Universidad de Valladolid

Grado en Ingeniería Informática: Mención I.S.

Trabajo Fin de Grado

CRONOVISOR

App para la visualización in situ de imágenes antiguas de Valladolid

Autor: Adrián Martín Gómez

Tutor: Joaquín Adiego Rodríguez

Resumen

El objetivo fundamental del presente documento es exponer el proceso y metodología llevados a cabo para realizar el Trabajo Fin de Grado. El mismo, consiste en el desarrollo de una aplicación Android para la visualización de imágenes historias de la ciudad de Valladolid en su localización aproximada. Para ello, se han investigado apps relacionadas y se ha realizado un trabajo de búsqueda de imágenes. A su vez, se ha llevado un trabajo de aprendizaje y puesta en práctica de conocimiento Android.

Contenido

TABLA DE ILUSTRACIONES Y TABLAS

Capítulo 1: Introducción y objetivos.

1. Introducción.

Cronovisor es una App móvil desarrollada en Android que permite la visualización de imágenes antiguas históricas y compararlas con la realidad actual del lugar. Está pensada para ser un complemento a la exploración y descubrimiento por parte de los visitantes de nuestra ciudad. Sin embargo, también es una app interesante para aquellas personas, residentes en la propia ciudad, que quieran tener una nueva visión y descubrir los cambios y transformaciones que han tenido lugar en la misma.

Cronovisor está soportada por la API de Google Maps, que permite la geolocalización del usuario, la visualización de marcadores personalizados y el mapa más completo de la actualidad. Además, permite la apertura de la aplicación propia de Google Maps si se quiere saber cómo llegar a la ubicación mediante el navegador del propio dispositivo.

2. Motivación.

Actualmente, con el auge de los dispositivos móviles, prácticamente todo el mundo dispone de un dispositivo móvil con acceso a internet. Con esta premisa en la cabeza, la idea y la motivación de esta aplicación es dar a conocer a la mayor cantidad posible de personas la gran cantidad de imágenes históricas y de cambios que se han llevado a cabo en Valladolid.

La idea original surge con mi amigo Rubén Justo, historiador, que me comentó la posibilidad de crear un modelo de la ciudad de Valladolid a partir de las fotos historias existentes. Esto sería posible, pero muy complejo, y más para un TFG. Sin embargo, simplificándolo, acabamos pensando en una aplicación que mostrara dichas imágenes en sus puntos originales para descubrir el cambio que se ha producido en la ciudad.

Antes de decidir si llevarlo a cabo o no, preguntamos en el Archivo Municipal de Valladolid. Al ser las imágenes públicas, no había ningún problema siempre y cuando se mencionara su procedencia, por lo que decidí llevar a cabo este proyecto.

3. Objetivos.

El objetivo principal de la aplicación es claros, crear una aplicación móvil fácil de usar y que muestre una gran cantidad de imágenes históricas con sus respectivas descripciones.

Sin embargo, también existen varios objetivos secundarios:

- Geolocalización del usuario.
- Búsqueda de calles, tanto en la galería de imágenes como en el propio mapa.
- Notificar al usuario cuando se encuentre cerca de una localización con imágenes.
- Descarga de datos para poder visualizar la aplicación sin conexión.
- Muestra de marcadores para señalizar las imágenes.
- Facilidad de uso y visualización pensando en el posible uso de personas mayores (mayores de 50 años).

Capítulo 2: Estado del arte

1. Android.

En octubre de 2003, en la localidad de Palo Alto, Andy Rubin, Rich Miner, Chris White y Nick Sears fundan Android Inc. con el objetivo de desarrollar un sistema operativo para móviles basado en Linux. En julio de 2005, la multinacional Google compra Android Inc. El 5 de noviembre de 2007 se crea la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. El mismo día se anuncia la primera versión del sistema operativo: Android 1.0 Apple Pie. Los terminales con Android no estarían disponibles hasta el año 2008 cuando salió el primer móvil con sistema operativo Android, el HTC Dream. Actualmente los dispositivos con Android venden más que las ventas combinadas de Windows Phone e IOS. Android tiene una gran comunidad de desarrolladores creando aplicaciones para extender la funcionalidad de los dispositivos. Actualmente ya se han superado el 1.000.000 de aplicaciones disponibles en la Google Play. El nombre de Android (androide) y la marca Nexus One hacen referencia a la novela ¿Sueñan los androides con ovejas eléctricas?, adaptada al cine como Blade Runner.

1.1. Historial de versiones.

1.1.1. Android 1.0 Apple Pie.

Fue la primera versión comercial del software. Su lanzado se realizó el 23 septiembre de 2008. El primer dispositivo con Android fue el HTC.

1.1.2. Android 1.1 Banana Bread.

El 9 de febrero de 2009, la actualización Android 1.1 Banana Bread (Pan de plátano) fue lanzada, inicialmente sólo para el HTC Dream. Android 1.1 fue conocido como "Petit Four" internamente, aunque este nombre no se utilizó oficialmente. La actualización resolvió fallos, cambio la API y agregó una serie de características.

1.1.3. Android 1.5 Cupcake.

El 30 de abril de 2009, la actualización de Android 1.5 Cupcake fue lanzada, basada en el núcleo Linux 2.6.27. La actualización incluye varias nuevas características y correcciones de interfaz de usuario. Con esta versión, Android empezó a recibir el apoyo de muchos fabricantes que empezaron a diseñar dispositivos para esta versión.

1.1.4. Android 1.6 Donut.

El 15 de septiembre de 2009, fue lanzado el SDK de Android 1.6 Donut, basado en el núcleo Linux 2.6.29. En la actualización se incluyen numerosas características nuevas como por ejemplo el reconocimiento de voz.

1.1.5. Android 2.0/2.1 Eclair

El 26 de octubre de 2009, el SDK de Android 2.0 fue lanzado, basado en el núcleo de linux 2.6.29. Esta versión tuvo 2 subversiones.

1.1.6. Android 2.2.x Froyo.

El 20 de mayo de 2010, El SDK de Android 2.2 Froyo (Yogur helado) fue lanzado, basado en el núcleo Linux 2.6. Disponía de un nuevo motor para el navegador o instalación de apps en memoria externa fueron algunas de las novedades y tuvo un gran éxito y aceptación en el público.

1.1.7. Android 2.3.x Gingerbread.

El 6 de diciembre de 2010, el SDK de Android 2.3 Gingerbread (Pan de Jengibre) fue lanzado, basado en el núcleo Linux 2.6.35. Esta versión tuvo 7 actualizaciones a lo largo de su vida.

1.1.8. Android 3.x Honeycomb.

El 22 de febrero de 2011, sale el SDK de Android 3.0 Honeycomb (Panal de Miel). Fue la primera actualización exclusiva para tablet, lo que quiere decir que sólo es apta para tablets y no para teléfonos Android. Está basada en el núcleo de Linux 2.6.36. El primer dispositivo con esta versión fue la tableta Motorola Xoom, lanzado el 24 de febrero de 2011. Esta versión tuvo 8 actualizaciones en 2 API´s diferentes.

1.1.9. Android 4.0.x Ice Cream Sandwich

El SDK para Android 4.0.0 Ice Cream Sandwich (Sandwich de Helado), basado en el núcleo de Linux 3.0.1, fue lanzado públicamente el 12 de octubre de 2011. Gabe Cohen de Google declaró que Android 4.0 era "teóricamente compatible" con cualquier dispositivo Android 2.3 en producción en ese momento, pero sólo si su procesador y memoria ram lo soportaban. El código fuente para Android 4.0 se puso a disposición del público el 14 de noviembre de 2011. La actualización incluye numerosas novedades entre las que destaca la renovación de la interfaz de Android, nuevos botones del dispositivo y las notificaciones.

1.1.10. Android 4.1 Jelly Bean.

Google anunció Android 4.1 Jelly Bean (Gomita Confitada o Gominola) en la conferencia del 30 de junio de 2012. Basado en el núcleo de linux 3.0.31, Bean fue una actualización incremental con el enfoque primario de mejorar la funcionalidad y el rendimiento de la interfaz de usuario. La mejora de rendimiento involucró el "Proyecto Butter", el cual usa anticipación táctil, triple buffer, latencia vsync extendida y un arreglo en la velocidad de cuadros de 60 fps para crear una fluida y "mantecosa" suavidad de la interfaz de usuario. Android 4.1 Jelly Bean fue lanzado bajo AOSP el 9 de julio de 2012 y el Nexus 7, el primer dispositivo en correr Jelly Bean, fue lanzado el 13 de julio de 2012. Esta versión estaba disponible para tablets y smartphones y es una de las más longevas, ya que en la actualidad aún se utiliza.

1.1.11. Android 4.4 KitKat.

Google sorprendió a todo el mundo llegando a un acuerdo con la famosa marca de chocolatinas, es una de las versiones que más cambios ha traído tanto a nivel de interfaz como, a nivel interno y con vistas al futuro. Hoy en día la versión 4.4 que pertenece a KitKat, es la versión más utilizada de Android en todo el mundo.

1.1.12. Android 5.0 Lollipop.

Esta versión ha traído cambios muy importantes en la interfaz, se ha redefinido al completo para incluir el denominado Material Design. Una de las nuevas funcionalidades que trajo fue la unificación entre los sistemas Smartphone, tablet, smartwatch y TV. Actualmente es la versión más reciente del sistema de Google.

1.1.13. Android 6.0 Marshmallow.

Se presentó en mayo del año pasado, la novedad más importante es la introducción de la plataforma de pagos de Google, Android Pay.

1.1.14. Android 6.x N.

Android N es la versión actual en desarrollo de Google. Fue presentada el 18 de mayo de 2016 en el evento Google I/O.

1.1.15. Resumen.

Nombre código	**Número de versión**	**Fecha de lanzamiento**	**Nivel de API**
	1.0	23 de Septiembre, 2008	1
	1.1	9 de Febrero, 2009	2
Cupcake	1.5	27 de Abril, 2009	3
Donut	1.6	15 de Septiembre, 2009	4
Eclair	2.0–2.1	26 de Octubre, 2009	5–7
Froyo	2.2–2.2.3	20 de Mayo, 2010	8
Gingerbread	2.3–2.3.7	6 de Diciembre, 2010	9–10
Honeycomb1	3.0–3.2.6	22 de Febrero, 2011	11–13
Ice Cream Sandwich	4.0–4.0.4	18 de Octubre, 2011	14–15
Jelly Bean	4.1–4.3.1	9 de Julio, 2012	16–18
KitKat	4.4–4.4.4, 4.4W–4.4W.2	31 de Octubre, 2013	19–20
Lollipop	5.0–5.1.1	12 de Noviembre, 2014	21–22
Marshmallow	6.0–6.0.1	5 de Octubre, 2015	23
N	6.x *Previa de desarrollador 4*	15 de junio, 2016	24

Tabla 1: Resumen versiones Android

1.2. Arquitectura.

Android es un sistema operativo creado para ser independiente de cualquier tipo de arquitectura de hardware en los dispositivos móviles. Esta característica hace que sea tan atractivo ante los fabricantes y desarrolladores.

Adicionalmente su portabilidad, flexibilidad y seguridad les da ese toque de simpatía a las personas interesadas en los sistemas de código abierto.

Android está construido con una arquitectura de 4 capas o niveles relacionados entre sí. A continuación veremos un diagrama ilustrativo extraído del libro Learning Android escrito por Marko Gargenta y Masumi Nakamura:

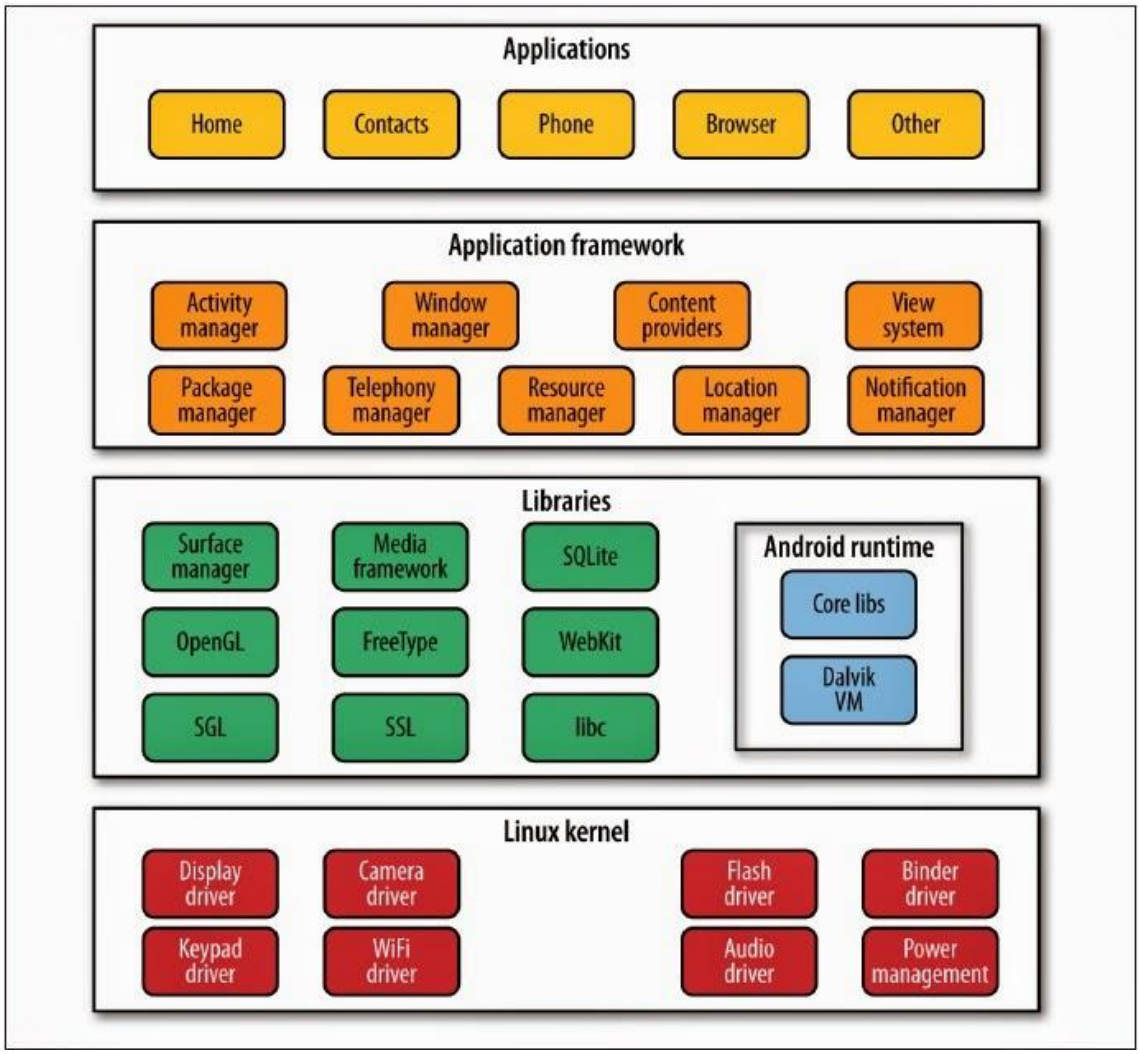

Ilustración 1: Arquitectura Android.

El diagrama indica que la estructura de Android se encuentra construida sobre el Kernel de Linux. Luego hay una capa de Librerías relacionadas con una estructura administradora en Tiempo de ejecución. En el siguiente nivel encontramos un Framework de apoyo para construcción de aplicaciones y posteriormente vemos a la capa de Aplicaciones.

1.2.1. Kernel de Linux.

Android está construido sobre el núcleo de Linux, pero se ha modificado dramáticamente para adaptarse a dispositivos móviles. Esta elección está basada en la excelente potabilidad, flexibilidad y seguridad que Linux presenta. Recuerda que el Kernel de Linux está bajo la licencia GPL, así que en consecuencia Android también.

1.2.2. Capa de librerías o capa nativa.

En esta capa se encuentran partes como la HAL, librerías nativas, demonios, las herramientas de consola y manejadores en tiempo de ejecución. Veamos un poco el propósito de estos conceptos:

- **Hardware Abstraction Layer (HAL**): Este componente es aquel que permite la independencia del hardware. Quiere decir que Android está construido para ejecutarse en cualquier dispositivo móvil sin importar su arquitectura física. El HAL actúa como una arquitectura genérica que representa a todos los posibles tipos de hardware existentes en el mercado. Aunque por el momento no hay estándares de construcción en el hardware de dispositivos móviles, el HAL permite que cada fabricante ajuste sus preferencias para que Android sea funcional sobre su tecnología.
- **Librerías nativas:** Aquí encontramos interfaces de código abierto como OpenGL para el renderizado de gráficos 3D, SQLite para la gestión de bases de datos, WebKit para el renderizado de los browsers, etc. También librerías para soportar los servicios del sistema como Wifi, posicionamiento, telefonía, y muchos más.
- **Demonios (Daemons):** Los demonios son códigos que se ejecutan para ayudar a un servicio del sistema. Por ejemplo cuando se requiere instalar o actualizar una aplicación, el demonio de instalación “installd” es ejecutado para administrar todo el proceso. O cuando los desarrolladores vamos a ejecutar en modo de depuración nuestro teléfono desde un PC, se ejecuta un demonio llamado adbd (Android Debug Bridge Daemon) para auxiliar a dicho proceso.
- **Consola:** Al igual que otros sistemas operativos, Android permite que empleemos comandos de línea para la ejecución de procesos del sistema o explorar el sistema operativo.
- **Manejadores en tiempo de ejecución:** Si bien las aplicaciones Android están escritas en lenguaje Java y son traducidas a bytecodes, estas no son interpretadas por la Máquina virtual de Java. Android tiene su propia máquina virtual interpretadora de bytecodes llamada Dalvik. Esta herramienta fue diseñada para ser flexible ante el diseño de hardware de un dispositivo móvil. Además JVM no es de licencia GPL, así que Google decidió generar su propia herramienta.

1.2.3. Dalvik.

Dalvik no cambia nada en el proceso de compilación, sencillamente interviene al final como receptor de un archivo ejecutable producto de una recopilación de los archivos .class de java.

Recuerda las fases de la construcción de una aplicación Java. El primer paso es generar el código fuente (arhivos .java), luego este es traducido por el Java Compiler (javac) y obtenemos un fichero tipo byte code (archivos .class). Finalmente la máquina virtual de Java (JVM) interpreta en tiempo real este archivo y la aplicación es ejecutada.

La ejecución de Dalvik es ingeniosa, simplemente espera que javac traduzca la aplicación a byte codes, cuando están listos los archivos, estos son compilados por el compilador Dex. Esta herramienta traduce los byte codes de java a un estilo de byte codes nativos que serán convertidos a un ejecutable .dex. Finalmente este archivo es ejecutado por una instancia de Dalvik VM.

