



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**título del TFG  
Documentación Técnica**



Presentado por nombre alumno  
en Universidad de Burgos — 6 de febrero  
de 2020

Tutor: nombre tutor



---

# Índice general

---

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>III</b>
<b>Índice de tablas</b>	<b>IV</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	2
A.3. Estudio de viabilidad . . . . .	7
<b>Apéndice B Especificación de Requisitos</b>	<b>11</b>
B.1. Introducción . . . . .	11
B.2. Objetivos generales . . . . .	11
B.3. Catalogo de requisitos . . . . .	11
B.4. Especificación de requisitos . . . . .	11
<b>Apéndice C Especificación de diseño</b>	<b>13</b>
C.1. Introducción . . . . .	13
C.2. Diseño de datos . . . . .	13
C.3. Diseño procedimental . . . . .	15
C.4. Diseño arquitectónico . . . . .	15
<b>Apéndice D Documentación técnica de programación</b>	<b>17</b>
D.1. Introducción . . . . .	17
D.2. Estructura de directorios . . . . .	18
D.3. Manual del programador . . . . .	19

D.4. Compilación, instalación y ejecución del proyecto . . . . .	30
D.5. Pruebas del sistema . . . . .	30
<b>Apéndice E Documentación de usuario</b>	<b>31</b>
E.1. Introducción . . . . .	31
E.2. Requisitos de usuarios . . . . .	31
E.3. Instalación . . . . .	32
E.4. Manual del usuario . . . . .	35

---

## Índice de figuras

---

---

## Índice de tablas

---

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

### **A.1. Introducción**

En este apartado trataremos varios temas importantes a la hora de realizar un trabajo.

Uno de ellos será la planificación temporal, a través de la cual sabremos la asignación de tiempos que tendremos que llevar durante todo el trascurso del proyecto.

Todo el trabajo se basa en el desarrollo de metodologías de trabajo como SCRUM, con la cual se puede conseguir una identificación mas sencilla de la asignación de tareas y tiempos y la inversión de tiempo en proporción al total de horas que llevará el proyecto.

Otro es el estudio de la viabilidad, donde trataremos dos puntos esenciales, la viabilidad económica y la viabilidad legal. Por la parte económica se tratará de simular las consecuencias que conllevan tratar unas herramientas u otras a la hora de invertir dinero en el proyecto o de que otras personas vayan a invertirlo cuando se trate venderlo.

La viabilidad legal se centrará un poco más en el estudio de los derechos de copyright, así como el trato que haremos de los datos. Por último también se tendrá en cuenta la seguridad en cuando a protección de los datos y las capas de seguridad del sistema que tratamos.

## A.2. Planificación temporal

El objetivo inicial al planificar un proyecto es definir lo más claro posible el marco de tiempo y de recursos que se llevarán a cabo durante el desarrollo del proyecto. Como ya mencionábamos antes, se basa en la metodología SCRUM.

Dado que este proyecto ha sufrido bastantes cambios sobre la planificación y desarrollo del mismo desde que se comenzó hasta la actualidad, haremos una comparación entre lo que inicialmente se planeó y lo que finalmente resulta, a fin de poder ver todo lo que un trabajo como este, donde no se definen unos objetivos claros desde un inicio, o se definen objetivos sin planificar las consecuencias puede variar y evolucionar.

El proyecto comienza el día 5 de abril de 2019 y finaliza el día 13 de febrero de 2020, con una duración bruta de 10 meses. Ésta duración también se debe, como mencionábamos antes, a los cambios que ha ido sufriendo el desarrollo del proyecto. Igualmente, también hay que tener en cuenta que no se ha dedicado todo el tiempo a la realización del proyecto, ya que también era necesario aprobar las asignaturas restantes así como realizar prácticas laborales durante todo este período. También se terminó aplazando de la entrega esperada (Septiembre) debido a la imposibilidad de presentar en esta convocatoria por no cumplir los requisitos de créditos máximos presentables en la misma.

Por todo esto que se expone, la realización del proyecto ha tenido una evolución irregular, con meses más activos que otros debido a la complicación para realizar reuniones entre los tutores y los alumnos participantes del mismo, ya que Ignacio se desplazó durante este proceso a Málaga a realizar estudios superiores y las reuniones eran a través de plataformas electrónicas por obligación.

Igualmente, se trata desde el inicio del proyecto de utilizar la metodología SCRUM y por ello vamos a desarrollar las partes que ha conllevado:

- SCRUM se basa en la división de las tareas y recursos en intervalos de tiempo, por lo que será como se desarrolle.
- Cada uno de los sprints conllevará unas tareas a realizar durante ese intervalo. Se trató de que las reuniones se realizaran cada 2 semanas, o una vez al mes.



El comienzo del proyecto se realizó con una reunión estándar con el tutor Bruno Baruque, el cual nos comentó el punto de partida y los objetivos iniciales que podríamos tener para comenzar el trabajo.

### **Sprint 1 (05/04/19 - 19/04/19)**

En esta primera reunión el tutor nos explicó los objetivos iniciales y tuvimos las siguientes tareas:

- Búsqueda de documentación sobre Android.
- Desarrollo de objetivos iniciales de cara a la memoria.
- Consulta inicial del proyecto base del que partíamos.

### **Sprint 2 (19/04/19 - 03/05/19)**

Tras haber tenido esta primera reunión, pudimos comentar el conocimiento que cada uno tenía sobre la plataforma sobre la que trabajaríamos así como el lenguaje de programación. En mi caso, nunca había utilizado el programa de Android Studio ni programado para un entorno Android, aunque si que conocía el lenguaje de programación Java.

Se propuso lo siguiente:

- Comenzar a nutrirse de tutoriales y vídeos explicativos sobre Android.
- Instalar la herramienta de Android Studio.
- Pensar en la viabilidad de actualizar la versión en la que se encuentra el proyecto, así como las posibilidades de cambiar de herramienta de desarrollo, porque en ese momento el proyecto se había desarrollado en Eclipse, con un plugin para Android debido a la inexistencia de Android Studio en esa época.

### **Sprint 3 (03/05/19 - 17/05/19)**

En este encuentro, se trata las posibilidades de lo comentado en la reunión anterior, concluyendo que el cambio de herramienta de desarrollo será algo positivo y relativamente fácil de hacer, pero sin tener claro las posibilidades sobre la actualización de versiones.

Se trata de desarrollar lo siguiente para el siguiente sprint:

- En caso de plantear la actualización, buscar la versión actual más óptima hacia la que se debería hacer, para evitar versiones inestables y con bugs.
- Instalar el proyecto inicial a través de maquinas virtuales y probar todas las funcionalidades que contiene, para conocer bien sobre lo que vamos a trabajar.
- Seguir desarrollando apartados referentes de la memoria para ir quitando carga de trabajo al final.

### **Sprint 4 (17/05/19 - 07/06/19)**

En esta reunión, tras tratar los temas anteriores, se concluye que la versión a la que actualizar debe ser la API 23, viniendo desde la API 11. Supone un gran salto pero si queremos que la aplicación sea funcional con la tecnología de hoy en día habrá que usar una versión viable. También se comenta acerca del desconocimiento inicial sobre la versión que tenía el proyecto, ya que ésta tenía 7 años de antigüedad, lo cual ya hace pensar que tendremos dificultades.

Se intenta realizar lo siguiente:

- Instalación del proyecto base sobre la nueva herramienta de desarrollo, para intentar que este quede ya funcional en Android Studio.
- Documentación sobre cambios de versiones 7 años atrás en Android hasta la versión sobre la que se desarrolla.

### **Sprint 5 (07/06/19 - 05/07/19)**

Reunión cercana a los meses de verano, por lo que se intenta dejar objetivos más a largo plazo, aunque también comentando todas las dificultades que se van encontrando a la hora de instalar y hacer que funcione el proyecto sobre la nueva herramienta:

- Conseguir que definitivamente la aplicación alcance un estado de funcionamiento correcto sobre Android Studio.
- Aplicar el cambio de versión sobre el proyecto y por consiguiente comenzar a solucionar todos los errores de base que se encuentren.

### **Sprint 6 (05/07/19 - 03/09/19)**

En esta reunión, tras observar la inmensa cantidad de problemas que supone hacer que funcione la aplicación sobre Android Studio y por consiguiente actualizarla, los objetivos de la anterior reunión siguen siendo los mismos con algunos añadidos.

El problema principal parece ser que la gran mayoría de librerías que se utilizaron en el proyecto inicial han dejado de tener soporte y han dejado de funcionar, por lo que todo ese código es ahora inservible y habrá que buscar sustitución.

Se plantean los objetivos anteriores añadiendo:

- Desarrollo de la memoria, tratando de quitar carga a futuro.
- Arreglo de errores con bibliotecas.
- Documentación sobre nuevas APIs utilizables para sustituir todas las que ya están obsoletas.

### **Sprint 7 (03/09/19 - 04/11/19)**

Una reunión tratando de ver los avances del verano, para después tratar de poner objetivos a largo plazo. En este momento ya se presentan más dificultades para trabajar en el proyecto y tener reuniones habituales, ya que en mi caso comienzo a trabajar y mi compañero Ignacio se traslada en este momento a Málaga para continuar sus estudios.

