va dirigida la acción.

- <number>type: este campo indica el tipo de acción y se codificación es la siguiente:
    - 1: like
    - 2: comment
    - 3: inappropriate vídeo
    - 4: follow

- <string>content: este campo solo es útil cuando la actividad es de tipo 2, es decir cuando se trata de un comentario. Es aquí donde se almacena el contenido de ese comentario.

- <date>createdAt: por último este campo se utiliza para guardar la fecha de creación del objeto, la cual servirá para mostrar en el tiempo cuando se produjo la actividad.

Esta ha sido la forma de implementar todas las actividades de nuestro sistema.

Orientación de la interfaz
Esta aplicación ha sido diseñada para que pueda ser usada en diferentes orientaciones, permitiendo al usuario trabajar con ella de forma vertical como horizontal. El proceso para crear una interfaz requiere bastante trabajo.

Existen dos métodos para ajustar el contenido de una vista a la orientación del dispositivo:

- Mediante código: cuando el dispositivo detecta un cambio de orientación se ejecutan una serie de métodos donde se debe cambiar la posición de los objetos para que se adapten a la nueva orientación.
- Mediante restricciones (constraints): esta es la manera más fácil ya que se hace de forma visual a través del interface builder.

A continuación se muestra como hacer que un botón permanezca en el centro de la pantalla independientemente de la orientación de la interfaz.

Mediante código:

```
-(void)viewWillLayoutSubviews{
  self.youtubeLoginButton.frame = CGRectMake(
                                  (self.view.frame.width / 2.0)-(280.0/2.0),
                                  (self.view.frame.height/ 2.0)-(30.0/2.0),
                                   280.0,
                                   30.0);
}
```

Mediante restricciones:

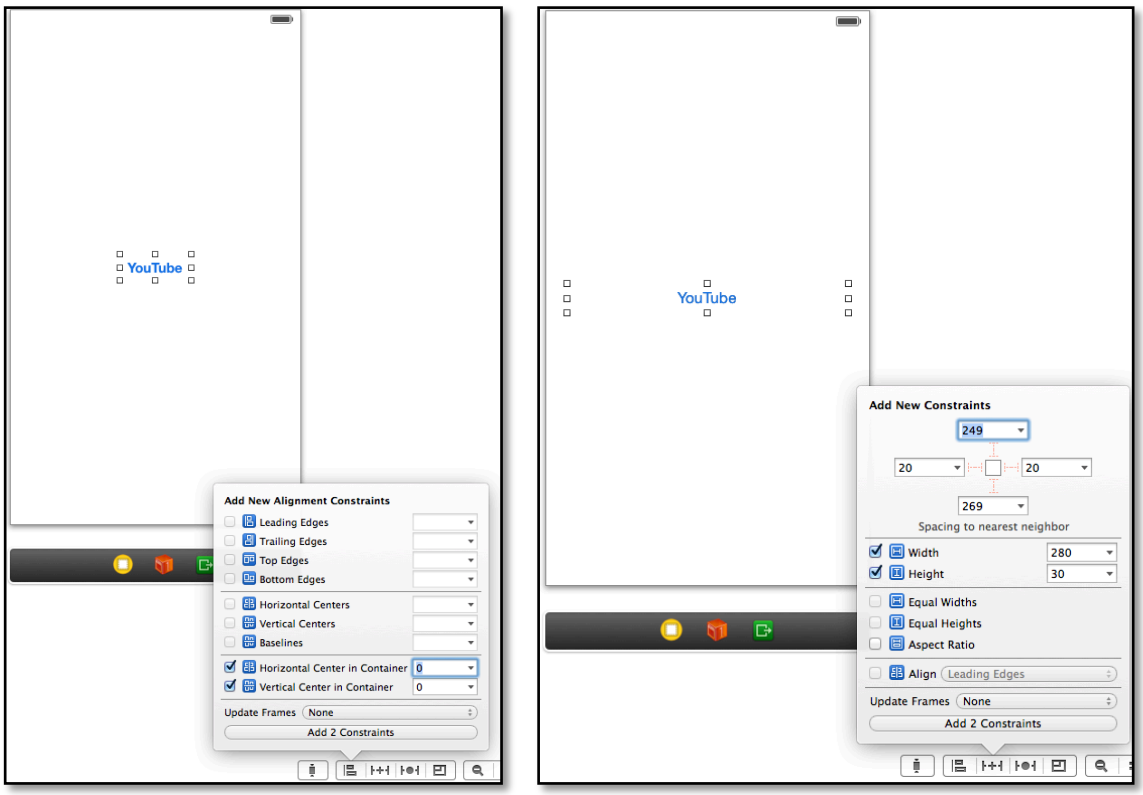


Imagen 6.30: Interface builder

Resultado:

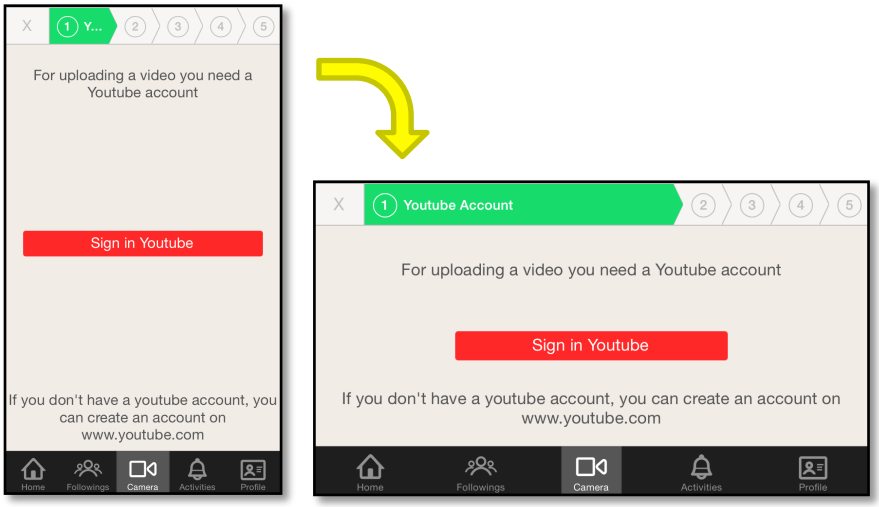


Imagen 6.31: Cambio de orientación

A primera vista parece más fácil y rápido hacer uso de las restricciones, pero si se quieren hacer cosas un poco más complejas hay que recurrir siempre al uso de código. Un ejemplo claro es el siguiente:

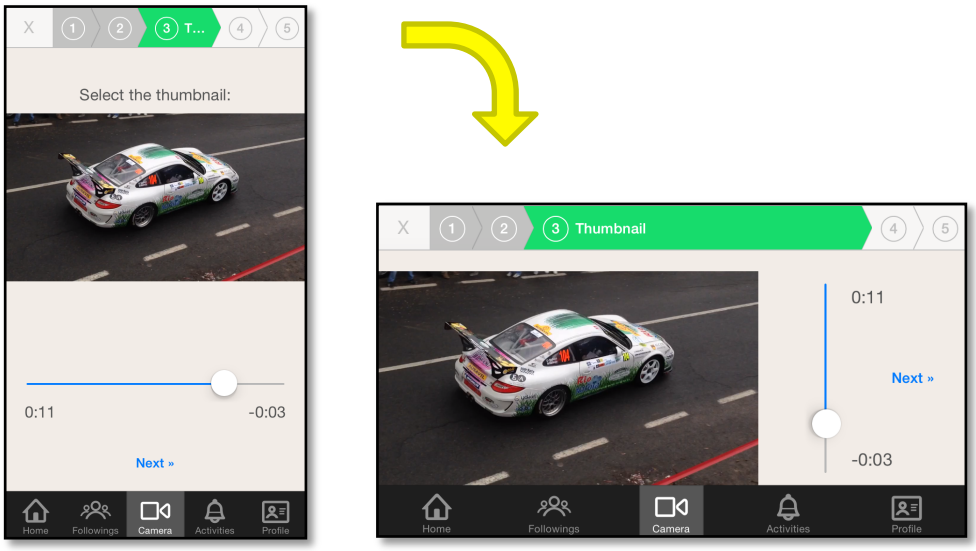


Imagen 6.32: Cambio de orientación

```
-(void)viewWillLayoutSubviews{

    if (UIInterfaceOrientationIsPortrait(self.interfaceOrientation)){

        self.topLabel.hidden = NO;
        self.slider.transform = CGAffineTransformMakeRotation(0);

        [self.topLabel setFrame:CGRectMake(self.view.frame.size.width / 2
                                                                  - 100.0,
                                                  topContainerView + 10.0,
                                                                    200.0,
                                                                   40.0)];

        [self.thumbnailImage setFrame:CGRectMake(thumbnailImageX,
                                                 thumbnailImageY,
                                            thumbnailImageWidth,
                                            thumbnailImageHeight)];

        [self.slider setFrame:CGRectMake(sliderX, sliderY, sliderWidth,
                                         sliderHeight)];

        [self.labelTime1 setFrame:CGRectMake(labelTime1X, labelTime1Y,
                                        labelTimeWidth, labelTimeHeigth)];

        [self.labelTime2 setFrame:CGRectMake(labelTime2X, labelTime2Y,
                                    labelTimeWidth, labelTimeHeigth)];

        [self.nextButton setFrame:CGRectMake(nextButtonX, nextButtonY,
                                    nextButtonWidth, nextButtonHeight)];

        [self.labelTime2 setTextAlignment:NSTextAlignmentRight];

    }else{

        self.topLabel.hidden = YES;
```

```
        self.slider.transform = CGAffineTransformMakeRotation(M_PI * 0.5);

        [self.thumbnailImage setFrame:CGRectMake(hThumbnailImageX,
                                                 hThumbnailImageY,
                                                 hThumbnailImageWidth,
                                                  hThumbnailImageHeight)];

        [self.labelTime2 setTextAlignment:NSTextAlignmentLeft];

        if (self.view.frame.size.width > 480.0) {

            [self.slider setFrame:CGRectMake((hSliderX)-50, hSliderY,
                                             hSliderWidth, hSliderHeight)];

            [self.labelTime1 setFrame:CGRectMake((hLabelTime1X)-50,
                     hLabelTime1Y, labelTimeWidth, labelTimeHeigth)];

            [self.labelTime2 setFrame:CGRectMake((hLabelTime2X)-50,
                     hLabelTime2Y, labelTimeWidth, labelTimeHeigth)];

            [self.nextButton setFrame:CGRectMake((hNextButtonX)-10,
                    hNextButtonY, nextButtonWidth, nextButtonHeight)];
        }else{

            [self.slider setFrame:CGRectMake(hSliderX, hSliderY,
                                     hSliderWidth, hSliderHeight)];

            [self.labelTime1 setFrame:CGRectMake(hLabelTime1X,
                  hLabelTime1Y, labelTimeWidth, labelTimeHeigth)];

            [self.labelTime2 setFrame:CGRectMake(hLabelTime2X,
                 hLabelTime2Y, labelTimeWidth, labelTimeHeigth)];

            [self.nextButton setFrame:CGRectMake(hNextButtonX,
                hNextButtonY, nextButtonWidth, nextButtonHeight)];
        }
    }
}
```

6.2.5.- Notificaciones Push.

La tecnología push es un tipo de comunicación en la que es el servidor el que inicia la petición al cliente cuando tiene una información nueva, permitiendo un importante ahorro de recursos y tiempo respecto a la tecnología convencional pull (el cliente pregunta al servidor si existe nueva información). Por lo tanto una notificación push es, básicamente, un mensaje enviado por un servidor a un cliente que está suscrito a sus notificaciones.

Para poder trabajar con este tipo de notificaciones se necesita de un servidor que envíe las notificaciones, para ello Apple proporciona el Apple Push Notification Service (APNS), un servicio que se encarga de enviar las notificaciones a las aplicaciones.

Ciclo de vida de las notificaciones push:

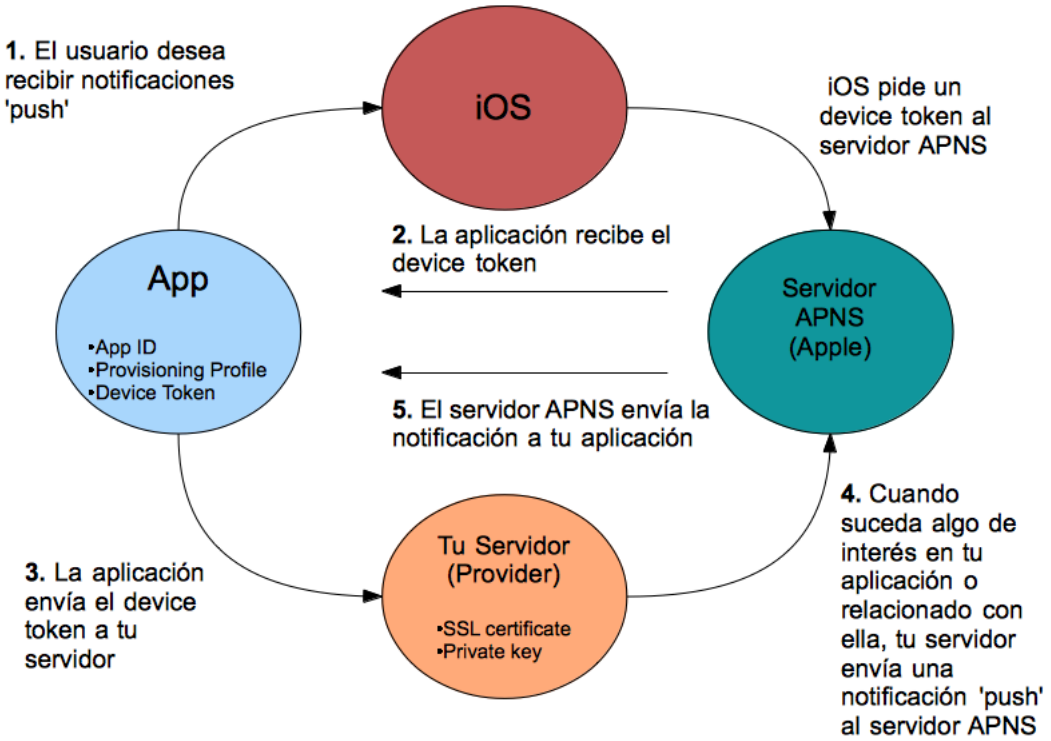


Figura 6.8.: Ciclo de vida de las notificaciones push

1. La aplicación tiene habilitada las notificaciones Push. El usuario tiene que confirmar que quiere recibir estas notificaciones.

```
//Habilitar la recepción de notificaciones push
[[UIApplication sharedApplication] registerForRemoteNotificationTypes:
(UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeSound |
 UIRemoteNotificationTypeAlert)];
```

2. La aplicación recibe el “device token” del servidor de Apple. Se puede ver este “device token” como la dirección que utilizará Apple para hacer llegar las notificaciones al dispositivo.
3. La aplicación envía el “device token” al servidor y este lo almacena.

```
//Guardar “device token” y enviarlo al servidor (Parse)
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    PFInstallation *currentInstallation = [PFInstallation
currentInstallation];
    [currentInstallation setDeviceTokenFromData:deviceToken];
    [currentInstallation saveInBackground];
}
```

4. Cuando suceda algo de interés en el sistema, el servidor externo le enviará la notificación Push al APNS.

```
Parse.Cloud.afterSave("Vídeo", function(request) {
  if (!request.object.existed()){
    var rallyDay = request.object.get("nDay")
    query = new Parse.Query("Rallye");
    query.get(request.object.get("rallyId").id, {
    success: function(rallye) {
      var rallyName = rallye.get("name")
      var rallyId = rallye.id;
      var pushQuery = new Parse.Query(Parse.Installation);
      pushQuery.equalTo('channels','NewVideos');

      Parse.Push.send({
        where: pushQuery,
        data:{
        alert: "New vídeo added to: " + rallyName,
        badge: "Increment",
        type:"0",
        r:rallyId,
        d:rallyDay
      }
      }, {
      success: function() {
        console.log("Notificación enviada correctamente");
      },
      error: function(error) {
        console.log("Error al enviar notificación después de añadir
vídeo”);
      }
      });
    },
    error: function(error) {
      console.error("Error al buscar rallye para enviar
notificación");
    }
    });
  }
});
```

5. APNS envía la notificación Push al dispositivo del usuario.

```
- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo {

    //Tratamiento de las notificaciones cuando la aplicación está
cerrada.
    if ([UIApplication sharedApplication].applicationState !=
UIApplicationStateActive) {
        [self handlePushForCloseApp:userInfo];
    }else{ //Tratamiento de las notificaciones cuando la aplicación
está abierta.
         [self handlePushForOpenApp:userInfo];
    }

    //Gestión de los contadores
    PFInstallation *currentInstallation = [PFInstallation
currentInstallation];
    if (currentInstallation.badge != 0) {
        currentInstallation.badge = 0;
        application.applicationIconBadgeNumber = 0;
        [currentInstallation saveEventually];
    }
}
```

A grandes rasgos este es el flujo de vida de las notificaciones push.