A continuación se muestra un diagrama comparativo de ambos procesos:

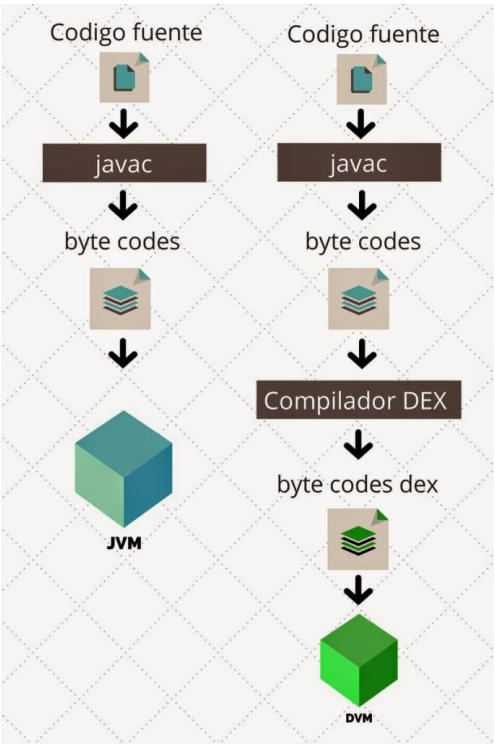

Ilustración 2: Modelo Dalvik.

Aunque el proceso añade unos cuantos pasos más, no debes preocuparte por ello, ya que esta tarea se le delega a la herramienta Gradle.

Actualmente, Dalvik VM está siendo reemplazada por una nueva máquina virtual llamada ART en su versión L (Lollipop).

1.2.4. Framework para aplicaciones.

Esta es la capa que nos interesa a los desarrolladores, ya que en ella encontramos todas las librerías Java que necesitamos para programar nuestras aplicaciones. Los paquetes con más preponderancia son los android.*, en ellos se alojan todas las características necesarias para construir una aplicación Android.

No obstante es posible acceder a clases como java.utils.*, java.net.*, etc. Aunque hay librerías Java excluidas como la java.awt.* y java.swing.*.

En esta capa también encontraremos manejadores, servicios y proveedores de contenido que soportaran la comunicación de nuestra aplicación con el ecosistema de Android.

1.2.5. Capa de aplicaciones.

Es la última instancia de funcionamiento de Android. Se centra en la ejecución, comunicación y estabilidad de las aplicaciones preinstaladas por el fabricante o las que nosotros vamos a construir. A ella acceden todos los usuarios Android debido a su alto nivel de compresión y simplicidad.

1.3. Componentes.

1.3.1. Activity

Sin duda es el componente más habitual de las aplicaciones para Android. Un componente Activity refleja una determinada actividad llevada a cabo por una aplicación, y que lleva asociada típicamente una ventana o interfaz de usuario; es importante señalar que no contempla únicamente el aspecto gráfico, sino que éste forma parte del componente Activity a través de vistas representadas por clases como View y sus derivadas. Este componente se implementa mediante la clase de mismo nombre Activity. La mayoría de las aplicaciones permiten la ejecución de varias acciones a través de la existencia de una o más pantallas. Por ejemplo, piénsese en una aplicación de mensajes de texto. En ella, la lista de contactos se muestra en una ventana. Mediante el despliegue de una segunda ventana, el usuario puede escribir el mensaje al contacto elegido, y en otra tercera puede repasar su historial de mensajes enviados o recibidos. Cada una de estas ventanas debería estar representada a través de un componente Activity, de forma que navegar de una ventana a otra implica lanzar una actividad o dormir otra. Android permite controlar por completo el ciclo de vida de los componentes Activity.

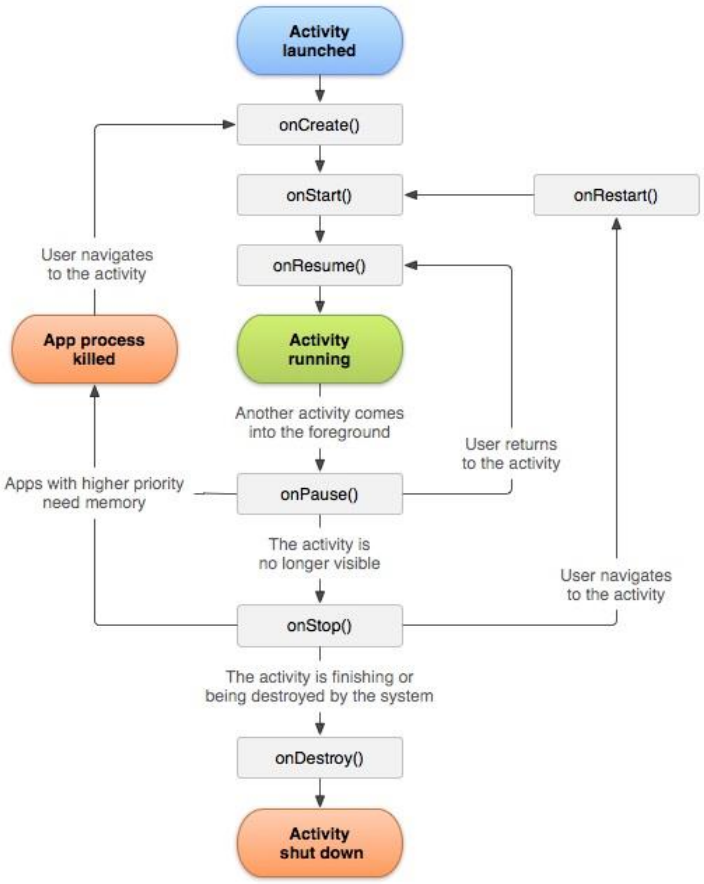

Ilustración 3: Ciclo de vida de actividad

Tal y como se puede ver en la imagen, una actividad tiene un ciclo de vida muy definido, que será igual para todas las actividades. Este ciclo de vida es impuesto por el SDK de Android. Las actividades tienen cuatro posibles estados: Activa, pausada, parada y reiniciada.

A la hora de diseñar una aplicación en Android hay que tener en cuenta su ciclo de vida.

1.3.2. Intent.

Un Intent consiste básicamente en la voluntad de realizar alguna acción, generalmente asociada a unos datos. Lanzando un Intent, una aplicación puede delegar el trabajo en otra, de forma que el sistema se encarga de buscar qué aplicación de entre las instaladas, es la que puede llevar a cabo la acción solicitada. Por ejemplo, abrir una URL en algún navegador web, o escribir un correo electrónico desde algún cliente de correo. Los Intents están incluidos en el AndroidManifest porque describen dónde y cuándo puede comenzar una actividad. Cuando una actividad crea un Intent, éste puede tener descriptores de lo que se quiere hacer. Una vez se está ejecutando la aplicación, Android compara esta información del Intent con los Intents de cada aplicación, eligiendo el más adecuado para realizar la operación especificada por el llamante.

1.3.3. Broadcast Intent Receiver.

Un componente Broadcast Intent Receiver se utiliza para lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produzca (generalmente, abrir un componente Activity). Por ejemplo, una llamada entrante o un SMS recibido. Este componente no tiene interfaz de usuario asociada, pero puede utilizar el API Notification Manager para avisar al usuario del evento producido a través de la barra de estado del dispositivo móvil. Este componente se implementa a través de una clase de nombre BroadcastReceiver. Para que Broadcast Intent Receiver funcione, no es necesario que la aplicación en cuestión sea la aplicación activa en el momento de producirse el evento. El sistema lanzará la aplicación si es necesario cuando el evento monotorizado tenga lugar.

1.3.4. Service.

Un componente Service representa una aplicación ejecutada sin interfaz de usuario, y que generalmente tiene lugar en segundo plano mientras otras aplicaciones (éstas con interfaz) son las que están activas en la pantalla del dispositivo. Un ejemplo típico de este componente es un reproductor de música. La interfaz del reproductor muestra al usuario las distintas canciones disponibles, así como los típicos botones de reproducción, pausa, volumen, etc. En el momento en el que el usuario reproduce una canción, ésta se escucha mientras se siguen visualizando todas las acciones anteriores, e incluso puede ejecutar una aplicación distinta sin que la música deje de sonar. La interfaz de usuario del reproductor sería un componente Activity, pero la música en reproducción sería un componente Service, porque se ejecuta en background. Este elemento está implementado por la clase de mismo nombre Service.

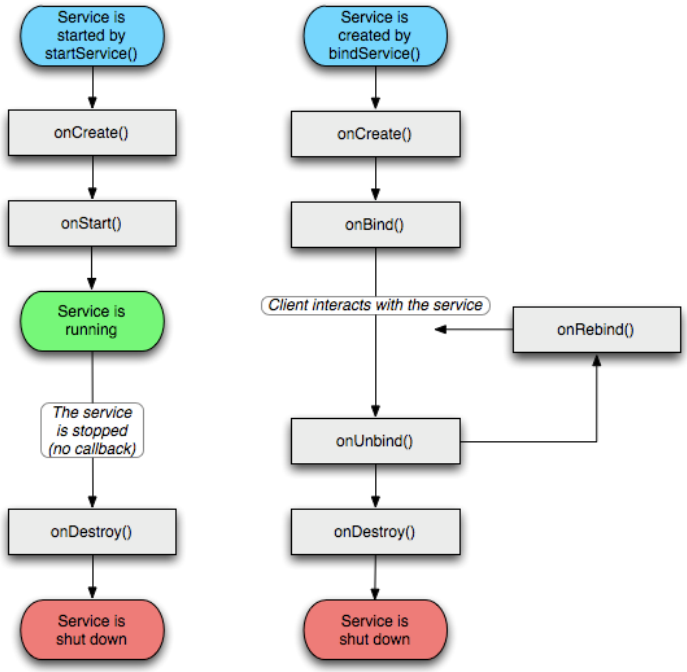

Ilustración 4: Ciclo de vida servicio

1.3.5. Content Provider

Con el componente Content Provider, cualquier aplicación en Android puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro formato que considere. Además, estos datos pueden ser compartidos entre distintas aplicaciones. Una clase que implemente el componente Content Provider contendrá una serie de métodos que permiten almacenar, recuperar, actualizar y compartir los datos de una aplicación.

1.4. Entorno de desarrollo: Android Studio.

Android Studio es el IDE (Entorno de desarrollo integrado) oficial para desarrollar aplicaciones para Android, utiliza licencia Apache 2.0, está basado en Java y es multiplataforma. Sustituye al plugin ADT (Android Developer Tools) para Eclipse.

2. Cronovisor.

El 2 de mayo de 1972, el semanario italiano Domenica del Corriere sorprendía a sus lectores con un insólito titular: "Inventada la máquina que fotografía el pasado". Esta asombrosa maquina era el Cronovisor, desarrollada en secreto por el monje benedictino Marcello Pellegrino Ernetti junto a 12 físicos. Ernitti afirmaba que había conseguido captar imágenes del pasado como la destrucción de Roma o la destrucción de Sodoma y Gomorra. Sin embargo, fueran pocas las imágenes que salieron a la luz, y entre ellas, las más polémica fueron una supuesta imagen de Jesucristo clavado en la cruz y una supuesta copia del texto del Thyestes. Ernetti aseguraba haber construido su máquina basándose en el concepto de que las ondas sonoras y visuales son energía, y por tanto, están sometidas a las mismas leyes físicas que la materia (la energía ni se crea ni se destruye, solo se transforma). Igual que desde las partículas más ínfimas se puede recomponer un elemento en su forma primitiva, el artefacto sería capaz de acceder a las ondas luminosas y sonoras del pasado, reorganizándolas en las mismas imágenes y sonidos que las integraron en su origen.

Ilustración 5: Imagen Cronovisor

Sin embargo, rápidamente se descubrió el engaño. La foto de Cristo en la cruz no era más que una fotografía invertida del Cristo venerado en el santuario del Amor Misericordioso de Collevalenza. Así mismo, se descubrió que palabra usadas en el texto de Thyestes no fueron usadas hasta 2 siglos después. Después de la muerte del padre Ernetti, los medios de comunicación locales recibieron un documento de alguien que decía ser un pariente de Ernetti pero que deseaba permanecer en el anonimato. El documento hablaba de como Ernetti antes de morir confesó de que había inventado la obra Tiestes y falsificado la fotografía.

3. Aplicaciones existentes.

En la actualidad están surgiendo aplicaciones móviles que tratan de ofrecer una alternativa viable e interesante a las guías turísticas en papel. Las ciudades también están desarrollando aplicaciones de este estilo. Vamos a analizar algunas de ellas.

3.1. Historias.

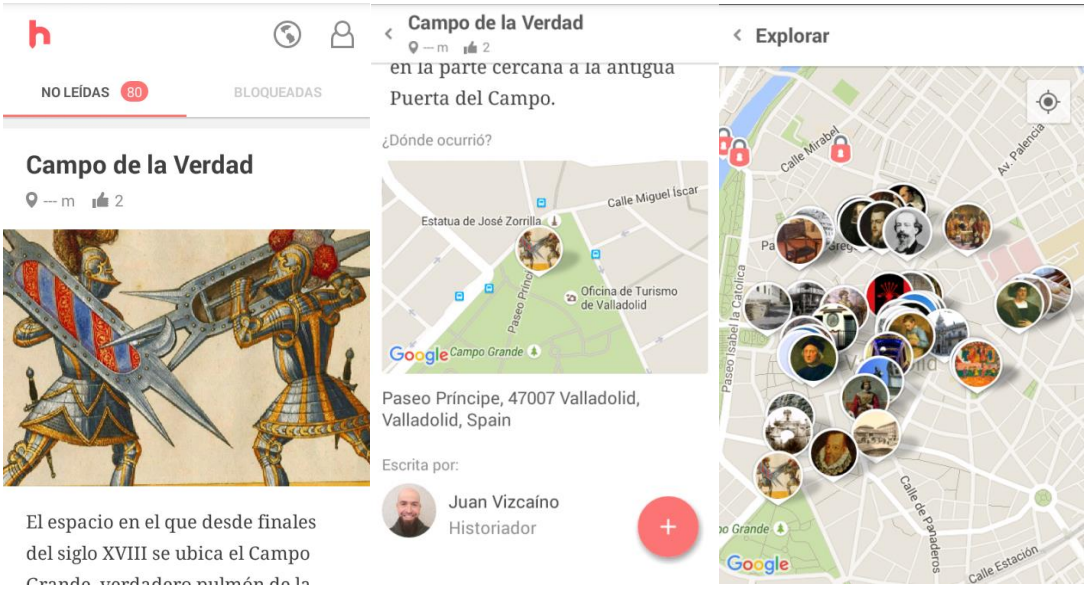

Ilustración 6: App Historias

Aplicación móvil que interactúa con el usuario cuando está a menos de 200 metros de un hecho histórico importante. Pretende descubrir a las personas la gran cantidad de hechos históricos que existen en la ciudad. Tiene la pega de que las historias se compran con monedas. Al descargarla contamos con 500 monedas y gastamos 10 por cada historia que descubramos y leamos. Si nos quedamos sin monedas podemos esperar un mes para conseguir 100 monedas más o comprar dichas monedas.

Es una aplicación muy interesante y un competidor directo de Cronovisor. Sin embargo, son bastante diferentes. Historias está centrado en dar conocimiento de unos hechos acontecidos en el lugar que nos muestra, utiliza algunas fotos antiguas para ilustrarlo, pero también utiliza imágenes actuales. Por otra parte, Cronovisor se centra en las imágenes antiguas y descubrir la evolución de calles y monumentos de la ciudad.

3.2. Eye of Shakspeare.

Ilustración 7: App Eye of Shakspeare

La aplicación utiliza la tecnología de realidad aumentada y escaneo de códigos QR para facilitar al usuario una experiencia única mientras se pasea por las calles de Stratford-upon-Avon. Podrá realizar un tour en el que pasará por la casa natal de Shakespeare y otros sitios relacionados con el escritor.

Esta aplicación es parecida a Cronovisor, pero cuenta con una realidad aumentada que genera un punto de vista totalmente diferente.

4. Servicios REST.

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP.

Los sistemas que siguen los principios REST se llaman con frecuencia sistemas *RESTful*.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un **protocolo cliente/servidor sin estado:** cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)
- Un conjunto de **operaciones bien definidas** que se aplican a todos los *recursos* de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son **POST**, **GET**, **PUT** y **DELETE**. Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (ABMC en castellano: crear, leer, actualizar, borrar) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
- Una **sintaxis universal** para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- El **uso de hipermedios**, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

5. Django y Django REST Framework.

5.1. Django.

Django es un Framework de desarrollo web de código abierto escrito en Python y que se basa en el patrón de diseño Modelo-Vista-Controlador. En su origen, fue desarrollado para gestión varias páginas web orientadas a noticias. Fue libreado al público bajo una licencia BSD en 2005. El objetivo fundamental de Django es facilitar la creación de sitios web complejos y llevarlos de idea a la realidad lo más rápido posible. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio no te repitas (DRY, Don´t Repeat Yourself).

Django fue pensado en su desarrollo para administrar páginas de noticias, y ese origen se ve reflejado en su diseño. Características principales:

- Un mapeador objeto-relacional.
- Sistemas de comentarios.
- Aplicación incorporada para administrar contenidos.
- Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- Una API de base de datos robusta.
- Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.
- Un sistema "middleware" para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

5.1.1. Arquitectura.

Aunque Django está fuertemente inspirado en la filosofía de desarrollo Modelo Vista Controlador, sus desarrolladores declaran públicamente que no se sienten especialmente atados a observar estrictamente ningún paradigma particular, y en cambio prefieren hacer "lo que les parece correcto". Como resultado, por ejemplo, lo que se llamaría "controlador" en un "verdadero" framework MVC se llama en Django "vista", y lo que se llamaría "vista" se llama "plantilla".

Gracias al poder de las capas *mediator* y *foundation*, Django permite que los desarrolladores se dediquen a construir los objetos Entity y la lógica de presentación y control para ellos.

- **Presentación:** aquí se maneja la interacción entre el usuario y el computador. En Django, ésta tarea la realizan el *template engine* y el *template loader* que toman la información y la presentan al usuario (vía HTML, por ejemplo). El sistema de configuración de URLs es también parte de la capa de presentación...
- **Control:** en esta capa reside el programa o la lógica de aplicación en sí. En Django son representados por las views y manipulators. La capa de presentación depende de ésta y a su vez ésta lo hace de la capa de dominio.
- **Mediator:** es el encargado de manejar la interacción entre el subsistema Entity y foundation. Aquí se realiza el mapeo objeto-relacional a cargo del motor de Django.
- **Entity:** el subsistema entity maneja los objetos de negocio. El mapeo objeto-relacional de Django permite escribir objetos de tipo entity de una forma fácil y estándar.
- **Foundation:** la principal tarea del subsistema foundation es la de manejar a bajo nivel el trabajo con la base de datos. Se provee soporte a nivel de foundation para varias bases de datos y otras están en etapa de prueba.

