Tagoror: Un espacio virtual geolocalizado para escuchar historias

Proyecto creado por Carlos Gustavo Pérez Almécija

Supervisado por:
Modesto Castrillón Santana y José Javier Lorenzo Navarro

Junio - 2019

Agradecimientos

A mis padres y mi hermano, por apoyarme en todo lo que hago.

A mi pareja, porque ella también es mi mejor amiga, que me ayuda en todos los momentos difíciles.

A mi familia, que siempre han estado conmigo.

A mi abuelo, porque aunque ya no esté aquí, sé que está orgulloso de mí.

A mis amigos, por acompañarme a lo largo del camino.

A mis tutores, Modesto Castrillón Santana y José Javier Lorenzo Navarro, por ayudarme a lo largo del proyecto.

Resumen

Tagoror es un proyecto software basado en tecnologías móviles en el cual los usuarios pueden subir a un servidor localizado en la nube, una historia creada por ellos mismos con cualquier contenido en formato audiovisual, de forma que asocian dicho contenido a algún punto del mapa físico. Por otro lado, los usuarios deberán acercarse físicamente al lugar en el que se guardó la historia, utilizando para ello el sistema de geolocalización del dispositivo móvil. Una vez en las proximidades del lugar, se podrá acceder a ver la historia y valorarla. De esta forma, se crea una comunidad de usuarios en la que ellos mismos sean los creadores del contenido que decidan compartir con los demás usuarios.

Abstract

Tagoror is a software project based on mobile technologies in which users can upload to a server located in the cloud, a story created by themselves with any content in audiovisual format, so that they deposit that content somewhere on the physical map. On the other hand, users must physically approach the place where the story was saved, using the geolocation system of the mobile device. Once they are close to the place, the user can reproduce the history and rate it, creating a community of users in which they themselves are the creators of the content they decide to share with other users.

Índice general

Índice de figuras

Índice de cuadros

Índice deAlgoritmos

Capítulo 1

Introducción

> En el pasado eras lo que tenías, ahora eres lo que compartes
>
> Godfried Boogard

En la *Era de la Información*, donde todo lo que nos rodea está relacionado con las *Tecnologías de la Información*, la sociedad actual está acostumbrada a tener mucha información al alcance de su mano, de esta manera las redes sociales hacen que un usuario pueda conocer lo que ocurre en su círculo social.

Tagoror es un proyecto móvil que se basa en, a través de la geolocalización del dispositivo, recolectar historias que el usuario quiera subir, de cualquier tipo y en formato de vídeo o audio, y marcar su localización en un punto del mapa. Para poder acceder a dichas historias, los usuarios de la aplicación deberán acercarse físicamente al punto en el que se encuentre la historia. Una vez se encuentre lo suficientemente cerca, podrá disfrutar del contenido de la historia, y se quedará marcada como leída para ese usuario.

Tagoror pretende acercar a las personas en un sentido más *físico*, en donde para conocer lo que ocurre en su entorno sea necesario acercarse a observar la historia que otro usuario ha colocado en ese punto del mapa. No sólo eso, sino que se da la posibilidad a que dichas historias contengan un amplio abanico de conocimiento listo a darse a conocer a la sociedad en su conjunto.

1.1. Motivaciones a empezar el proyecto

Como se puede percibir, este Trabajo de Fin de Grado tiene por objetivo principal el poder compartir una historia en formato audiovisual entre usuarios con la sociedad que le rodea, gracias a las facilidades que ofrecen las tecnologías móviles.

Una de las principales motivaciones reside en la posibilidad de ofrecer a la sociedad en su

totalidad una forma de expresarse con su entorno, de manera que los usuarios necesitan moverse de manera física al punto en donde el usuario ha «depositado» su historia, generando de esta manera que aquellos que quieran ver ciertas historias obtengan una alternativa de realizar ejercicio físico mientras disfrutan observando contenido en formato audiovisual.

Otra motivación para realizar el proyecto es aquella que, a lo largo de la carrera no se ha planteado contenido relacionado con tecnologías Android, las cuales están adquiriendo mayor fuerza cada día que pasa, generando de esta forma cierto atractivo para poder empezar a conocerlas y cómo trabajar con ellas de manera que pueda realizar un proyecto asegurando optimización en cuánto a calidad, rendimiento y seguridad óptimos para la aplicación.

Capítulo 2

Fundamentos relacionados

2.1. Qué es Android

Android es un sistema operativo desarrollado por *Google* y se encuentra basado en Linux. Está diseñado para dispositivos móviles como smartphones. A finales del año 2018, Android es el sistema operativo móvil más utilizado del mundo, con una cuota de mercado cercana al 90 % (véase [13]). Podemos ver en la Ilustración 2.1 cómo aumentó la tasa de mercado de sistemas móviles entre el último trimestre año 2017 y el del año 2018. Si se desea obtener más datos sobre qué es Android, podemos observar la siguiente referencia [19].

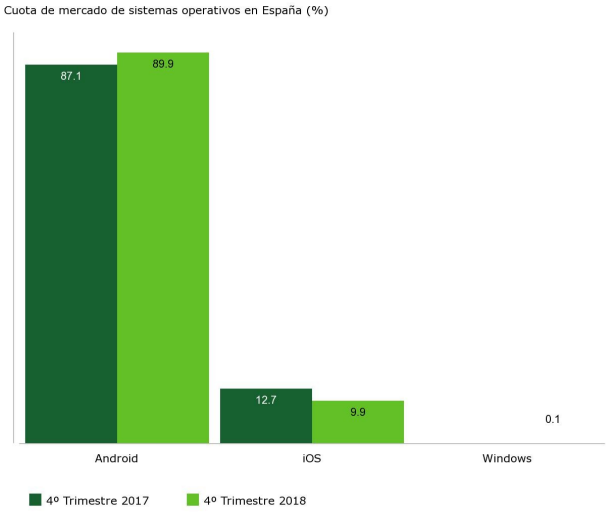

Ilustración 2.1: Cuota de mercado de ***Android*** en España. Fuente : [13]

2.1.1. Arquitectura Android

Los componentes del sistema operativo Adnroid son:

- ✔ **Aplicaciones.** En Android y en su arquitectura se encuentran por defecto aplicaciones base las cuales se encuentran escritas en *Java*: mapa, navegador, contactos del teléfono, etc.
- ✔ **Marco de trabajo de aplicaciones.** En el cual los desarrolladores tienen acceso a las mismas API usadas por las aplicaciones. Ésta arquitectura está diseñada para simplificar la reutilización de los componentes en Android.
- ✔ **Bibliotecas.** Android incluye un conjunto de librerías basadas en *C/C++* usadas por varios componentes del sistema.
- ✔ **Runtime de Android.** Android incluye un conjunto de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas del lenguaje *Java*. Cada aplicación realiza su propio proceso, con su propia instancia en la máquina virtual *Dalvik*, que ha sido escrito de manera que un dispositivo puede correr múltiples máquinas de manera eficiente.
- ✔ **Núcleo Linux.** Android depende de *Linux* para los servicios base del sistema: seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. También actúa como capa de abstracción entre el hardware y el software.

En la ilustración 2.2 podemos ver un ejemplo gráfico de la arquitectura de Android.

Ilustración 2.2: Arquitectura de ***Android***. Fuente: [20]

2.1.2. Desarrollo de aplicaciones

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK), pero están disponibles otras herramientas de desarrollo para aplicaciones o extensiones en *C o C++*.

El desarrollo de aplicaciones para Android no requiere aprender lenguajes completos de programación, ya que para realizar una aplicación en Android sólo es necesario tener conoci-

mientos de *Java* y adquirir e instalar Android SDK, que se puede obtener de forma gratuita.

Todas las aplicaciones Android están comprimidas en formato APK, que se pueden instalar sin dificultad en la mayoría de dispositivos. Para mayor información, se puede consultar la siguiente referencia [20].

2.1.3. APK

Un archivo con extensión *.apk* (Android Application Package) es un paquete útil para un sistema Android. Es una variante del *JAR* en *Java* y es usado para distribuir e instalar componentes empaquetado para la plataforma Android.

Para crear un archivo APK, primero se compila un programa para Android y luego todas sus partes se empaquetan en un sólo archivo. Un archivo APK contiene todo el código de ese programa, recursos, activos, certificados y archivos de manifiesto.

2.1.4. Evolución de las versiones en Android

Android se inició con el lanzamiento de *Android beta* en noviembre de 2007. La primera versión, *Android 1.0* fue lanzada en septiembre de 2008, y desde ese momento este sistema operativo ha visto una serie de actualizaciones. Normalmente, dichas actualizaciones corrigen fallos de programas y agregan nuevas funcionalidades y cabe destacar que las versiones de Android son desarrolladas bajo cierto nombre y siguen un orden alfabético. Podemos observar la evolución de las versiones en el cuadro 2.1.

Nombre de versión	Número de versión	Fecha de lanzamiento	API
Apple Pie	1.0	23 de septiembre de 2008	1
Banana Bread	1.1	9 de febrero de 2009	2
Cupcake	1.5	25 de abril de 2009	3
Donut	1.6	15 de septiembre de 2009	4
Eclair	2.0 - 2.1	26 de octubre de 2009	5 - 7
Froyo	2.2 - 2.2.3	20 de mayo de 2010	8
GingerBread	2.3 - 2.3.7	6 de diciembre de 2010	9 - 10
HoneyComb	3.0 - 3.2.6	22 de febrero de 2011	11-13
Ice Cream Sandwich	4.0 - 4.0.5	18 de octubre de 2011	14 - 15
Jelly Bean	4.1 - 4.3.1	9 de julio de 2012	16 - 18
KitKat	4.4 - 4.4.4	31 de octubre de 2013	19 - 20
Lollipop	5.0 - 5.1.1	12 de noviembre de 2014	21 - 22
Marshmallow	6.0 - 6.0.1	5 de octubre de 2015	23
Nougat	7.0 - 7.1.2	15 de junio de 2016	24 - 25
Oreo	8.0 - 8.1	21 de agosto de 2017	26 - 27
Pie	9.0	6 de agosto de 2018	28
Q	10.0	Agosto de 2019	29

Cuadro 2.1: Listado de versiones de Android

Para más información de las respectivas versiones de Android, se puede ver la referencia [15].

2.2. Componentes de Android

Una vez conocido lo más básico de qué es Android, proseguimos mostrando los componentes más importantes para el desarrollo de aplicaciones en Android.

2.2.1. Permisos en Android

Android es un sistema operativo con privilegios independientes, en el que cada aplicación se ejecuta con una identidad de sistema distinta, de la misma forma que en Linux existe un identificador de usuario y de grupo.

Por tanto, obtenemos funcionalidades de seguridad más precisas mediante un mecanismo denomidado “permisos”, que aplica restricciones en algunas operaciones específicas, dependiendo de si el usuario acepta dichos permisos o no. Una aplicación básica en Android no tiene permisos asociados de manera predeterminada. Esto implica que no se puede hacer nada que afecte negativamente a la experiencia del usuario o datos del dispositivo. Estos permisos se dividen en diferentes niveles de protección.

Los dos niveles de permisos más importantes son el nivel normal y el que presenta riesgos.

- ✔ **Permisos normales.** Recogen áreas en las cuales una aplicación tiene que acceder a datos del dispositivo, pero donde existe un riesgo mínimo para la privacidad del usuario o funcionamiento de otras aplicaciones.
- ✔ **Permisos con riesgo.** Recogen datos en los que se incluye información privada del usaurio, o bien podrían alterar el funcionamiento de otras aplicaciones. Si una aplicación necesita un permiso con riesgo, el usuario debe otorgar explícitamente permiso. Algunos permisos de riesgo son:

  - Relacionados con la cámara.

  - Relacionados con la geolocalización.

  - Relacionados con el micrófono del dispositivo.

  - Relacionados con el almacenamiento, tanto en la lectura como en la escritura de archivos.

De hecho, cabe destacar que en el proyecto serán necesarios, entre otros, los permisos de riesgo mencionados para el correcto manejo de la aplicación. Podemos encontrar más información sobre los permisos en la referencia [6].

2.2.2. Manifiesto

Todas las aplicaciones deben tener un archivo llamado *"AndroidManifiest.xml"* en el directorio raíz. Este archivo proporciona información esencial sobre la aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la aplicación. Podemos ver más información del archivo de manifiesto en la siguiente referencia [5].

El archivo de manifiesto es útil para:

- ✔ **Nombrar el paquete Java del proyecto.** Haciendo de identificador para la aplicación.
- ✔ **Describe otros componentes, como las actividades, receptores de mensajes, etc.** También nombra las clases que implementa cada componente y publica sus capacidades. Estas declaraciones notifican al sistema los componentes y condiciones para su lanzamiento.
- ✔ **Determina los procesos que alojan a los componentes de la aplicación.**
- ✔ **Declara los permisos que debe tener la aplicación para poder interactuar con el usuario.**
- ✔ **Declara el nível mínimo de la API de Android que requiere la aplicación.**

En definitiva, es un archivo necesario para poder manejar correctamente datos necesarios que requiera nuestra aplicación en la interacción con el usuario.

2.2.3. Actividades

Una actividad es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción. En el caso de nuestra aplicación encontramos varias actividades, entre ellas, aquellas relacionadas con la visualización de un mapa.