Canales de notificaciones en nuestro sistema:

Como podemos observar en el código que se ejecuta cuando se publica un nuevo vídeo, se envía una notificación a todos los usuarios que se encuentren suscritos al canal "NewVideos" (`pushQuery.equalTo('channels','NewVideos')`). Por defecto todo usuario cuando se crea se suscribe automáticamente a este canal.

Además, también cuando se crea un usuario, se crea un nuevo canal, que se llamará "IN_"+ objectId del usuario recién creado. Este será el "canal de entrada", por el cual se envíen las notificaciones relacionadas con dicho usuario. Por ejemplo, cada vez que alguien comente uno de sus vídeos, se enviará una notificación a ese canal indicando que un determinado usuario ha comentado su vídeo. De esta forma el usuario puede enterarse de todo lo relacionado con su perfil.

Pero ¿cómo se entera de las actividades de los usuarios a los que sigue?, para lograr hacer esto, todo usuario tiene un "canal de salida" asociado, este tendrá el nombre "OUT_" + objectId, y cada vez que otro usuario comienza a seguirle, también se suscribe a su "canal de salida". De esta manera cada vez que yo le de a me gusta, estaré enviado una notificación a todos mis seguidores, a través de mi canal de salida, donde indico que me gusta un determinado vídeo.

A continuación se muestra una tabla de ejemplo:

Usuario : objectId	Siguientes	Canales a los que esta suscrito
Peter : xCf	Asimo	- NewVideos - IN_xCf - Out_AcD
Asimo : AcD	Lisa Peter	- NewVideos - IN_AcD - OUT_Mkj - OUT_xCF
Lisa : Mkj		- NewVideos - IN_Mkj

Una vez que sabemos como funcionan las notificaciones vamos a ver en que consiste lo de activar o desactivar las notificaciones desde los ajustes de la aplicación.

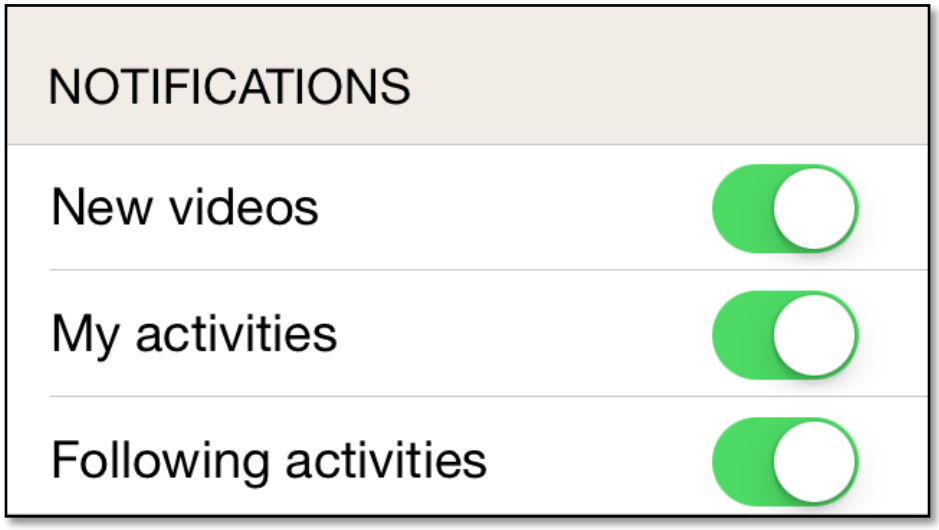


Imagen 6.33: Ajustes de notificaciones

Como se puede ver en la imagen anterior, podemos manejar tres tipos de notificaciones, la llamadas “New vídeos”, las llamadas “My activities” y las llamadas “Following activities”.

- New vídeos: activar o desactivar estas notificaciones implican suscribirse o darse de baja en el canal llamado “NewVideos”

- My activities: este caso es similar al anterior lo que debemos suscribirnos o darnos de bajar en el canal llamado “IN_” + objectID propio.

- Following activities: Este caso es un tanto diferente, ya que ahora no solo hay que suscribirse o darse de baja en un canal, si no que hay que hacerlo en tantos canales como personas estemos siguiendo. Darse de baja es la acción más fácil ya que únicamente hay que eliminar todos los canales excepto los llamados “NewVideos” y “IN_...”. En cambio para suscribirnos hay que saber primero quienes son esas

personas a las que seguimos; para ello hay que realizar una consulta a la base de datos y luego, una vez conocidos esos usuarios, suscribirnos a sus canales de salida.

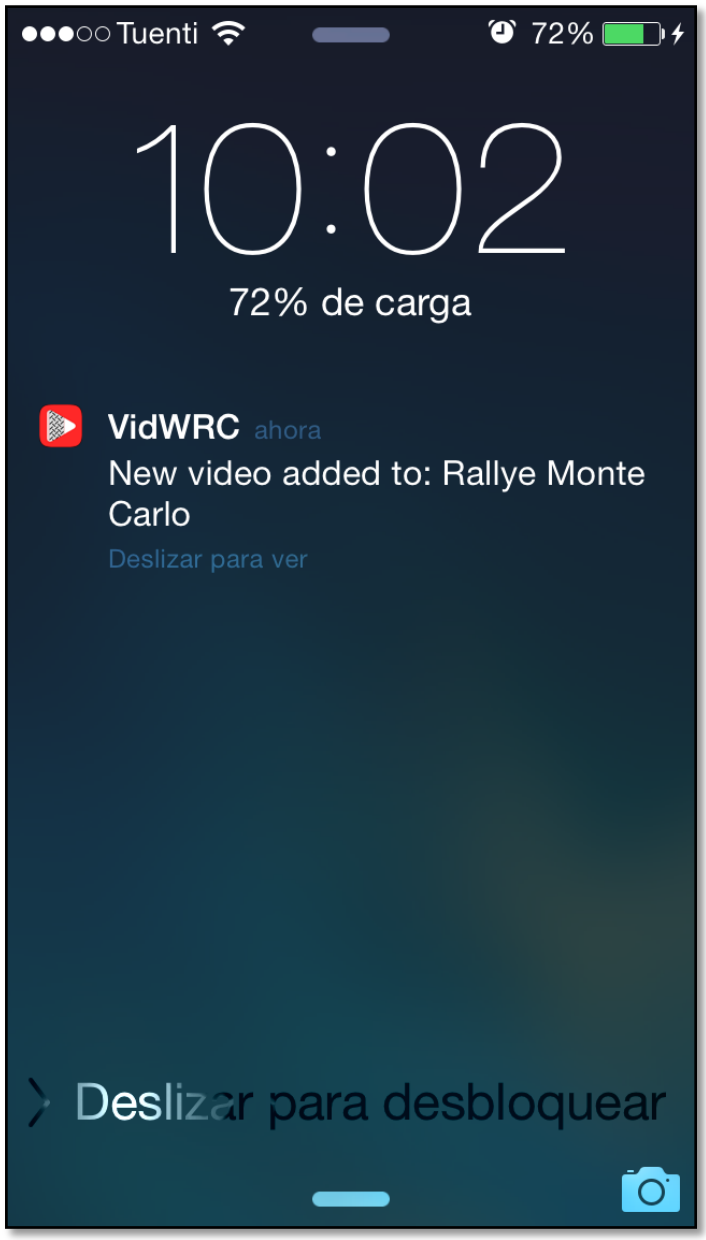


Imagen 6.34: Recepción de una notificación

7.- Resultados y conclusiones.