5.1.2. Soporte de bases de datos.

Respecto a la base de datos, la recomendada es PostgreSQL, pero también son soportadas MySQL y SQLite 3. Se encuentra en desarrollo un adaptador para Microsoft SQL Server. Una vez creados los data models, Django proporciona una abstracción de la base de datos a través de su API que permite crear, recuperar, actualizar y borrar objetos. También es posible que el usuario ejecute sus propias consultas SQL directamente. En el modelo de datos de Django, una clase representa un registro de una tabla en la base de datos y las instancias de esta serán las filas en la tabla.

5.1.3. Soporte de servidores Web.

Como mencionamos en los requisitos, Django incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. En la etapa de producción, sin embargo, se recomienda Apache 2 con mod_python. Aunque Django soporta la especificación WSGI, por lo que puede correr sobre una gran variedad de servidores comoFastCGI o SCGI en Apache u otros servidores (particularmente Lighttpd).

5.2. Django REST Framework.

Django Rest Framework es una aplicación Django que permite construir proyectos software bajo la arquitectura REST, incluye gran cantidad de código para reutilizar (Views, Resources, etc.) y una interfaz administrativa desde la cual es posible realizar pruebas sobre las operaciones HTTP como lo son: POST y GET.

Cabe resaltar que los mentores de este proyecto hacen uso intensivo de las Generic Views, las cuales desde Django 1.3 se basan en clases (class) y no en funciones (def), esto con el objetivo de aprovechar las ventajas de la programación orientada a objetos.

6. SQLite.

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

El autor de SQLite ofrece formación, contratos de soporte técnico y características adicionales como compresión y cifrado.

7. Archivo municipal de Valladolid.

El Archivo Municipal de Valladolid conserva la documentación producida por los órganos de gobierno de la ciudad desde la Edad Media hasta nuestros días. Tiene como objetivos principales dar soporte a la gestión de las oficinas, garantizar el derecho de los ciudadanos a acceder a la documentación, salvaguardar el patrimonio documental de la ciudad y difundirlo a la sociedad.

7.1. Fondos documentales.

El Archivo Municipal de Valladolid conserva más de cuatro kilómetros lineales de documentación fechada entre finales del siglo XII y el siglo XXI. Junto a los documentos en papel, de presencia mayoritaria, destacan por su singularidad los pergaminos medievales (1191-1393); los fondos fotográficos (más de 52000 unidades catalogadas); los fondos cartográficos (planes de urbanismo desde 1863 hasta la actualidad), y los carteles, bandos y postales de los siglos XIX y XX. A estos documentos se suman los que en soporte electrónico han comenzado a recibirse en el archivo en los últimos años.

La mayoría de estos documentos han sido producidos por los órganos de gobierno municipal (fondo Ayuntamiento de Valladolid), aunque también por otras instituciones y personas de la ciudad o relacionados con ella.

7.1.1. Fondo ayuntamiento de Valladolid.

Documentación producida por los órganos de gobierno de la ciudad desde la aparición del concejo en la Edad Media hasta la actualidad. Se trata, por tanto, de un fondo abierto, cuyo documento más antiguo se remonta al año 1191.

Su organización es funcional, en torno a cuatro secciones: Gobierno, Administración, Servicios y Hacienda, que abarcan toda la gestión municipal (actuaciones políticas, jurídicas, administrativas, económicas y sociales del Ayuntamiento), reflejando más de ocho siglos de vida municipal. Entre las series más significativas destacan los Libros de Actas, que recogen las sesiones del Concejo desde 1497 hasta la actualidad; los padrones de habitantes, que reflejan la evolución de la población de la ciudad; y la rica documentación de obras y urbanismo, que permite conocer las transformaciones del casco urbano desde la Edad Moderna hasta nuestros días.

- FONDO CONSORCIO CONMEMORACIÓN V CENTENARIO PRESENCIA ESPAÑOLA EN AMÉRICA: Consorcio constituido el 3 de octubre de 1988 por el municipio de Valladolid, la provincia, las Cortes de Castilla y León, la Universidad y la Cámara de Comercio de Valladolid para la promoción de la conmemoración en Valladolid del V Centenario de la presencia española en América. Según el decreto 5394 de 9 de junio de 1995, una vez disuelto el consorcio toda su documentación (seis cajas de documentos de los años 1988 a 1995) pasó a formar parte de los fondos del Archivo Municipal.

- FONDO CONSORCIO CONMEMORACIÓN IV CENTENARIO TÍTULO DE CIUDAD: Consorcio constituido por Ayuntamiento, Diputación Provincial, Junta de Castilla y León, Universidad de Valladolid, Cámara de Comercio e Industria, Arzobispado y Asociación de la prensa de Valladolid por acuerdo del pleno el 11 de mayo de 1994 con el fin de promover la conmemoración del IV Centenario de la Ciudad de Valladolid (1596 – 1996). Por acuerdo de pleno de 4 de marzo de 1998 se aprobó la disolución del consorcio, y con fecha 31 de enero de 2000 se realizó la transferencia de su documentación al archivo (64 cajas correspondientes a los años 1994 a 1996) para su custodia definitiva.

7.1.2. Fondo Hospital de Esgueva.

El Hospital de Santa María de Esgueva, fundado durante el reinado de Alfonso VI (1073-1109) por los condes de Castilla Pedro Ansúrez y Doña Eylo, se dedicó a la asistencia de los enfermos, pobres y ancianos de Valladolid desde sus orígenes hasta su conversión en hospital municipal en la segunda mitad del siglo XIX.

Su documentación, en 430 cajas, está organizada en las secciones de Gobierno, Jurisdicción Privativa, Administración de Servicios, Patrimonio, Hacienda y Contabilidad, Junta Municipal de Beneficencia y Archivos Personales, es esencial para el estudio de la beneficencia y la asistencia social en Valladolid a lo largo de buena parte de la historia de la ciudad, ya que sus fondos abarcan desde la Edad Media hasta finales del siglo XIX (1387-1887).

7.1.3. Fondo Asociación de la prensa.

Fondo integrado por 70.000 fotografías procedentes de la desaparecida Hoja del Lunes, editada por la Asociación de la Prensa, y publicada entre los años 1947 y 1985. Durante esos años fue el único periódico matutino de carácter general que salía los lunes (día en que descansaba la plantilla del resto de los periódicos) hasta que, en 1985, El Norte de Castilla comienza a salir también ese día. Sus contenidos se refieren a la actualidad local y provincial, coincidiendo con su ámbito de distribución, dedicando gran número de páginas a la sección deportiva y menos a las de nacional e internacional, cuya información procede en su totalidad de las agencias Efe, Europa Press y Cifra Gráfica.

Las fotografías de ámbito local están firmadas por profesionales con una dilatada trayectoria en la ciudad de Valladolid: Cacho, Urbano, Carvajal, Santiago Benito Cacho y Cantalapiedra, entre otros.

7.1.4. Fondo Sociedad Industrial Castellana.

La Sociedad Industrial Castellana se constituyó en 1898 con la finalidad de instalar en Valladolid una fábrica azucarera (la Azucarera Santa Victoria). En 1900 adquiere la concesión y propiedad del Canal del Duero con el fin de promover el regadío y garantizar el abastecimiento de remolacha para la fábrica, lo que con el tiempo acaba propiciando que se convierta en abastecedora de agua para la ciudad.

Los 80 planos y 5 fotografías que integran el fondo corresponden a la fábrica y a diversos proyectos de abastecimiento de aguas de la ciudad, y están datados entre los años 1893 y 1968.

7.1.5. Fondo Carvajal.

Primitivo Carvajal Santiago [Lanseros (Zamora), 1886–Valladolid, 1953] fue un fotógrafo autodidacta que adquirió fama por sus trabajos sobre la Semana Santa vallisoletana. Prestó servicios en el departamento de cardio-radiología de la facultad de Medicina de la Universidad, ilustrando con sus fotos las patologías de los enfermos.

Fue corresponsal gráfico de periódico ABC y del semanario Blanco y Negro, trabajando como redactor gráfico en el periódico Libertad desde 1931. A lo largo de los treinta años de su actividad los temas que abordó fueron el paisaje urbano (edificios civiles y religiosos) y pueblos de la provincia y de la región.

En 1940 funda los Laboratorios Carvajal, empresa en la que colaboró su familia. Sus hijos, Primitivo (1920-1995) y José Guillermo (1922-1977) dieron continuidad a la firma Carvajal.

El libro "Valladolid en Castilla", publicado en 1969 con textos de Félix Antonio González, se ilustra con imágenes procedentes de los archivos fotográficos del Museo Nacional de Escultura y del Seminario de Estudios de Arte y Arqueología de la Universidad de Valladolid, con los que Primitivo Carvajal colaboró estrechamente.

Aunque no consta documentalmente, el archivo Carvajal formaba parte de los fondos que ingresaron en el Archivo Municipal procedentes de la extinguida Sociedad Industrial Castellana. Esta Sociedad se constituyó en 1898 con la finalidad de instalar en Valladolid una fábrica de azúcar (la Azucarera Santa Victoria). En 1900 adquiere la concesión y propiedad del Canal del Duero con el fin de promover el regadío y garantizar el

abastecimiento de remolacha para la fábrica, lo que con el tiempo acabapropiciando que se convierta en abastecedora de agua para la ciudad.

El fondo llega al Archivo Municipal de Valladolid en junio de 1998 por donación de María Jesús Puente Aparicio. Las fotografías del fondo Carvajal son de carácter institucional y comercial (actos públicos de autoridades, Feria de Muestras, Semana de Cine, reportajes de empresas).

7.1.6. Fondo Teatro Calderón.

La Sociedad Pérez Calderón y Compañía fue creada en 1863 con la finalidad de contar con un marco legal para la construcción y gestión de un teatro (el Calderón) acorde con las necesidades de la ciudad, teatro que, después de muchas vicisitudes, acabará siendo adquirido por el Ayuntamiento (1986).

Los documentos que integran el fondo (160 cajas y 90 libros más dibujos, programas de mano, planos y fotografías comprendidos entre 1863 y 1990) reflejan tanto la historia del teatro Calderón (arquitectura, sistemas constructivos, motivos decorativos, dotación del teatro, personal, compañías, actores, obras representadas) como la vida cultural de la ciudad y sus dimensiones sociológica, económica y política.

Se trata de un fondo de singular importancia para el estudio de la historia de la música y la literatura que, además de incluir un rico fondo musical y libretos de las distintas representaciones tiene un importante complemento en el fondo bibliográfico de la Biblioteca del Círculo del Calderón.

7.1.7. Fondos Fotográficos de donaciones privadas.

Conjunto de fondos fotográficos de distinta procedencia (instituciones y particulares) recibidos en donación por el Archivo Municipal. Comprende un total de 5600 unidades fechadas desde el año 1880 hasta la actualidad, de temática variada, aunque principalmente referentes a personajes, edificios, y las transformaciones de la vida de la ciudad.

Colecciones: César Aguirre Viani, Ciro Crespo Cortejoso, José María Campos Setién, Fondo Carvajal, Joaquín Martín de Uña, Colegio de los Ingleses (San Albano), Ulises Asimov (seudónimo), José Ramón Santos Martínez, Gilardi, Óscar Campillo Madrigal, Jesús Sanz Silla, Gloria Jiménez Rodríguez-Vila, familia Marciel Conde, Olga López Soto, Ángel Villaverde Villa, Víctor Muñoz Adrián, Miguel Ángel Soria.

Hay además otras donaciones de fondos fotográficos de menos de 25 unidades, que se agrupan bajo la denominación de Fondos fotográficos de procedencia variada.

Capítulo 3: Planificación del proyecto.

1. Metodología.

La experiencia propia en el desarrollo de aplicaciones es bastante escasa, debido a que únicamente hemos tenido una asignatura en la carrera y no ha sido lo suficientemente extensa (debido al tiempo de la misma) como para tener un buen conocimiento y dominio de como plantear el desarrollo de una aplicación Android.

Por ello, he considerado que la mejor manera de desarrollar dicho proyecto es realizar un diseño ágil. Todos los roles de desarrollo son llevados a cabo por mí mismo. Sin embargo, el cliente serán 2 personas. Una parte del cliente es mi tutor, Joaquín, que sabiendo a que tiene que ir dirigido el proyecto valora los resultados presentados. La segunda parte la realizo yo mí al ser quien prueba y valora la aplicación además de ser quien la dirige según los objetivos.

2. Tecnologías a utilizar.

Tarea	Tecnología	Conocimientos previos
Desarrollo DB	Django, Django REST Framework, Python, SQLite	Nivel bajo
Desarrollo Back-End	Java	Nivel alto
Desarrollo Front-End	Android, xml	Nivel medio
Tratamiento de datos	JSON	Nivel medio

Tabla 1: Tecnologías a utilizar.

3. Planificación y resultados.

La planificación ser realiza contando que cada día se trabajan 6 horas y que se trabaja de lunes a viernes. La fecha de inicio establecida es el 18 de abril y la fecha de fin estimada el 1 de julio. Lo cual quiere que decir que serían 12 semana de trabajo. La planificación de las tareas es la siguiente.

Nombre de la tarea	Duración estimada (días)	Duración real (días)
Análisis de requisitos	2	2
Generación de casos de uso	4	3
Esqueleto de la aplicación	2	2
Geolocalización de imágenes	2	2
Modelo de dominio	2	4
Implementación del servidor	4	6
Funcionalidad del mapa	5	8
Funcionalidad de la galería	5	6
Detalle de imágenes	5	6
Creación de servicio Android	5	7
Documentación	10	10
Manual de usuario	3	4
Total	**300**	**360**

Tabla 2: Planificación de tareas.

Como vemos en la tabla, prácticamente todas las tareas sufrieron retrasos. Esto se debe, como hemos dicho antes, a la falta de experiencia en el desarrollo Android, y por ello, las fechas de entrega eran holgadas.

Capítulo 4: Proceso de análisis, desarrollo e implementación.

1. Análisis de requisitos.

1.1. Requisitos funcionales.

RF-001 Geolocalización del usuario

Descripción: El sistema deberá conocer la ubicación del usuario en todo momento para poder localizarlo en el mapa, siempre y cuando, el usuario de su consentimiento.

Justificación: Objetivo principal del sistema.

RF-002 Mostrar marcadores en el mapa

Descripción: El sistema deberá dibujar en el mapa los diferentes marcadores correspondientes a ubicaciones con imágenes históricas en su localización exacta. Para ello, el servidor debe estar operativo o descargados los datos en el dispositivo móvil de antemano.

Justificación: Objetivo principal del sistema.

RF-003 Visualización de imágenes

Descripción: El sistema debe permitir al usuario visualizar las imágenes históricas con sus respectivas descripciones.

Justificación: Objetivo prioritario del sistema.

RF-004 Consulta de calles

Descripción: El sistema mostrará un listado de todas las calles con una o más imágenes existentes en el sistema.

Justificación: Facilidad de búsqueda de imágenes.

RF-005 Búsqueda de calles.

Descripción: El sistema deberá permitir al usuario realizar una búsqueda por nombre de aquellas calles con una o más imágenes, tanto en la galería de calles como en el propio mapa.

Justificación: Facilidad de búsqueda de calles.

RF-006 Descarga de datos

Descripción: El sistema deberá permitir al usuario realizar una descarga de todos los datos disponibles en el servidor para que la aplicación pueda cargar los datos off-line.

Justificación: Complemento para la aplicación.

RF-007 Notificación de geolocalización

Descripción: El sistema deberá mostrar notificaciones al usuario para avisarle de que se encuentra cerca de un punto con imagen.

Justificación: Complemento para la aplicación.

1.2. Requisitos no funcionales.

RNF-001 Accesibilidad

Descripción: El acceso a la aplicación será exclusivamente mediante Smartphone o tableta con sistema operativo Android con versión 16 o superior.

RNF-002 Ubicación de datos

Descripción: Todos los datos de la aplicación estarán alojados en un servidor REST Django con base de datos SQLite. A su vez, los datos pueden ser replicados en el dispositivo móvil en la base de datos interna SQLite.

RNF-003 Disponibilidad

Descripción: La aplicación debe estar operativa en cualquier instante, por lo que el servidor tiene que estar operativo el mayor tiempo posible.

RNF-004 Extensibilidad

Descripción: La aplicación estará abierta a futuros cambios o funcionalidades.

RNF-005 Escalabilidad

Descripción: El servidor debe tener la capacidad de responder de forma eficiente ante un gran número de usuarios realizando peticiones de descarga de datos de forma simultánea.

RNF-006 Agilidad

Descripción: El tiempo de respuesta debe de ser lo más rápido posible.

RNF-007 Usabilidad

Descripción: La aplicación debe resultar lo más intuitiva posible para todos aquellos usuarios familiarizados con dispositivos Android que interactúen con la aplicación.

RNF-007 Usabilidad

Descripción: La aplicación debe adaptarse al tamaño de texto del usuario que tenga predefinido en su dispositivo móvil.

RNF-008 Documentación

Descripción: La aplicación debe disponer de un manual de usuario para facilitar su uso.

2. Casos de uso.

2.1. Diagrama de casos de uso.

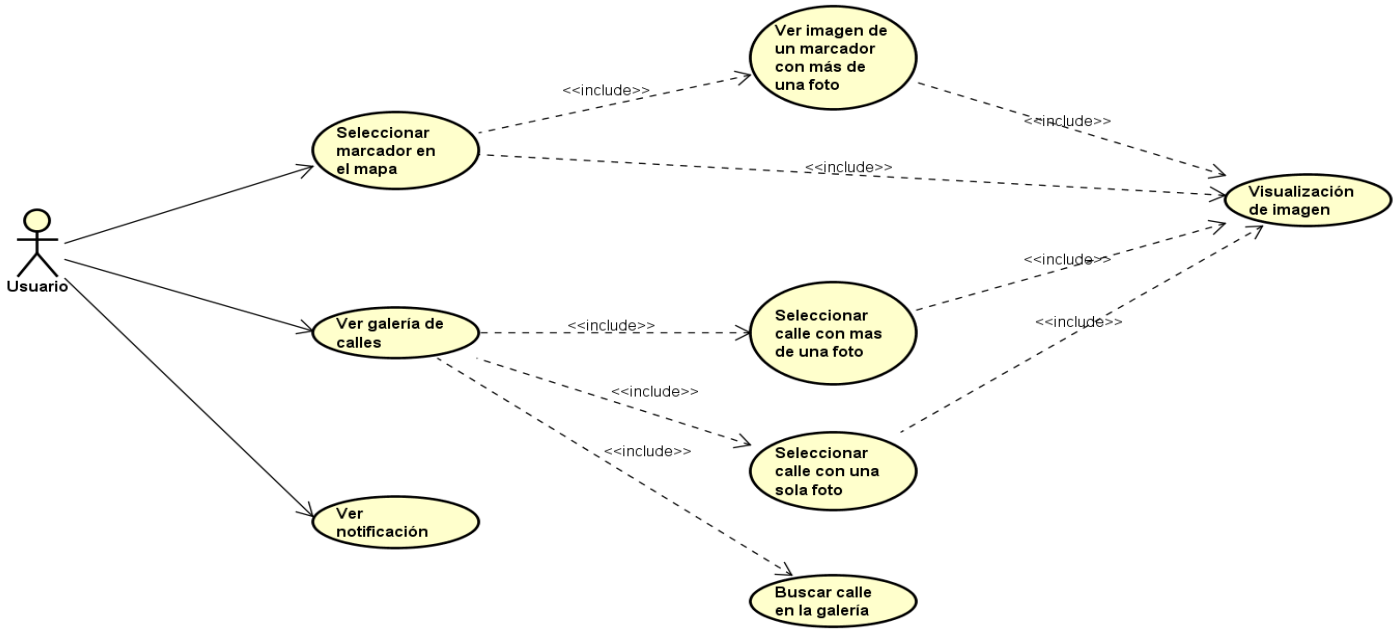

Ilustración 8: Diagrama de casos de uso.