Una aplicación consiste generalmente en múltiples actividades vinculadas entre sí. Existe una aplicación especificada como principal en el manifiesto de Android, haciendo que cada vez que se ejecute la aplicación aparezca dicha actividad. Cada actividad puede a su vez iniciar otra actividad para poder realizar diferentes acciones, de esta forma, se crean actividades para interacciones específicas por parte del usuario. El ciclo de vida de una actividad se puede resumir en la ilustración 2.3. Para más información, podemos ir a la referencia [4].

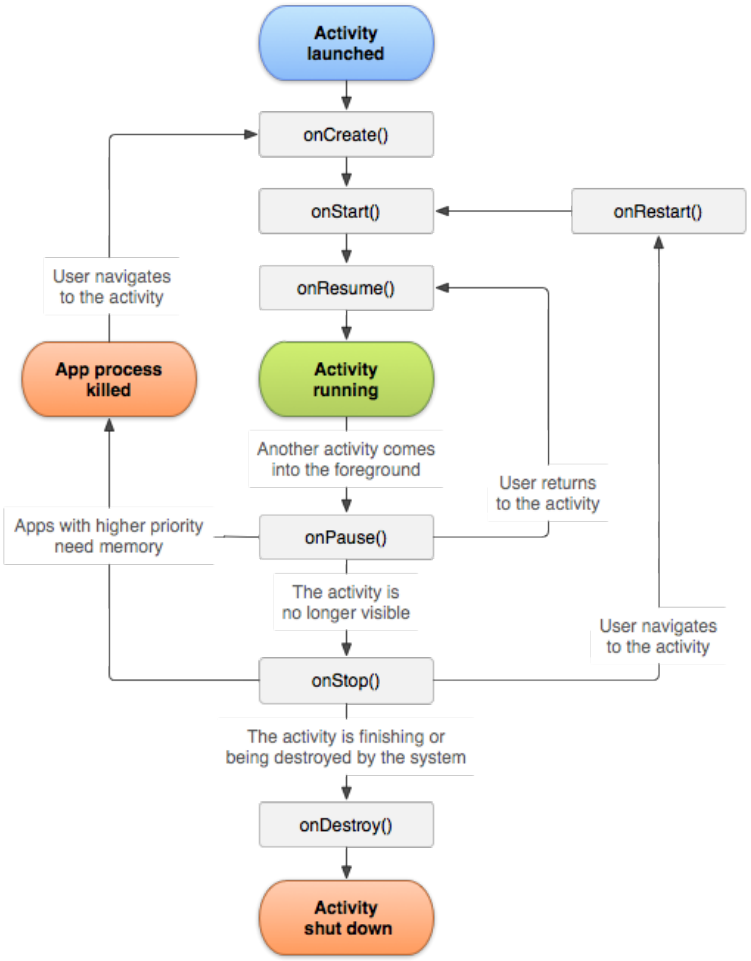

Ilustración 2.3: Ciclo de vida de una Actividad ***Android***. Fuente: [4]

Capítulo 3

Estado del arte

Actualmente en el mercado podemos encontrar aplicaciones móviles con objetivos similares, cuyo propósito es el de informar sobre la historia local al usuario. En el caso de la aplicación *Historias*, a través de un sistema de geolocalización en el que mientras se pasea por el lugar, aparecen puntos en el mapa y en el momento en el que el usuario esté a menos de 200 metros, se mostrará la historia del mismo. El formato de la aplicación es el que aparece en la Ilustración 3.1. Cabe destacar que aunque se ha podido encontrar la referencia a esta aplicación, a la hora de ir a la PlayStore de Google, dicha aplicación no aparece, por lo que podemos deducir que abandonó el mercado dicho programa. Un artículo de periódico que hace referencia a dicha aplicación lo podemos encontrar en la siguiente referencia [1].

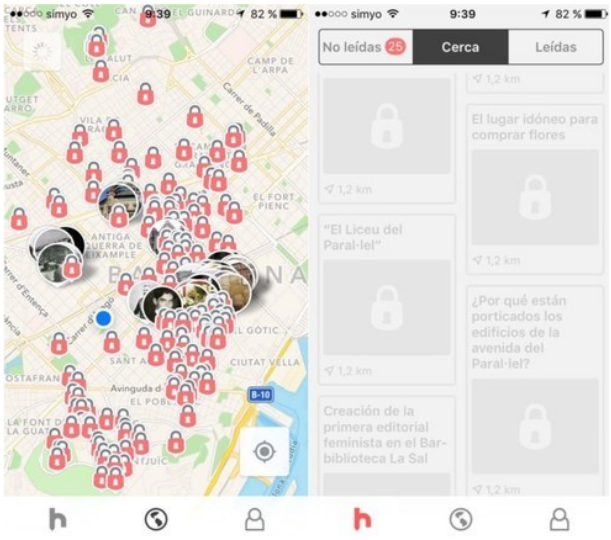

Ilustración 3.1: Interfaz de ***Historias***. Fuente: [1]

Además, Google está trabajando en una aplicación en la que cualquiera puede compartir una historia: *Bulletin*. El formato del mensaje será texto y podrá acompañarse de una fotografía. La compañía apuesta en que los usuarios sean los informantes de la comunidad. Se basa más

en contar el día a día que en contar una historia cualquiera proporcionada por el usuario, un ejemplo de ello es el de la Ilustración 3.2. Podemos encontrar información en el siguiente artículo [8] y en la página de Bulletin [9].

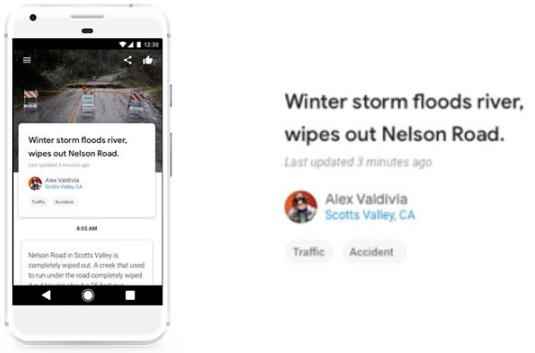

Ilustración 3.2: Interfaz de ***Bulletin***. Fuente: [9]

También podemos encontrar aplicaciones del tipo cultural centrado en el sector artístico, como es *ArtCity*, en la que se basa más en explorar la ciudad y observar diferentes puntos relacionados con el arte, con el añadido de que notifica de las diferentes exposiciones que estén a una distancia que puede ser modificada por el usuario, la interfaz de la aplicación la podemos ver en a continuación en la Ilustración 3.3. Siguiendo la siguiente referencia encontramos información de ArtCity [2].

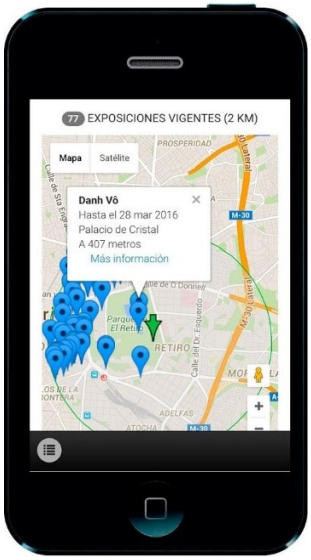

Ilustración 3.3: Interfaz de ***ArtCity***. Fuente: [2]

Para seguir hablando del estado del arte, no debemos olvidar aplicaciones como *Tales GPS Storytelling*, que se basan en contar historias locales relacionadas con la zona, en la cual el usuario pasea y si se acerca a un punto con contenido, su móvil hará sonar una historia en formato de texto. Además, aquellas historias que se hayan leído se podrán volver a leer en

la librería personal del usuario. Podemos ver a grandes rasgos cómo funciona la aplicación en la Ilustración 3.4. Si seguimos la siguiente referencia encontraremos la página web de la aplicación [12].

Ilustración 3.4: Interfaz de ***Tales GPS Storytelling***. Fuente: [12]

Cabe destacar otro proyecto, como *Contour*, que se basa en que se puedan subir historias a una página web, relacionadas con el género de la aventura: Bicicleta, Motocicletas, *Skateboarding*, *Snowboarding*, etc. Cuentan, además, con una tienda en donde se puede adquirir productos como cámaras para grabar el vídeo. Cabe destacar que este proyecto software parece ser que su ciclo de vida llegó a su fin, ya que, aunque la página web aún sigue, la última noticia reciente de ellos data del 15 de enero de 2015. A principios del año 2019, la interfaz de la web es tal y como se observa en la Ilustración 3.5. Para llegar a esta página web, nos podemos apoyar en la siguiente referencia [3].

Ilustración 3.5: Inicio de ***Contour***. Fuente: [3]

Echoes, es un sistema software el cual es similar a nuestro proyecto, de forma que el usuario puede pasear por el lugar en el que se encuentre y a través de la geolocalización, poder oír historias por parte de otros usuarios. Sin embargo, estas historias se basan más bien en términos turísticos, en donde los creadores de contenido suben tours para mostrar al usuario

cualquier cosa relacionada con el lugar. Cabe resaltar que existen dos aplicaciones para el uso de este proyecto: el que usan los usuarios que disfrutan de las historias, y el de aquellos que suben contenido a la aplicación, por lo que será necesario usar las dos si nos encontramos en ambos tipos de usuarios. Para el primer tipo de usuario, podemos observar la interfaz de la aplicación en la Ilustracion 3.6. Si ademas, deseamos buscar más información, es posible acceder a través de esta referencia [7].

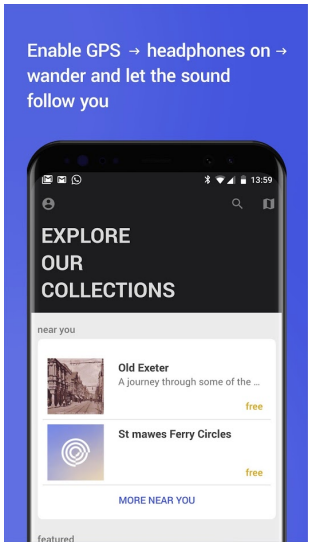

Ilustración 3.6: Interfaz de ***Echoes***. Fuente: [7]

Si continuamos por la línea turística, podemos tambien ubicar a *TravelStorys*, una aplicación que *convierte tu dispositivo en una guía online multimedia*. Las historias que almacena esta aplicación se encuentran en formato de audio, en las que el usuario puede explorar ciertos lugares y disfrutar de información relacionada con el lugar por el que se pasea. Podemos ver un ejemplo del sistema en la Ilustración 3.7. *TravelStorys* tiene una página web a la que se accede a través de esta referencia [17].

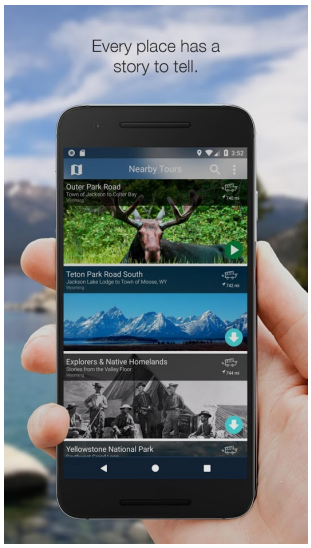

Ilustración 3.7: Interfaz de ***TravelStorys***. Fuente: [17]

VoiceMap es una aplicación en la cual se muestran diversos tours realizados por guías turísticos, normalmente son tours en formato audio y suelen ser dichos audios de pago, aunque también encontramos tours gratis. La aplicación cuenta con un sistema de votaciones para saber si merece la pena oír la historia. Además, cuenta con un sistema en el cual los tres primeros puntos por los cuales vas pasando por la aplicación son gratis. *VoiceMap* tiene por objetivo crear un tour de calidad en el que el usuario pasee por los diferentes puntos mientras escucha la historia local. Nos podemos hacer una idea del uso de la aplicación si observamos la Ilustración 3.8. Más información en el siguiente artículo [14], y además, aquí encontramos su página web [18].

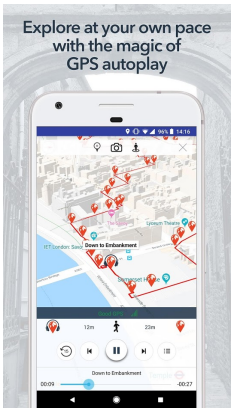

Ilustración 3.8: Interfaz de ***VoiceMap***. Fuente: [18]

Capítulo 4

Objetivos

En todo proyecto software es necesario plantearse una serie de objetivos para determinar el alcance de la aplicación en sí. Es por ello, que se han decidido ciertos objetivos que se han realizado a lo largo del ciclo de desarrollo del proyecto. Estos objetivos los podemos dividir dependiendo de la fase de recorrido del proyecto.

4.1. Objetivos iniciales

Los diferentes objetivos iniciales del proyecto son los siguientes:

- ✔ Diseñar un prototipo que permita acceder a contenido multimedia basado en geolocalización.
- ✔ Desarrollar un prototipo que permita compartir contenido entre usuarios.
- ✔ Crear un prototipo que permita al usuario subir contenido multimedia a un servidor.
- ✔ Implementar un prototipo que mantenga un sistema de sesiones de usuarios, en donde los mismos puedan editar sus respectivos perfiles.
- ✔ Diseñar un prototipo que en tiempo real muestre la geolocalización del usuario.
- ✔ Desarrollar un prototipo con contenido culturales.
- ✔ Diseñar una interfaz de usuario intuitiva para facilitar el uso de la aplicación al usuario.