La elaboración de este proyecto ha supuesto un reto para mí, cuando comencé con todo esto no tenia ni idea de cómo empezar pero si tenia claro lo que quería hacer. Lo primero de todo fue comenzar a hacer pequeñas aplicaciones para ir familiarizándome con el nuevo lenguaje de programación (Objectice-C), con las diferentes librerías y con su entorno de desarrollo (Xcode). Comprender como funcionaba todo el ecosistema iOS me llevó bastante tiempo, en primer lugar, porque nunca había trabajo con una tecnología como esta y en segundo lugar, porque estaba acostumbrado a la creación de programas de menor complejidad.

Una vez que dominaba con cierta soltura las herramientas básicas, empecé a barajar las diferentes opciones para suplir las necesidades de un servidor donde centralizar todo. Tras unas semanas de trabajo de campo, opté por usar el servicio en la nube Parse. Ahora tocaba aprender a usar este backend en la nube, la curva de aprendizaje no fue muy dura y gracias a su buena documentación en unas semanas ya era capaz de hacer prácticamente todo lo que necesitaba.

Con todas las funcionalidades básicas implementadas, era hora de implementar los detalles que haría que la aplicación tomara un aspecto más profesional, esta fue sin duda una de las partes más tediosas de todo el desarrollo, ya que se emplea muchísimo tiempo en conseguir dejar todo a nuestro gusto.

Como resultado final de todo este proceso, se ha obtenido una red social con su correspondiente cliente para dispositivos móviles iOS. Como comentaba anteriormente, partiendo desde cero, se ha conseguido crear una aplicación totalmente funcional y con un diseño bastante logrado acorde a los conocimientos de diseño con los que contaba.

Aparte del resultado más evidente, el desarrollo de este proyecto me ha hecho adquirir nuevos conocimientos y me ha ayudado a desarrollar nuevas habilidades en el manejo de herramientas. Se puede decir con toda certeza, que el objeto principal de este proyecto se ha cumplido.

Como problemas surgidos durante la realización de este proyecto cabe destacar la portabilidad de la aplicación de una versión de iOS a otra, ya que comencé el desarrollo en iOS 6 y lo finalice en iOS 7. Este cambio provocó diversos errores ya que varios métodos que usaba habían quedado en desuso.

Como parte positiva, mencionar las facilidades que ofrece el entorno iOS para los desarrolladores, hay infinidad de librerías y todo lo que se nos ocurra hacer casi seguro que ya alguien lo ha hecho antes. Además de esto, hay mucha información disponible en internet y su entorno de desarrollo cuenta con todas las herramientas necesarias para la creación de una aplicación.

8.- Trabajo futuro.

Se puede seguir trabajando en muchos aspectos para mejorar la aplicación así como también se pueden crear nuevos clientes para la red social que se ha implementado.

¿Qué mejorar o que características añadir a la aplicación?

- **Interfaz multi-idioma**: hoy en día es algo básico para cualquier aplicación ya que se distribuyen en tiendas que operan a nivel mundial.
- **Interfaz para tablets:** el uso de las tablets ha aumentado significativamente en los últimos tiempos y se está convirtiendo en un sustito de los ordenadores actuales, es por esto que sería interesante crear una interfaz para este tipo de dispositivos ya que permitiría aumentar el público de destino.
- **Mejoras en los servicios que ofrece:** aparte de vídeos también se podrían ofrecer las clasificaciones de los rallyes para mantener informados a los usuarios.
- **Mejora de sistema de clasificación de los vídeos:** actualmente, los vídeos se clasifican según el rally al que pertenecen y al día dentro del rally, esta nueva propuesta consistiría en afinar aún mas el sistema y permitir a los usuarios geoposicionar los vídeos dentro del mapa de tramos que forman un rally.

Imagen 8.1: Geoposicionamiento de vídeos

- **Mejora para insertar vídeos que ya han sido subidos a YouTube**: esta mejora consistiría en crear una interfaz donde el usuario pueda publicar vídeos en el sistema desde su librería de vídeos en YouTube, es decir el usuario puede añadir vídeos que no han sido subidos a través de la interfaz que proporciona la aplicación.

Nuevos clientes para la red social: otro aspecto en el que trabajar es la creación de nuevos clientes para distintas plataformas móviles, como pueden ser Android o Windows Phone. Además se podría crear una versión web reducida y pública, donde cualquier persona pueda ver el contenido pero no pueda interactuar, este cliente podría añadir publicidad, lo que podría significar una fuente de ingresos y una motivación para que el usuario se instale la aplicación en el dispositivo móvil.

9.- Bibliografía

- https://developer.apple.com/library/IOS/navigation/
- https://sites.google.com/a/2coders.com/curso-ios-iniciacion/
- http://objective-c.es
- http://www.raywenderlich.com/es/
- http://www.manzanamagica.com/desarrollo/
- http://stackoverflow.com
- http://www.startcapps.com/
- https://parse.com
- http://www.javierrodriguez.com.es/blog/2011/10/24/streaming-adaptativo-en-internet-2/
- https://developers.facebook.com
- https://dev.twitter.com
- https://developers.google.com/?hl=es
- https://github.com/0xced/XCDYouTubeVideoPlayerViewController
- https://github.com/jdg/MBProgressHUD
- https://github.com/luugiathuy/GTScrollNavigationBar
- https://github.com/CooperRS/RMStepsController
- https://github.com/CooperRS/RMPickerViewController.
- https://github.com/CooperRS/RMMultipleViewsController.
- https://github.com/jessesquires/JSQFlatButton.
- https://github.com/mattt/FormatterKit
- https://github.com/tonymillion/Reachability
- https://github.com/benilovj/UIImage-Categories
- http://www.dit.upm.es/~santiago/docencia/ios/
- http://www.techotopia.com/index.php/The_iPhone_OS_Architecture_and_Frameworks
- http://eve-ingsistemas-u.blogspot.com.es/2012/04/sistemas-operativos-moviles-ios.html
- http://code.tutsplus.com/tutorials/ios-succinctly-hello-ios--mobile-18813

10.-Anexos

Manual de usuario

Inicio de sesión y registro

Existen varias formas de iniciar sesión en el sistema, bien a través de redes sociales como Twitter y Facebook, o bien a través de un nombre de usuario y una contraseña. Si se quiere acceder haciendo uso de una red social habrá que pulsar en el correspondiente botón y si se quiere acceder mediante un nombre de usuario y contraseña, pulsando en el botón “login”. Si no se dispone de una cuenta de usuario, esta se puede crear accediendo a un formulario de registro a través del botón “Sign Up”.