2.2. Definición de casos de uso.

CU-001 Ver galería de calles

Requisitos asociados: RF-003

Pre-condición: El actor debe encontrarse en MapsActivity.

Eventos: 1 - El usuario abre el Drawer (menú).

2 - El sistema muestra el Drawer.

3 - El usuario selecciona “Gallería”.

4 - El sistema lanza la nueva actividad GalleryStreetActiviy.

5 – El sistema descarga los datos y los muestra al usuario.

Excepciones: 1 - El usuario no selecciona la opción y el caso de uso queda sin efecto.

2 - El usuario cancela en cualquier momento.

Post-condición: El usuario se encuentra en la actividad GalleryStreetActivity.

CU-002 Buscar calle en la galería

Requisitos asociados: RF-005

Pre-condición: Debe haberse cumplido el caso de uso CU-001.

Eventos: 1 - El usuario hace click en el icono de lupa.

2 - El sistema muestra el SearchView y activa el teclado.

3 - El usuario escribe el nombre de la calle.

4 - El sistema muestra todas aquellas calles que coincidan con el nombre buscado.

Excepciones: 1 - El usuario hace click en retorno o presiona el botón “back” del dispositivo y el caso de uso queda sin efecto.

Post-condición: El usuario ha filtrado la calle que quería.

CU-003 Seleccionar calle con más de una foto

Requisitos asociados: RF-003

Pre-condición: Debe haberse cumplido el caso de uso CU-001.

Eventos: 1 - El usuario hace click en una calle.

2 - El sistema comprueba el número de imágenes de la calle y descarga las mismas.

3 - El sistema lanza la nueva actividad GalleryImageActivity.

4 - El usuario hace click en una imagen.

5 - El sistema lanza la nueva actividad DetailActivity.

6 - El sistema descarga y muestra la imagen.

Excepciones: 1 - El usuario no selecciona ninguna calle y el caso de uso queda sin efecto.

2 - El usuario no selecciona ninguna calle y el caso de uso queda sin efecto.

3 - El usuario cancela en cualquier momento.

Post-condición: El usuario se encuentra en la actividad DetailActivity.

CU-004 Seleccionar calle con una sola foto

Requisitos asociados: RF-003

Pre-condición: Debe haberse cumplido el caso de uso CU-001.

Eventos: 1 - El usuario hace click en una calle.

2 - El sistema comprueba el número de imágenes de la calle y descarga la misma.

3 - El sistema lanza la nueva actividad DetailActivity.

4 - El sistema descarga y muestra la imagen.

Excepciones: 1 - El usuario no selecciona ninguna calle y el caso de uso queda sin efecto.

2 - El usuario cancela en cualquier momento.

Post-condición: El usuario se encuentra en la actividad DetailActivity.

CU-005 Seleccionar marcador en el mapa

Requisitos asociados: RF-002

Pre-condición: El actor debe encontrarse en MapsActivity.

Eventos: 1 - El usuario selecciona un marcador.

2 - El sistema muestra el cuadro de dialogo con la información del marcador.

3 - El usuario hace click en el cuadro de dialogo.

4 - El sistema comprueba el número de imágenes de dicho marcador

5 - El sistema lanza la nueva actividad GalleryImageActiviy o DetailActivity dependiendo del número de imágenes que tenga el marcador.

Excepciones: 1 - El usuario no hace click en el cuadro de dialogo y el caso de uso queda sin efecto.

Post-condición: El usuario se encuentra en la actividad GalleryImageActivity o DetailActivity.

CU-006 Ver imagen de un marcador con más de una foto.

Requisitos asociados: RF-009

Pre-condición: Se debe haber completado el caso de uso CU-005 con un marcador que tenga 2 o más imágenes y se encuentra en GalleryImageActivity.

Eventos: 1 - El usuario selecciona la imagen que quiere visualizar.

2 - El sistema lanza la actividad DetailActivity con dicha imagen.

3 - El sistema descarga y muestra la imagen.

Excepciones: 1 - El usuario hace click en retorno o presiona el botón “back” del dispositivo y el caso de uso queda sin efecto.

Post-condición: El usuario se encuentra en DetailActivity.

CU-007 Visualización de imagen.

Requisitos asociados: RF-009

Pre-condición: Se debe haber completado uno de los siguientes casos de uso: CU-003, CU-004, CU-005 Y CU-006.

Eventos: 1 - El usuario hace click en la imagen.

2 - El sistema expande la imagen para una mejor visualización.

3 - El usuario presiona el botón “back” y a continuación selecciona el texto.

4 - El sistema expande el texto para una mejor visualización.

Excepciones:

Post-condición: El usuario ha visualizado correctamente la imagen.

CU-008 Ver notificación

Requisitos asociados: RF-007

Pre-condición: El usuario tiene activada la geolocalización del dispositivo móvil y el servicio de Cronovisor activo.

Eventos: 1 - El usuario se encuentra en una localización cercana a una calle existente en el sistema.

2 - El sistema muestra una notificación al usuario.

3 - El usuario hace en uno de los 2 botones disponibles.

4 - El sistema muestra el mapa centrado en la calle que activó la notificación o muestra las imágenes de la misma.

Excepciones: 1 - El usuario desecha la notificación y este caso de uso queda sin efecto

Post-condición: El usuario se encuentra visualizando una calle en el mapa o las imágenes correspondientes a una calle

3. Diagrama de clases.

A continuación se muestra un diagrama de clases con las clases implementadas en la aplicación.

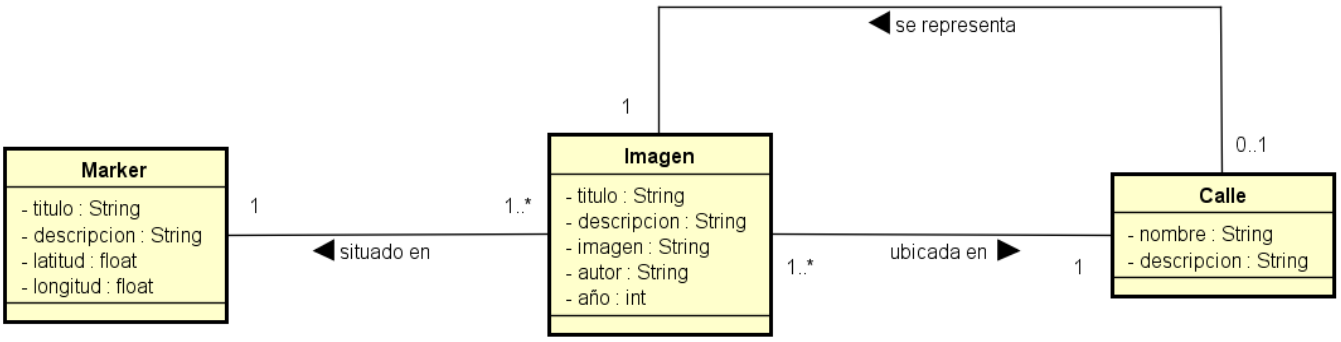

Ilustración 9: Diagrama de clases

3.1. Marker.

La clase Marker es la encargada de reflejar la existencia de imágenes en un punto concreto del mapa. La clase se transformara en la clase Marker propia de Google Maps. Como tal, tiene unas coordenadas geográficas que se utilizaran para ubicarlo en el mapa. Tiene un título y una breve descripción.

3.2. Imagen.

La clase Imagen es la encargada de reflejar las imágenes existentes en la base de datos. Cada imagen tiene un título, una descripción, un autor y un año aproximado en el que fue tomada. El atributo imagen es una url que contiene la imagen concreta en el servidor. La imagen se relaciona con los marcadores y las calles. La imagen está ubicada en un marcador y está contenida en una calle.

3.3. Calle.

La clase Calle es la encargada de reflejar las calles en las cuales se encuentra al menos una imagen en la base de datos. Una calle tiene un nombre de calle y una breve descripción de la misma. A su vez, una calle se relaciona con las imágenes, siendo la imagen relacionada la que aparecerá como imagen representativa de la misma.

4. Diagramas de secuencia.

4.1. Ver galería de calles.

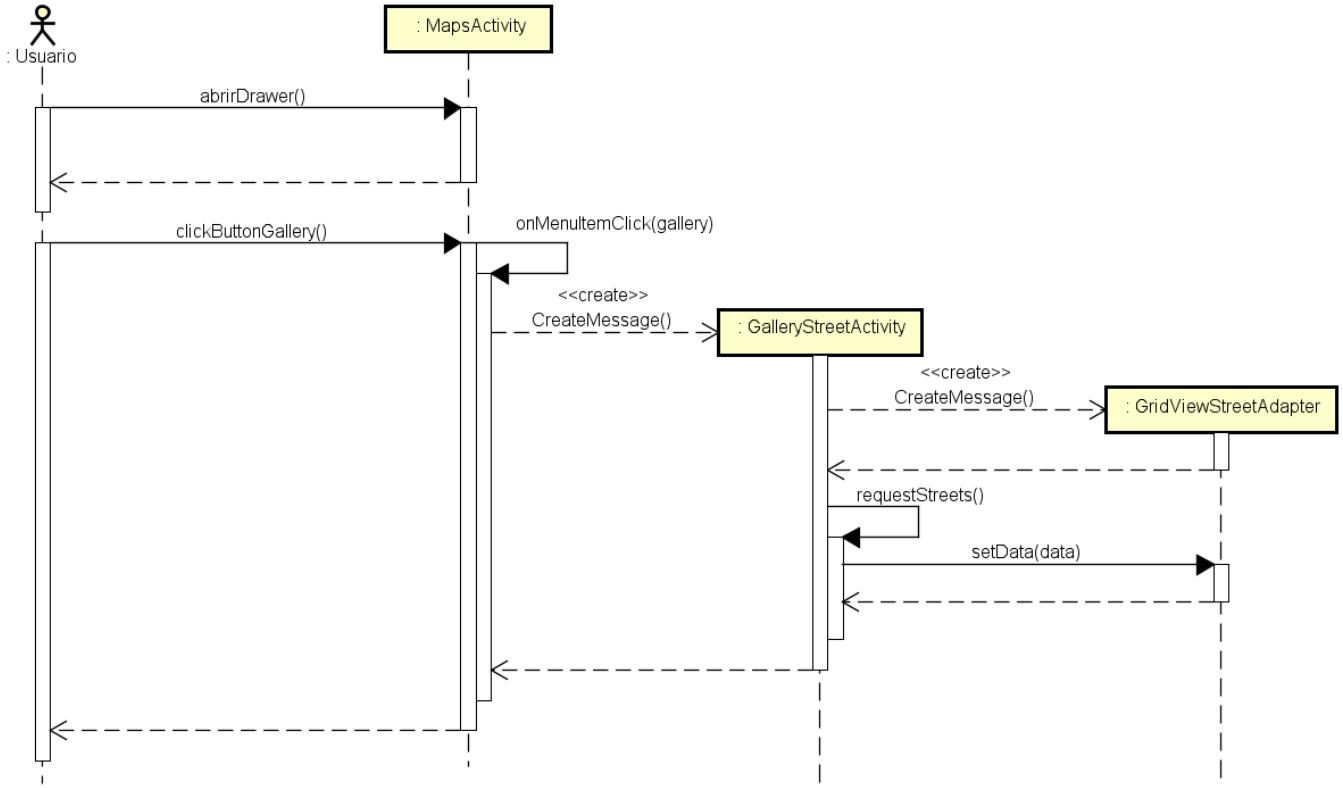

Ilustración 10: Diagrama ver galería de calles.

4.2. Seleccionar calle.

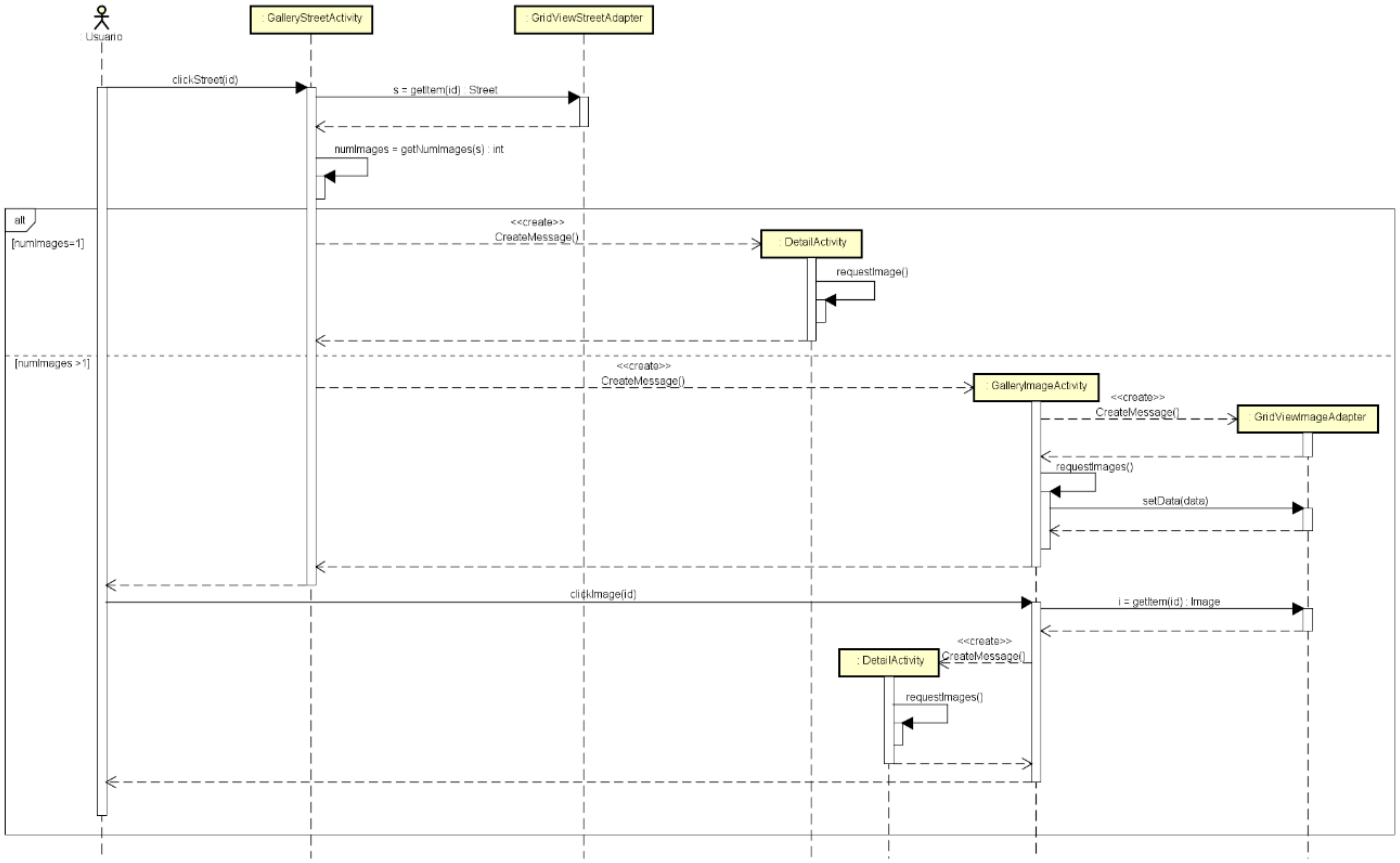

Ilustración 11: Diagrama seleccionar calle.