4.2. Objetivos conseguidos a lo largo del proceso de desarrollo

Si bien es cierto que esos son los objetivos planteados en primera instancia al iniciar el proyecto, se ha de comentar que dichos objetivos aumentaron conforme las necesidades del

proyecto y las del usuario final. De esta forma, nos planteamos una serie de objetivos mínimos para la iniciación del proyecto, y obtenemos unos objetivos nuevos, añadidos a los iniciales, teniendo en cuenta que se puede editar esta pila de necesidades. Al final del proceso de desarrollo del proyecto, se completaron diferentes objetivos en la aplicación:

- ✔ Se puede acceder al apartado del perfil del usuario, en donde se podrán ver las diferentes características del mismo: foto de usuario, nombre o correo de usuario y las historias que ha visto y/o creado.
- ✔ Podemos iniciar sesión con nuestro usuario de tres formas posibles.

  -Vía Google.

  -Vía Facebook.

  -Vía Correo electrónico y contraseña.
- ✔ Un usuario puede registrarse en la aplicación a través de las tres vías comentadas en el punto anterior.
- ✔ Un usuario es capaz de reportar una historia para marcarla como contenido no adecuado.
- ✔ Los marcadores relacionados con las historias van apareciendo en función de la porción del mapa que visualice el usuario.
- ✔ Con respecto al apartado en el que podemos visualizar una historia, se han añadido aspectos de interfaz como botones de «Me gusta» o «No me gusta» para que el usuario valore la misma.
- ✔ Se añadió una barra de búsqueda y botones de acceso directo a la ubicación actual en el apartado de marcar la historia en el mapa para facilitar la interacción del usuario con la aplicación.

Con esta actulización de los objetivos iniciales, se ha conseguido desarrollar correctamente el proyecto que nos hemos determinado realizar.

Capítulo 5

Herramientas usadas

Para la realización de este proyecto se han hecho uso de diferentes herramientas, que se pueden dividir dependiendo de las diferentes necesidades:

5.1. Entorno de desarrollo

En términos de desarrollo de código, se ha utilizado «Android Studio» en su versión entorno de desarrollo oficial para los sistemas Android. Es un entorno desarrollado por JetBrains y está basado en IntelliJ. En la Ilustración 5.1, se puede observar el logotipo de Android Studio.

Ilustración 5.1: Logotipo de ***Android Studio***

Además, se utilizó para realizar las pruebas del sistema un dispositivo virtual Android, un Nexus 5X con la API en su versión 24. Con ello, gracias a que se ofrece este dispositivo y junto con el desarrollo del proyecto, se puede realizar pruebas a la vez que se implementan funcionalidades. En la Ilustración 5.2 se puede observar un ejemplo de interfaz de Android Studio.

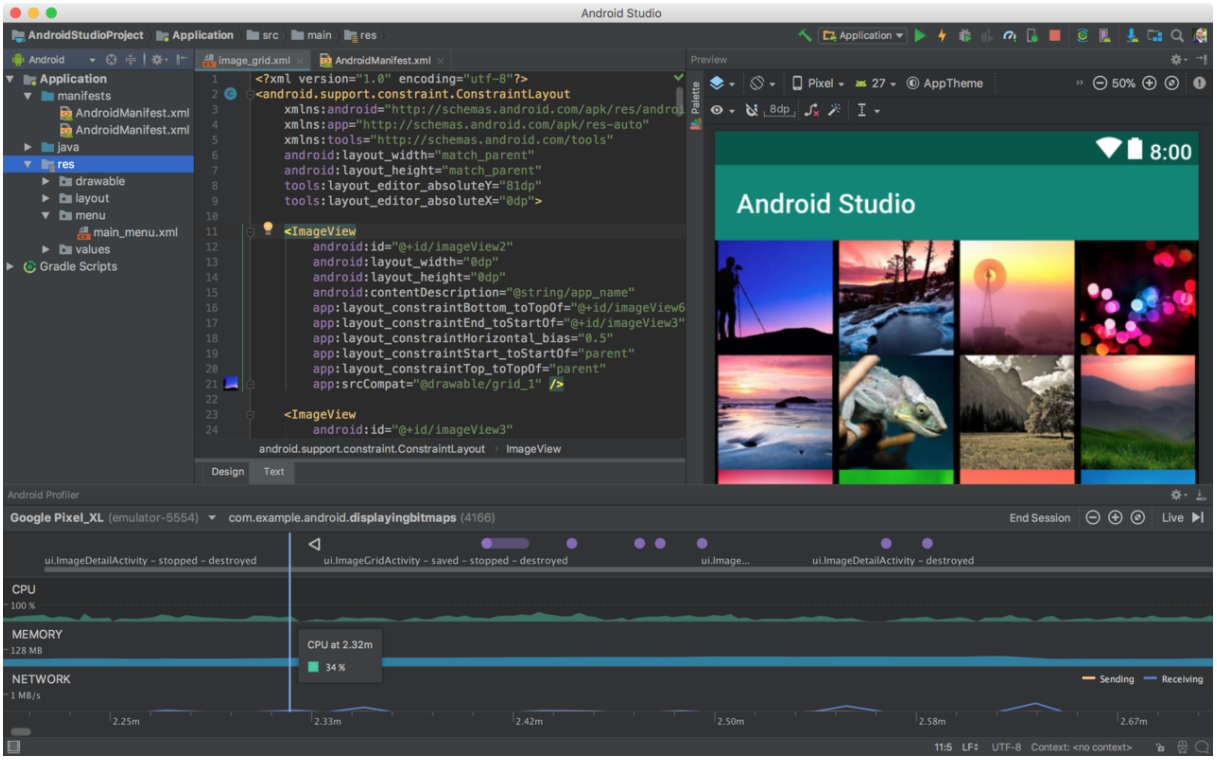

Ilustración 5.2: Interfaz de ***Android Studio***

5.2. Gestión de historias y usuarios

Para la autentificación de usuarios, almacenamiento de vídeos y base de datos en tiempo real y online se ha utilizado la plataforma *Firebase*. El logo de *Firebase* lo podemos encontrar en la siguiente ilustración 5.3. Además, podemos encontrar más información de la que se encuentra a continuación en su página web, que se encuentra en la referencia [10].

Ilustración 5.3: Logotipo de ***Firebase***

Procedamos a desglosar las diferentes subcategorías.

5.2.1. Gestión de usuarios

Firebase Auth es un servicio que autentica los usuarios utilizando código del lado del cliente. Presenta diferentes formas de autentificación, como pueden ser:

- ✔ Facebook
- ✔ Google
- ✔ Correo electrónico y contraseña
- ✔ GitHub
- ✔ Twitter
- ✔ Teléfono
- ✔ Play Juegos
- ✔ Yahoo
- ✔ Game Center
- ✔ Autentificación de forma anónima

5.2.2. Base de datos

Cloud Firestore es una base de datos NoSQL. Se organiza en forma de documentos agrupados en colecciones, y de ellos se pueden incluir campos de diversos tipos, incluidos objetos propios, en este caso por ejemplo se pueden insertar objetos del tipo «Historia».Un ejemplo de modelo de datos en *Firestore* es como el que se recoge en la Ilustración 5.4.

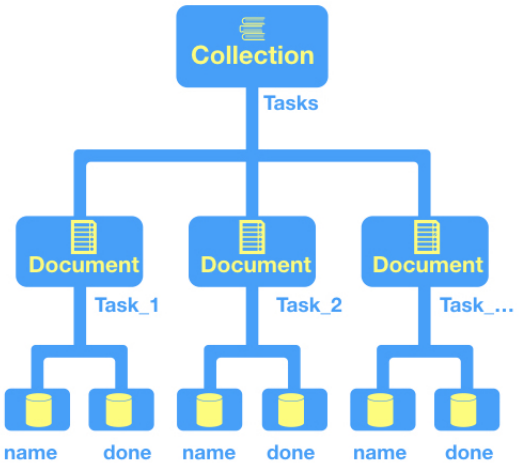

Ilustración 5.4: Modelo de datos de ***Firestore***

5.2.3. Almacenamiento de vídeo

Firebase Storage es un módulo que proporciona subida y descarga de archivos. Allí, se almacenan por tanto, los vídeos de cada usuario bajo la estructura de archivos:

«**/videos/IDUSUARIO/IDHISTORIA**»

Donde IDUSUARIO corresponde al identificador único de cada usuario, y IDHISTORIA el identificador de la historia, ya comentado anteriormente.

Con este módulo se asegura que las historias puedan ser subidas de forma éxitosa y sin problemas de seguridad. Además, se usa la plataforma para realizar «streaming» de los vídeos de las diferentes historias, optimizando el uso de datos móviles en caso de no estar conectado el dispositivo a una red WiFi.

5.3. Repositorio en la nube

Para guardar el código durante el desarrollo del mismo se ha optado por utilizar un repositorio en **GitHub**, de manera que se realizan varias confirmaciones en el momento en el que se han implementados diferentes funcionalidades, haciendo que no se pierda el código en caso de indisponibilidad del equipo de desarrollo y manteniendo un registro del estado del proyecto.

5.4. Herramientas organizativas

Un proyecto sin organización aboga al fracaso, por ello se ha optado usar una herramienta como **Trello**, que es una aplicación cuya utilidad reside en administrar un proyecto usando el sistema *kanban*. Un ejemplo de interfaz de Trello es como el que podemos encontrar en la Ilustración 5.5.

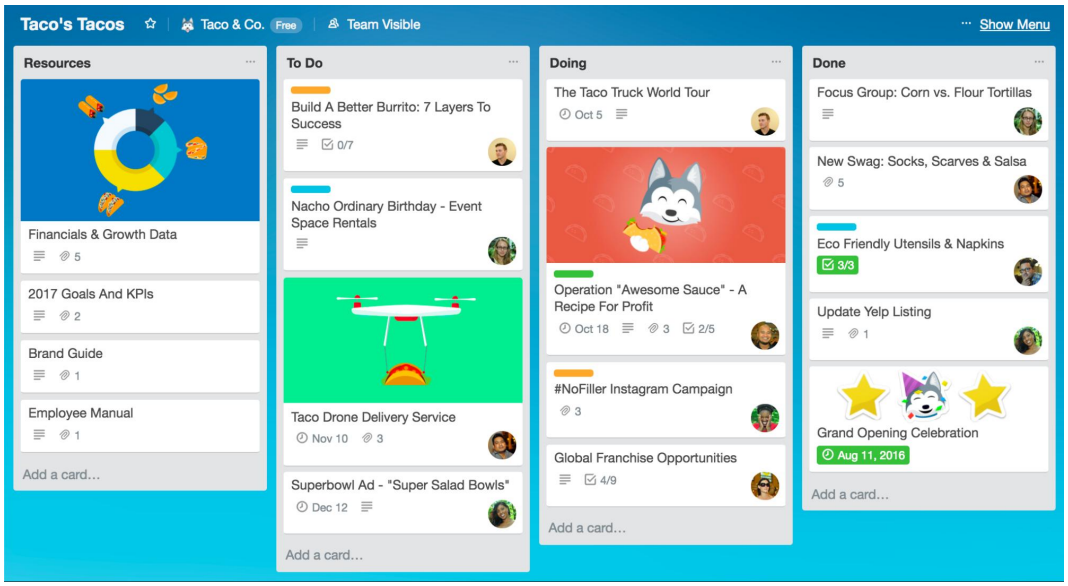

Ilustración 5.5: Ejemplo de interfaz de ***Trello***

5.5. Creación de *mockups*

Para la creación de maquetas iniciales, se ha usado la aplicación *Balsamiq*. Esta herramienta ha sido útil para el desarrollo inicial del diseño del proyecto, sobretodo con respecto a la realización de bocetos de la interfaz de usuario. De esta forma, tenemos una idea preliminar de lo que puede llegar a ser la aplicación, manteniendo un orden desde una fase temprana del ciclo de vida del proyecto. El logotipo de la aplicación y la compañía lo podemos encontrar en la siguiente Ilustración 5.6.

Ilustración 5.6: Logotipo de ***Balsamiq***

Podemos obtener más información de lo que es *Balsamiq* a través de las referencias [16] y [11].

Capítulo 6

Competencias específicas cubiertas

Las competencias cubiertas en este proyecto son las siguientes:

- ✔ *" IS01. Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software."*

  A través del uso de diferentes herramientas y servicios que se comentarán más adelante, se ha logrado desarrollar un producto software que satisface las necesidades básicas e incluso algunas más específicas que el cliente tiene. Además, el sistema presenta diferentes módulos y clases que hacen que el mismo sea fácil de mantener y desarrollar, haciendo uso de buenas prácticas de programación.

- ✔ *" IS03. Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles. "*

  Ya que se han necesitado diferentes tecnologías para dar vida al sistema informático aquí planteado, ha sido necesario realizar un esfuerzo en solucionar los diferentes problemas encontrados a lo largo del desarrollo del mismo, como por ejemplo, problemas de compatibilidad de librerías. De esta forma, al final se ha logrado integrar los diferentes módulos que presenta el sistema en uno sólo, haciendo que nos encontremos ante una aplicación flexible en términos de flexibilidad al añadir nuevos módulos.

- ✔ *" IS04. Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales."*

  Haciendo uso de los conocimientos adquiridos a lo largo del Grado en Ingeniería Informática y gracias a la experiencia adquirida durante el recorrido del mismo, ha sido posible observar e identificar diferentes problemas en el sistema, de forma que se plantean diferentes soluciones a los mismos en función de las características del inconveniente, de forma que se solvente de manera correcta.