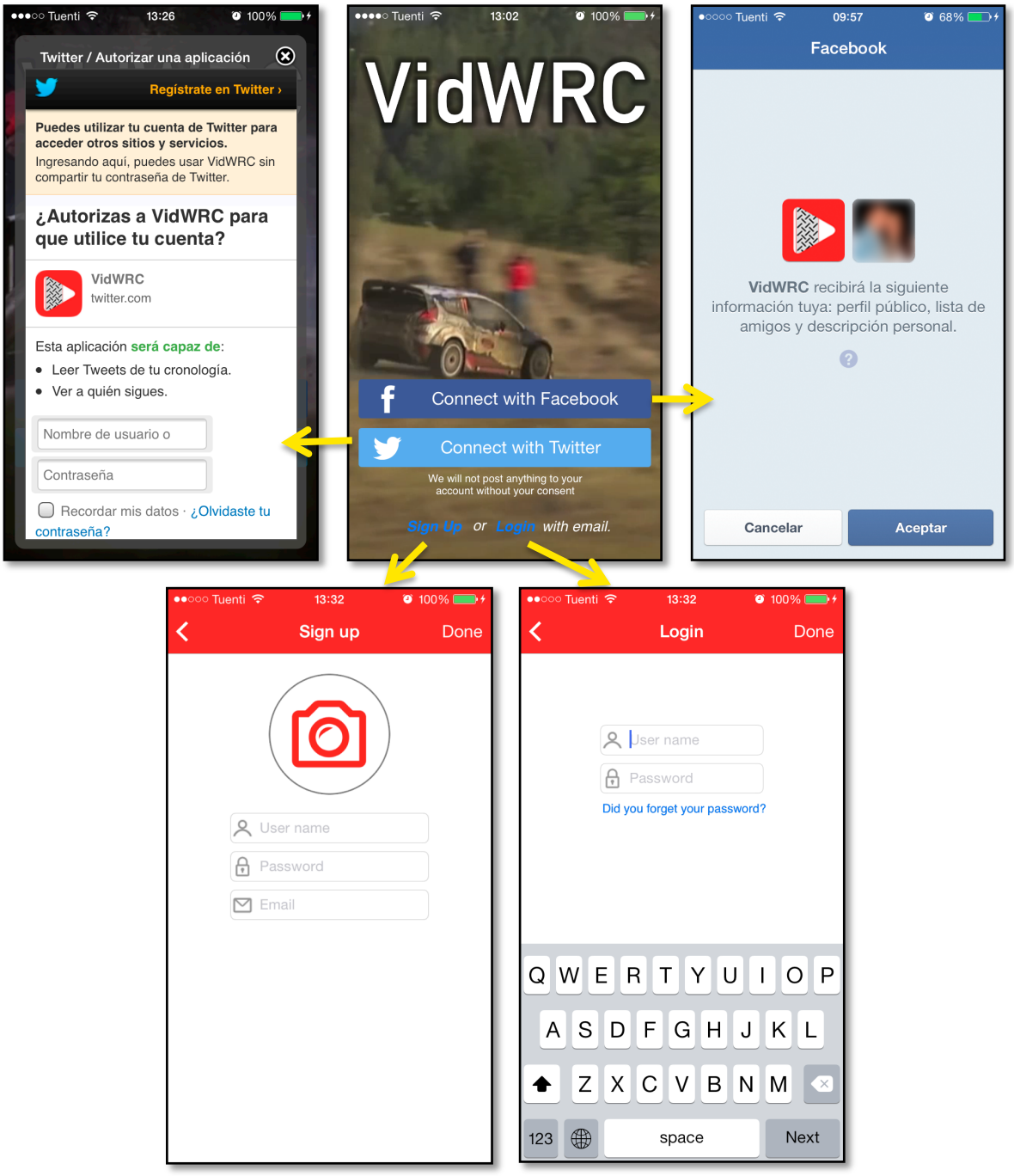


Imagen 10.1: Inicio de sesión

Panel principal y pestaña Home

Esta es la vista principal de la aplicación y es donde se encuentra el listado completo de rallyes que forman el campeonato. Para acceder los vídeos de cada rallye basta con pulsar sobre el rallye deseado y seleccionar uno de los días en los que se divide el rallye.

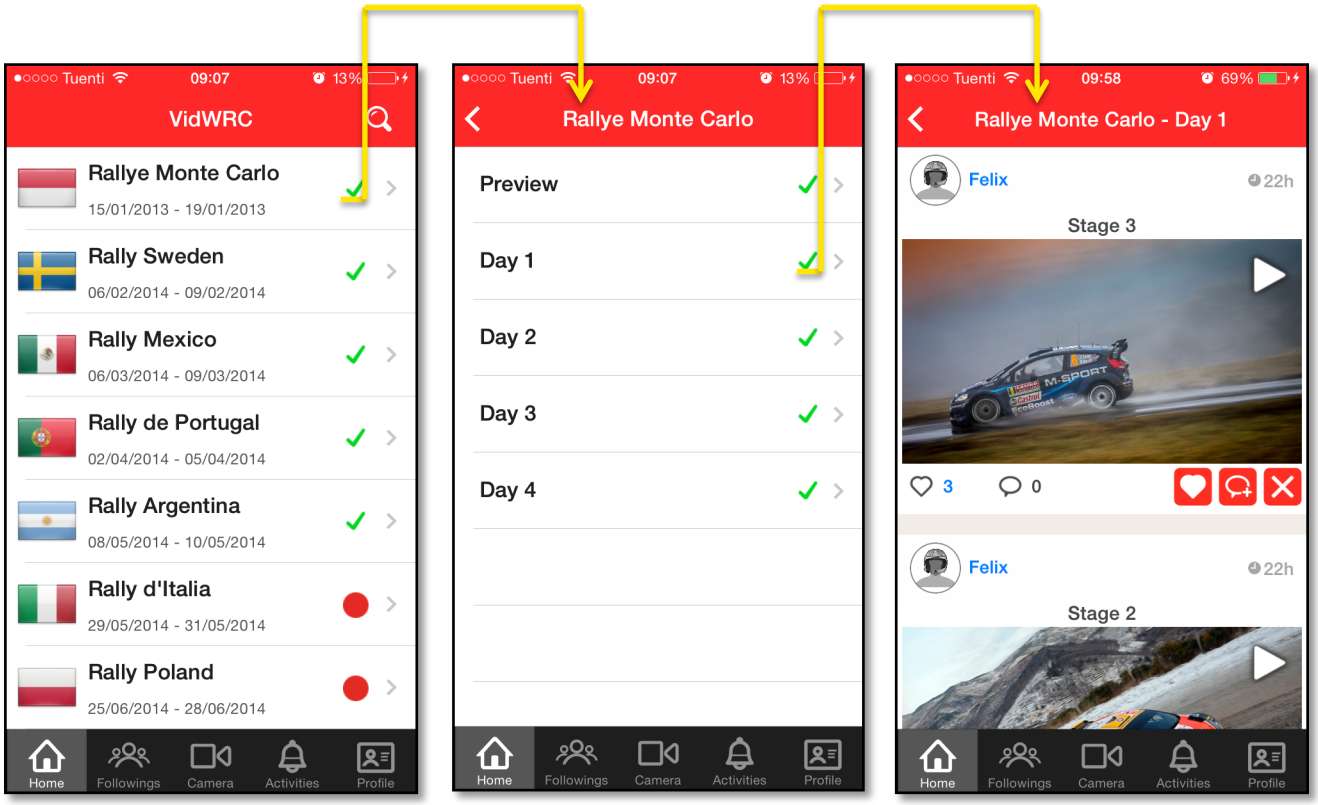


Imagen 10.2: Pestaña "Home"

Como se aprecia en las tablas anteriores existen unos símbolos en cada uno de los rallyes así como en cada uno de los días que componen un rallye, estos símbolos sirven para indicar diferentes estados:

Simbología	Significado
	Apartado inaccesible.
	Apartado accesible, falta menos de una semana para el inicio del rallye.
	Apartado accesible, rallye disputándose actualmente.
	Apartado accesible, rallye disputado.

Búsquedas:

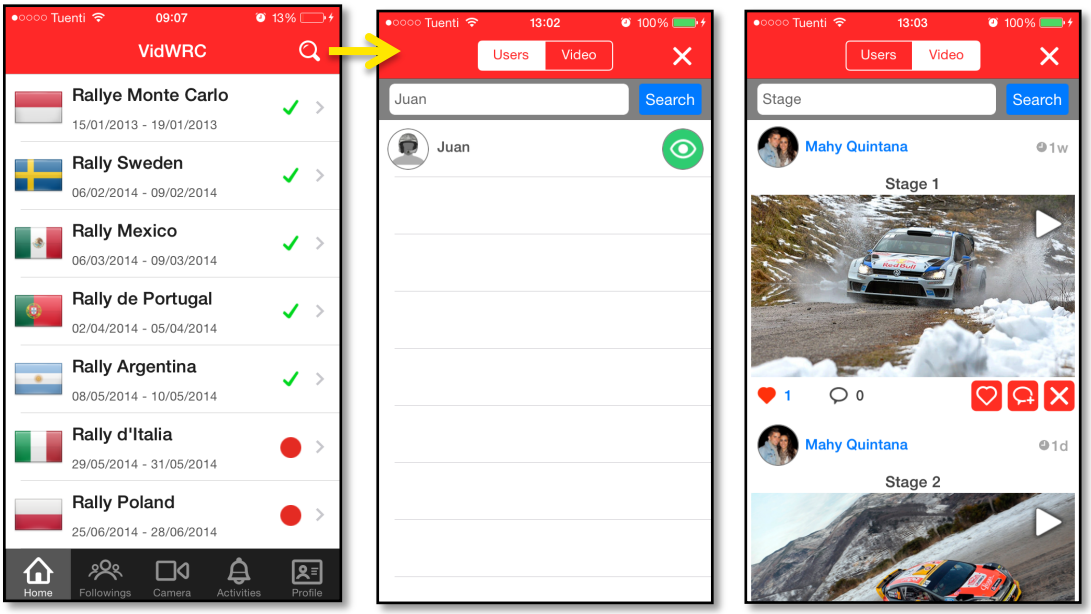
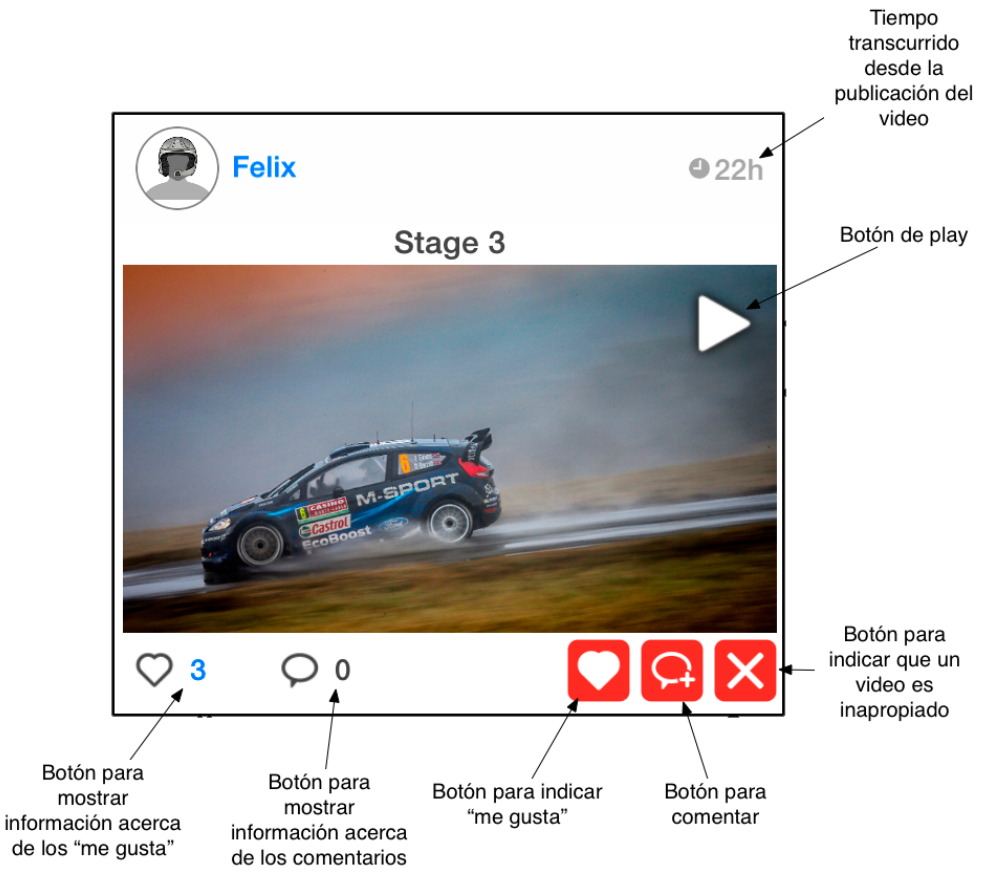


Imagen 10.3: Búsquedas

Vista detallada de un vídeo:



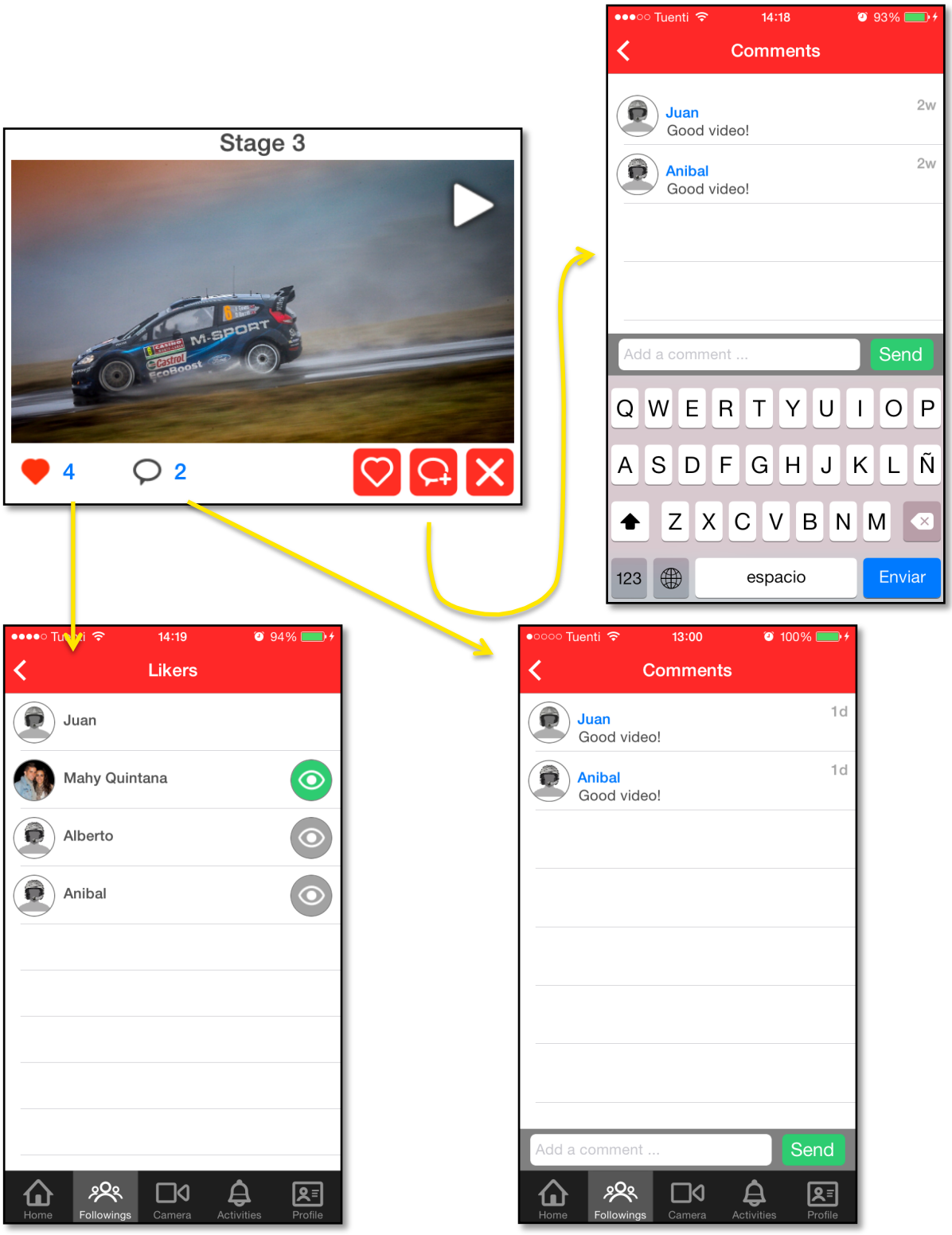
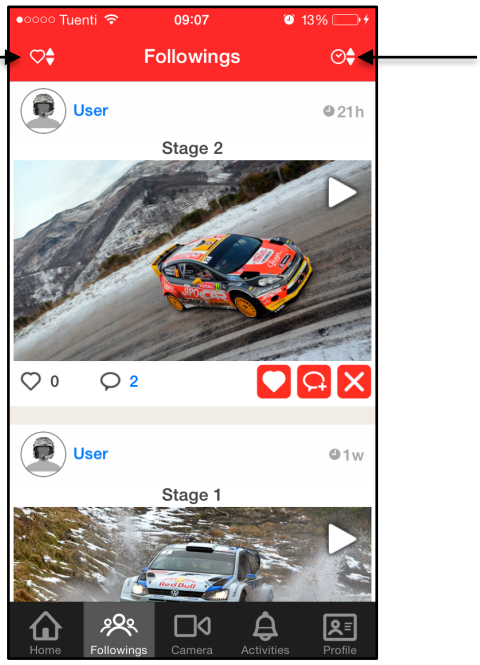


Imagen 10.4: Vídeo

Pestaña Followings

En esta pestaña se muestran las publicaciones de los usuarios a los que el usuario actual está siguiendo, así como sus propios vídeos.