4.3. Buscar calle en la galería.

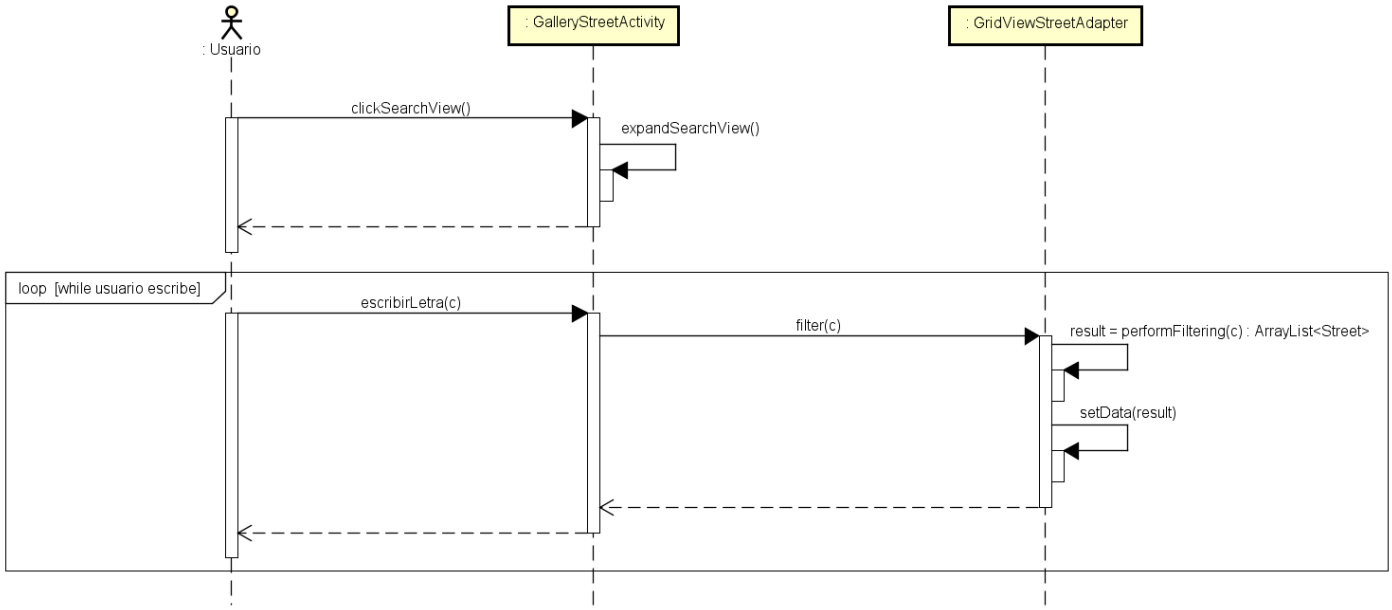

Ilustración 12: Diagrama buscar calle en la galería

4.4. Seleccionar marcador en mapa.

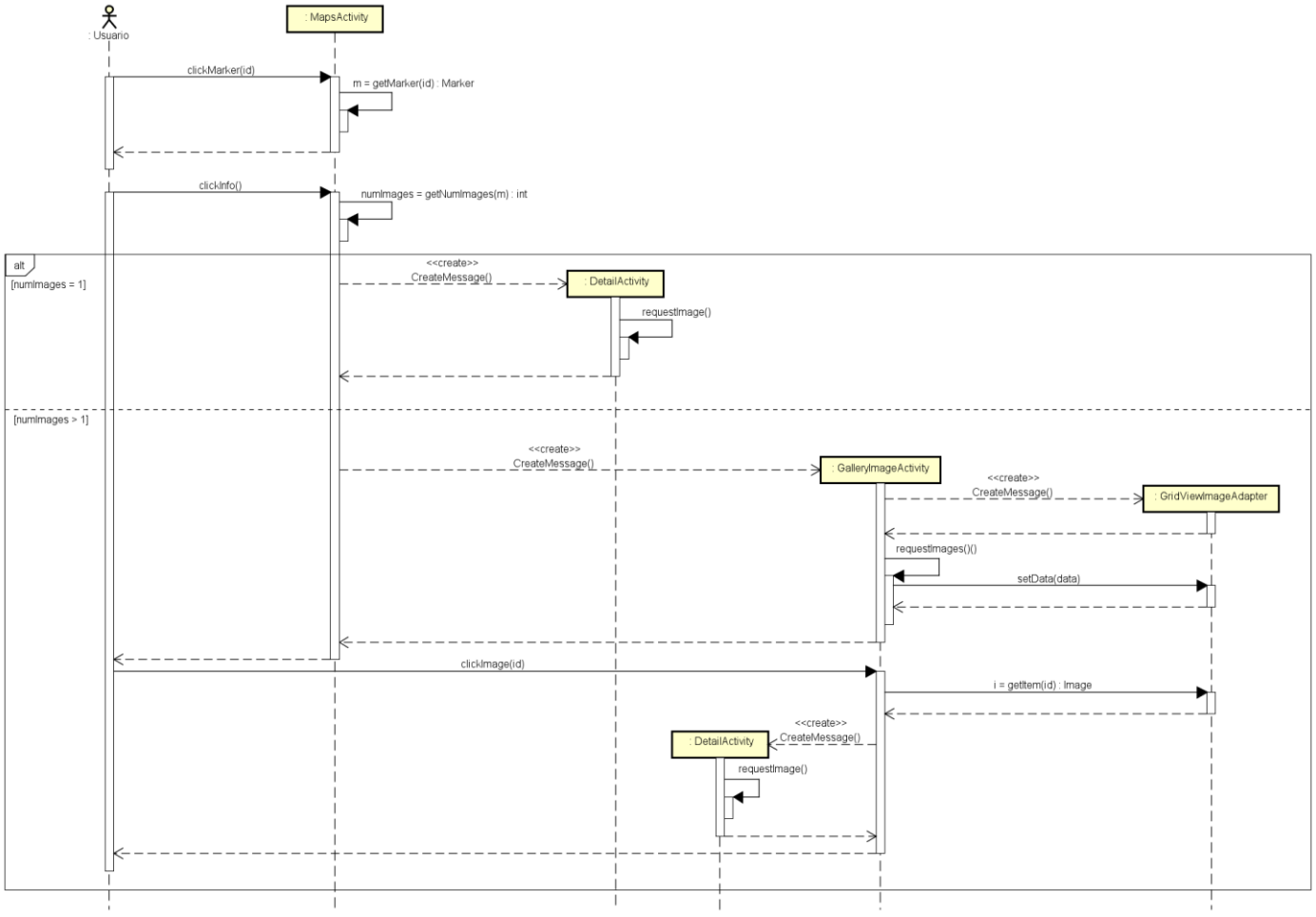

Ilustración 13: Diagrama seleccionar marcador en mapa.

4.5. Ver imagen.

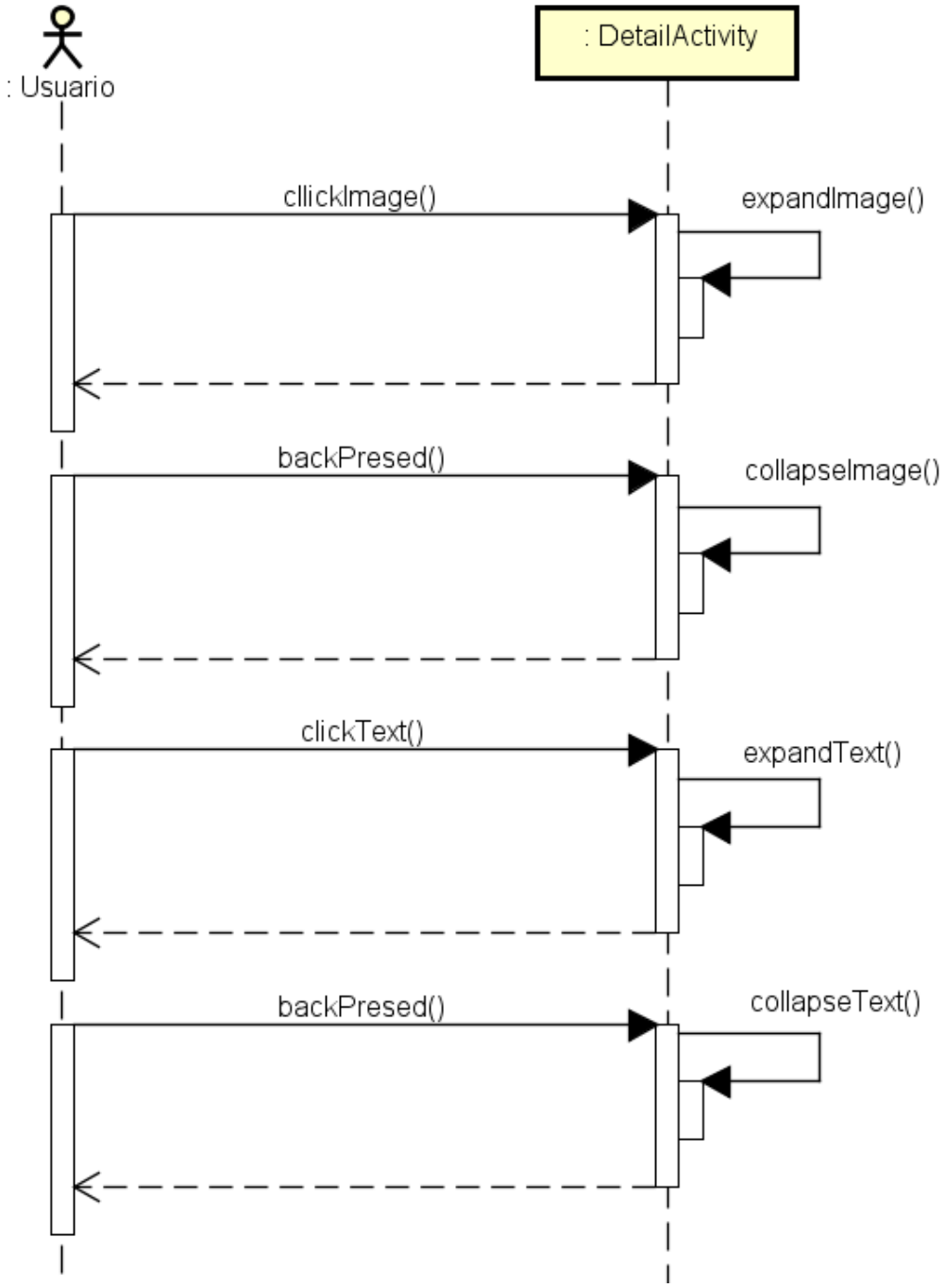

Ilustración 14: Diagrama ver imagen.

4.6. Ver notificación.

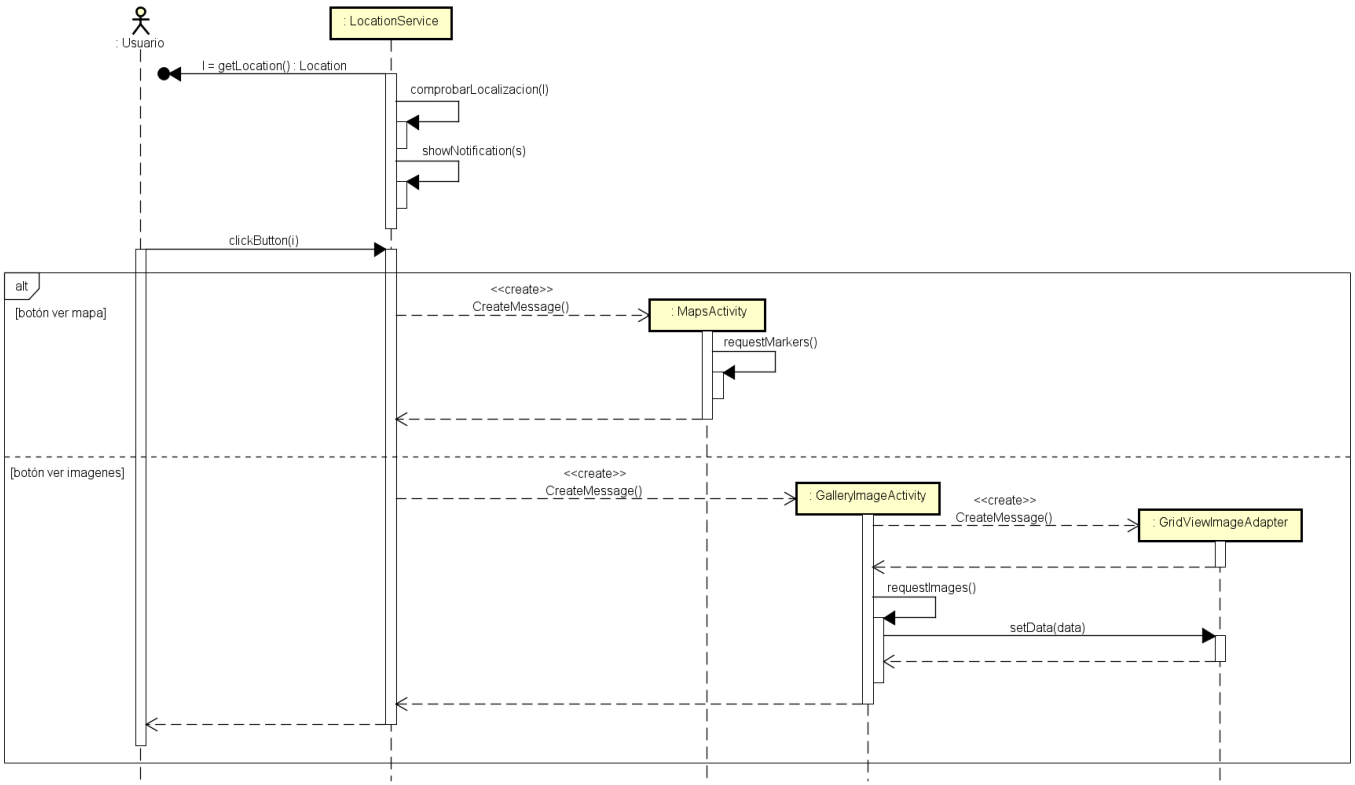

Ilustración 15: Diagrama ver notificación.

5. Diagrama Entidad-Relación.

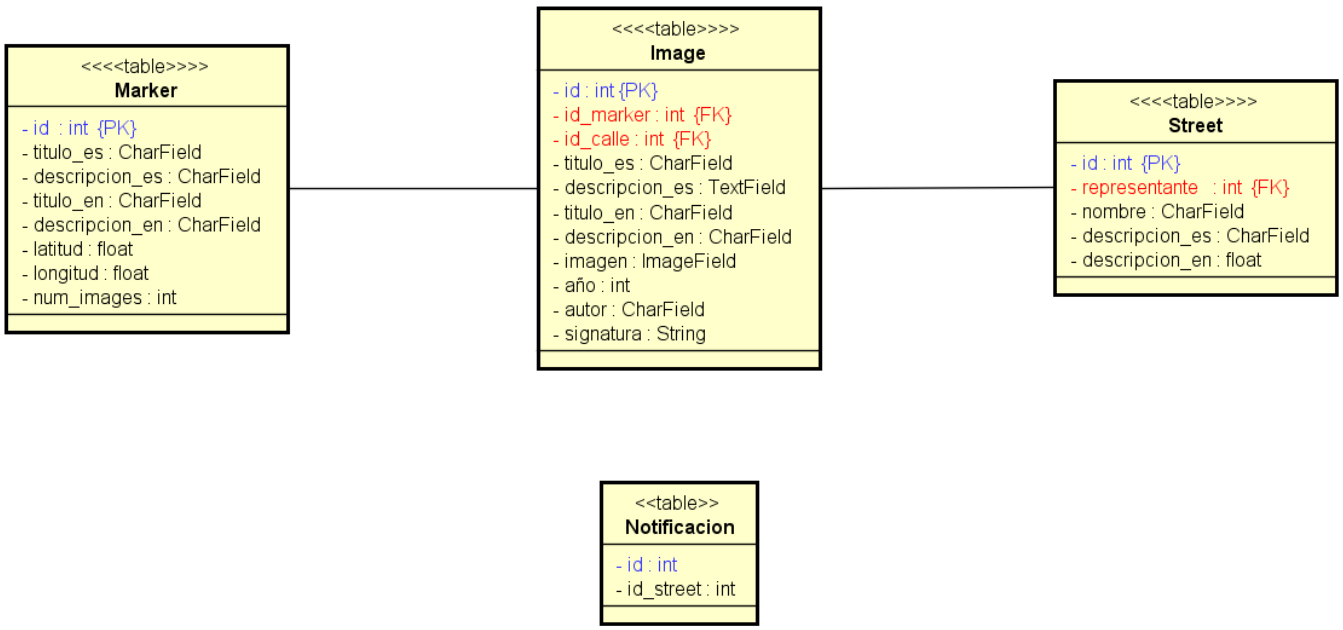

Ilustración 16: Diagrama Entidad-Relación.

Las clases Marker, Image y Street coinciden con las mostradas anteriormente en el diagrama de clases. Sin embargo, la clase Notificación es nueva. Cabe destacar que las tablas Marker, Image y Street tienen replicados los valores escritos que se mostraran al usuario. Esto se debe a que Android traduce los textos propios de la aplicación, pero no los descargados, por lo que es necesario tener las traducciones de los textos en base de datos.

- **Marker:** un marcador tiene un identificador único autoincremental (id). Tambien tiene un titulo de marcador, una breve descripción y sus coordenadas gps en decimal, siendo estas los 2 valores latitud y longitud. Ademas, tiene el campo num_images, que inserta el número de imágenes que tiene asociadas. Este valor es calculable haciendo una consulta a base de datos, pero para una mayor rapidez de computo y sobre todo, realizar menos consultas al servidor para ahorrar datos, este valor está en la tabla Marker.
- **Image:** una imagen tiene un identificador unico autoincrementar (id). Tambien tiene 2 claves foraneas que apuntan a un marcador (id_marker) y a una calle (id_street). A su vez, una imagen tiene un titulo, una descripcion, un año aproximado y un autor. El atributo signatura se utiliza para tener control sobre que imagen es la representativa y no es un valor que se utilice en la aplicación. El atributo imagen es una referencia a la imagen en el servidor. Las imágenes son guardadas en una estructura de carpetas predefinidas.
- **Street:** al igual que los marcadores y la imagen, una calle tiene un identificador unico (id). En lugar de tener una clave foranea a una imagen, una calle tiene la referencia a la imagen que es su representante. De esta manera, la descarga y la puesta a punto de la galería es mucho mas rapida y eficaz, ya que aunque no tenemos una correcta implementacion de clave foranea, el resto de valores de la imagen no nos interesa, y por ello los descartamos guardando unicamente el valor de la imagen.
- **Notificación:** la tabla notificación es la unica tabla de la base de datos que unicamente se guarda en el dispositivo. Se debe a que guarda una referencia a las calles ya visitadas por el usuario. Por ello, al ser personal, se guarda en el propio dispositivo. Dicha tabla ser referencia con un id unico y referencia a una calle.

6. Implementación del servidor.

Al estar pensada como una aplicación escalable, es preferible que los datos de la aplicación estén ubicados en un servidor externo. Además así no sobrecargamos la memoria del dispositivo móvil, lo cual haría de la aplicación indeseable para gran cantidad de personas.

De entre las opciones disponibles, la más acertada para los requisitos de la aplicación es un servidor REST. Para su implementación se ha utilizado Django REST Framework que ofrece una gran facilidad de configuración y gran cantidad de opciones. El servidor está alojado en un servidor interno de la Universidad de Valladolid, dedicado a dicho fin.

En Django hay que configurar 4 archivos clave:

- **Models.py:** representa las tablas que se guardaran en la base de datos. Las clases son todas propias de Django, que las transforma a valores correctos para el tipo de implementación de base de datos (en nuestro caso, SQLite). El campo ImageField requiere una pequeña explicación, ya que no es un campo usual. Django automatiza las operaciones sobre las imágenes, tanto la subida como la descarga de la misma, lo único que hay que indicar es en que ruta queremos guardar las imágenes. Para ello, se utiliza el método upload_to.
- **Serializers.py**: realiza la transformación de la clase models para poder ser transmitidos. Utilizamos la serialización básica de Django REST Framework.
- **Views.py:** modela los datos que serán mostrados. Utilizamos la modelación básica de Django REST Framework, indicando que campos queremos mostrar.
- **Urls.py:** indica en que direcciones url se mostraran los datos. También hay que hacer visibles las urls de las imágenes para que sean accesibles desde fuera.