- ✔ *“ IS05. Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.”*

  De la misma forma que en el punto anterior, se ha logrado registrar los diferentes riesgos que pueda presentar la aplicación y solventar, haciendo uso de las tecnologías que se usaron para desarrollar el sistema.

- ✔ *“ IS06. Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos.”*

  La aplicación planteada está asociada fuertemente a aspectos sociales, legales y éticos, como pueden ser, la integración con otras redes sociales, comunicación entre diferentes usuarios a través de las historias que encuentran entre ellos, etc. Además, para evitar que un usuario suba al sistema contenido que no es del agrado de otros usuarios, se ha insertado un sistema de reportes de la historia para así evitar historias que no son legales, éticas, o que simplemente son inadecuadas desde el punto de vista de los demás usuarios.

Capítulo 7

Desarrollo

La metodología de trabajo seguida durante el desarrollo del proyecto se puede establecer en diversas categorías, no sin antes comentar que existe una metodología general para todo el desarrollo. Las diversas categorías las podemos clasificar dependiendo del ciclo de vida en el que nos encontremos.

7.1. Metodología general

La metodología de trabajo que se siguió en términos generales fue la de reuniones semanales con los tutores para declarar un pequeño reporte del estado del proyecto: Avances, problemas encontrados, soluciones, dudas, etc. De forma que se establecía de esta manera un trabajo semanal que hacía que el sistema avanzara de manera gradual.

Cabe destacar que la metodología está basada en realizar un prototipo funcional de la aplicación, en el que se irán añadiendo funcionalidades a medida que se obtienen buenos resultados del proyecto.

Todo esto añadido a la reuniones semanales hacen que aparezca una metodología de trabajo en la que se permite al alumno trabajar sin sobrecarga de trabajo.

Podemos resumir la planificación del trabajo realizado en la tabla 7.1:

Fases	Duración Estimada (horas)	Tareas a realizar
Estudio previo y Análisis	60	Tarea 1.1.
		Tarea 1.2.
Diseño / Desarrollo / Implementación	140	Tarea 2.1.
		Tarea 2.2.
		Tarea 2.3.
		Tarea 2.4.
Evaluación / Validación / Prueba	60	Tarea 3.1.
		Tarea 3.2.
Documentación / Presentación	40	Tarea 4.1.
		Tarea 4.2.
		Tarea 4.3.

Cuadro 7.1: Planificación de trabajo durante el proyecto

A continuación se recoge la descripción en cada tarea.

- ✔ Tarea 1.1: Estudio de las tecnologías para creación de un servidor con contenido multimedia geolocalizado.
- ✔ Tarea 1.2: Familiarización con el entorno de desarrollo para plataformas móviles.
- ✔ Tarea 2.1: Captura de un repertorio de historias/anécdotas.
- ✔ Tarea 2.2: Desarrollo del servidor de contenido multimedia.
- ✔ Tarea 2.3: Integración de reproducción por geolocalización mediante GPS.
- ✔ Tarea 2.4: Desarrollo del prototipo de servidor.
- ✔ Tarea 3.1: Diseño de experimentos de validación.
- ✔ Tarea 3.2: Evaluación de resultados.
- ✔ Tarea 4.1: Redacción de la memoria del TFG.
- ✔ Tarea 4.2: Realización de la presentación.
- ✔ Tarea 4.3: : Ensayos presentación.

7.2. Estudio previo y Análisis

Es importante señalar que al tratarse de un sistema basado en dispositivos móviles, en este caso Android, ha sido necesario realizar una investigación propia de cómo darle uso correctamente las herramientas utilizadas, intentando conocer la existencia de otras que realicen un trabajo más eficiente. Por tanto, podemos dividir el desarrollo en esta sección en dos partes:

- ✔ **Estudio de tecnologías.** En el que es necesario conocer cuales pueden llegar a ser las herramientas necesarias para montar el sistema de forma correcta y sin encontrar problemas de compatibilidad entre ellas.
- ✔ **Familarización con el entorno.** En este caso, ha sido necesario incluso realizar pequeños ejemplos para entender correctamente el funcionamiento completo del entorno de trabajo utilizado. Al tratarse de un sistema basado en Java, existen muchos conceptos que se conocen porque se imparten a lo largo de la carrera, pero hay que tener en cuenta nuevos conceptos, como por ejemplo:

  - Aspectos de interfaz, que se basa en xml

  - Android lanza múltiples hilos de forma paralela, añadiendo mayor dificultad para realizar una secuencia de comandos.

  - Uso de permisos en el manifiesto del proyecto.

7.3. Diseño, Desarrollo e Implementación

En esta sección, se han focalizado los esfuerzos en diferentes apartados:

7.3.1. Diseño inicial de la aplicación

Se realizó un diseño inicial, basado en *mockups*, para tener una idea más o menos clara de lo que se pretendía obtener al final del desarrollo del proyecto. Estos *mockups* los podemos diferenciar dependiendo de los diferentes momentos en los que el usuario utilice nuestra aplicación.

7.3.1.1. Iniciar sesión

Una visión de lo que sería el inicio de sesión de un usuario sería el que se puede observar en la Ilustración 7.1 :

Ilustración 7.1: Diseño inicial al iniciar sesión en ***Tagoror***

Aunque si es cierto que el diseño inicial es similar a lo que podemos encontrar cuando un usuario inicia sesión a través de su correo electrónico y contraseña, cabe destacar que se aumentaron las funcionalidades relacionadas con el inicio de sesión con Facebook y Google.

7.3.1.2. Mapa principal

Con el desconocimiento aún de cómo sería el manejo de un mapa en Android, se intentó realizar una aproximación inicial de lo que podría ser el uso de dicho mapa y su relación con las historias. El diseño preliminar, que se puede ver en la Ilustración 7.2, muestra cómo se pensó desde el primer momento de que manera sería una de las *Actividades* más importantes del proyecto.

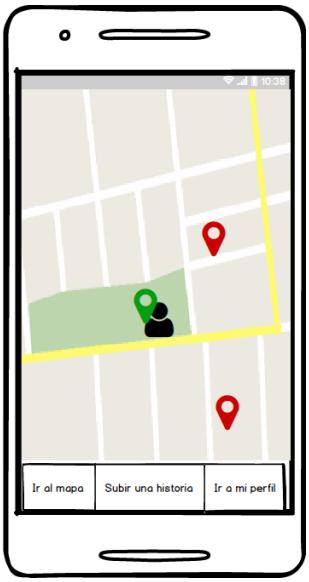

Ilustración 7.2: Diseño inicial al buscar historias en ***Tagoror***

Como bien se puede examinar en la Ilustración, existen pequeños cambios con respecto a la versión final, que se puede observar en la Ilustración 7.8. El cambio más destacado es el de que en el diseño preliminar no existen círculos que delimiten el rango de acción del usuario, haciendo confuso cuanto debe acercarse a una historia. Por ello, en la versión final se añadió el círculo exterior, de manera que el usuario conoce qué historias puede o no observar.

7.3.1.3. Ver una historia

Una vez pensado cómo será la interfaz al buscar historias, es necesario saber cómo se podrá ver la información de las mismas. Es por ello que se pensó en una idea preliminar antes de empezar el ciclo de desarrollo del proyecto. Esta idea la podemos ver en la Ilustración 7.3

Ilustración 7.3: Diseño inicial al ver información de una historia en ***Tagoror***

Si observamos la Ilustracion 7.9, podemos observar que no hay grandes cambios en el diseño del proyecto, sobre todo los podemos encontrar en el orden de los componentes y en el que el diseño inicial presenta un modal, el cual realmente no proporciona ninguna utilidad en el manejo de la aplicación por parte del usuario.

7.3.1.4. Ver el vídeo relacionado con una historia

Al pulsar en la historia, se puede ver el vídeo con el que la misma se encuentra relacionada. Esta idea ya se encontraba pensada desde un inicio, pero era necesario pensar de qué forma debería ser la interfaz de la aplicación para facilitar el trabajo de desarrollo. Un boceto inicial de cómo sería lo podemos encontrar en la siguiente Ilustración 7.4

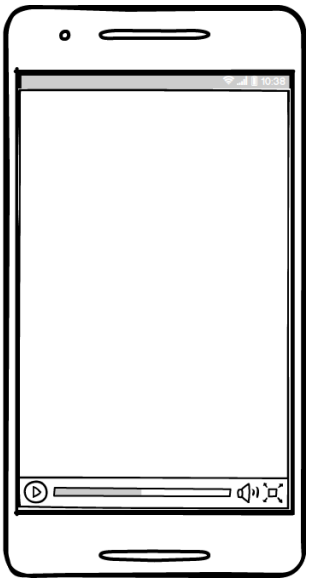

Ilustración 7.4: Diseño inicial al ver el vídeo de una historia en ***Tagoror***

En este apartado sí que se encontraron cambios en la versión inicial con respecto a la versión final, que se puede ver en la Ilustración 7.10. Dichos cambios se pueden encontrar sobre todo en que en la versión final existe una barra de menú en la cual encontramos la opción de *Reportar una historia.*

Por otro lado, se encuentran algunos cambios más relacionados con el controlador de vídeo: Se añadió el botón de *Me gusta* y *No me gusta* a dicho controlador. Esto fue algo que no se pensó en la fase inicial del proyecto. Pero que conforme se desarrollaba el proyecto y aumentaba el conocimiento del manejo de la base de datos y de las herramientas utilizadas en sí, se decidió que sería útil un sistema de *me gusta* para mayor interacción entre usuarios. Es más, en un inicio no se pensó en un sistema de reporte de una historia, de forma que se pensó que sería útil para el manejo de historias que contengan contenido inapropiado.

7.3.1.5. Subir una historia

Para subir una historia en su totalidad, se pensó cómo idea inicial crear un formulario en el cual se rellenen ciertos datos como podían ser:

- ✔ Un título y su descripción.
- ✔ Un punto en el mapa
- ✔ Un vídeo en el cual se encontrara el contenido de la historia.

Estos tres puntos fueron los que se consideraron en un inicio en la aplicación, y se mantendrían bajo el diseño preliminar que se puede observar en la Ilustración 7.5.

Ilustración 7.5: Diseño inicial al crear una historia en ***Tagoror***

Como se puede observar en la Ilustración 7.11, no existen grandes cambios. Los más importantes son aquellos en los cuales se elige el contenido del vídeo: O bien a través de la galería, o bien a través de la cámara del dispositivo. Cabe decir que se ha conseguido una aproximación muy precisa de lo que se pensaba en un principio con respecto al diseño de interfaz de usuario respecto a este apartado.

Por otro lado, es necesario señalar que se han añadido unos iconos de comprobación que informan al usuario de que los diferentes apartados para subir una historia se han rellenado correctamente o no, de forma que el usuario conoce en todo momento si se está realizando de forma correcta la creación de una historia.

7.3.1.6. Marcar una historia para poder crearla

En este caso se pensó que para poder marcar una historia en el mapa en el momento de querer completar el formulario de creación de la misma, sería más intuitivo si se utilizara un mapa en el cual el usuario determine la localización de la historia en cuestión. Esto sería así si se mantuviera pulsado en la pantalla dicho lugar, de forma que el diseño de la aplicación sería un mapa, como el que se puede observar en la Ilustración 7.6.

Ilustración 7.6: Diseño inicial al marcar una historia para poder crearla en ***Tagoror***

Si bien es cierto que con esto ya se podría señalar la localización de la historia en el mapa, se hace pesado el tener que encontrar "a mano" la localización de la historia. Es por ello, que se pensó en añadir una barra de búsqueda en la que se pudiera buscar una ubicación en concreto y se insertó un botón en el cual aparece un diálogo en el que se puede realizar lo siguiente:

- ✔ Por un lado, se puede acceder directamente a la ubicación actual.
- ✔ Por otro, se puede marcar la historia directamente en la ubicación del usuario.

Como se puede ver en la Ilustración 7.13, se ha añadido lo anterior al mapa, añadiendo valor al sistema para el manejo del usuario con la aplicación.

7.3.1.7. Ver el perfil de usuario

.

Como último apartado en el diseño del proyecto, se esbozó la funcionalidad de ver el perfil de un usuario. Para ello, bastaría con entrar en dicho apartado y se podrían recoger diferentes datos del usuario, entre ellos:

- ✔ El correo electrónico del usuario
- ✔ El nombre del usuario (si hubiera)
- ✔ Las historias creadas

En la Ilustración 7.7 se observa una maqueta de lo que podría ser el apartado de perfil del usuario

Ilustración 7.7: Diseño inicial al ver el perfil de un usuario en ***Tagoror***

Sin embargo, como se puede observar en la Ilustración 7.16, se han añadido diferentes cambios en este apartado. Por ejemplo, la funcionalidad relacionada con poder ver el contenido creado de un usuario se ha recogido en otra actividad diferente y, además, no sólo se pueden ver las historias propias, sino aquellas ya vistas.

Siguiendo la misma línea comparativa, en el diseño final, se puede observar todas las historias vistas y las creadas, dando la posibilidad al usuario de poder repetir una que ya vió sin tener que acercarse de nuevo.