Botón para ordenar los vídeos, primero los que tengan mayor cantidad de “me gusta”.



Botón para ordenar los vídeos en orden cronológico.

Imagen 10.5: Pestaña “Followings”

Pestaña Camera

En esta pestaña es donde se le da la posibilidad al usuario de publicar un nuevo vídeo, para ello debe seguir una secuencia que consta de 5 pasos:

- El primer paso consiste en identificarse con una cuenta de YouTube para poder subir el vídeo.
- El segundo de los pasos es obtener el vídeo, este se podrá seleccionar de la biblioteca o se podrá crear en ese instante haciendo uso de la cámara del dispositivo.
- Lo siguiente es seleccionar el fotograma del vídeo que se mostrará como previsualización del vídeo.
- El en cuarto paso se le dará un título al vídeo, se etiquetará en un rallye y día y se podrá elegir si compartir o no este vídeo en Facebook.
- El quinto y último paso consiste en seleccionar la visibilidad del vídeo dentro de YouTube e iniciar la publicación del vídeo.

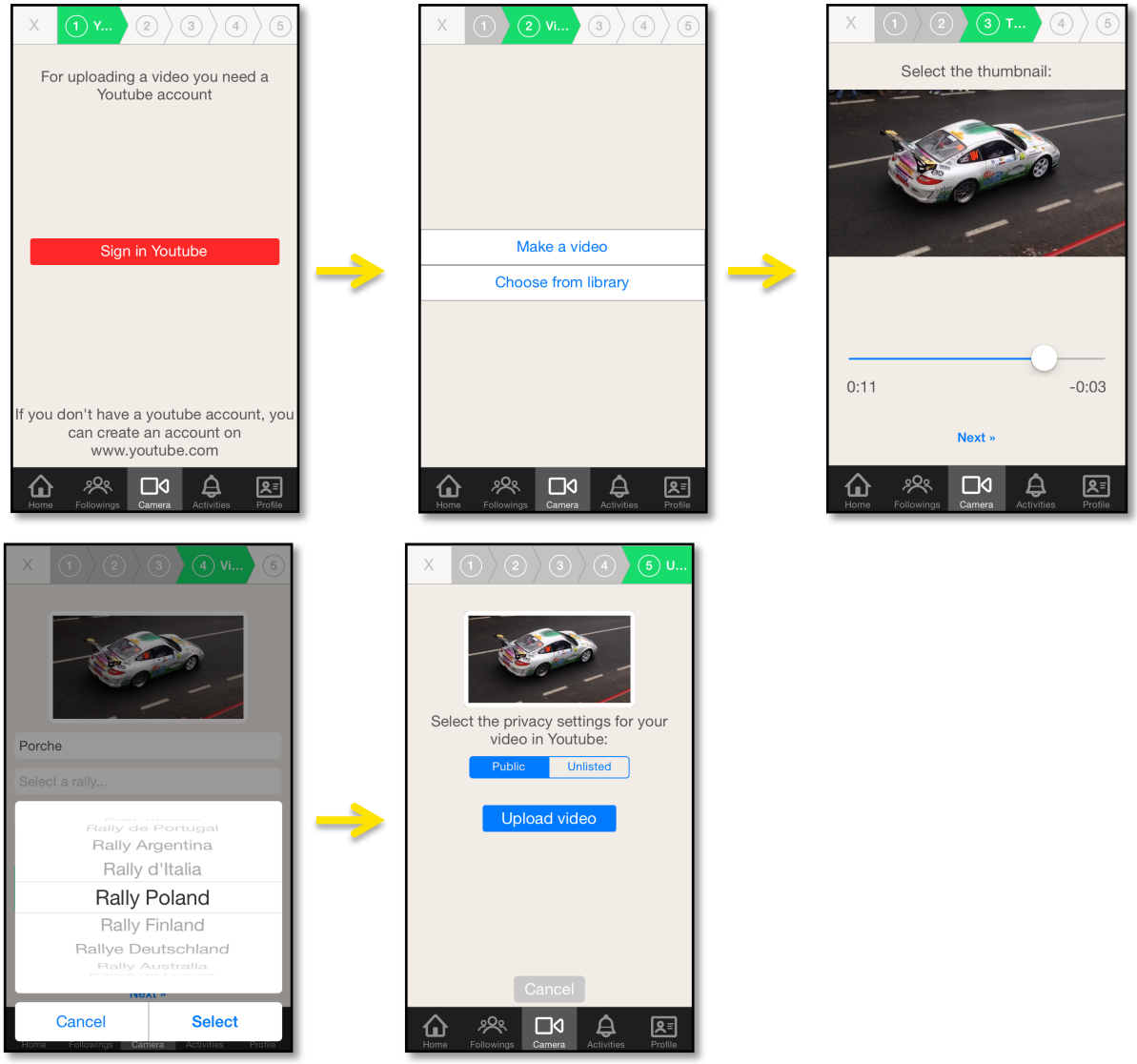


Imagen 10.6: Pestaña "Camera"

Pestaña Activities

Esta pestaña de subdivide en tres apartados distintos:

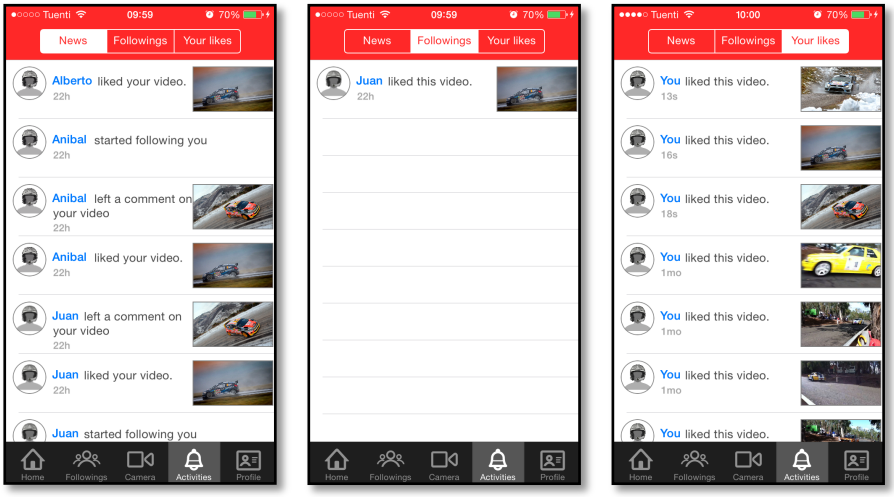


Imagen 10.7: Pestaña "Activities"

- En el primer apartado, denominado “News”, es donde se pueden ver las actividades relacionadas con los vídeos del usuario, así como con su perfil. Existen tres actividades posibles:
    - Otro usuario comienza a seguir al usuario actual.
    - Alguien comenta un vídeo del usuario.
    - A alguien le gusta un vídeo del usuario.
- En el segundo apartado, denominado “Followings”, es donde se mostrarán los vídeos que les gusta a nuestros “siguientes”.

Por último, en el apartado denominado “Your likes ”, se pueden ver todos los vídeos que han gustado al usuario actual.