Mediante peticiones a la url a cada una de las tablas se nos muestra una página web con los datos. Sin embargo, las peticiones desde la aplicación se solicitan con un formato fácil de tratas como es json:

- Todas las imágenes: http://virtual.lab.inf.uva.es:20202/image/?format=json
- Imagen con id 1: http://virtual.lab.inf.uva.es:20202/image/1/?format=json
- Todas las calles: http://virtual.lab.inf.uva.es:20202/street/?format=json
- Calle con id 1: http://virtual.lab.inf.uva.es:20202/street/1/?format=json
- Todos los marcadores: http://virtual.lab.inf.uva.es:20202/marker/?format=json
- Marcador con id 1: http://virtual.lab.inf.uva.es:20202/marker/1/?format=json
- Imágenes pertenecientes a la calle con id 1: http://virtual.lab.inf.uva.es:20202/imagesstreet/1/
- Imágenes pertenecientes al marcador con id 1: http://virtual.lab.inf.uva.es:20202/imagesmarker/1/
- La solicitud de imágenes se realiza directamente a la url en la que está ubicada dentro del servidor.

Django dispone de un servidor básico pero no muy potente. Sin embargo, para las peticiones que recibirá ahora mismo (1 o 2 simultáneamente) no hace falta un servidor más complejo. Sin embargo, si el número de peticiones aumentara, habría que implementar un servidor Apache integrándolo con Django.

El acceso a los datos es libre, cualquiera con el url puede ver los mismos, sin embargo, solo el usuario administrador podrá modificar o añadir nuevos datos. Para acceder como administrador, primero debemos loguearnos en el siguiente url http://virtual.lab.inf.uva.es:20202/admin/. Una vez estamos logueados, podemos acceder a las url anteriores (sin ¿format=json) y añadir o modificar los datos.

7. Actividades.

7.1. SplashScreenActivity.

Una splash screen es una pantalla que muestra una imagen al usuario con el logo de la aplicación, con lo que se consigue una mejor inmersión del usuario en nuestra aplicación. Es la primera actividad que se ejecuta de la aplicación, ya que es la actividad que tiene el intent de lanzamiento.

Tiene una funcionalidad adicional, iniciar el servicio Android que detallaremos más adelante.

7.2. MapsActivity.

Es la segunda actividad que el usuario ve al iniciar la aplicación. Es la encargada de mostrar el mapa al usuario con los marcadores que indican las localizaciones de las imágenes.

7.3. GalleryStreetActivity.

Esta actividad muestra un GridView con todas las calles que tienen una o más fotos en nuestra aplicación. Es accesible desde el menú de la aplicación. Las calles se pueden filtrar con la opción de búsqueda.

7.4. GalleryImageActivity.

La actividad no es accesible directamente, solo se puede acceder a ella desde un marcador que tenga más de una foto o desde la galería de calles con la misma condición. Sirve de tránsito para ver las imágenes.

7.5. DetailActivity.

Es la actividad que muestra la imagen concreta junto a su descripción al usuario. AL igual que GalleryImageActivity no es accesible directamente, debe accederse a ella desde una de las 2 opciones siguientes. O bien desde un marcador que solo tenga una imagen o bien desde la galería de imágenes. La imagen mostrada es ampliable para una mejor visualización.

8. Servicio Android.

Un elemento importante de las aplicaciones es como interaccionan con el usuario. La mayor parte de las aplicaciones actuales muestra notificaciones al usuario para comunicarles diferente información, ya sea el tiempo actual o que tiene un nuevo mensaje en la bandeja de entrada.

Cronovisor cuenta con un servicio Android cuya función es notificar al usuario cuando se encuentra en una calle con una o varias imágenes en nuestra aplicación. Con ello conseguimos 2 objetivos, uno, dar mayor visibilidad a la aplicación para que al usuario no se le olvide que está en su dispositivo, y dos, que el usuario no se pierda información por no entrar en la aplicación.

Sin embargo, una gran parte de los usuarios detesta las notificaciones recurrentes, por lo que es necesario tener un buen control sobre cuando y como se notifica al usuario.

El servicio web es iniciado la primera vez que se ejecuta la aplicación, y permanecerá activo hasta que el usuario lo para de forma manual o hasta que el sistema lo para por necesidad de recursos. Si el servicio ha sido parado, la próxima vez que se ejecute la aplicación, volverá a activarse. Al volver a iniciarse, los datos del servidor son descargados de nuevo, y por ello, podríamos recibir notificación de calles que ya hemos visitado. Para evitarlo, utilizamos la clase Notificación descrita anteriormente. Con ella, nos aseguramos de que a un mismo usuario no se le notifique 2 veces la misma calle. Además, para no consumir gran cantidad de recursos ni consumo de batería, el servicio solo está activo unos pocos segundos cada 5 minutos. Así, no solo ahorramos recursos, sino que damos tiempo al usuario a desplazarse entre diferentes calles.

A su vez, si el servicio no obtiene respuesta del servidor, dormirá durante 5 minutos y volverá a intentarlo.

Capítulo 5: Conclusiones y trabajo futuro.

1. Conclusiones.

El desarrollo de Cronovisor ha sido un aprendizaje continuo, sobre todo en el modo de desarrollo en Android, lo cual no se aprende en suficiente medida en la asignatura Aplicaciones móviles de la carrera.
Se ha conseguido realizar una aplicación que cumpla con los objetivos propuestos en un primer momento. Sin embargo, por la naturaleza de misma durante el desarrollo de la misma surgió gran cantidad de funcionalidad que se podría añadir a la misma. Algunas de esa funcionalidad son factibles y se expondrán en el punto siguiente. En cambio, algunas otras ideas no son posibles de llevar a cabo en una aplicación móvil y fueron descartadas.

A parte de lo aprendido respecto al aspecto tecnológico, la creación de Cronovisor me ha brindado la oportunidad de conocer puntos de Valladolid desconocidos para mí, y posiblemente, para gran cantidad de personas que viven en dicha ciudad.

Por todo ello, creo que elegir este tema para realizar el TFG ha sido un gran acierto y no lo cambiaria.

2. Trabajo futuro.

Existe una gran cantidad de contenido que se podría añadir a Cronovisor en el futuro:

- Añadir una opción de rutas para recorrer ciertos marcadores en un orden concreto, y a la vez, poder crear historias de edificios y localizaciones.
- Añadir nuevos lenguajes a la aplicación para tener una mejor interactuación con usuario que no hablen español o inglés.
- Añadir una descarga de datos más efectiva que la actual. Entre otras cosas, la comunicación entre la aplicación y el servidor para poder conocer si los datos descargados al dispositivo están o no actualizados.
- Añadir más contenido de imágenes y con descripciones interesantes.
- Modificación del modelo de clases y modelo de dominio para una mejor ampliación de la aplicación. También entraría en este apartado la traducción de las descripciones a una mayor cantidad de idioma.

Anexo.

1. Manual de usuario.

1.1. Inicio de la aplicación.

Una vez instalada la aplicación, buscaremos nuestra aplicación en el menú del dispositivo y la ejecutaremos. Una vez ejecutada, veremos la pantalla de carga, y automáticamente después aparecerá el mapa.

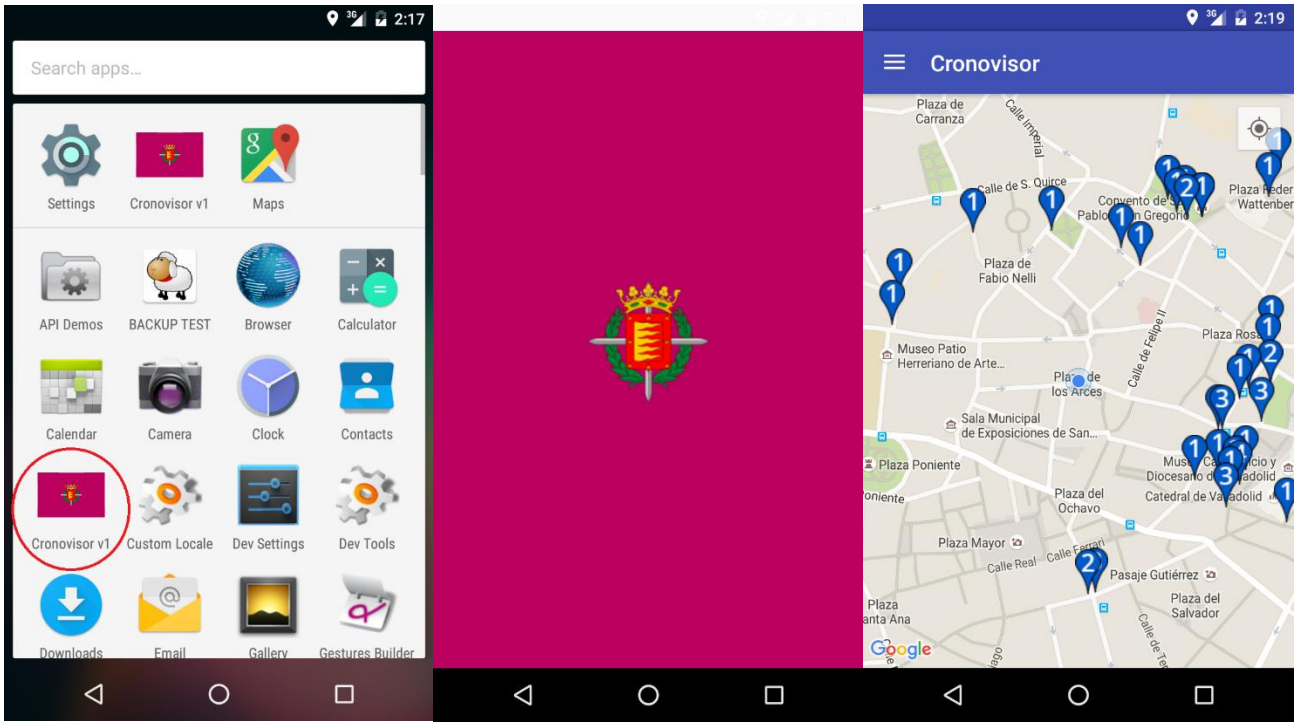

Ilustración 17: Inicio de App

1.2. Solicitud de localización GPS.

Si la versión de su dispositivo es Marshmallow (6.0.x), al iniciar la aplicación deberás de otorgar permiso tal y como se muestra en las siguientes imágenes.

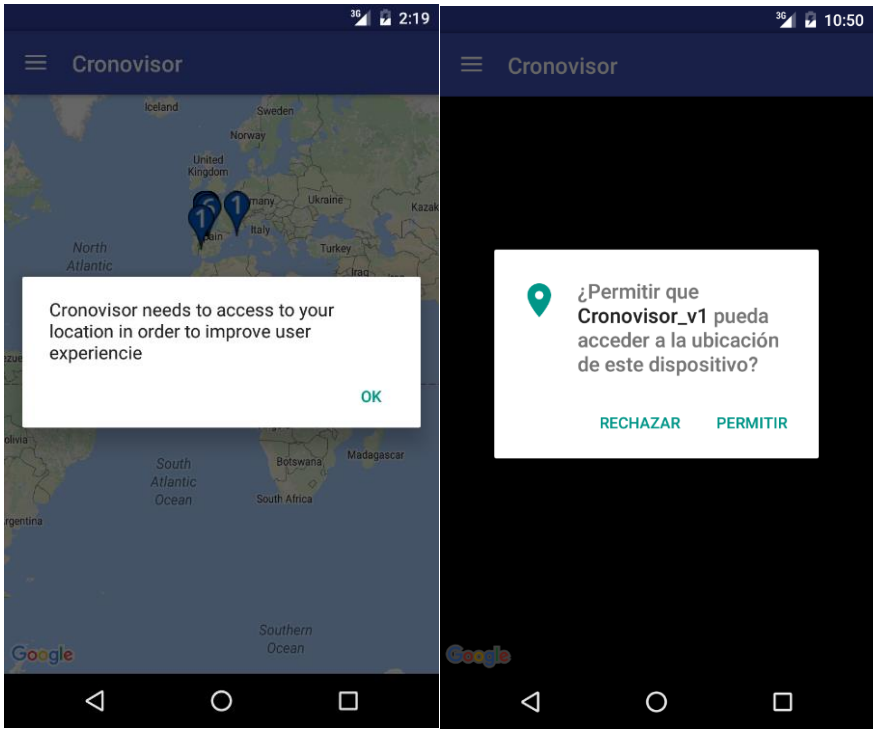

Ilustración 18: Solicitud autorización GPS.

Para el resto de dispositivos, el permiso a la ubicación se otorga al instalar la aplicación. Si nuestro dispositivo no tiene activada la localización GPS, la aplicación nos informara de ello y nos pedirá activarlo. Si no damos permiso para activar el sistema GPS, la aplicación continuara funcionando, pero no podremos aprovecharla al 100%. Por ello, es recomendable tener activada la localización GPS. El cuadro de dialogo es el siguiente, pudiendo variar dependiendo de la versión Android.

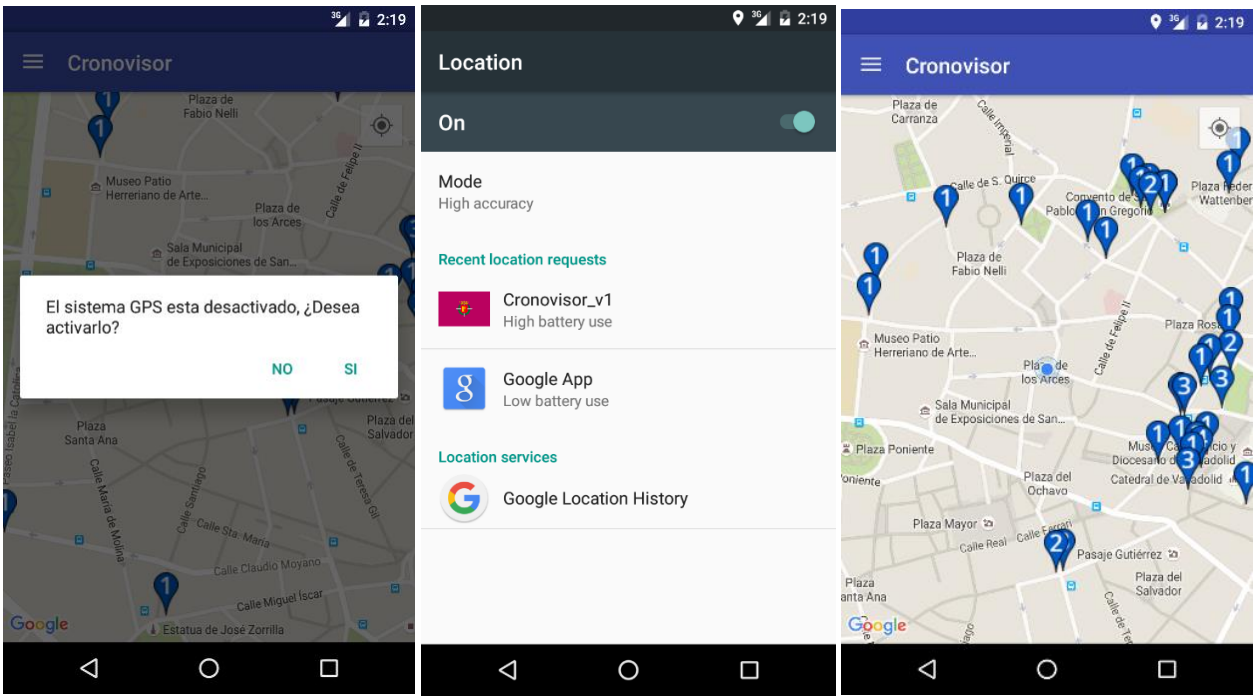

Ilustración 19: Activación GPS.

1.3. Selección de marcador.

Una vez en el mapa, si hemos dado permiso para acceder a la localización GPS, veremos que el mapa está centrado en nuestra ubicación. Si no es así, podemos hacer click en el recuadro para centrar el mapa en nuestra localización. Si no hemos otorgado permiso o no tenemos la ubicación GPS activa, el mapa aparecerá centrado en la plaza mayor de Valladolid. Veremos en el mapa una colección de marcadores, cada uno con un valor numérico. Estos valores representan el número de imágenes que encontraremos en dicho marcador.

Para visualizar las imágenes de un marcador, hacemos click en el mismo. De esta manera, aparecerá un pequeño cuadro con la descripción del marcador. Si hacemos click de nuevo en la ventana, iremos a las imágenes propias de dicho marcador.

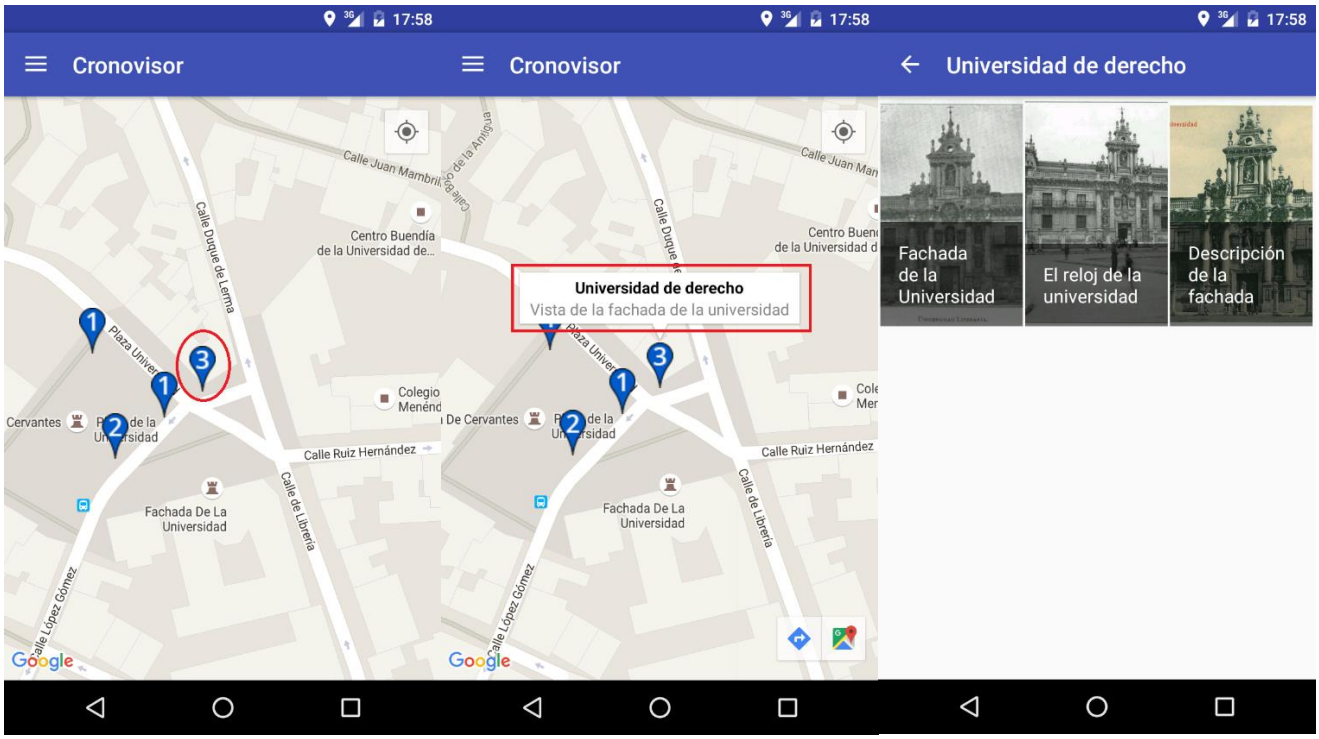

Ilustración 20: Selección de marcador.