7.3.2. Integración de geolocalización mediante GPS.

Ilustración 7.8: Mapa principal en ***Tagoror***

Como preparación, para poder realizar correctamente este apartado, cabe señalar que hay que declarar en el manifiesto de la aplicación que se va a requerir al usuario los permisos de ubicación, en Android existen dos tipos permisos de ubicación: permisos de ubicación de baja precisión y los de alta precisión. En el primer caso son aquellos en las que la ubicación del dispositivo la aportan los datos móviles y/o la red WiFi del dispositivo y en el segundo caso, añadido a lo anterior, se le permite al sistema determinar lo más preciso posible la localización del dispositivo a través de los proveedores de localización, incluyendo GPS. En el proyecto, se ha optado por utilizar los dos tipos de permisos en conjunto para precisar lo máximo posible la geolocalización del dispositivo Android.

Una vez, gestionados los permisos y haciendo uso de las diferentes API de Google «Maps SDK for Android», «Geocoding API», «Geolocation API» y «Places API», y utilizando el sistema de geolocalización de los dispositivos Android, se ha creado una interfaz en la que el elemento principal es un mapa similar a «Google Maps» en la que podemos ver los diferentes lugares en los que se guarda una historia. Cabe destacar que para poder usar estas APIs, se ha requerido utilizar una contraseña *hash* para identificar la aplicación en los servidores de Google, dicha contraseña se encuentra localizada en el manifiesto de la aplicación, por razones de seguridad, oculta para los usuarios. En la imagen 7.8, se puede ver una captura de pantalla de la página en la cual se encuentra el mapa donde se observan las diferentes historias. Caben destacar varios aspectos:

- ✔ En relación al usuario se muestran dos círculos: uno más pequeño que es el punto exacto donde se encuentra, y otro más grande, que es el rango de acción del usuario. El rango de acción se refiere aquella distancia máxima a la cual se puede visualizar una historia.
- ✔ En relación a la historia, se marcan en tres dependiendo de tres posibles casos:

- La historia no ha sido vista y no está en el rango de acción. Se marcará en color rojo y si el usuario intenta visualizar la historia, se avisará al usuario de que no se puede visulizar.

- La historia no ha sido vista, pero está en el rango de acción. Se marcará en color azul y el usuario podrá visulizar la historia si pulsa en ella.

- La historia ha sido vista, independientemente del rango de acción. Se marcará en color verde y se añadirá a la lista de historias vista en el apartado del perfil del usuario.

Los círculos, tanto el interno como el externo se han creado personalmente para crear una interfaz más amigable con el usuario. De esta forma el usuario sabe en qué momento se encuentra y la distancia de su rango de acción, que está establecida a 500 metros a la redonda del usuario, fácilmente modificable ya que se establece en una variable.

Para saber si una historia se encuentra, o no en su rango de acción se ha utilizado el método «distanceBetween» que se encuentra en la clase que proporciona Android, *Location*, que indica la distancia a la que se encuentran dos puntos en el mapa, integrándolo así en un método que se ha establecido para que devuelva verdadero o falso en función de si se encuentra o no en el rango.

Si un usuario pulsa sobre una historia, se verá algunas características, como se puede observar en la imagen 7.9.

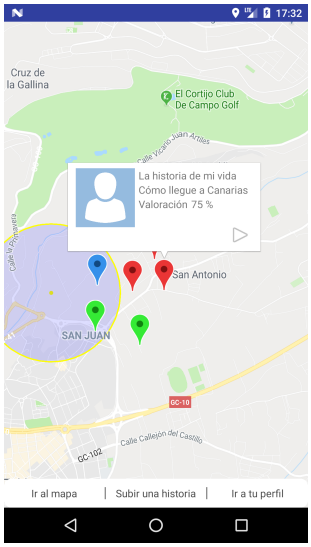

Ilustración 7.9: Usuario pulsando una historia en ***Tagoror***

En este caso, se observan la foto de perfil del usuario, el título, una descripción y un porcentaje fruto de la relación de «Me gusta» y «No me gusta» en la historia en cuestión. Este nuevo componente es un InfoWindow que se ha personalizado en función de las necesidades del proyecto para mostrar un resumen rápido de la historia pulsada.

Al volver a pulsar en la historia, desembocaremos en un reproductor de vídeo mostrando el vídeo fruto del usuario que ha subido el contenido al servidor, que se comentarán en los

siguientes apartados. Esto hace que se pase de una actividad hacia otra, que es básicamente, movernos a una nueva interfaz en la que aparece un nuevo abanico de acciones a tomar, en este caso, usar un reproductor de vídeo. Para hacer que funcione correctamente, ha sido necesario que entre actividades se pasen información entre ellas, sobre todo el enlace al vídeo que va a ser visualizado.

7.3.3. Gestión de usuarios

Se ha utilizado *FirebaseAuth*, que ya se comentó en el siguiente apartado 5.2.1. En este proyecto, se utiliza *FirebaseAuth* para dar opción a elegir tres formas de inicio de sesión: Google, Facebook y por una cuenta de correo y contraseña proporcionados. Para este último será necesario registrarse en la aplicación a través de un formulario insertando el correo y la contraseña de forma correcta. En esta última opción, se ha manejado la funcionalidad de que se pueda enviar un correo al usuario para validar su correo electrónico. En las demás opciones, simplemente será requerido tener una cuenta de Google o Facebook válidas.

7.3.4. Creación de un repositorio dónde se guarden historias.

Para ello, se ha requerido utilizar *Firebase*, como ya se explicó en el apartado 5.2.2. Se eligió *Firebase* ya que es una plataforma que proporciona la gestión de una base de datos en la nube sin añadirle demasiada complejidad a la lógica del sistema. Dentro de *Firestore* dividimos la base de datos en dos colecciones:

- ✔ Por un lado, tenemos la colección «History», que guarda, como su propio nombre dice, las historias de los usuarios. Allí encontraremos diferentes documentos, que tendrán el identificador único con el siguiente formato: «m» continuado del número de la historia en cuestión, por ejemplo: «m0». En cada documento se podrán observar campos como el título, descripción, posición de la historia, URL donde se almacena el vídeo relacionado, etc., que tiene cada historia.
- ✔ Por otro lado, tenemos la colección «User», que guarda todo lo relacionado con los usuarios, en concreto, las historias que han creado y aquellas que han llegado a ver. Dentro de cada documento encontraremos una lista con los identificadores únicos de cada historia, de manera que se evita la duplicación de datos.

Así, creamos una base de datos en la nube que actualiza los datos en tiempo real de forma fácil y comoda sin recurrir en costes de rendimiento que afecten a la experiencia del usuario final.

En el algoritmo 7.1, podemos ver la manera en la que se sube una historia a la base de datos. Para ello, se hace uso del método *addMarkers*, en la que, como se puede ver, lo primero de todo es hacer uso de la colección *History*, seguido del documento con nombre del identificador del marcador asociada a la nueva historia y por último, se añade dicha historia al documento.

```
        private FirebaseFirestore firestore;
        [...]
        firestore = FirebaseFirestore.getInstance();
        [...]
        public void addMarkers(History history) {
        firestore.collection("History").document(history.getMarker().getId())
                .set(history).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                Log.i("addMarkers",getResources().getString(R.string.successive));
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Log.i("addMarkers",getResources().getString(R.string.failure));
            }
        });
    }
```

Algoritmo 7.1: Método addMarkers

7.3.5. Desarrollo del servidor de contenido multimedia.

En este apartado fue necesario utilizar *FireStorage*, un módulo de *Firebase*, que ya se comentó en el apartado 5.2.3. Fue muy importante conocer cómo funciona correctamente esta herramienta, ya que, proporciona servicio de *streaming* del contenido multimedia que se encuentre dentro. Para hacer un correcto uso del sistema, se enlazaron los módules de *Firebase Storage* y *Cloud Firestore*, de forma que en este último se guarda una referencia al contenido multimedia, y no se almacena el contenido en su totalidad. Esto hace que el rendimiento en cuanto a lectura y escritura en base de datos aumente. Además, al utilizar un servicio de *streaming* de los vídeos, disminuye el uso de datos móviles de los diferentes dispositivos. En la imagen 7.10, se observa qué es lo que encuentra el usuario al ver una historia, pudiendo darle a «Me gusta» o «No me gusta» en función de los gustos del usuario.

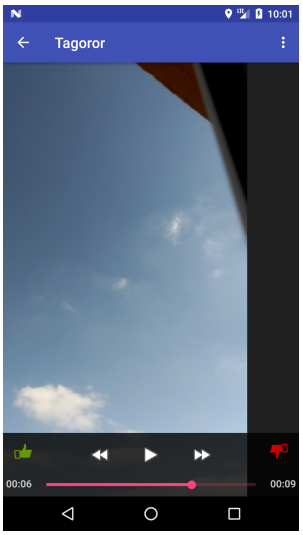

Ilustración 7.10: Usuario observando una historia en ***Tagoror***

Además, para evitar historias de contenido inadecuado, se ha añadido la posibilidad de «Reportar una historia» que se puede encontrar en el apartado de opciones en la misma interfaz. Una historia se marca como reportada cuando llega a cierto número de reportes por parte de diferentes usuarios. En este caso, la historia se deshabilita para que otros no puedan verla y se envían dos mensajes, uno para el administrador del sistema, y otro para el creador de la historia, avisando a ambos de que se ha reportado la historia. El rol de administrador se encargará de determinar si la historia es o no válida para los demás usuarios.

En el algoritmo 7.2 podemos ver la forma en la que se añade un reporte a una historia en concreto, si el tamaño de la lista es mayor o igual a diez, dicha historia se marcará como no visible y se mandará un correo al administrador de la aplicación y al correo del creador del contenido.

```
    public void updateReportList(History history) {
    DocumentReference noteRef = firestore.collection("History").document(history.getMarker().getId());
    noteRef.update("reportList",history.getReportList()).addOnSuccessListener(
        new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            Log.i("updateReportList","Success␣"+ history.getMarker().getId());
            if(history.getReportList().size() >= 10){
                history.setVisible(false);
                updateVisibleHistory(history, false);
            }
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Log.i("updateReportList","Fail"+ history.getMarker().getId());
        }
    });
}
```

Algoritmo 7.2: Método updateReportList

A continuación en el algoritmo 7.3, si la historia ha alcanzado el número de reportes necesarios, esta se marca como no visible para los demás usuarios, y una vez terminado este proceso se manda los correos pertinentes ya comentados anteriormente.

```
private void updateVisibleHistory(History history, boolean visible) {
    DocumentReference noteRef = firestore.collection("History").document(history.getMarker().getId());
    noteRef.update("visible",visible).addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            Log.i("updateVisibleHistory","Success"+ history.getMarker().getId());
            videoActivity.reportHistory(history);
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Log.i("updateVisibleHistory","Fail"+ history.getMarker().getId());
        }
    });
}
```

Algoritmo 7.3: Método updateVisibleHistory

7.3.6. Desarrollo del prototipo del servidor

Para poder enviar una historia, se hará uso de un formulario personalizado, que presentará una interfaz similar a la imagen 7.11. Cabe decir que aparecerán iconos de comprobación para hacer que el usuario sepa en qué momento ha cumplimentado cada apartado, haciendo que la interfaz sea más intuitiva para el cliente. Esta se divide en cuatro apartados.

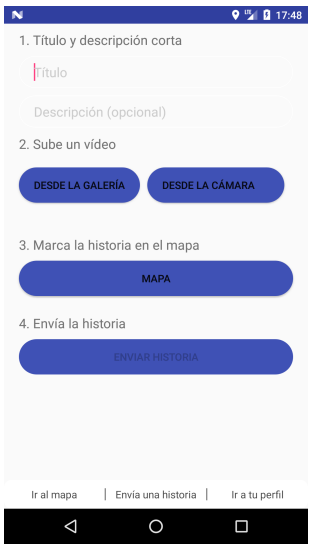

Ilustración 7.11: Formulario inicial para subir una historia en ***Tagoror***

- ✔ En el primer apartado se escribirá un título y descripción que defina la historia en cuestión

✔ En el segundo apartado, se eligirá el vídeo a subir. Se puede realizar de dos formas:

- Desde la galería. Se eligirá la aplicación a usar y desde la memoria del teléfono se seleccionará un vídeo.

- Desde la cámara. Se usará la cámara del dispositivo para grabar un vídeo y poder usarlo para subir la historia al servidor.

✔ En la tercera sección, se marca la historia en el mapa. Existen tres posibles acciones que se pueden realizar en este apartado, que se pueden observar en la imagen 7.12:

- Seleccionar libremente el lugar donde se quiere depositar la historia. Para ello se puede usar el buscador que se encuentra en la parte superior para acceder rápidamente a diferentes ubicaciones que el usuario quiera ir. Para marcarlo, será necesario mantener pulsada la pantalla unos segundos en el lugar el cual quiera insertar la historia.

- Usar el botón ubicado en la esquina superior derecha para marcar directamente la historia en el lugar en el que el usuario se encuentra actualmente

- También se puede, utilizando el botón comentado anteriormente, acercar la vista del mapa a la posición actual del usuario.