Para poder ver el vídeo relacionado con la actividad solo hay que pulsar sobre su imagen, esta acción hará que aparezca una nueva vista donde se muestra el vídeo y sus comentarios:

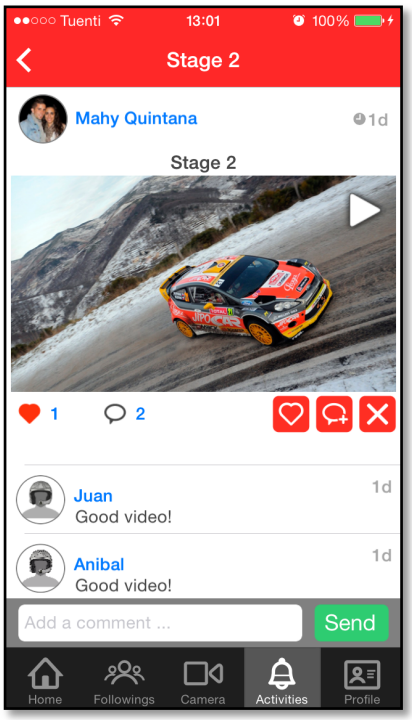


Imagen 10.8: Pestaña “Activities”

Pestaña Profile

En esta pestaña se mostrará información relacionada con el usuario y se le permitirá gestionar sus vídeos.

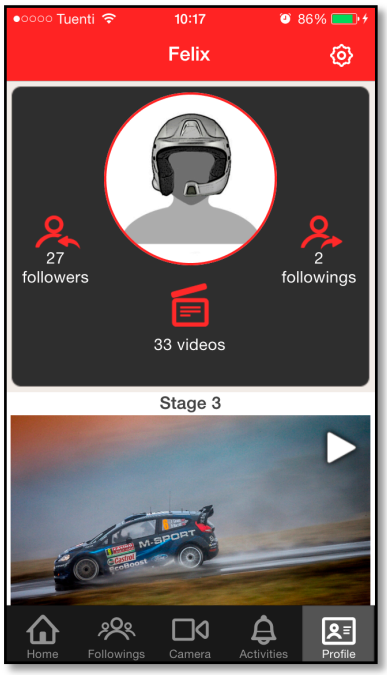


Imagen 10.9: Pestaña "Profile"

Dentro de la sección "Profile" el usuario puede acceder a los ajustes, desde esta nueva vista, el usuario puede gestionar aspectos relacionados con su cuenta y modificar otros parámetros relacionados con la aplicación como pueden ser las notificaciones. Esta vista será diferente para unos u otros usuarios, todo depende de la forma en la que hallan iniciado sesión. A los usuario que usen las redes sociales para iniciar sesión solo se les permitirá cambiar su foto y eliminar su cuenta, en cambio, a los usuarios que inicien sesión haciendo uso de un nombre de usuario y una contraseña, podrán modificar también su dirección de correo y cambiar su contraseña.

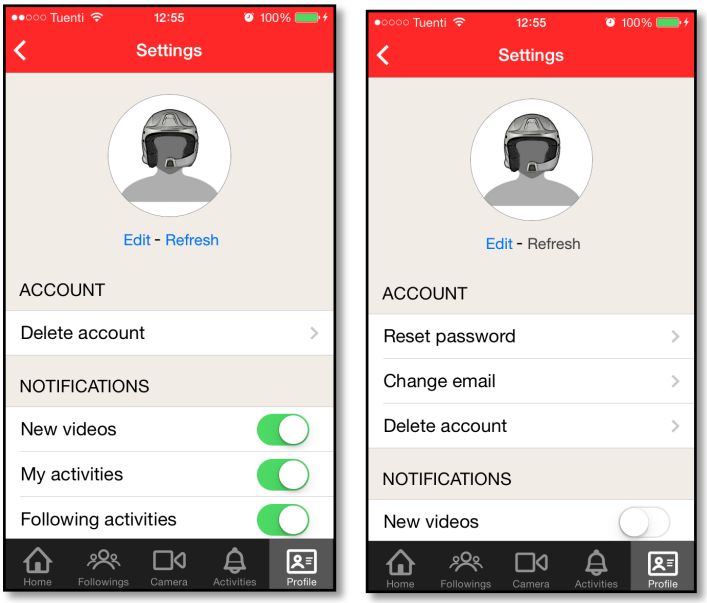
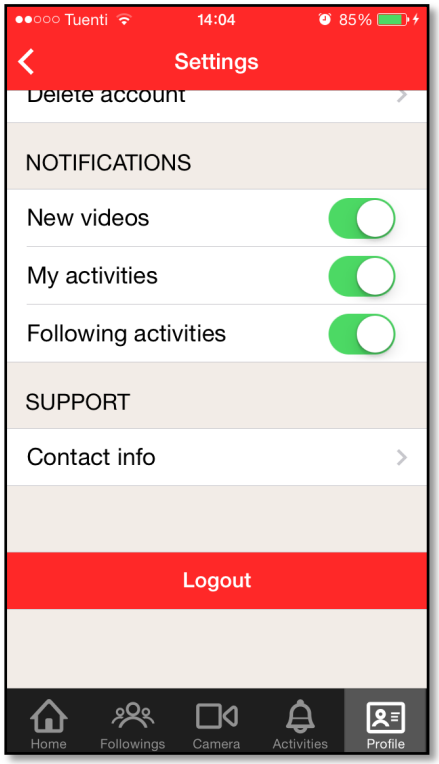


Imagen 10.10: Pestaña "Profile" -> "Settings"

Es también en esta vista donde el usuario puede cerrar su sesión a través del botón “Logout”.



¿Cómo seguir a otros usuarios?

Comenzar a seguir a otros usuarios es muy sencillo, solo debemos pulsar sobre el icono  . Al pulsar sobre el icono, este deberá cambiar de color, lo cual nos indica que ya estamos siguiendo a ese usuario  . Este icono se verá siempre asociado a un usuario como por ejemplo cuando visitamos su perfil o cuando vemos una lista de usuarios:

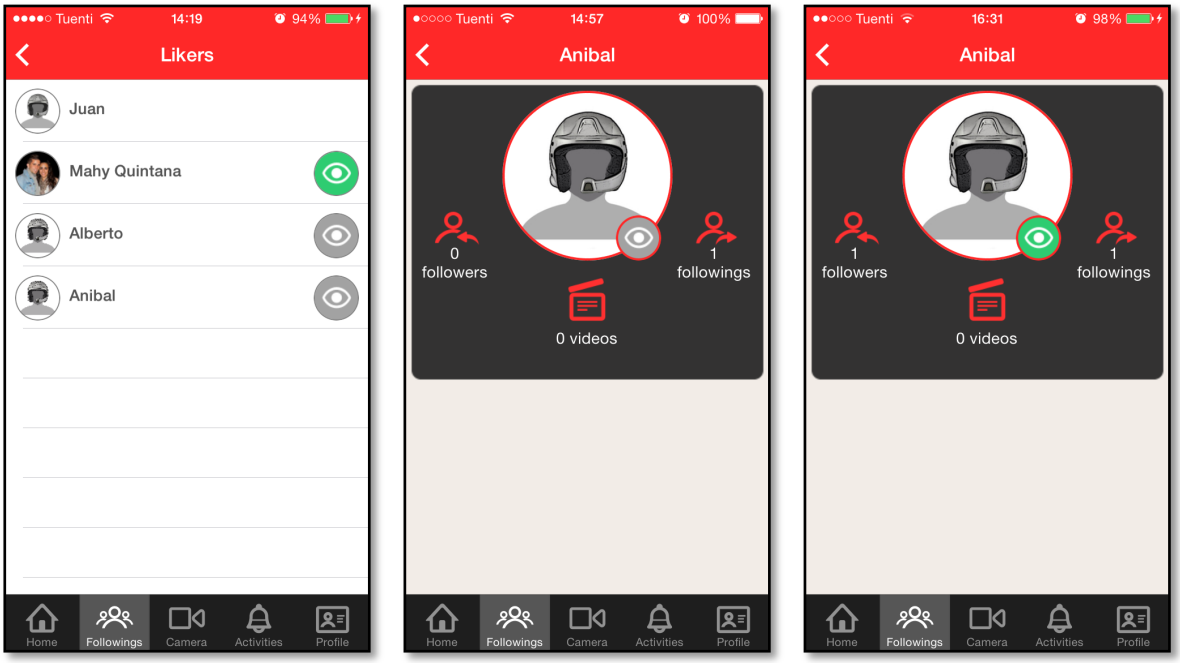


Imagen 10.11: Seguir a otros usuarios