Por estos motivos, a partir de los objetivos que se tenían inicialmente, se obtienen unos más realistas teniendo en cuenta la situación:

- Continuar con todos los arreglos necesarios sobre la aplicación, ya que según se van solucionando algunas incompatibilidades se van descubriendo nuevas.
- Intentar continuar el desarrollo de la memoria.

### **Sprint 8 (04/11/19 - 18/11/19)**

Primera reunión en la que contamos con nuestro compañero Ignacio a través de videollamada. Tras el tiempo que ha pasado, se intenta tener un seguimiento sobre las tareas que habían sido asignadas. Por desgracia,

estas tareas no varían demasiado, ya que supone un trabajo importante la documentación sobre los métodos que tienen que ser sustituidos para incluir otros nuevos, sobre los que también es necesario documentarse previamente, sobre un lenguaje de programación relativamente nuevo.

### **Sprint 9 (18/11/19 - 02/12/19)**

Tras haber ido trabajando, en esta nueva reunión se presenta un problema que ha ido siendo recurrente durante el desarrollo del proyecto, ya que aunque se han solucionado muchos problemas, la conexión entre la parte del cliente y la del servidor no parece funcionar, lo cual genera bastante incertidumbre porque desconocemos si la causa puede venir desde el cliente o desde el servidor y este problema se va arrastrando.

Objetivos anteriores añadiendo:

- Tratar de arreglar el problema de la conexión, identificando el origen y tratándolo.

### **Sprint 10 (02/12/19 - 16/12/19)**

Reunión sin demasiado contenido, ya que se sigue tratando de realizar las mismas tareas, por lo que los objetivos se trasladan al siguiente sprint.

### **Sprint 11 (16/12/19 - 13/01/20)**

Una de las últimas reuniones que se producen, tratando de ver los últimos objetivos que se tienen de cara a finalizar el trabajo, así como viendo si es posible arreglar la conexión.

Por mi parte, durante este periodo detecto varios problemas que pueden llegar a ser la causa, aparte de la evidente diferencia de versiones entre la aplicación y el servidor que, a la hora de comunicarse generan problemas.

- Tratándose de fechas cercanas a la entrega, desarrollar la memoria.
- Solucionar conexión.
- Depurar el código.
- Cambios visuales propuestos sobre la aplicación para mejorar la accesibilidad de la misma.

**Sprint 12 (13/01/20 - 07/02/20)**

En esta reunión se establece los últimos objetivos, así como una fecha para entregar la memoria a los tutores para que puedan revisarla y proponer mejoras.

- Continuar depurando código.
- Finalizar memoria.

**Sprint 13 (07/02/20 - Fin de proyecto)**

Para la última reunión se proponen cambios en la memoria y se realiza una visión general sobre el proyecto.

- Finalizar código.
- Realizar mejoras en memoria.
- Pancarta.

Ya hemos podido ver que el trascurso de este proyecto ha tenido épocas más activas que otras, debido a las complicaciones que comentaba. De todas formas se ha tratado de seguir la metodología de la mejor manera posible.

**A.3. Estudio de viabilidad**

Como ya hemos comentado antes, en este apartado se trata de la viabilidad del proyecto, de manera económica y legal, ya que cabe la posibilidad de que un proyecto posea las cualidades para ser interesante de realizar, pero no ser viable económicamente o legalmente debido a protección de ciertos u otros factores.

Hay otro factor, que en el caso concreto de este proyecto adquiere mucha importancia, que es la viabilidad de realizar el proyecto a nivel de magnitud y conocimiento. Lo trataremos inicialmente:

**Viabilidad sobre la magnitud y conocimiento del proyecto**

Hay que tener en cuenta una serie de factores en este apartado:

- Conocimiento sobre el lenguaje que se va a utilizar.
- Conocimiento sobre la herramienta que se va a utilizar.
- Análisis sobre la viabilidad de desarrollo del proyecto en cuestión, estableciendo límites de tiempos y recursos.

Con respecto a esta parte, es algo que a la hora de planificar un proyecto hay que tener muy en cuenta, porque aunque pueda resultar algo muy interesante, puede no ser viable debido a la complejidad del mismo.

En nuestro caso, si se hubiera realizado de forma más minuciosa este apartado, probablemente se habría llegado a la conclusión de que es un proyecto de final de carrera muy poco viable, porque supone obtener conocimiento sobre herramientas y lenguajes nunca utilizados, análisis de un código que no es propio, desarrollado en versiones con más de 7 años de antigüedad, tratar de actualizarlo sin conocer perfectamente todas las dificultades que se pueden encontrar, etc.

Por eso todo lo mencionado nos hace plantear este apartado, para que una persona que comience un proyecto, trate minuciosamente la información de la que dispone, y analice las ventajas e inconvenientes que supone, así como el tiempo del que dispondrá para su desarrollo.

## Viabilidad económica

Dado que partimos de un proyecto anterior, es conveniente mencionarlo, ya que en él se realiza un análisis sobre la viabilidad económica bastante acertado y que contiene información perfectamente válida para el estudio de la viabilidad de nuestro proyecto.

### Análisis de costes

Tal como se menciona en el proyecto anterior, tendremos en cuenta 4 tipos de costes:

- Coste de personal

La persona que se encargará de realizar el proyecto será un desarrollador de software, y como hemos comentado que el proyecto tiene una duración de unos 10 meses, con una media de 4 horas diarias trabajadas tendremos una estimación bastante decente, con épocas más activas y otras menos. En este

caso tomaremos de referencia coste/hora del proyecto anterior que suponía 12 euros/hora.

## **Viabilidad legal**

Para la viabilidad legal del proyecto, nos tendremos que fijar en la utilización de los datos y las licencias que se utilicen para desarrollar el trabajo.

Tal como se menciona aquí <https://developer.android.com/studio/terms?hl=es-419> , para poder utilizar todas las funcionalidades que se nos ofrecen debemos aceptar ciertas condiciones y hacer un uso responsable de las mismas, para no infringir causas legales.

Para ello, dentro del propio proyecto desarrollado en Android Studio contaremos con archivos que harán referencia a estas licencias, y que todo proyecto que se desarrolle con estas tecnologías deberá incluir como parte de su estructura.





## *Apéndice B*

---

# **Especificación de Requisitos**

---

### **B.1. Introducción**

En este anexo contemplaremos varios aspectos, donde tenemos que tener en cuenta lo que quiere conseguir el cliente de este proyecto y lo que comprende el desarrollador sobre el mismo, para que todo le mundo acabe con una visión general lo más parecida posible. De esta forma evitaremos posibles decepciones por parte del cliente o demasiada carga de trabajo para el programador en cuestión.

Se ayuda de elementos gráficos para poder visualizar la información sin tener que tener un conocimiento tecnico elevado y donde se podrán ver los requisitos generales del sistema, así como las funciones que debería cumplir.

Por último, se verán los diagramas de casos de uso, donde se podrá observar aun mejor cómo será el manejo de la aplicación.

### **B.2. Objetivos generales**

### **B.3. Catalogo de requisitos**

### **B.4. Especificación de requisitos**



## *Apéndice C*

---

# Especificación de diseño

---

### C.1. Introducción

### C.2. Diseño de datos

En este apartado se hará referencia a cómo están organizados los datos y cada una de

#### **Base de datos geo-espacial**

Debido a que seguimos desarrollando el proyecto a partir de uno anterior, utilizaremos la misma base de datos geoespacial, proporcionada por OpenStreetMaps y con la cual podremos acceder a datos necesarios para el trabajo.

La estructura de la base de datos ha sido modificada ligeramente para que los algoritmos se puedan nutrir de ventanas de tiempo. Para el resto de la estructura de la base de datos, así como el conjunto de tablas, se ha mantenido tal y como estaba. Si es necesario analizar cualquiera de estas partes, será preferible observar la memoria del trabajo anterior a este.

Igualmente, el modelo de datos, las tablas generales y la tabla que almacena las rutas, se mantienen sin cambios.

#### **Estructura de paquetes del servidor**

La estructura de paquetes del servidor ha sido la parte de la que se ha encargado mi compañero Ignacio, para la cual ha realizado modificaciones

con respecto a lo anterior, aunque manteniendo una estructura general bastante parecida.

Los cambios más relevantes en esta estructura son los referentes al añadido de nuevos algoritmos de cálculo de rutas y también a la implementación de nuevas estructura con herencia. Estos cambios se pueden ver en las siguientes imágenes que pertenecen al trabajo de mi compañero, por lo que si se quiere información más detallada es preferible visitar su proyecto.

## **Estructura de paquetes del cliente**

Ésta parte tendrá mas detalle que la anterior, ya que este proyecto se ha centrado en la interfaz de la aplicación y por tanto también se han hecho modificaciones en la estructura de paquetes de la misma.