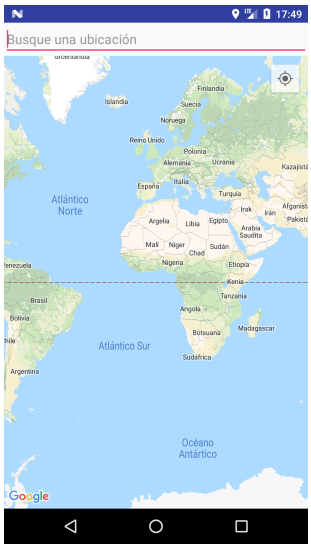

Ilustración 7.12: Interfaz inicial al marcar el lugar de la historia en ***Tagoror***

✔ Para que se pueda utilizar el último apartado, es necesario haber rellenado correctamente todos los campos anteriores, de forma que se habilitará el botón para poder subir una historia. Un ejemplo de interfaz en la que el usuario ha validado todos los campos y está listo para subir una historia lo podemos encontrar en la imagen 7.13.

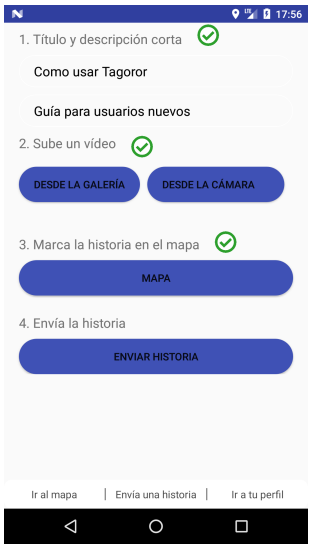

Ilustración 7.13: Interfaz inicial al marcar el lugar de la historia en ***Tagoror***

De forma que sólo será necesario pulsar en el botón para que la historia se suba al servidor. La lógica interna recogerá por tanto, el título y descripción escritos en el formulario, el vídeo seleccionado y la posición geográfica de la historia creada. Con estos datos, se utiliza *FireCloud* para subir el vídeo a la nube y almacenar los datos necesarios en *FireStore*. Además, se añadieron componentes que notifican al usuario de que la historia se ha subido al servidor y es entonces cuando los usuarios podrán ver la historia que se ha creado por parte del cliente. Un extracto del código como es el Algoritmo 7.4 muestra cómo se sube un vídeo a *FireStore*, y cómo se enlaza a la historia.

```
private void uploadFile(Uri file) {
    StorageReference mStorageReference
        = FirebaseStorage.getInstance().getReference();
    //Guardamos el contenido en FireStore en formato ".mp4".
    StorageReference fileRef = mStorageReference
        .child("videos/"+currentUser.getUid()+"/"+history.getMarker().getId()+".mp4");
    fileRef.putFile(file).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            fileRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                @Override
                public void onSuccess(Uri uri) {
                    //Se enlaza el campo de ruta de una historia con
                    //la Uri del contenido subido a FireCloud.
                    history.setPath(uri.toString());
                    firebaseMarkers.addMarkers(history);
                    Toast.makeText(getApplicationContext(),
                        getResources().getString(R.string.success), Toast.LENGTH_SHORT).show();
                    alertDialog.dismiss();
                    //Agregamos la nueva historia a la lista del usuario.
                    firebaseUsers.addHistoryCreatedToUser(history);
                }
            });

        }
```

```
        });
    }
```

Algoritmo 7.4: Método uploadFile

7.3.7. Estructuración de clases

Para la correcta realización de todas las funcionalidades realizadas a lo largo de la fase de desarrollo, se ha decidido estructurar el proyecto en diferentes componentes, diviendo por un lado las actividades y por otro lado las clases relacionadas con el manejo de la aplicación.

7.3.7.1. Clases utilizadas en el proyecto

Almacenamos las diferentes clases en la carpeta denomida "*Classes*", para que de esta forma obtener una buena organización de los ficheros java. La descripción de las diferentes clases es:

- ✔ **CustomMarker.** Es la clase que se ha creado de forma personalizada para que sea capaz de almacenar los diferentes datos de un marcador de la Api de Google Maps: latitud, longitud,etc. A la hora de almacenar un *Custom Marker* en *FireStore*, primero será necesario enlazar dicho *Custom Marker* a la clase *History*.
- ✔ **FirebaseMarkers.** Es la clase que se encarga de almacenar tanto las historias como los marcadores asociados a las mismas. Para ello, primero será necesario enlazar esta clase con una instancia de la clase ya creada por *Firebase*, *FirebaseFirestore*. El algoritmo 7.5 es uno de los métodos más importantes en esta clase, *readMarkersFromDB()*, con ella se obtienen sólo las historias que se encuentren dentro de la porción del mapa visible por el usuario, y las coloca en el mismo, haciendo que el rendimiento de la aplicación sea mayor:

```
public void readMarkersFromDB(){
    LatLngBounds bounds = mMap.getProjection().getVisibleRegion().latLngBounds;
    //Recogemos la parte visible del mapa.
    for (Marker marker: markerList) {
        //Iteramos en la lista de marcadores que obtenemos de base de datos.
        if(!bounds.contains(marker.getPosition()))
            //Si la historia no se encuentra en la parte visible del mapa
            marker.remove();
            //Borramos el marcador del mapa.
    }
    this.customMarkerList.clear();
    this.markerList.clear();
    firestore.collection("History")
            .get()
            .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
                @Override
                public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                    if(!queryDocumentSnapshots.isEmpty()){
                        List<DocumentSnapshot> list = queryDocumentSnapshots.getDocuments();
```

```
                        for(DocumentSnapshot d: list){
                            History history = d.toObject(History.class);
                            if(history.isVisible()){
                                if(bounds.contains(new LatLng(history.getMarker().getLatitude(),
                                        history.getMarker().getLongitude()))){
                                    customMarkerList.add(history);
                                    Marker marker = mMap.addMarker(new MarkerOptions()
                                            .position(history.getMarker().getPosition())
                                            .title(resources.getString(R.string.marker)));
                                    marker.setTag(history.getMarker().getId());
                                    markerList.add(marker);
                                }
                            }
                        }
                        mainActivity.iteratingMarkers();
                    }
                }
    });
}
```

Algoritmo 7.5: Método readMarkersFromDB

- ✔ **FirebaseUsers.** Esta clase la usamos para, entre otros, relacionar una historia creada por parte de un usuario con la base de datos, de manera que al añadir una historia a *FireStore* se hará uso de esta clase.
- ✔ **History.** Es la clase en la que se almacenan todos los datos que son necesarias para lo que entedemos por historia: título, descripción, valoración, usuario que la ha creado, su marcador asociado, si es visible o no..., etc.
- ✔ **HistoryCreateList.** Es una clase auxiliar que nos sirve para almacenar para cierto usuario, la lista de *History* creadas por el mismo.
- ✔ **HistorySeenList.** Análogamente a la clase anterior, esta sirve para almacenar la lista de *History* vistas por un usuario.
- ✔ **MailJob.** Esta clase es utilizada para enviar notificaciones vía correo electrónico a, por un lado al usuario administrador adjuntando un mensaje de que cierta historia ha sido reportada, y por otro lado al usuario dueño de la historia reportada comentando el estado de la misma.
- ✔ **VideoController**. Esta clase se encarga de manejar la interfaz de usuario relacionada con el vídeo asociada a una historia: dar a *Me gusta*, *No me gusta*..., etc.

7.3.7.2. Actividades

Diferenciamos las diferentes actividades en función de las necesidades de interacción con el usuario, de esta forma, podemos encontrar múltiples actividades para nuestra aplicación.

- ✔ **FormHistoryActivity**. Es la actividad en la cual encontramos el formulario relativo a la posibilidad de subir una historia por parte del usuario. La interfaz de esta actividad la podemos encontrar en la Ilustración 7.11.

✔ **HistorySeenActivity**. En esta actividad, encontramos las historias que ha visto el usuario. Podemos acceder a ella desde la actividad *ProfileActivity*. Aquí, el usuario es puede ver tanto las historias creadas como las vistas y reproducir su contenido. En la Ilustración 7.14 podemos observar cómo se ven las historias subidas y las ya vistas.

Ilustración 7.14: Interfaz de ***HistorySeenActivity***

✔ **LoginActivity**. La actividad por defecto de la aplicación. En ella, podemos iniciar sesión con el usuario por Facebook, Google o vía correo electrónico y contraseña, o registrarse en la base de datos del sistema, todo ello posible gracias a los botones pertinentes que llevarán a distintas actividades o métodos. Observamos cómo es esta actividad en la Ilustración 7.15.

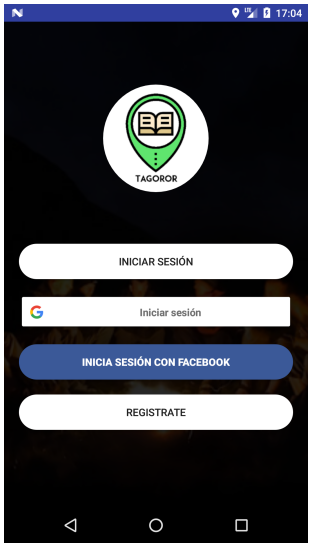

Ilustración 7.15: Interfaz de ***LoginActivity***

- ✔ **MainActivity.** La actividad en la cual aparece el mapa principal y es donde se pueden ver las historias depositadas por otros usuarios. Aquí es donde se maneja todo lo relacionado con la geolocalización del usuario, círculos de posición, marcadores y sus respectivas historias, etc. En la Ilustración 7.8 se observa cómo es la interfaz de la actividad en cuestión.
- ✔ **MapActivity.** Esta actividad es accesible desde *FormHistoryActivity* y es necesaria para que el usuario pueda marcar la posición de la historia a crear. En ella encontramos botones y una barra de búsqueda que simplifican la tarea de buscar una localización en concreto. Un ejemplo de cómo es la interfaz de la actividad es el que podemos ver en la Ilustración 7.13.
- ✔ **ProfileActivity.** En ella podemos ver el perfil de un usuario. Se encuentran componentes como la foto de perfil del usuario, correo electrónico, o nombre, etc. Además, en el apartado de opciones se puede examinar las historias que dicho usuario ha visto. Podemos observar cómo es la interfaz en la Ilustración 7.16.

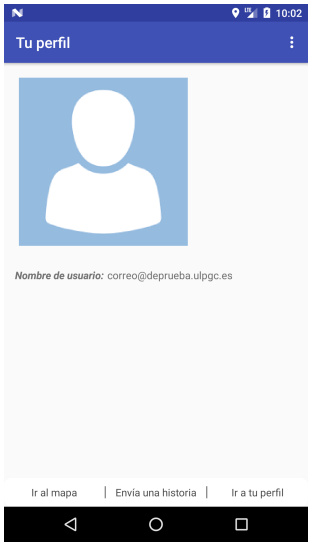

Ilustración 7.16: Interfaz de ***ProfileActivity***

En esta actividad se encuentra una barra de menú desde la cual se puede acceder a *HistorySeenActivity* y se puede cerrar la sesión de usuario.

- ✔ **RegisterActivity.** Maneja todo lo relacionado con el registro de un usuario al sistema. Presenta un formulario con varios campos que recopilan datos como un correo electrónico y una contraseña. Un usuario observaría esta interfaz si accediera a esta actividad.

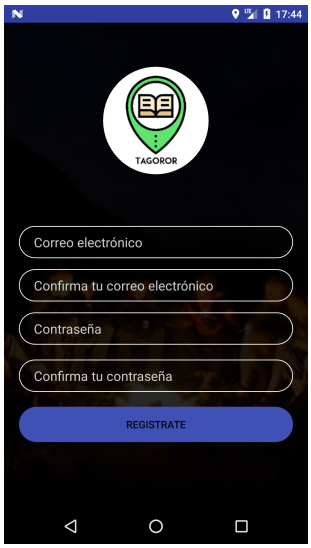

Ilustración 7.17: Interfaz de ***RegisterActivity***

En el algoritmo 7.6 se puede observar la manera en la que un usuario se registra en el sistema. Para ello, simplemente se recogen los valores obtenidos en los campos de correo electrónico y contraseña, ya confirmados que se escribieron de forma correcta, y se realiza el registro en el sistema. Además, se enviará un correo de verificación al usuario de manera que se valide su cuenta de usuario en el sistema.

```
    private void attemptRegister() {
    mAuth.createUserWithEmailAndPassword(
        mEmailView.getText().toString(), mPasswordView.getText().toString())
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        FirebaseUser user = mAuth.getCurrentUser();
                        user.sendEmailVerification();
                        Toast.makeText(getApplicationContext(),
                                getResources().getString(
                                    com.ulpgc.gustavo.tagoror.R.string.info_register),
                                Toast.LENGTH_SHORT).show();
                    }
                }
            }).addOnFailureListener(this, new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(getApplicationContext(),
                    getResources().getString(
                        com.ulpgc.gustavo.tagoror.R.string.info_already_register),
                    Toast.LENGTH_SHORT).show();
        }
    });
}
```

Algoritmo 7.6: Método attemptRegister

- ✔ **SignInActivity.** La actividad que recoge la interfaz por la que el usuario se mueve cuando quiere iniciar sesión con un correo electrónico y contraseña. Recoge un pequeño formulario pidiendo los datos correspondientes y un botón en el cual se puede reestablecer la contraseña del usuario. Todo ello se recoge, como se puede ver en la Ilustración 7.18.