1.4. Visualización de galería de calles.

Para acceder a la galería de imágenes, hacemos click en el icono de menú o bien desde la ventana de mapa, arrastramos el dedo de izquierda a derecha. Con cualquiera de las 2 opciones, accederemos al menú de la aplicación. En el aparecen todas las opciones disponibles. Hacemos click en galería para acceder a la misma.

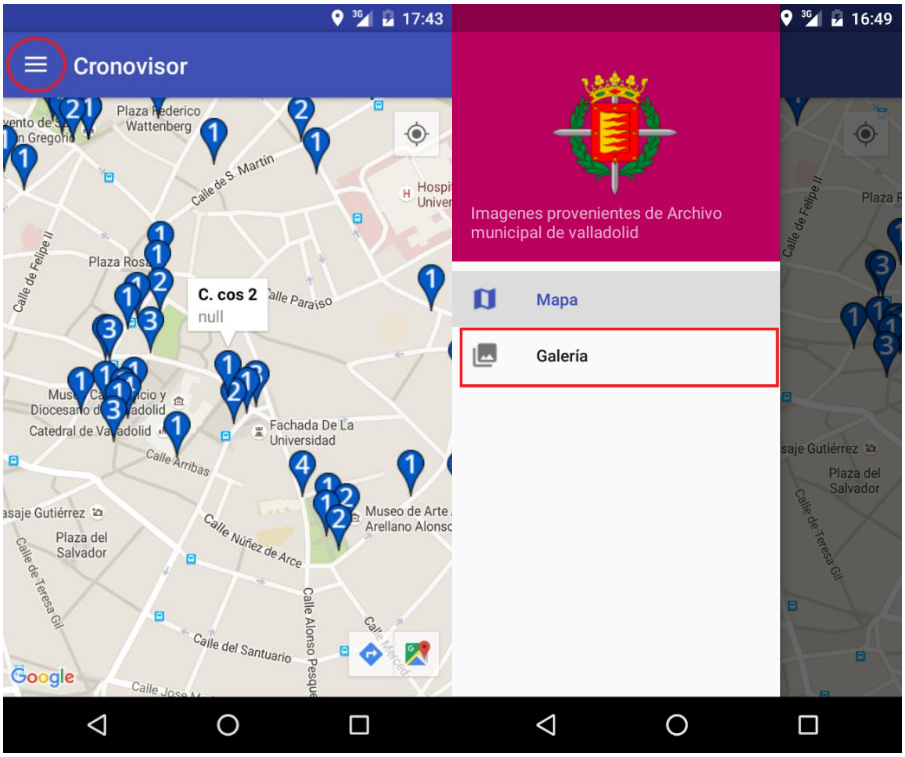

Ilustración 21: Selección galería.

Una vez en la galería, si hacemos click en el icono de opciones (3 puntos), nos aparecerá la opción “Vista en línea”. Si hacemos click en esta opción, cambiaremos el modo en el que se visualizan las calles, y en vez de ver las mismas en cuadricula, la veremos en línea. En la vista en línea veremos una breve descripción de la calle (no apreciable en la ilustración.

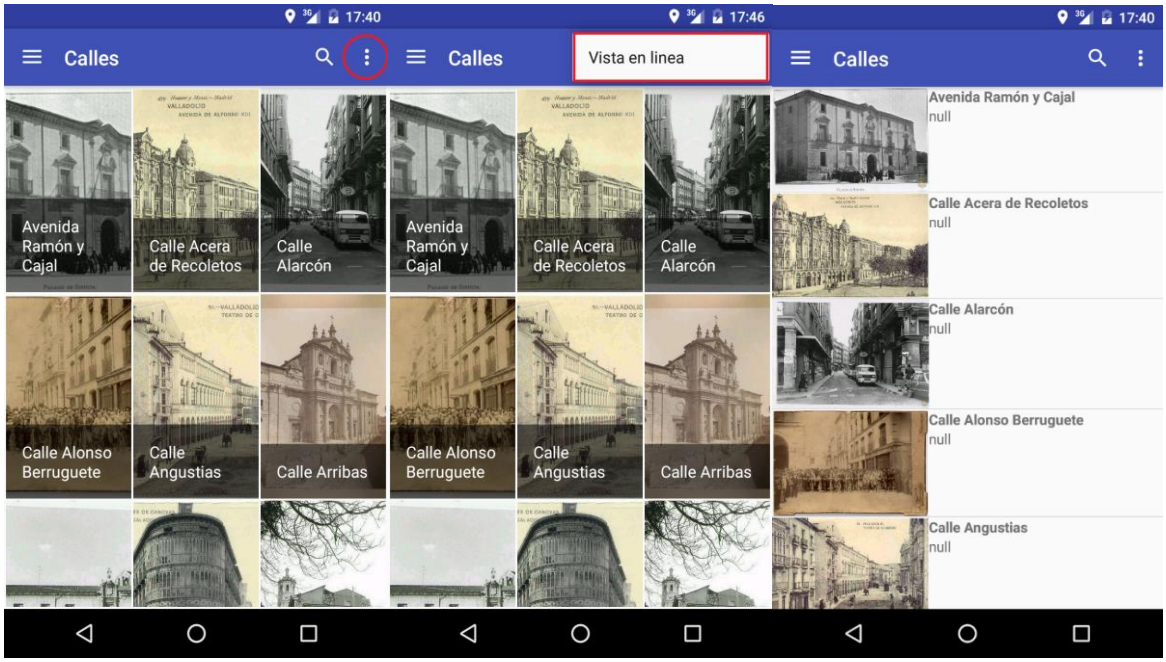

Ilustración 22: Galería de calles.

En cualquiera de los 2 modos de vista, podremos realizar una búsqueda en la galería por nombre de calle. Para ello, hacemos click en el icono de búsqueda (lupa) y escribimos el nombre de la calle. La aplicación realizará la búsqueda cada vez que introduzcamos una letra y nos mostrara aquellas calles que concuerden en nombre.

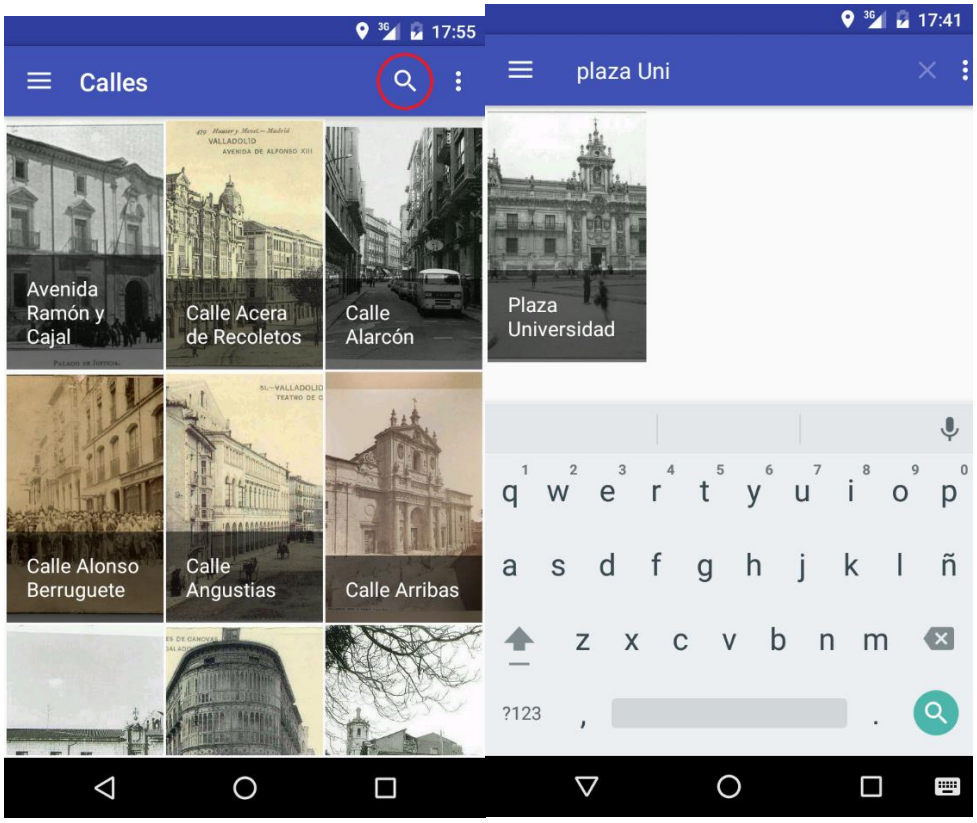

Ilustración 23: Búsqueda de calles.

Al hacer click en una de las calles, podemos ir a la imagen directamente si la única existente de dicha calle, o podemos ir a la galería de imágenes si tiene más de una foto.

1.5. Visualización de galería de imágenes.

A esta ventana se puede acceder de 2 formas diferentes. Una forma de acceder es desde la galería de calles, haciendo click en una de las mismas. Si hemos llegado de esta forma, la ventana que encontraremos será la siguiente. Si hacemos click en el icono de marcador podremos centrar el mapa en la calle en la que nos encontremos.

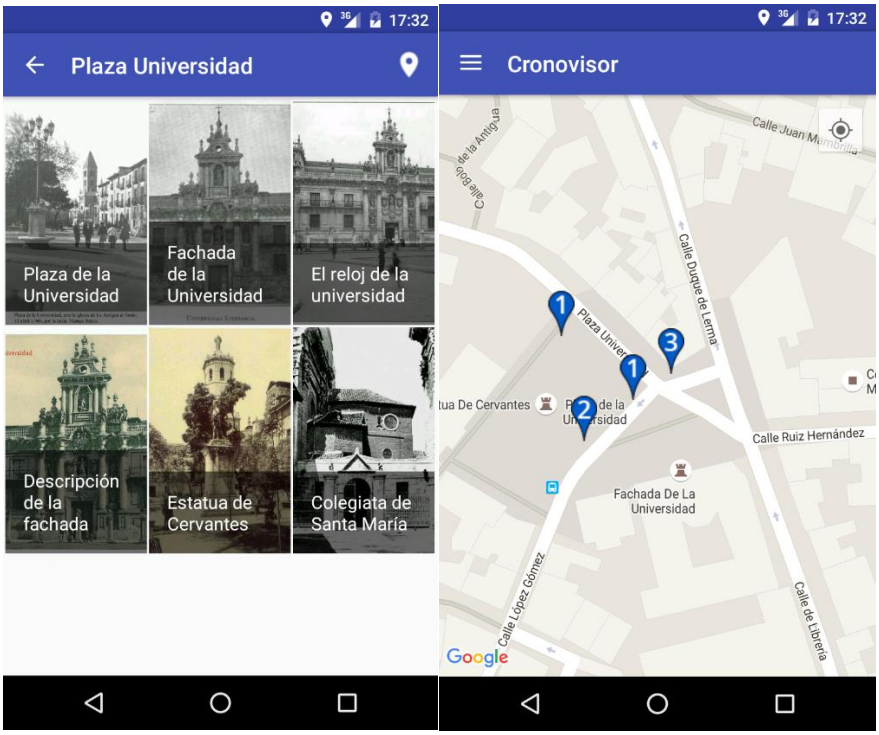

Ilustración 24: Galería de imágenes y ver en mapa

En cambio, si hemos accedido desde un marcador con más de una imagen, no dispondremos de la opción "ver en mapa", ya que procedemos del mismo y si quisiéramos volver a él, bastaría con pulsar el botón retroceso o la flecha de retorno de la parte superior izquierda.

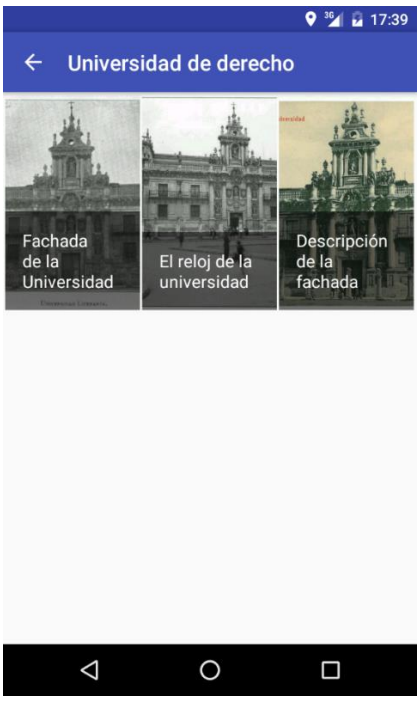

Ilustración 25: Galería de imágenes de marcador.

1.6. Visualización de imágenes.

Como hemos visto, a la visualización de imágenes hay 3 formas de llegar. O bien desde un marcador o bien, desde una calle con varias imágenes, o bien desde la galería de imágenes. Sea como sea, una vez en esta ventana veremos lo siguiente. La pantalla se bloquea automáticamente en horizontal para una mejor visualización.

Ilustración 26: Detalle de imagen.

Podemos ver la imagen a la izquierda de la pantalla y una descripción a la derecha. Para ver más de cerca y poder hacer ampliar donde queramos, basta con hacer click en la imagen para ampliarla.

Ilustración 27: Imagen expandida.

Si queremos ampliar el texto, la operación es la misma que con la imagen, hacemos click en el texto.

Ilustración 28: Expandir texto.

1.7. Notificaciones.

Cronovisor usa un servicio Android para realizar un seguimiento de la ubicación del dispositivo si el GPS está activo. Por ello, recursivamente nos aparecerán notificaciones. Estas notificaciones avisan de que el usuario se encuentra en una calle con imágenes históricas. Es importante recalcar que una vez hayamos recibido una notificación de una calle, no volverá a aparecernos la misma notificación a no ser que borremos los datos de la aplicación, ya que esta acción la reiniciaría completamente. La notificación es la siguiente.

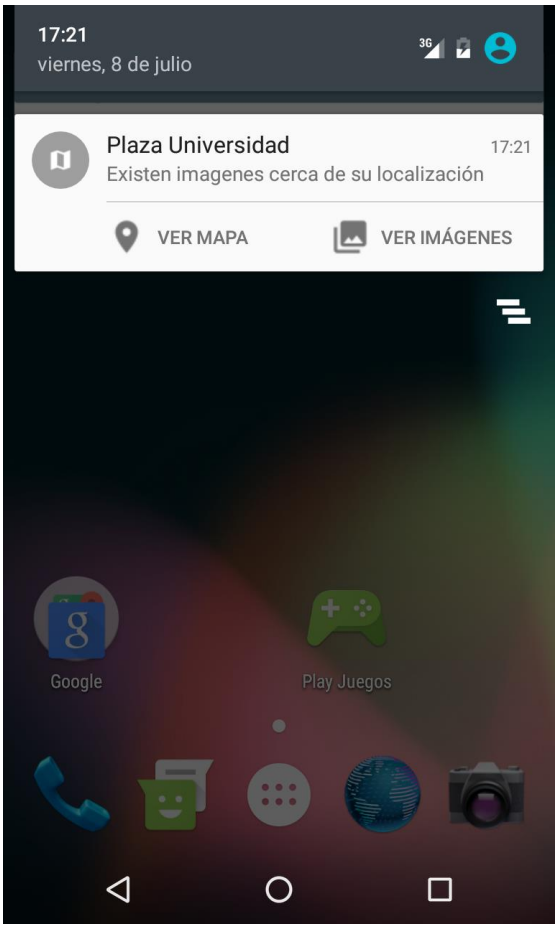

Ilustración 29: Notificación

La notificación permite 3 opciones. Descartarla de la manera tradicional, ver el mapa haciendo click en “Ver mapa” o ver las imágenes de la calle en “Ver imágenes”. Si hacemos click en la primera opción, la aplicación se ejecutara y centrará el mapa en la calle que activo la notificación. Si por el contrario hacemos click en la segunda opción, “Ver imágenes”, la aplicación se ejecutará y nos mostrará la galería de imágenes de la calle que activo la notificación.

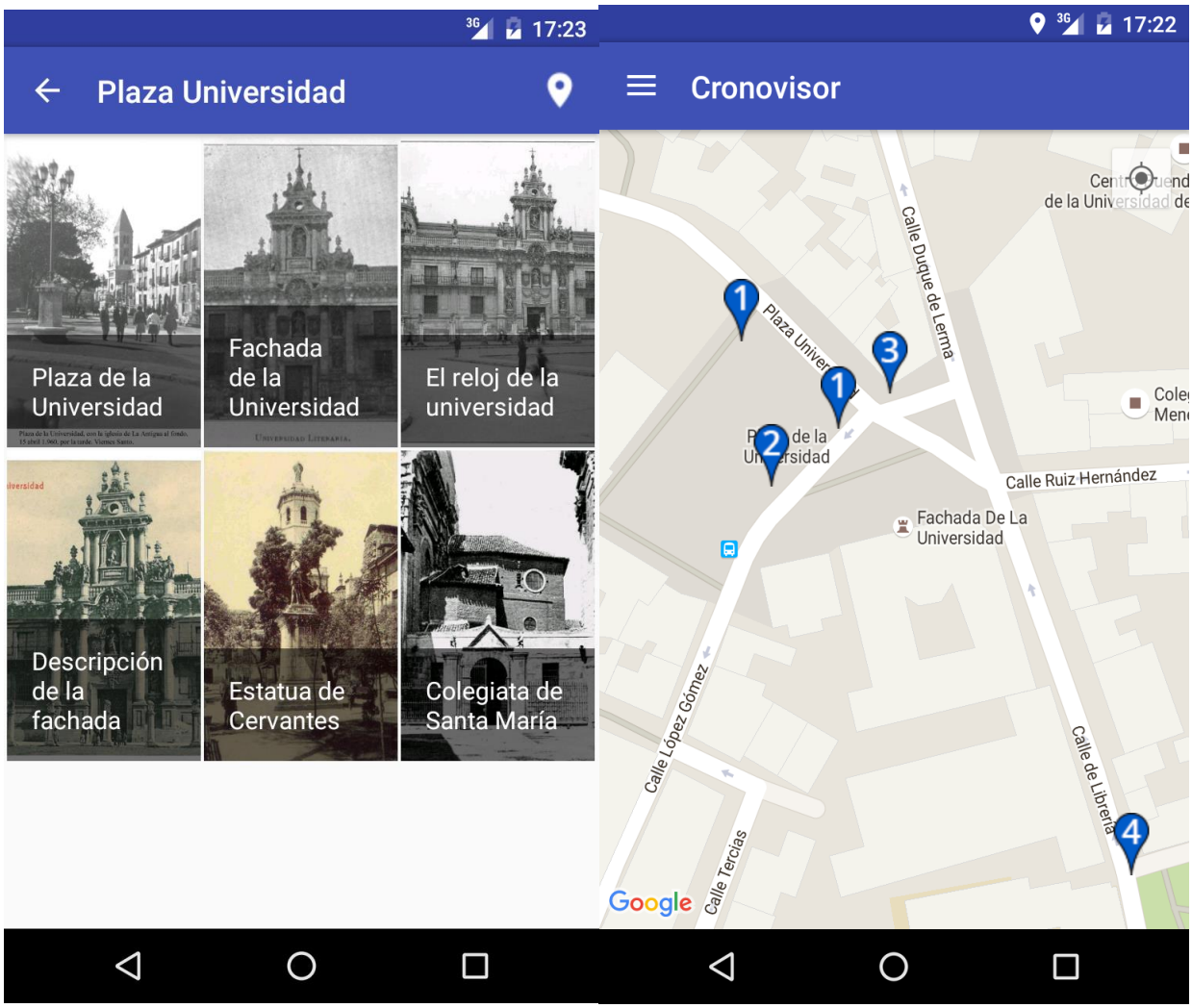

Ilustración 30: Notificación ver imágenes y ver mapa.