**Paquete activities**

**Paquete adapters**

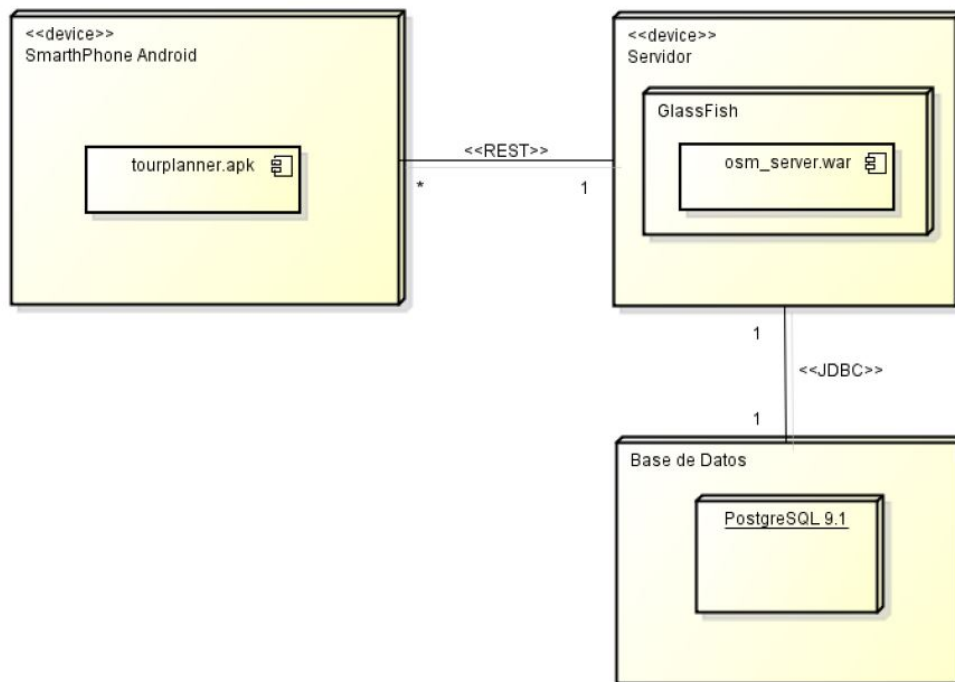
**Paquete communication**

**Paquete útil**

## **Diagrama de despliegue**

Tal como en el proyecto anterior, haremos referencia al trabajo original a partir del cual se desarrolló este diagrama sobre el funcionamiento a rasgos generales de como se comportan el cliente y el servidor a la hora de enviar y recibir datos entre ellos.

El cliente realiza peticiones del tipo REST hacia el servidor. Para que esa petición pueda avanzar, pasará por un "servlet" se ejecutará dentro de Glassfish. En este momento será cuando haga falta una consulta de tipo SQL hacia la base de datos, para obtener la información que se solicite.



## C.3. Diseño procedimental

### Diagramas de secuencia

## C.4. Diseño arquitectónico

En este apartado se pueden ver los distintos patrones de diseño que han sido aplicados sobre el código para lograr una estructura mucho mas homogénea y reutilizable a futuro.

Como se comentó en el proyecto anterior, se aplicaron los patrones Singleton y Facade para dos partes distintas del desarrollo de la estructura del proyecto.

Por desgracia, el patrón Facade se implementó para acceder al servicio de Panoramio, el cual ya no se encuentra activo, por lo que objetivamente no se está poniendo en práctica real para este proyecto. De todas formas, es un patrón perfectamente válido y una vez localizada una API que pueda sustituir los servicios prestados por Panoramio, se podría volver a aplicar sin ningún problema.

El patrón Singleton se sigue utilizando para el manejo de la clase `SSLFactory`, ya que seguimos necesitando validar y asegurar que se está realizando una conexión HTTPS con el servidor y este patrón nos ayuda bastante para poder hacerlo de manera ordenada.

## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

En este apartado de la memoria, nos centraremos en la parte más técnica del trabajo, entrando a detalle sobre los cambios realizados y los problemas diversos que se han encontrado y solucionado.

Mencionaremos algunas partes que hacen referencia al servidor, teniendo en cuenta que esta parte no la hemos desarrollado, ya que lo hizo mi compañero Ignacio.

Dentro de este apartado, veremos las distintas bibliotecas que se han utilizado, las APIs, el manual del programador y también las pruebas realizadas.

Cabe destacar que en este proyecto la mayor parte de desarrollo ha sido sobre la resolución de errores provocados por el cambio de versión y la antigüedad del proyecto, sin poder mostrar implementaciones nuevas con respecto a funcionalidades. Es un trabajo dedicado al mantenimiento y actualización de una aplicación que contaba con una versión y recursos con 7 años de antigüedad y por eso ha habido tanta base de problemas.

Del mismo modo que hemos hecho en algunos apartados, se mencionarán partes que ya fueron desarrolladas anteriormente y que no hayan sufrido cambios importantes.

## D.2. Estructura de directorios

En este apartado vamos a enumerar distintos contenidos que tendrá el dispositivo entregable, tanto del cliente como del servidor:

- Cliente
  - Aplicación: dentro de esta subcarpeta tendremos el archivo Tour-Planner.apk, para que pueda ser instalado en cualquier dispositivo.
  - Código fuente: Contiene el código de la aplicación Android.
  - Javadoc: documentación sobre la aplicación.
  - Datos: contiene el archivo con las referencias utilizadas en el proyecto y un backup con la base de datos PostgreSQL de Burgos configurada.
  - Máquina virtual: Dentro tendremos la máquina virtual donde estará instalado Android Studio y se podrá utilizar sin tener que hacer más instalaciones.
- Servidor
  - Aplicación: contiene el archivo osm server.war para que se pueda utilizar en cualquier momento.
  - Código fuente: Contiene el código fuente de la aplicación servidor.
  - Javadoc: Contiene la documentación de la aplicación servidor.
  - Datos: Archivo con las referencias utilizadas en el proyecto y un backup con la base de datos PostgreSQL de Burgos configurada.
  - contiene la máquina virtual del servidor configurada para ser utilizada.
- Documentación Memoria
  - PDF: Contiene la memoria en formato PDF.
  - Latex: Contiene la memoria en formato de latex.
- Software
  - Esta carpeta contendrá los distintos programas que se han ido utilizando en el desarrollo de la aplicación.



## **D.3. Manual del programador**

Dentro de este apartado veremos todo el proceso de desarrollo que se ha realizado en el proyecto, explicando los cambios, mejoras y arreglos sobre el mismo. Lo dividiremos en distintos puntos, ya que es necesario cubrir todos los aspectos que hacen referencia al desarrollo del código y de la aplicación.

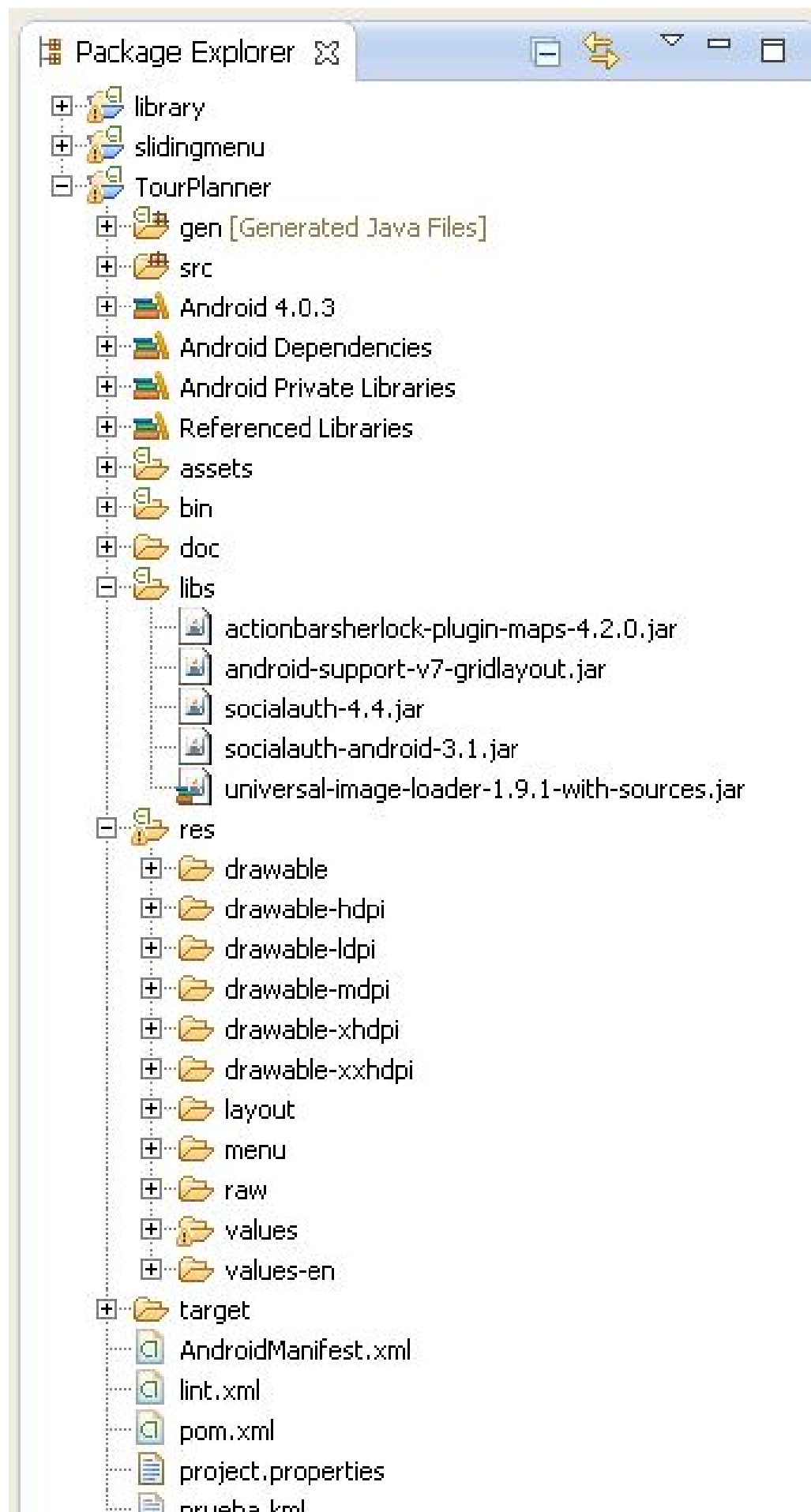
Antes de comenzar, cabe destacar que en este proyecto se ha tratado de mejorar el código de una aplicación con una antigüedad de 7 años, actualizándola a versiones compatibles con los dispositivos de hoy en día y a su vez tratando de que sea lo más estable posible. Ha habido funcionalidades que no se han podido actualizar tal y como estaban desarrolladas, por lo que se mencionarán para un futuro desarrollo.