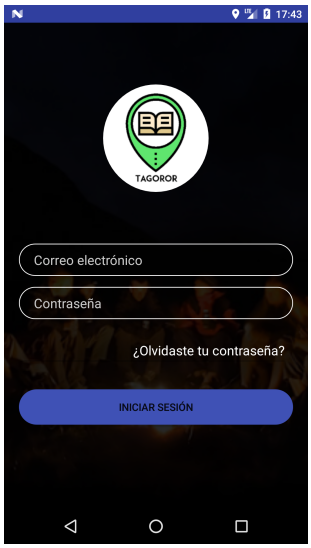

Ilustración 7.18: Interfaz de ***SignInActivity***

- ✔ **VideoActivity.** La actividad en la cual se encuntra la interfaz relacionada con el vídeo asociado a una historia. Accedemos a esta actividad cuando pulsamos una historia que se encuentra en el mapa principal y esté en el rango de acción del usuario. La interfaz asociada a esta actividad la podemos ver en la Ilustración 7.10.

Con este desarrollo y estructuración de clases, nace un sistema en la que el usuario es capaz de subir una historia de forma sencilla e intuitiva, además de poder ver el contenido que otros usuarios han creado.

7.4. Evaluación, Validación y Prueba

Para la prueba de las diferentes funcionalidades nos hemos basado en realizarlas a la par que se iban creando, para más tarde realizar una prueba integral que compruebe que los diferentes módulos siguen funcionando. Para ello, se ha debido probar todos los componentes a la vez, ya que no se pueden hacer pruebas unitarias o de integración por separado ya que se trata de un sistema móvil, en el que las funcionalidades van funcionando en relación con el dispositivo. Caben destacar otros aspectos en cuanto al rendimiento de la aplicación.

7.4.1. Subir una historia completa al servidor

Se han realizado pruebas de rendimiento sobre todo al realizar la subida de diferentes vídeos, en relación con su tamaño y el tiempo que ha tardado en subir un vídeo. Los resultados se pueden observar en la imagen 7.19:

Ilustración 7.19: Comparativa de tiempo en función del tamaño al subir un vídeo en ***Tagoror***

Estos datos están cronometrados a partir de que el usuario pulsa el botón para subir una historia. Como se puede comprobar, el tiempo medio de subida aumenta conforme el tamaño del vídeo es mayor, aspecto que parece lógico ya que debe procesar un mayor tamaño de bytes para poder subir sin fallos el vídeo que viene enlazado a la historia. Por tanto, podemos considerar que los resultados de rendimiento para subir una historia presentan concordancia a la hora de subir una historia al servidor.

7.4.2. Consumo de datos móviles y batería del dispositivo

Es importante conocer que se han creado métodos que disminuyen el gasto de estos factores, aunque no existan pruebas que verifiquen cuantitativamente estas mejoras, se pueden deducir estas optimizaciones gracias a las mejoras que se han introducido de forma iterativa en la aplicación.

- ✔ **En función del gasto de batería.** Se ha realizado métodos que hacen que reduzcan el gasto de batería, como por ejemplo aquellos métodos que hacen que si la aplicación no esté en primer plano, deje de hacer operaciones, o funciones que muestren las historias que se encuentran en el área que el usuario ve en el mapa, de forma que no aparecen más historias que las necesarias.
- ✔ **En función del uso de datos móviles.** En un primer momento se pensó en realizar el servicio de visualización de una historia descargando el vídeo y reproducir dicha descarga. Esto hacía que aumentara el uso de datos móviles alrededor de un 50 por ciento. Visto el gasto excesivo de datos móviles, se optimizó el sistema para que se reproduciese a través del servicio de *streaming* que ofrece *FireCloud* y presentar al usuario una aplicación óptima para su uso.

Capítulo 8

Normativa y legislación

8.1. Ley de protección de Datos y Garantía de Derechos Digitales

En estos tiempos, donde la información de una misma persona fluctua en diferentes puntos de internet, es necesario conocer ésta ley en proyectos software. En este caso, observamos cómo nos afecta a nuestro sistema, y si cumple con la legislación actual. Como se recoge en el Boletín Oficial del Estado, en el apartado de Protección de Datos de Carácter Personal, siendo la edición actualizada a 19 de marzo de 2019:

"La protección de las personas físicas en relación con el tratamiento de datos personales es un derecho fundamental protegido por el artículo 18.4 de la Constitución española"

Por tanto, tenemos que tener en cuenta que la protección de identidad de las personas físicas es necesaria para nuestra aplicación. No podemos tener ninguna forma de conocer qué persona utiliza nuestra aplicación, o los datos de la misma, ni siquiera (y esto es importante para nuestro sistema) en qué momento se haya la persona en sí. Es por ello que Tagoror no utiliza ningún sistema de rastreo de usuarios, o algún tipo de histórico por el cual haya visto diversas historias. Además, el sistema de geolocalización es usado únicamente en el dispositivo del usuario, por lo que no se puede saber en qué lugar se encuentra el mismo.

Hay que tener en cuenta otros aspectos como la información que identifique al usuario en sí. Como se recoge en el *Artículo 11. Transparencia e información al afectado.*:

" *1. Cuando los datos personales sean obtenidos del afectado el responsable del tratamiento podrá dar cumplimiento al deber de información establecido en el artículo 13 del Reglamento (UE) 2016/679 facilitando al afectado la información básica a la que se refiere el apartado siguiente e indicándole una dirección electrónica u otro medio que permita acceder de forma sencilla e inmediata a la restante información.*

2. La información básica a la que se refiere el apartado anterior deberá contener, al menos:

a) La identidad del responsable del tratamiento y de su representante, en su caso.

b) La finalidad del tratamiento.

c) La posibilidad de ejercer los derechos establecidos en los artículos 15 a 22 del Reglamento (UE) 2016/679.

Si los datos obtenidos del afectado fueran a ser tratados para la elaboración de perfiles, la información básica comprenderá asimismo esta circunstancia. En este caso, el afectado deberá ser informado de su derecho a oponerse a la adopción de decisiones individuales. "Si entendemos bien estos puntos, podemos observar que ésta ley puede afectarnos en la gestión de un usuario, sobretodo al ofrecer una cuenta de correo, o una cuenta de Facebook o Google. Para entrar en mayor detalle, desglosamos estos aspectos en varios grupos:

- ✔ **En función de los usuarios que inicien a través del correo electrónico.** Debemos de tener en cuenta que el sistema cumple con la legislación ya que sólo se ofrece un correo electrónico, que no reune ningún tipo de información básica en base de datos, luego no se identifica al usuario de ninguna manera. De esta forma, no es necesario preguntar al cliente un permiso por tratar datos que sean sensibles para el usuario.
- ✔ **En función de servicios de terceros.** Como bien se pudo ver, la legislación habla de que si se trata de datos que identifiquen al usuario será necesario pedir al usuario permiso para el uso de estos. Sin embargo, estos datos son ofrecidos por terceros, como Google o Facebook, aplicaciones en el que ya se dió permiso para el uso de estos datos, de los cuales son los terceros aquellos que se encargan de gestionar la información que pueden llegar a ofrecer, haciendo que no sea necesario por parte del sistema pedir datos al usuario.

De esta forma, esta ley de protección de datos queda cumplimentada para los usuarios que utilicen la aplicación, de tal manera que sus datos quedan o bien ocultos para el resto de usuarios, o bien son gestionados por aplicaciones de terceros, de los cuales el sistema no se hace cargo de la administración de los datos.

8.2. Artículo 18 de la Constitución

Como se recoge en el artículo 18, apartado 1, de la *Constitución Española*:

" *1. Se garantiza el derecho al honor, a la intimidad personal y familiar y a la propia imagen.*

[...]

4. La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos. "

Con esto, debemos asegurar que la privacidad de los usuarios quede asegurada por nuestra parte. Ya que la aplicación no requiere o recoge información sensible del usuario, podemos confirmar que se hace cumplir este artículo.

8.3. Real Decreto 381/2015 de 14 de mayo

Este Real Decreto trata sobre el tráfico irregular con fines fraudulentos en comunicaciones electrónicas. En él, hay que asegurar ciertos puntos, que se citan a continuación:

"*a) Garantizar la integridad de las redes y la seguridad de las redes y servicios de comunicaciones electrónicas.*

b) Garantizar la calidad en la prestación de los servicios de comunicaciones electrónicas.

c) Garantizar los derechos específicos de los usuarios de telecomunicaciones.

d) Controlar el uso de la numeración asignada, en particular para garantizar el cumplimiento de las condiciones ligadas al uso de los recursos públicos de numeración establecidas en los planes e instrucciones de esta ley. "

Es decir, que la aplicación ha de ser capaz de identificar los servicios que se prestan y garantizar que cada servicio cumple con los puntos comentados anteriormente del Real Decreto, además de asegurar los derechos de los usuarios de la aplicación.

Tagoror es una aplicación que satisface múltiples necesidades a los usuarios, apoyándose en módulos software proporcionados por Google o Facebook, entre otros. Dichos módulos garantizan que los derechos de los usuarios se cumplan, haciendo que el sistema en su totalidad y en la de los servicios que ofrece aseguren la seguridad del usuario.

8.4. Artículo 13 en Europa

En un artículo del periódico *La Vanguardia*, que podemos encontrar en la referencia [23], tiene el titular:

"*Europa aprueba el borrador de ley que quiere acabar con Internet tal y como lo conocemos*"

Podemos observar que la Unión Europea tiene planteado realizar una nueva ley que afectará a que los usuarios deban publicar contenido libre de derechos de autor que no sea del usuario, el llamado Artículo 13 que intenta implantar la *Directiva de Copyright de la Unión Europea*. Esto podría tener repercusiones en el proyecto con respecto a las historias que suban los usuarios al servidor ya que toda historia está potencialmente expuesta a contener publicaciones con derechos de autor, de forma que si dicho Artículo se acepta, el desarrollo del proyecto deberá continuar en que las historias deberán ser procesadas en primer lugar, bien sea por algún módulo software que se añada al sistema para que distingue las historias que sean válidas, bien personas físicas que comprueben dicha validez para cada historia, o ambos.

Capítulo 9

Conclusiones y trabajo futuro

Por último, en este apartado vamos a centrar nuestra atención en las conclusiones finales que hemos recapitulado al finalizar el proyecto y se realizará un vistazo al trabajo futuro que se puede realizar desde el punto en el que se encuentra actualmente Tagoror.

9.1. Trabajo futuro

Un proyecto software nunca acaba. Es por esto que, aunque se entregue una versión prototipada de la aplicación, las funcionalidades nunca dejan de aumentar. De esta manera, podemos dividir el trabajo futuro en diferentes funcionalidades, en las que encontramos de manera adjunta, un boceto inicial de cómo quedaría la interfaz relacionada.

9.1.1. Añadir un botón en el *MainActivity* para encontrar las historias más cercanas a la ubicación del usuario

El usuario no quiere pensar demasiado cual es la siguiente historia que puede observar. Es por ello, que se puede añadir en futuras versiones un botón en el cual el usuario encuentra la historia más cercana a su ubicación. Siguiendo esta línea, se pensó que lo útil sería que en el momento en el que dicho usuario pulsará el botón comentado, se buscara aquella historia no sólo más cercana, sino aquella historia que tenga otras más historias a su alrededor, es decir, la historia que tenga mejor relación número de historias / distancia. Un ejemplo de interfaz de usuario es como la que se recoge en la Ilustración 9.1.

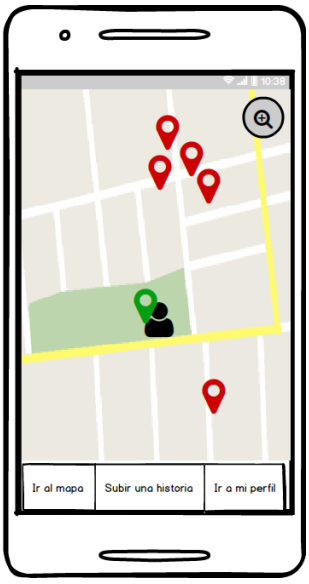

Ilustración 9.1: Esbozo inicial en el momento en el que un usuario busca una historia cercana

En él, se puede ver dicho botón comentado, y es entonces cuando un usuario pulsa dicho botón que ocurre que la historia con mejor la relación citada anteriormente se vuelve de color naranja, dando a conocer al usuario cual es la historia más cercana. Además, la vista del mapa se centraría entre dicha historia y el usuario, como se puede ver en la Ilustración 9.2.

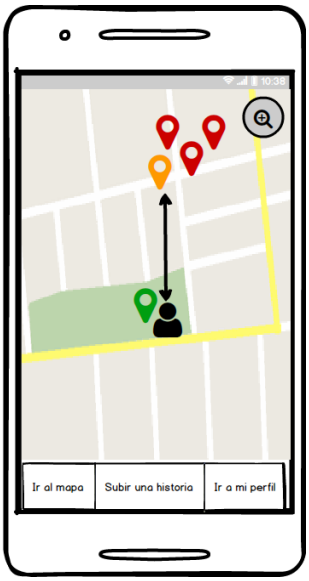

Ilustración 9.2: Momento en el que un usuario encuentra una historia cercana

9.1.2. Borrar una historia

Se hace necesaria la posibilidad de poder eliminar una historia que un usuario no quiera ya que exista. Es por esto que se debería poder eliminar una historia de manera permanente y que los demás usuarios no puedan verlo. Se plantea pues, que en el apartados de las historias creadas por un usuario se pueda acceder a un botón en el cual se elimine la historia, no sin antes un diálogo de confirmación por parte del usuario de que se desea borrar dicha historia. Para entender el flujo de actividades de cómo se borra una historia, debemos partir de la interfaz encontrada en la Ilustración 7.7, de manera que se puede observar la historia de ejemplo que se desea borrar. Un *mockup* de cómo sería la interfaz sería el de la Ilustración 9.3.