1.8. Mensajes de error.

Existen varios tipos de mensajes de error que nos podremos encontrar mientras utilizamos la aplicación. El más común, es aquel que nos indica que el servidor de la aplicación no está activo. Si nos encontramos con ese mensaje de error, deberemos esperar a que el servidor vuelva a estar operativo. Los motivos por los que el servidor puede estar inactivo son variados, desde una caída en el mismo hasta una tarea de mantenimiento.

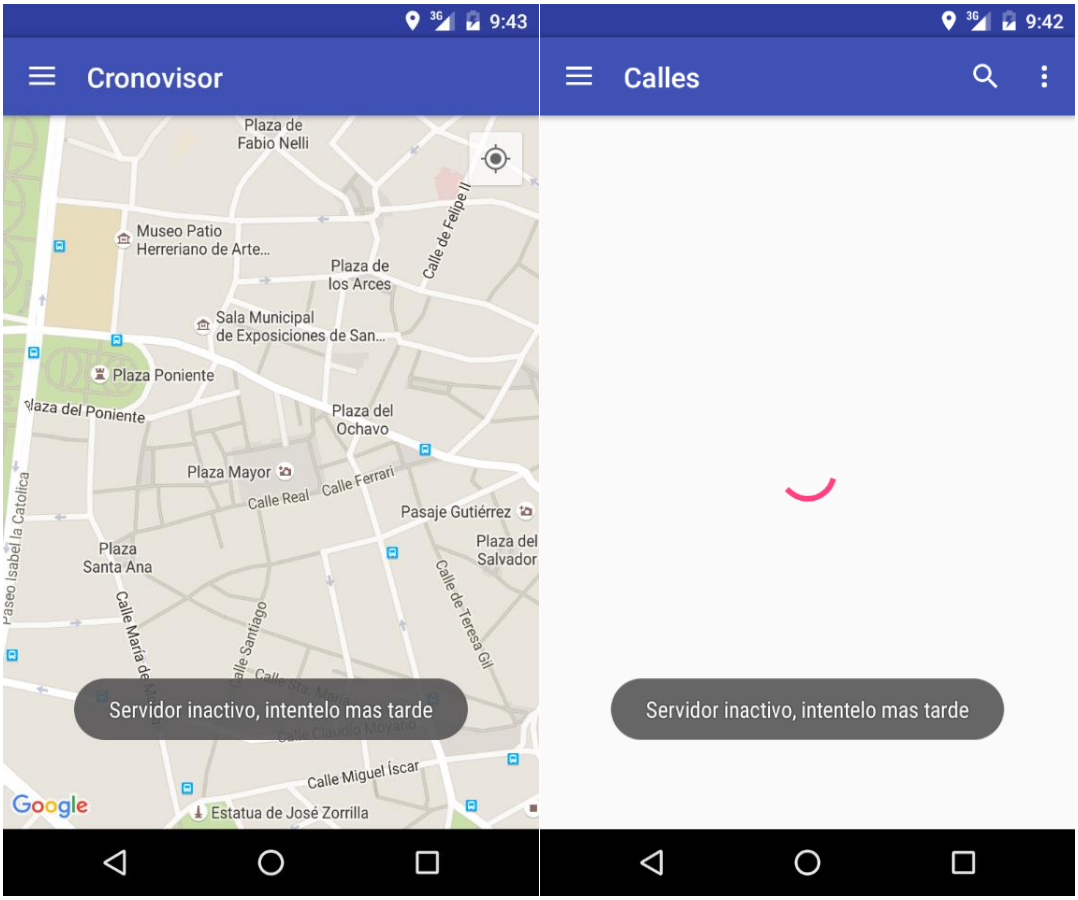

Ilustración 31: Mensaje de error servidor inactivo.

Pero también, puede haber ocurrido un error interno. Estos errores se producen cuando los datos descargados no son correctos, debido a errores en los mismos, o cuando ha ocurrido un error puntual en el sistema al procesar los datos. Este error puede ocurrir en cualquiera de las pantallas de la aplicación, ya que todas descargan y procesan datos del servidor.

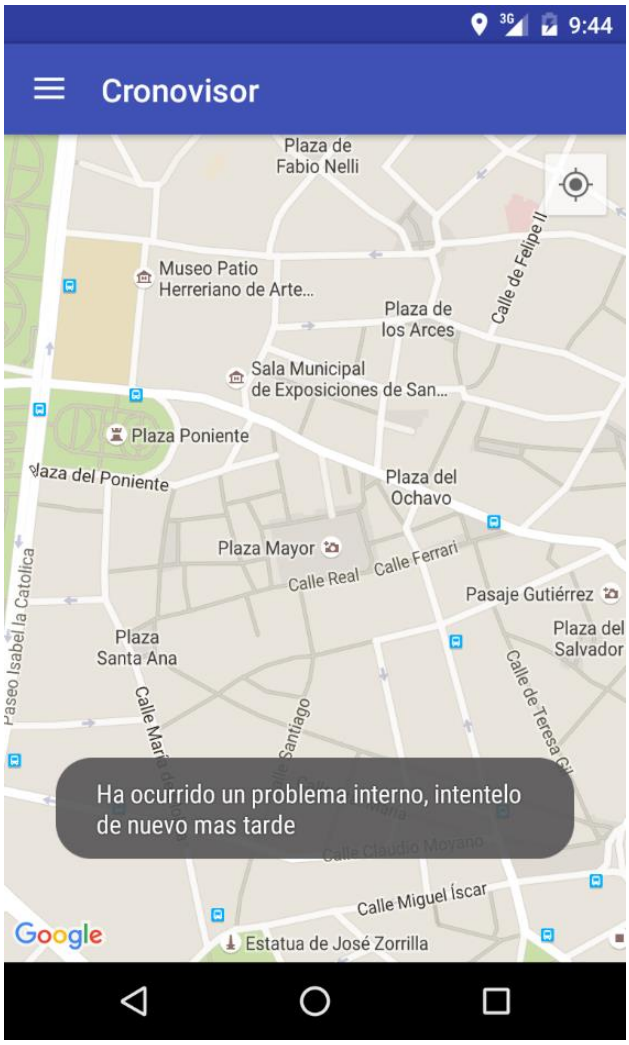

Ilustración 32: Mensaje de error problema interno.

2. Manual de instalación del servidor.

2.1. Instalación en Linux.

A continuación se detalla el proceso de instalación necesario para poder poner en marcha el servidor de la aplicación Cronovisor en distribuciones Linux.

1.1. Instalamos Python 2.7:

```
$ sudo apt-get install Python
```

1.2. Instalamos la herramienta de control de versiones de Git:

```
$ sudo apt-get install git
```

1.3. Clonamos el repositorio del proyecto desde GitHub:

```
$ git clone https://github.com/adrmart/CronovisorServer
```

1.4. Instalamos pip:

```
$ sudo apt-get install python-pip
```

1.5. Instalamos las librerías de Python necesarias:

```
$ pip install django
```

```
$ pip install django-rest-framework
```

1.6. Ejecutamos la migración de la base de datos:

```
$ python manage.py makemigrations
```

1.7. Ejecutamos la migración de la base de datos asociada al proyecto, proceso que creará o actualizará dicha base de datos:

```
$ python manage.py migrate
```

1.8. Creamos un usuario administrador (en caso de no existir ya uno):

```
$ python manage.py createsuperuser
```

1.9. Ejecutamos el servidor web:

```
$ python manage.py runserver 0.0.0.0:80
```

Nota I: si se desea que la aplicación se ejecute en una dirección IP diferente o en un puerto diferente, basta con cambiar los valores numéricos antes expuestos.
Nota II: para que la aplicación funcione correctamente, será necesario cambiar las urls si se instala en otro servidor.

2.2. Instalación en Windows.

A continuación se detalla el proceso de instalación necesario para poder poner en marcha un servidor con la aplicación Cronovisor en distribuciones Windows.

1.1. Instalamos Python 2.7:
Descargar desde la página oficial de Python: https://www.python.org/downloads/

1.2. Instalamos la herramienta de control de versiones de Git:
Descargar desde la página oficial de Git: https://git-scm.com/download/win

1.3. Clonamos el repositorio del proyecto desde GitHub:
$ git clone https://github.com/adrmart/CronovisorServer

1.4. Instalamos pip:
$ python get-pip.py

1.5. Instalamos las librerías de Python necesarias:

```
$ pip install django
$ pip install django-rest-framework
```

1.6. Ejecutamos la migración de la base de datos:
$ python manage.py makemigrations

1.7. Ejecutamos la migración de la base de datos asociada al proyecto, proceso que creará o actualizará dicha base de datos:

```
$ python manage.py migrate
```

1.8. Creamos un usuario administrador (en caso de no existir ya uno).

```
$ python manage.py createsuperuser
```

1.9. Ejecutamos el servidor web:
$ python manage.py runserver 0.0.0.0:80

Nota: si se desea que la aplicación se ejecute en una dirección IP diferente o en un puerto diferente, basta con cambiar los valores numéricos antes expuestos.
Nota II: para que la aplicación funcione correctamente, será necesario cambiar las urls si se instala en otro servidor.

3. Manual de instalación de la aplicación.

3.1. Play store.

La aplicación estará disponible a través de la Play Store siendo gratuita. Para instalarla, vamos la Play Store, buscamos Cronovisor y hacemos click en la opción Instalar. Con ello empezara el proceso de instalación.

3.2. APK.

Para instalar la aplicación mediante el archivo APK, basta con ejecutar el mismo en el dispositivo. Generalmente, el dispositivo nos preguntará si estamos seguros de que queremos instalar la aplicación de esta manera. Es recomendable volver a activar la medida de seguridad una vez se haya instalado la aplicación.

4. Pruebas.

En este apartado se recoge el resultado de los test de usabilidad realizados para probar la aplicación con usuarios reales.

4.1. Test de usabilidad.

Las pruebas se van a realizar directamente sobre la aplicación para detectar posibles fallos o confusiones de los usuarios durante la ejecución de las pruebas.

Además, de esta manera, podemos utilizar métricas tales como el recuerdo y la respuesta emocional, que de otra forma no podrían ser consideradas.

Los test que se realizaran son los siguientes:

- Iniciar aplicación activando GPS y centrando el mapa en uno mismo.
- Ver una imagen de marcador cercano ampliando texto e imagen.
- Ver una imagen de una calle ampliando texto e imagen.
- Buscar una calle y acceder a sus imágenes.
- Tratar con una notificación accediendo mediante una de las 2 opciones a la aplicación.

Las métricas de usabilidad que se tendrán en consideración son las siguientes:

- Tiempo de realización.
- Recuerdo de la operación.
- Valoración personal del usuario.

4.2. Resultados.

El test fue aplica cado a 2 personas. La primera es una persona joven (23 años) valorado como usuario de nivel medio de dispositivos y aplicaciones Android. El segundo usuario es una persona mayor (53 años) valorado como usuario de bajo nivel de dispositivos y aplicaciones Android.

Resultados del primer usuario:

Actividad	Tiempo (s)	Recuerdo (1-10)	Valoración (1-10)
Iniciar aplicación	20	9	8
Ver una imagen de marcador cercano	15	10	8
Ver una imagen de una calle	10	10	10
Buscar una calle	14	9	7
Tratar con una notificación	8	10	10

Tabla 3: Resultados test usabilidad usuario medio

Resultados del segundo usuario:

Actividad	Tiempo (s)	Recuerdo (1-10)	Valoración (1-10)
Iniciar aplicación	25	9	7
Ver una imagen de marcador cercano	19	9	8
Ver una imagen de una calle	13	10	10
Buscar una calle	20	9	9
Tratar con una notificación	10	10	9

Tabla 4: Resultados test usabilidad usuario bajo

4.3. Conclusiones.

Una vez concluidas las pruebas y analizados los resultados, podemos sacar varias conclusiones.

- La aplicación es intuitiva en sus operaciones básicas.
- La mayor dificultad encontrada está relacionada con los cambios de ventana a la hora de solicitar el acceso al GPS y en la ventana de búsqueda de calle al introducir el texto a buscar. Sin embargo, estas 2 opciones son las predefinidas por Google, por lo que no se deben modificar.
- El diseño de la aplicación se ha valorado de forma positiva y los textos se han leído correctamente.
- El recuerdo de las operaciones es sencillo, por lo que los usuarios rebajan el tiempo de realización de la tarea cuando ya la han realizado una vez.

5. Contenido del CD.

En el CD adjunto, podemos encontrar los siguientes elementos:

- Memoria: una copia completa de este documento en formato PDF.
- Diagramas: un fichero astah con los diagramas creados para este trabajo.
- Proyecto Android: proyecto Android completo. Sin embargo, el mismo proyecto se encuentra en github para mayor seguridad (https://github.com/adrmart/Cronovisor)
- Servidor: todos los archivos y carpetas necesarias para la puesta a punto del servidor. Para saber más ver la sección Manual de instalación del servidor.
- Fichero APK: contiene el fichero instalable en dispositivos Android. Si se desea instalar la aplicación, es necesario tener el servidor activo.

Bibliografía.

Maps.

1. https://developers.google.com/maps/documentation/android-api/intro?hl=es [Último acceso: Abril 2016]
2. http://www.androidcurso.com/index.php/tutoriales-android/41-unidad-7-seguridad-y-posicionamiento/223-google-maps-api-v2 [Último acceso: Abril 2016]
3. http://gestdb.piensayactua.com/blog/?x=entry:entry131219-182331 [Último acceso: Abril 2016]
4. http://developer.android.com/intl/es/guide/topics/location/strategies.html [Último acceso: Abril 2016]

Servidor y descarga de datos.

5. http://expocodetech.com/descargar-datos-de-un-servidor-en-android/ [Último acceso: Mayo 2016]
6. http://stackoverflow.com/questions/15549421/how-to-download-and-save-an-image-in-android [Último acceso: Mayo 2016]
7. http://www.genbetadev.com/desarrolloparastartups/como-crear-un-servicio-rest-en-30-lineas-de-codigo-de-django-y-python [Último acceso: Mayo 2016]
8. http://www.machinalis.com/blog/image-fields-with-django-rest-framework/ [Último acceso: Mayo 2016]
9. http://blog.enriqueoriol.com/2014/07/upload-de-imagenes-con-django.html [Último acceso: Mayo 2016]

Galerías y detalle de imágenes.

10. http://movalink.blogspot.com.es/2013/08/galeria-de-imagenes-usando-swipe.html [Último acceso: Junio 2016]
11. http://www.hermosaprogramacion.com/2015/07/tutorial-para-crear-un-gridview-en-android/ [Último acceso: Junio 2016]
12. http://www.androidbegin.com/tutorial/android-search-filter-listview-images-and-texts-tutorial/ [Último acceso: Junio 2016]
13. https://www.youtube.com/watch?v=kGcP61OQPW8 [Último acceso: Junio 2016]
14. https://developer.android.com/training/animation/zoom.html [Último acceso: Junio 2016]
15. http://javatechig.com/android/download-and-display-image-in-android-gridview [Último acceso: Junio 2016]
16. http://stackoverflow.com/questions/27378981/how-to-use-searchview-in-toolbar-android [Último acceso: Junio 2016]
17. http://stackoverflow.com/questions/31489109/android-searchview-in-toolbar [Último acceso: Junio 2016]

Servicio Android.

18. https://developer.android.com/reference/android/app/Service.html [Último acceso: Julio 2016]
19. http://www.hermosaprogramacion.com/2015/07/tutorial-para-crear-un-servicio-en-android/ [Último acceso: Julio 2016]
20. http://www.sgoliver.net/blog/tareas-en-segundo-plano-en-android-ii-intentservice/ [Último acceso: Julio 2016]
21. https://developer.android.com/reference/android/app/Notification.Builder.html#addAction%28android.app.Notification.Action%29 [Último acceso: Julio 2016]
22. http://stackoverflow.com/questions/35647821/android-notification-addaction-deprecated-in-api-23 [Último acceso: Julio 2016]
23. http://androcode.es/2012/09/notificaciones-metodo-tradicional-notification-builder-y-jelly-bean/ [Último acceso: Julio 2016]
24. http://stackoverflow.com/questions/6775257/android-location-providers-gps-or-network-provider [Último acceso: Julio 2016]
25. https://developer.android.com/reference/android/location/Address.html [Último acceso: Julio 2016]

26. https://developer.android.com/guide/topics/location/strategies.html?hl=es [Último acceso: Julio 2016]
27. https://developer.android.com/reference/android/location/Geocoder.html [Último acceso: Julio 2016]

Splash screen

28. https://amatellanes.wordpress.com/2013/08/27/android-crear-un-splash-screen-en-android/ [Último acceso: Mayo 2016]

Aplicaciones existentes.

29. http://www.ubicuostudio.com/es/resenas-de-apps/realidad-aumentada-apps-museos/ [Último acceso: Junio 2016]
30. http://findingshakespeare.co.uk/eye-shakespeare-v2 [Último acceso: Julio 2016]

Iconos.

31. https://design.google.com/icons/ [Último acceso: Julio 2016]

Android.

32. https://es.wikipedia.org/wiki/Android [Último acceso: Junio 2016]
33. https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android [Último acceso: Junio 2016]
34. http://www.hermosaprogramacion.com/2014/08/aprendiendo-la-arquitectura-de-android/ [Último acceso: Junio 2016]