### **Modificaciones sobre la estructura**

En este pequeño apartado mencionaremos las modificaciones que ha sufrido la estructura dentro de la herramienta de desarrollo, partiendo de la base de que la herramienta en sí es distinta y consideramos que para este propósito Android Studio consigue una estructura más clara y sencilla, donde se pueden localizar mejor las distintas partes de la que está compuesta la aplicación, porque está organizada de forma mas homogénea.

En el proyecto anterior contábamos con una estructura como la siguiente:

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

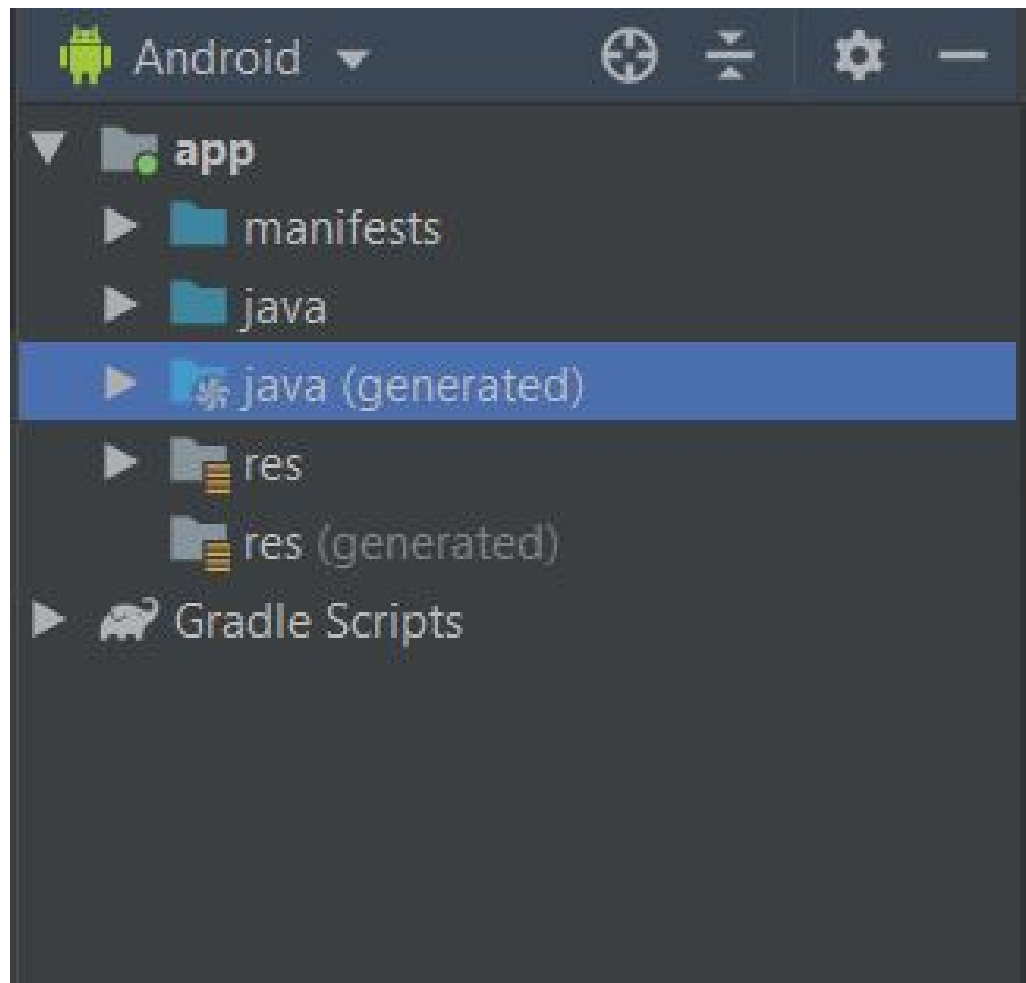


Aquí se puede observar que hay 3 carpetas principales:

- Tourplanner, la cual hace referencia a la aplicación. Dentro de esta carpeta esta la mayor parte del código de desarrollo y tambien se hace referencia a librerías y recursos.
  
- SlidingMenu, que en este proyecto es la biblioteca utilizada para generar el menú lateral con el cuenta la aplicación. Como se puede ver es necesario que esté implementada de manera individual, en vez de ser importada como el resto de librerias y también tiene código dentro de sus subcarpetas.
  
- libraries, que hace referencia a las librerias que estan utilizando en el proyecto, excepto SlidingMenu por lo que acabamos de ver.

Comparando esta estructura con la que tenemos actualmente podremos ver bastantes diferencias:

## *APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN*



En este caso, podemos ver una estructura mas sencilla, ya que sólo contamos con una carpeta principal de la que aparecen el resto de elementos.

La carpeta java, que será donde se encuentre todo el código de desarrollo de la aplicación. No tendremos que estar desplazandonos entre otros directorios para poder encontrar código, ya que estará al completo aquí.

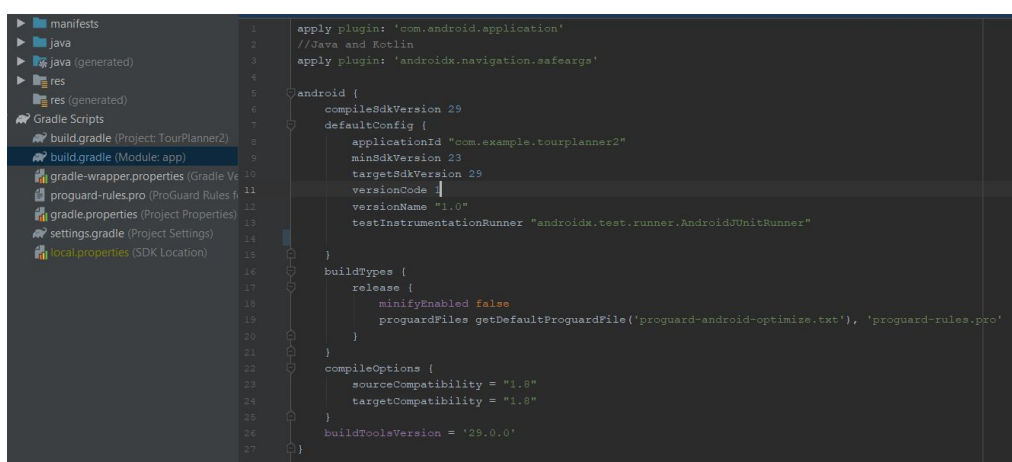
### **Gradle Scripts**

Por otro lado, el apartado de Gradle Scripts es algo que Android Studio implementa por defecto en todos los proyectos, y es una herramienta muy útil cuando se trabaja con un numero considerable de librerias, pudiendo manejar a la perfección qué se está importando para que sea utilizado, así como las versiones en las que lo implantamos.

El hecho de poder manejar las versiones de esta manera nos permite

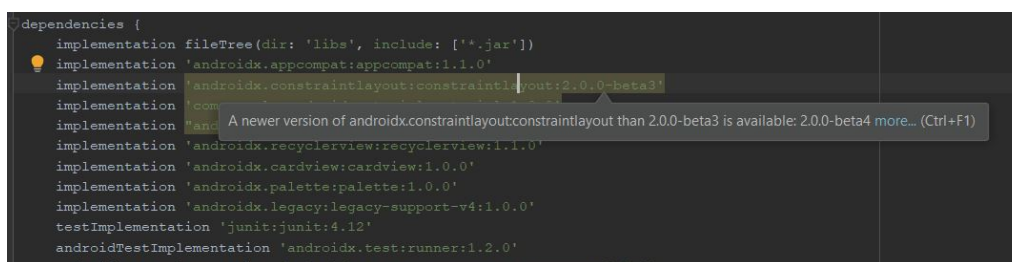
poder mantener la aplicación lo más actualizada posible continuamente, ya que además el propio Software nos advertirá siempre que tengamos versiones de repositorios para los cuales ya hay otras más nuevas. Esto ayuda a mantenerse siempre lo más alejado posible de trabajar con "bujsz errores de desarrollo de los propios repositorios, ya que suele ser lo que se corrige cuando se actualiza el Software.

El fichero más importante del apartado de Gradle será el llamado "build.gradle", ya que será con el que podremos añadir repositorios y bibliotecas para que despues podamos implementar en el código.