Ilustración 9.3: Momento en el que un usuario borra una historia

Una vez el usuario pulse en el botón de confirmación, la historia será borrada de la base de datos junto con el vídeo relacionado a dicha historia.

9.1.3. Editar una historia

Siguiendo la misma línea que el subapartado anterior, es útil la edición de una historia en concreto. Puede ser el título, la descripción, el vídeo asociado o la localización exacta de la historia. Para ello, volvemos a partir de la interfaz de la Ilustración 7.7, de forma que el usuario quiere editar dicha historia. Una vez pulsado el botón de editar dicha historia, se procede a un formulario donde el usuario puede editar los datos de la misma. Un ejemplo de la interfaz es el encontrado en la Ilustración 9.4.

Ilustración 9.4: Momento en el que un usuario edita una historia

Una vez el usuario haya terminado, se notificará al mismo de que los cambios se han realizado correctamente. En este punto es necesario pensar si la edición de una historia conlleva a que dicha historia vuelve a quedar como "no vista"por los demás usuarios, idea que sería necesario plantear a los usuarios finales para conocer qué piensan en este sentido.

9.1.4. Modificar perfil de usuario

En algún momento del uso de la aplicación un usuario, sobretodo aquel que haya iniciado sesión vía correo electrónico, querrá modificar algún dato que le sea necesario por ejemplo, la contraseña. Es por eso que se ha pensado que se debe editar el perfil de un usuario para poder tener control por parte del usuario de los datos que éste posee.

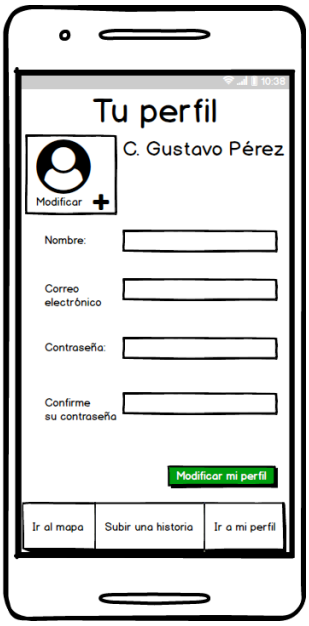

Ilustración 9.5: Usuario editando su perfil

Como se puede ver en la Ilustración 9.5, el usuario se encuentra ante este formulario en el que puede editar todos los datos que éste tenga, además de su foto de usuario. Una vez realizado algún cambio, se mostrará una notificación al usuario de que los cambios se han realizado con éxito.

Cabe tener en cuenta que los usuarios de Facebook o Google no podrán modificar sus datos ya que estos se encuentran en sus respectivos dominios.

9.1.5. Encontrar en el mapa las historias creadas o vistas

Esta funcionalidad se refiere básicamente a que en el momento de ver las historias creadas o ya vistas, se pueda acceder a ellas en el mapa a través de un botón. Ésto puede ser útil para mostrar a otros usuarios donde se encuentra dicha historia, de forma que se podría compartir la localización a la comunidad.

9.1.6. Tutorial dinámico para nuevos usuarios

Se ha pensado en, además de un manual de usuario, crear un tutorial para los usuarios, que aparecerá en el momento en que estos pidan ayuda, a través de un botón que maneje la lógica interna de dicho tutorial.

Para ello, se ha pensado utilizar un componente muy utilizado para este tipo de casos: *TourGuide*, que es un componente que podemos encontrar en un repositorio de GitHub (vease [22]) que sirve para crear tutoriales a los usuarios de una aplicación. Un ejemplo de cómo es este tipo de componente lo podemos ver en la Ilustración 9.6.

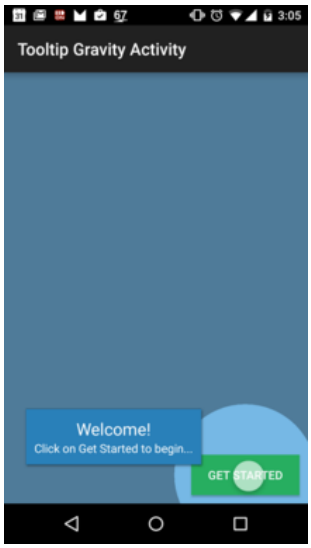

Ilustración 9.6: Ejemplo de interfaz de ***TourGuide***. Fuente [21]

9.2. Aportaciones

Tagoror es un sistema informático que afecta de manera global a la sociedad, haciendo que cualquier persona cuente una historia en formato multimedia a diferentes usuarios. Esto presenta un gran impacto social a nuestro entorno, de manera que se puede usar el sistema con múltiples propositos. Se podría usar, por ejemplo:

- ✔ Los ayuntamientos de diferentes ciudades podrían usar el sistema para realizar una ruta turística por diferentes zonas.
- ✔ Museos y otras instituciones pueden subir contenido para hacer vídeos explicativos de diferentes puntos importantes.
- ✔ A nivel individual, los usuarios podrían subir contenido que podrá contener cualquier cosa que quieran compartir con su entorno. Además, los usuarios pueden valorar un vídeo ya que la aplicación proporciona un sistema de «Me gusta» y «No me gusta» al acceder a las diferentes historias, de forma que pueden ver la valoración a priori de la misma. No sólo eso, si no que este proyecto requiere que el usuario realice un esfuerzo al moverse para visualizar las diferentes historias que puede encontrar por el lugar, proporcionando de esta forma una nueva alternativa para hacer ejercicio físico, ayudando al usuario a mantener un ritmo de vida saludable. Además, proporciona al usuario una manera de conocer nuevos lugares o conocimiento sobre el lugar en el que se encuentre marcada la historia.

De esta forma, Tagoror crea un ecosistema vivo que proporciona una nueva forma de comunicación entre los diferentes usuarios y el entorno que les rodea.

9.3. Conclusiones

Tagoror ha sido un proyecto muy amplio, en el que se cumplieron los objetivos establecidos de manera inicial. A través de una metodología de trabajo basada en reuniones semanales, se fijan pequeños objetivos que, aunque de por sí no son la creación del proyecto, van acercándonos a la creación de un proyecto de una envergadura ajustada a un Trabajo de Fin de Grado. Estas reuniones no eran una metodología ágil como puede ser Scrum, Crystal o Xtreme Programming, pero el concepto que es el de establecer objetivos semanales en los cuales se fija una reunión en la que se pueden plantear dudas y ayuda sobre la resolución de ciertos problemas, hace que el logro de los diferentes objetivos se haya cumplido en el tiempo establecido.

Además, se han utilizado herramientas que aportan facilidad al desarrollador a la hora de crear el proyecto, como puede ser *Firebase* y sus diferentes módulos. Gracias a la versatibilidad de estos componentes, se ha logrado gestionar parte de un proyecto móvil sin necesidad de aumentar demasiado la complejidad de la lógica del programa.

Cabe destacar que en primera instancia, este Trabajo de Fin de Grado ha pasado por diferentes etapas. En un primer momento, se encontraba como una idea, un concepto en el que aún se encontraban definidas las funcionalidades concretas del sistema. Más tarde, se focalizó el esfuerzo en el apartado de Análisis y Diseño, donde se fijaron ciertos objetivos y requisitos necesarios para hacer que Tagoror valiese el esfuerzo, en dónde se decidió qué dependencias software serían útiles para el sistema. Seguido de esta fase en el ciclo de vida de este Trabajo de Fin de Grado, viene el proceso de desarrollo, el cual una vez ya establecido todo lo necesario, se da comienzo a realizar funcionalidades una a una hasta completar todos los objetivos, teniendo en cuenta que un proyecto software nunca acaba y en el que suelen venir más y más necesidades por parte del usuario.

En definitiva, gracias a la elaboración de este proyecto se ha aprendido a elaborar desde el principio una aplicación basada en Android, usando diferentes herramientas para ello y con el aporte que supone una metodología "*ágil*", siendo este tipo de metodologías muy útiles a la hora de desarrollar un proyecto en el ámbito profesional, de manera que el desarrollador pueda realizar su trabajo sin generar una gran carga de trabajo.

9.3.1. Valoración personal

En mi opinión, Tagoror ha supuesto un nuevo descubrimiento personal en lo que a aplicaciones móviles se refiere. Al principio existía cierta incertidumbre por no conocer la mayoría de componentes de Android, pero gracias a que la aplicación está basada en Java y que existían muchos conceptos que ya conocía debido a mi recorrido por el grado, he podido resolver la mayoría de problemas que han podido llegar a ocurrir.

Por último, quiero decir que me siento realizado a nivel personal porque he realizado una aplicación que tiene un potencial muy alto para ser usado a nivel mundial, de manera que puede llegar a ser utilizado por cualquier persona del mundo que posea un *smartphone*, y

el hecho de que se relacionen a través de breves historias entre ellos, me hace pensar que las tecnologías móviles son muy útiles para aportar una herramienta de comunicación entre personas.

De esta forma y con este sentimiento, pienso que Tagoror puede ser un instrumento que puede proporcionar utilidad a nivel mundial entre las personas en el sector de ocio, y con ello, percibo que Tagoror puede llegar a ser un proyecto que nunca acabe; en el que no paren de llegar nuevas funcionalidades y actualizaciones, generando una especie de retroalimentación entre el usuario y los desarrolladores que incluso ellos mismos pueden ser usuarios de la misma aplicación en el que potencialmente se llega a interconectar los pensamientos de una persona con otras personas a nivel mundial a través de una historia.

Bibliografía

[1] APPS4CITIZENS (2016). Historias: la app para redescubrir tu ciudad. https://www.elperiodico.com/es/apps-para-el-ciudadano-comprometido/20160719/app-para-descubrir-ciudad-5277080. Accedido el día 2019-02-07.

[2] ArteInformado (2019). Artcity, arte en tu ciudad. https://play.google.com/store/apps/details?id=es.arteinformado&hl=es. Accedido el día 2019-02-07.

[3] Contour (2019). Página web de contour. http://contour.com/storyteller. Accedido el día 2019-02-13.

[4] Developers, A. (2019a). Actividades android. https://developer.android.com/guide/components/activities.html?hl=ES. Accedido el día 2019-05-14.

[5] Developers, A. (2019b). Manifiesto de android. https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419. Accedido el día 2019-05-14.

[6] Developers, A. (2019c). Permisos del sistema. https://developer.android.com/guide/topics/security/permissions.html?hl=es-419#normal-dangerous. Accedido el día 2019-05-14.

[7] Echoes (2019). Página web de echoes. https://echoes.xyz. Accedido el día 2019-02-13.

[8] González, J. C. (2018). Bulletin: así es la app con la que google quiere que publiquemos y leamos historias locales. https://www.xatakandroid.com/aplicaciones-android/bulletin-asi-es-la-app-con-la-que-google-quiere-que-publiquemos-y-leamos-historias-locales. Accedido el día 2019-02-07.

[9] Google (2019a). Bulletin, de google. https://posts.google.com/bulletin/share. Accedido el día 2019-02-07.

[10] Google (2019b). Página web de firebase. https://firebase.google.com/. Accedido el día 2019-05-22.

[11] ISDI (2014). La herramienta que te permite realizar prototipos de tus proyectos:balsamiq. https://www.isdi.education/es/isdigital-now/herramienta-te-permite-realizar-prototipos-de-tus-proyectos-balsamiq. Accedido el día 2019-05-21.

[12] JamCreativeStudios (2019). Vale tales gps storytelling app. http://jamcreativestudios.com/index.php/portfolio_page/vale-tales/. Accedido el día 2019-02-13.

[13] Neto, J. G. (2019). Android terminó 2018 rozando el 90 % de cuota de mercado en españa, según kantar. https://www.xatakandroid.com/moviles-android/android-termino-2018-rozando-90-cuota-mercado-espana-kantar. Accedido el día 2019-05-13.

[14] PatientRock (2016). Stories that follow. http://blog.patientrock.com/stories-that-follow-voicemap-immersive-storytelling/. Accedido el día 2019-02-13.

[15] Staff, V. (2018). Android: a 10-year visual history. https://www.theverge.com/2011/12/7/2585779/android-10th-anniversary-google-history-pie-oreo-nougat-cupcake. Accedido el día 2019-05-14.

[16] Studios, B. (2019). Página web de balsamiq. https://balsamiq.com/. Accedido el día 2019-05-21.

[17] TravelStorysGPS (2019). Página web de travelstorys. https://www.travelstorys.com/about/. Accedido el día 2019-02-13.

[18] VoiceMap (2019). Página web de voicemap. https://voicemap.me/. Accedido el día 2019-02-13.

[19] Wikipedia (2019a). Android. https://es.wikipedia.org/wiki/Android. Accedido el día 2019-05-14.

[20] Wikipedia (2019b). Desarrollo de programas para android. https://es.wikipedia.org/wiki/Desarrollo_de_programas_para_Android. Accedido el día 2019-05-14.

[21] worker8 (2019a). Tourguide. http://worker8.github.io/TourGuide/#/. Accedido el día 2019-05-22.

[22] worker8 (2019b). Tourguide. https://github.com/worker8/TourGuide. Accedido el día 2019-05-22.

[23] Álex Barredo (2018). Artículo de la vanguardia sobre el artículo 13. https://www.lavanguardia.com/tecnologia/20180621/45288972842/directiva-copyright-union-europea-articulo-11-articulo-13.html. Accedido el día 2019-05-13.