Como se puede observar, aquí configuraremos la versión de sdk con la que estamos trabajando y la versión mínima con la que podremos hacer funcionar la aplicación. La versión mínima será la API 23, como se indica en esta configuración y la versión objetivo que se menciona aquí también será la API 29, que es la última con la que cuenta Android. Nuestra aplicación funcionará para dispositivos con versión Android 6.0 (API 23) o superior.

Por último podemos ver cómo nos recomienda Android que mantengamos actualizado nuestro software y repositorios, ya que en el momento que alguno de ellos tiene una versión disponible superior, nos los marcará:



## **Modificaciones importantes sobre bibliotecas**

Como ya hemos mencionado en otros apartados de la memoria, se han realizado muchas modificaciones sobre las librerías que estaban siendo utilizadas previamente y que, por desgracia, han quedado totalmente obsoletas. Esto nos ha llevado a intentar descubrir todas las novedades que se han ido implementando en el software de Android desde el desarrollo del anterior proyecto.

### **Cambio de SlidingMenu**

Uno de los primeros descubrimientos que nos vimos obligados a realizar fue el de una nueva herramienta par obtener el menú "deslizante" con el que contaba la aplicación, o al menos un nuevo menú que cumpliera la función que ya cumplía previamente SlidingMenu.

Esto se debe a que la librería de SlidingMenu quedo obsoleta, y no se siguió desarrollando soporte para la misma a lo largo de las nuevas versiones, por lo que al intentar utilizarla en Android Studio con la versión para la API 23, ésta no era reconocida.

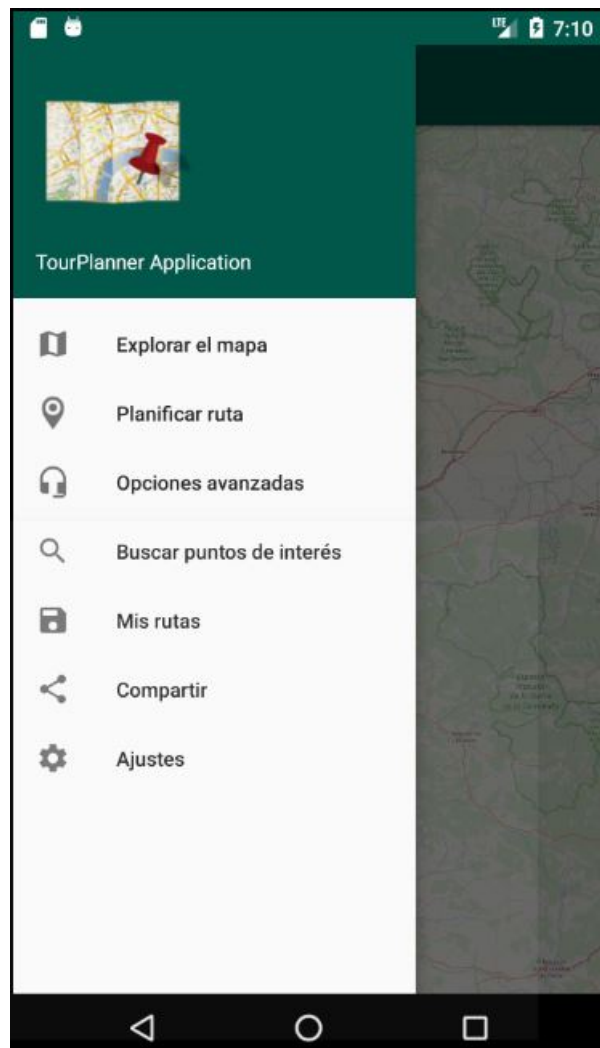
De esta forma, hemos desarrollado el nuevo menú de la aplicación, utilizando una versión más estándar proveniente de Android, lo cual nos asegurará más soporte de cara a futuro y es mucho menos probable que tenga que ser sustituida por completo en una futura versión. El menú en cuestión es NavigationMethod y es una funcionalidad de la biblioteca de androidx, la cual ha sido utilizada en muchas partes de este proyecto. Éste menú nos ofrece una funcionalidad muy parecida a lo que ya nos encontrábamos antes, pero mejorando la eficiencia y la visualización, siendo esta última bastante más parecida a la que nos podemos encontrar en cualquier dispositivo que utilice Android. Esto también ayuda a la accesibilidad de cara al usuario.

La interfaz del menú que teníamos antes era de esta forma:



La interfaz del menú que tenemos ahora es de la forma:

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN



Como podemos ver, hay un cambio considerable, tanto en facilidad a la hora de visualizarlo, como de desarrollarlo en código, ya que implementar este menú, al ser un estándar de Android, nos resulta una tarea más accesible y en caso de tener dudas, también es mas sencillo conseguir documentación al respecto. Estas son las ventajas que nos ofrece estar utilizando repositorios estándar para Android.

Las dos bibliotecas que han sustituido a SlidingMenu han sido:

- NavigationMethod, y los métodos que implementa.
- Android.view.Menu



También los colores es algo en lo que se ha puesto cierta atención a la hora de desarrollar este menú y el resto de la aplicación, ya que consideramos que la accesibilidad de una aplicación hoy en día es un valor añadido muy importante. Se ha tratado de manejar colores que faciliten a la lectura y visualización, siguiendo los estándares de accesibilidad de W3C.

### **Librería SherlockActivity**

Esta es otra de las bibliotecas que más han afectado al desarrollo, ya que la mayoría de `.activities` estaban construidas en torno a ella. Como Ocurrió con `SlidingMenu`, también nos encontramos con que la biblioteca estaba "deprecated" obsoleta pero, por desgracia, no se había continuado el desarrollo de la misma y por lo tanto no era una cuestión de actualizar a una versión más actualizada, porque ésta no existía. Por esto mismo se tuvo que buscar una alternativa y pasar a utilizar algo más estandarizable, siempre evitando que en un futuro suceda algo similar a lo que nos ha sucedido a nosotros.

Así fue como se comenzó a implementar la librería de `AppCompatActivity`, pero no en todas las actividades que conformaban la aplicación, sino en la principal. Para poder explicar esto es necesario mencionar la librería anterior de `NavigationMethod`, ya que en este sentido se ha conseguido solucionar los dos problemas utilizando un método estándar que está más optimizado para el manejo de aplicaciones como la nuestra, donde se maneja un número considerable de ventanas diferentes y estas poseen comunicación e interacción.

Hablamos del concepto de `Activity` y `Fragment`.

- **Activity.** Se podría definir como una ventana de la aplicación. En código corresponde a una clase de Java, que contiene todos los métodos que sirva para mostrar información, aunque por otro lado la parte más visual corresponde a un archivo `.xml` que será el que contenga la información puramente visual, configuración de colores, tamaños, etc.
- **Fragment.** A efectos visuales contiene lo mismo que una `Activity`, pero en este caso la verdadera diferencia surge en la comunicación entre ellas y la optimización de la aplicación en cuanto a tiempos a la hora de navegar entre ventanas. Por otro lado, para implantar correctamente el menú del que hablábamos en el punto anterior, la manera más óptima es tenerlo instaurado dentro de una `activity`, y que los distintos apartados (ventanas) que tengamos dentro de ese menú sean `Fragment`.

## ~~28~~ APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

De esta forma conseguimos navegar desde una ventana principal hacia el resto de apartados utilizando el `NavigationView`.

### **Librería `Org.Apache`**

La última de las bibliotecas mas conflictivas a la hora de desarrollar la aplicación ha sido la de `Org.Apache`. Esto se debe principalmente a que, como las dos anteriores supone una carga dentro del código bastante grande, pero a diferencia de estas no obtenemos un error directo por su parte a la hora de implantarlo en nuestro código actualizado, sino que obtuvimos los problemas más adelante.

De hecho, pese a que estuviera obsoleta, se decidió que en un principio no se modificaría, ya que suponía un cambio excesivamente grande y costoso como para invertir recursos en algo que aparentemente en un inicio funcionaba correctamente. Finalmente no se ha podido recurrir a otra opción, ya que hay métodos de esta biblioteca que no dan el resultado esperado a la hora de ejecutar todo el código.

El problema principal es que supone un cambio en todas las clases donde se realiza una petición de tipo `HTTP` o `HTTPS`, ya que son éstas la que utilizan los métodos provenientes de `Org.Apache`.

Los metodos que más utilizados estaban siendo eran:

- `DefaultHttpClient`. Éste es con diferencia el método que más implementaciones y llamadas recibía dentro de las clases de la aplicación. Era el método que servía para iniciar la conexión `HTTP`.
- `HttpsClient`. Ésta clase había sido implementada por los desarrolladores anteriores del proyecto y aportaba la posibilidad de realizar una conexión `HTTPS` o lo que es lo mismo, una conexión segura a través del protocolo `SSL`. El problema es que también dependía al completo de métodos obsoletos.

Finalmente, se consiguió aplicar otro cambio importante, también hacia la estandarización, ya que se han modificado todos los métodos y llamadas dentro de clases para que pasen a utilizar `HttpsURLConnection`, que es la librería estándar de `Android` que nos ofrece el desarrollo y manejo de peticiones y conexiones `HTTPS`.

Para realizar la conexión utilizando esta librería tendremos que seguir unos pasos mas sencillos que antes, aunque por otro lado bastante similares.

- Establecemos una dirección URL, que nos servirá para poder alcanzar el destino de nuestras peticiones.
- Instanciamos la conexión de tipo `URLConnection`, utilizando la URL mencionada.
- Establecemos todos los parámetros que consideremos necesarios para poder establecer la conexión, como por ejemplo el tiempo de "Timeout" para poder conectarnos y evitar malgasto de recursos.
- Una vez está todo lo anterior configurado, simplemente realizamos la conexión a través del método `connect()`.
- por último quedaría recibir respuesta, lo cual podremos obtenerlo con métodos como `getResponseCode()`, obteniendo la respuesta que hayamos recibido de parte del servidor, y pudiendo enviar nuevas peticiones en base a esta respuesta.

```
//Create a URL object holding our url
URL myUrl = new URL(url);

//Create a connection
URLConnection connection = (URLConnection)
    myUrl.openConnection();

//Set methods and timeouts
connection.setRequestMethod(REQUEST_METHOD);
connection.setReadTimeout(READ_TIMEOUT);
connection.setConnectTimeout(CONN_TIMEOUT);

//Connect to our url
connection.connect();

switch (connection.getResponseCode()) {
    case POST_TASK:
        connection.setRequestMethod("POST");
}
```

Como se puede ver en la imagen, realizar esta conexión supone bastante poca carga de código y al ser un método bastante parecido, si se utiliza varias veces en una misma clase, se podría realizar `EXTRACT METHOD` sobre varias partes, para evitar aun más la sobrecarga de código.

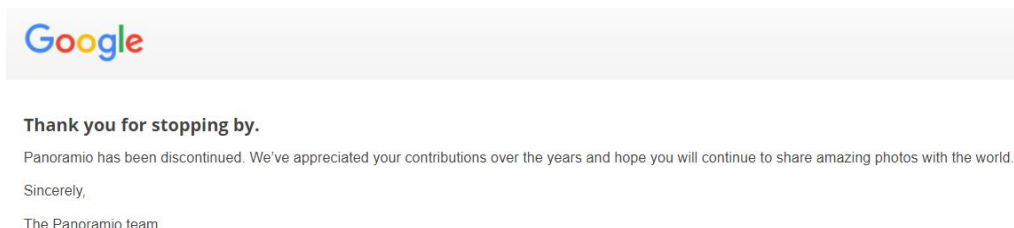
## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

### Cambios sobre APIs

En este caso, hablaremos sobre las APIs que antes eran utilizadas, pero con el motivo de haber quedado obsoletas, no ha sido posible seguir contando con su funcionalidad dentro de la aplicación.

#### Panoramio

Una de ellas es Panoramio, que nos proveía de imágenes para que fueran mostradas dentro de los detalles de un punto de interés. Desde que este proyecto Online fue absorbido por Google, ya no podemos contar con el servicio de obtención de imágenes gratuito y por tanto la API que nos permitía hacerlo ha ido quedando en el olvido, al no tener más sentido de existencia.



### D.4. Compilación, instalación y ejecución del proyecto

En este apartado se cubrirá todos los aspectos referentes a la utilización de la aplicación, por lo tanto veremos cómo instalar todo el software necesario, cómo compilar el proyecto de Android y también cómo ejecutarlo.

Cabe destacar que sólo se va a cubrir la instalación, compilación y ejecución de la aplicación de Android, ya que la parte referente al servidor se tratará en detalle en el proyecto de mi compañero Ignacio.

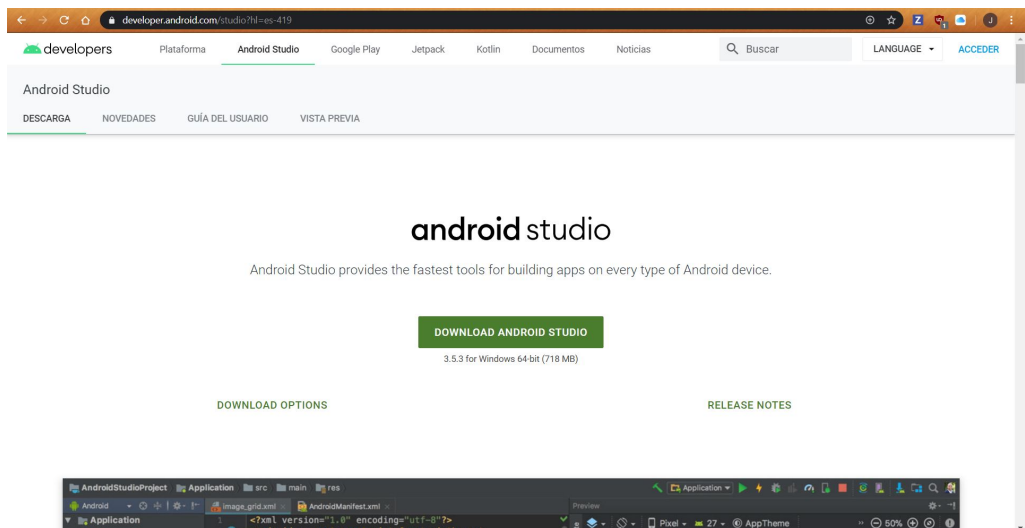
#### Instalación

Las herramientas que vamos a necesitar tener instaladas por parte de la aplicación Android serán, básicamente Android Studio y sus componentes internos. Vamos a observar cómo lo instalaremos.

#### D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO

### Android Studio

Simplemente necesitaremos acceder a la página oficial de Android, a través del siguiente enlace (), y veremos lo siguiente:



Cómo se puede observar, bastará con pulsar la opción para descargar Android Studio. En esta opción siempre nos ofrecerá la última versión oficial del producto. Excepto en casos excepcionales, esto es lo más recomendable, ya que al ser un software que exige bastantes recursos, cuanto más actual sea la versión que estamos utilizando mejor optimizada estará.

Si se da uno de los casos excepcionales que mencionábamos, también nos ofrecen diferentes versiones o incluso diferentes sistemas operativos con los que son compatibles.

Android Studio downloads			
Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	<a href="#">android-studio-ide-191.6010548-windows.exe</a> Recommended	718 MB	58b3728fc414602e17fd9827e5ad0c969e5942aff1ee82964eedf1686450265b
	<a href="#">android-studio-ide-191.6010548-windows.zip</a> No .exe installer	721 MB	d88d640b3444f0267d1900710911ca350db6ca27d07466039e25caf515d909fe
Windows (32-bit)	<a href="#">android-studio-ide-191.6010548-windows32.zip</a> No .exe installer	721 MB	2786400eb2f5d9cbe143fe02d4e711915c83f95a335e609a890e89775195b7
Mac (64-bit)	<a href="#">android-studio-ide-191.6010548-mac.dmg</a>	733 MB	6cb545c07ab4880513f4757579be7ae53a2de935435f8f22eb736ef72ecdffe
Linux (64-bit)	<a href="#">android-studio-ide-191.6010548-linux.tar.gz</a>	738 MB	af630d40f276bd169c6ac8c7663a989f562b0ac48a1d3f0d720f5b6472355db
Chrome OS	<a href="#">android-studio-ide-191.6010548-cros.deb</a>	620 MB	87ca5f17f808ecb909e62c80da3e578156563309ca24f0b820064cc786d1360f

See the [Android Studio release notes](#).

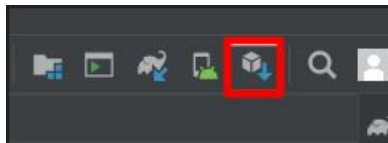
Como se puede observar, también nos ofrecen la opción de acceder a las release notes", lo cual resulta muy útil para poder descubrir más a fondo los

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

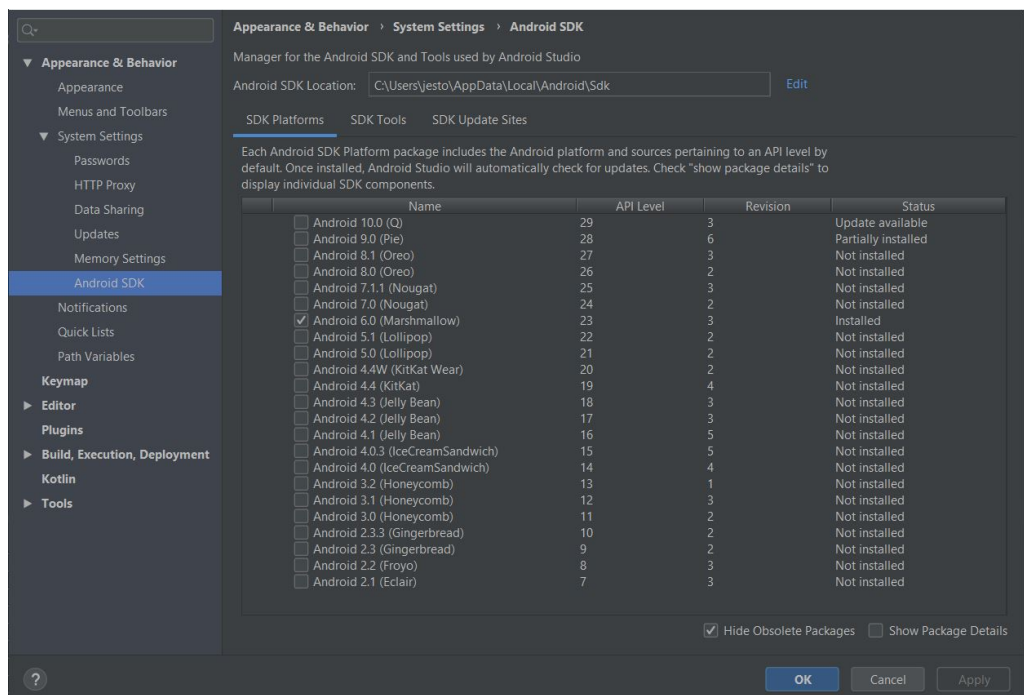
cambios que se han desarrollado en cada versión.

### SDK Manager

Esta herramienta es parte de Android Studio, pero necesitaremos configurarla por separado, para poder contar después con una experiencia fructífera:



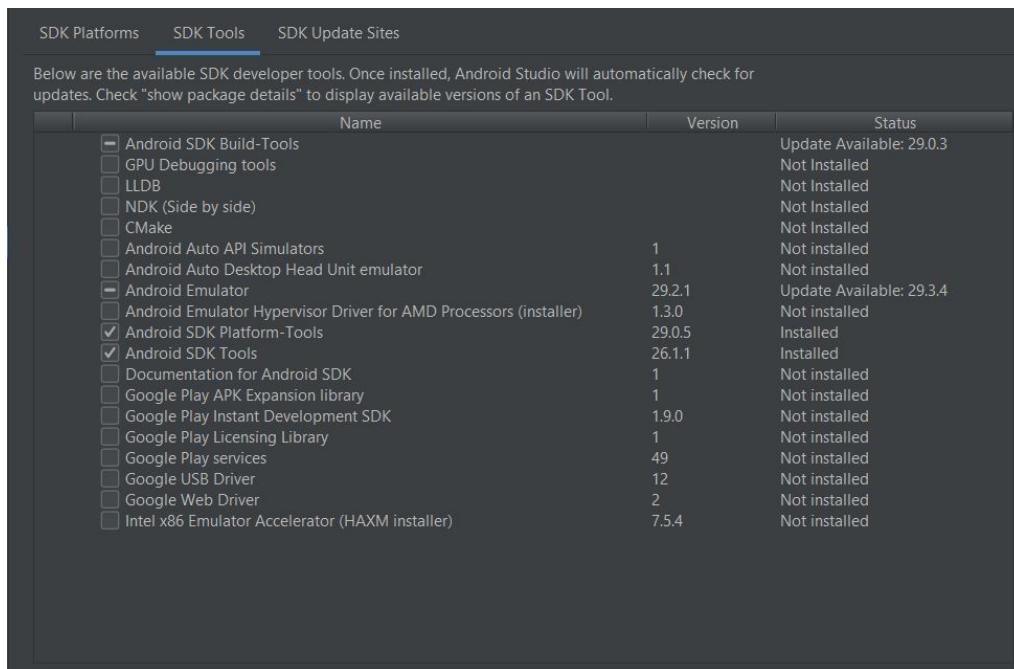
Una vez pulsemos ese botón remarcado en rojo, entraremos en la configuración del SDK Manager:



Como podemos ver, la primera ventana que nos ofrece este gestor, será para que seleccionemos la versión de Android sobre la que queremos trabajar en nuestro proyecto. Como ya hemos mencionado previamente, nosotros tenemos instalada la versión de Android 6.0.

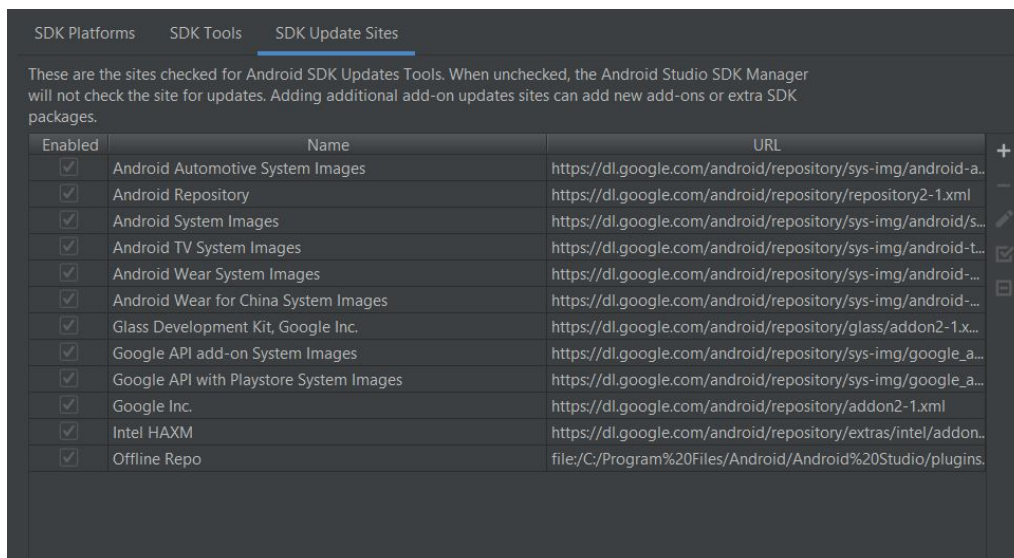
También contamos con otras dos ventanas:

#### D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO



Dentro de esta segunda ventana podremos configurar las herramientas adicionales que instalaremos para conseguir una mejor experiencia mientras desarrollamos las aplicaciones. En nuestro caso, tenemos instalado el emulador, que resulta totalmente necesario para hacer pruebas.

Por último contamos con la ventana de actualizaciones:



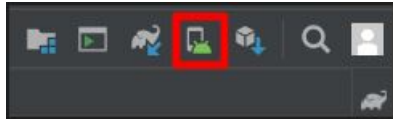
En esta ventana, tal como se describe, se podrá configurar los repositorios de los que se realizarán comprobaciones en busca de actualizaciones de

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

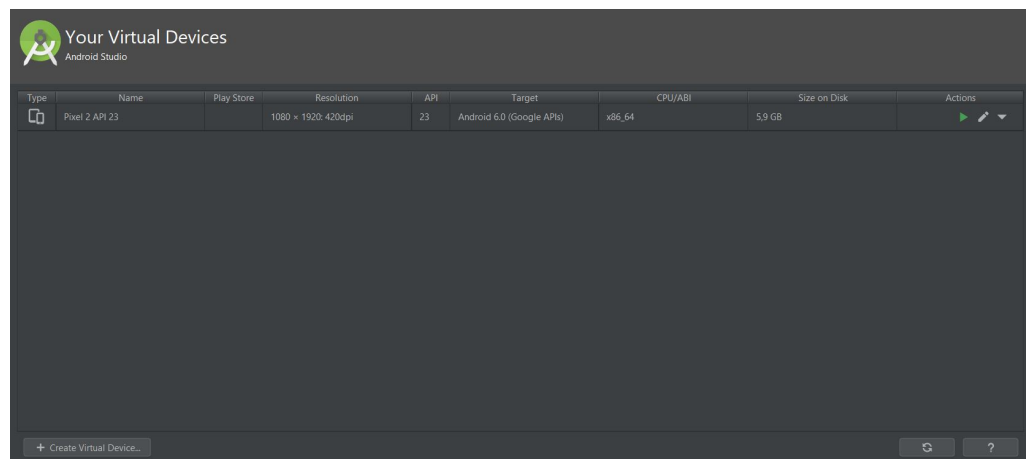
manera automática.

### AVD Manager

Este gestor será el que nos permita llevar un control sobre el emulador de Android, ya que aquí será donde instalaremos el dispositivo móvil virtual que después ejecutaremos para probar nuestra aplicación.



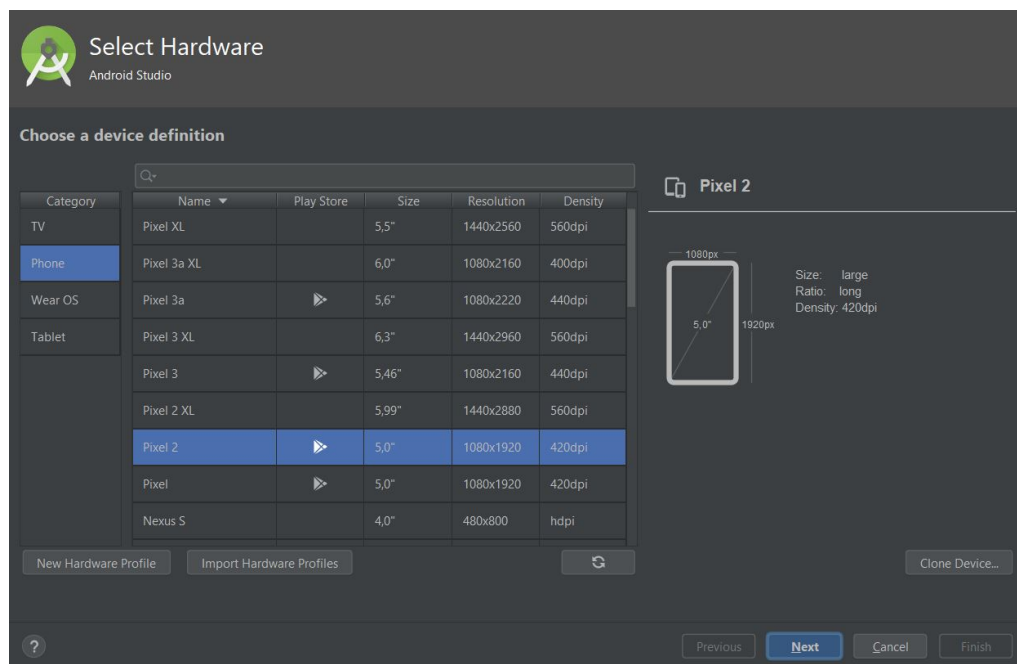
Una vez pulsemos tendremos acceso a la configuración de nuestros dispositivos virtuales:



Aquí ya podremos ver los dispositivos que tenemos instalados, ejecutarlo, eliminarlos o modificar algunas configuraciones.

Si queremos crear un dispositivo tendremos que seleccionar el botón de 'Create Virtual Device':





En esta ventana simplemente seleccionaremos que modelo de dispositivo móvil queremos simular, descargarlo y utilizarlo.

## Compilación

## Ejecución

## D.5. Pruebas del sistema



## *Apéndice E*

---

# Documentación de usuario

---

## E.1. Introducción

## E.2. Requisitos de usuarios

### Requisitos HardWare

Para poder realizar una ejecución correcta de la aplicación necesitaremos utilizar un emulador de Android que esté instalado en un ordenador con un rendimiento bastante alto, ya que este tipo de programas para emular un sistema Android al completo, exigen muchos recursos de CPU y RAM.

Otra exigencia básica para poder hacer funcionar el emulador de Android Studio es que la CPU cuente con estos requisitos mínimos:

- SDK Tools 26.1.1 o versiones posteriores.
- Procesador de 64 bits.
- CPU con compatibilidad para UG (invitado no restringido)
- HAXM 6.2.1 o posterior (se recomienda HAXM 7.2.0 o posterior)
- También hemos observado que en cuanto al tipo de procesador, en el caso de intel no cause ningún problema, pero con AMD, parece que sólo lo soportan los procesadores más nuevos. Esto fue probado en un procesador AMD Ryzen 2700x y no fue capaz de ejecutar el programa de emulador, en cambio en un procesador intel i7 de 8th generación funciona a la perfección.

Por supuesto, también cabe destacar que otra forma de hacer que la aplicación funcione, tratándose de Android, podremos usar un dispositivo que tenga una versión instalada de Android 6.0 o superior. De esta manera podremos observar cómo es la experiencia al completo de ésta aplicación en un dispositivo real.

La aplicación ha sido instalada en un dispositivo móvil Google Pixel 2 a través del emulador y en un BQ Aquaris E5 de manera física. Ambos funcionaron a la perfección bajo los requisitos mencionados anteriormente.

## Requisitos SoftWare

Estos son los requisitos software:

- Compilador de Java. Nos servirá para poder programar la aplicación de Android y también para ejecutar la máquina virtual Java.
- Android Studio, ya que esta nueva herramienta proporcionada por Google nos permite desarrollar la aplicación en un entorno totalmente compatible. También ofrece numerosos tooltips a la hora de programar.
- Android Developer Tools. Ésta herramienta viene incorporada en Android Studio y supone la base para poder desarrollar todo el código de Android.
- Oracle GlassFish Server 3.0, que contendrá el servidor encargado de las respuestas al cliente.
- Oracle VM VirtualBox, el cual nos servirá si no queremos instalar los programas mencionados, ya que con tenerlo todo listo en una máquina virtual podríamos ejecutar el cliente o servidor desde allí. Es importante mencionar que será necesario un ordenador con bastante potencia para poder aprovechar esta opción.

## E.3. Instalación

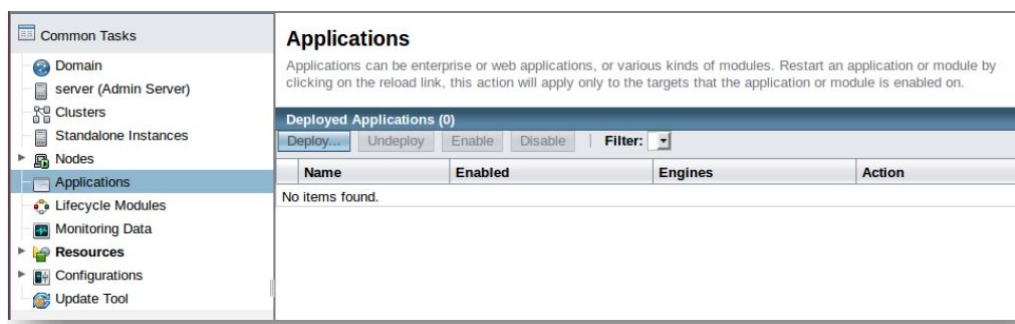
En esta sección se encuentra detallada la guía de instalación de la aplicación.

Cabe destacar que es necesario instalar tanto el cliente en un dispositivo Android (o emulador) como el servidor en GlassFish.

## Instalación de la aplicación servidor

Debido a que la versión de GlassFish utilizada es la misma que en anteriores versiones del proyecto, se seguirán los mismo pasos para la instalación del servidor. Dichos pasos son los siguientes:

Para instalar la aplicación en el servidor, hay que abrir la consola de administración de GlassFish tecleando en un navegador web la dirección del servidor con el puerto 4848 (<http://localhost:4848> en este caso) e ir a la opción Applications.



A continuación hay que hacer click sobre el botón Deploy e indicar la ruta en la que se encuentra el archivo `.osm_server.war`. Después, hacemos click sobre el botón OK y ya tendremos la aplicación publicada y funcionando.

OK

Cancel

Deploy Applications or Modules

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

\* Indicates required field

Location:

☒ Packaged File to Be Uploaded to the Server

Examinar...

☐ Local Packaged File or Directory That Is Accessible from GlassFish Server

Browse Files...

Browse Folders...

Type: \*

Web Application

Context Root:

Path relative to server's base URL.

Application Name: \*

Virtual Servers:

server

Associates an Internet domain name with a physical server.

Status:

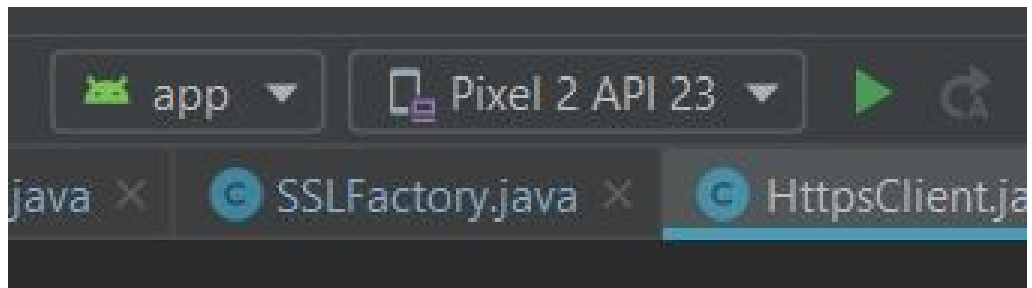
☒ Enabled

Allows users to access the application.

## Instalación de la aplicación cliente

Para realizar la instalación de nuestra aplicación Android en un dispositivo físico o un emulador, necesitaremos seguir estos pasos:

- Necesitaremos, en caso de un dispositivo físico, tenerlo conectado al ordenador en el que estamos desarrollando la aplicación.
- Una vez tengamos lista la aplicación, pulsaremos el botón de "play" en Android Studio.



- Android Studio será el encargado de instalar el archivo.apk dentro del dispositivo y lo ejecutará directamente.



Hay que tener en cuenta que para el correcto funcionamiento de la aplicación hay que seguir correctamente los pasos de compilación explicados en el apartado "Compilación del cliente", prestando especial atención al punto en el que se establece la dirección IP y puerto del servidor.

## Ejecución de la aplicación

Para lograr que la aplicación servidor se ejecute sin la necesidad de contar con un servidor y un cliente independientes se han incluido en el USB del proyecto las máquinas virtuales correspondientes al cliente y servidor. De esta forma podremos ejecutar la aplicación con un solo ordenador.

Cabe destacar que para lograr que ambas máquinas virtuales se comuniquen entre sí correctamente, se ha utilizado el programa Logmein Hamachi que nos permite crear redes virtuales entre ambas máquinas a través de Internet.

**Ejecución de la aplicación servidor**

Para ejecutar la aplicación servidor, hay que importar la máquina virtual Servidor - Ubuntu 18.04 LTS en un software de virtualización como VirtualBox y ejecutarla.

Una vez iniciada, introducimos la contraseña tourplanner y ejecutamos el script startGlassfish que se muestra en la ilustración, con lo que la aplicación estará ejecutándose y escuchando a la espera de peticiones.

**Ejecución de la aplicación cliente**

Para ejecutar la aplicación cliente, al igual que la aplicación servidor, hay que importar la máquina virtual Cliente - Windows 10 en un software de virtualización como VirtualBox y ejecutarla.

Una vez iniciada, hacer doble click sobre el acceso directo del entorno de desarrollo Android Studio. A continuación, hacemos click sobre el icono "play". Después de esto se abrirá el emulador de Android con la aplicación instalada.

**E.4. Manual del usuario**