UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Ampliación, actualización y mantenimiento de la aplicación *TourPlanner*. Parte FrontEnd

Presentado por Jesús Manuel Calvo Ruiz de Temiño
en Universidad de Burgos — 9 de febrero de 2020
Tutor: Bruno Baruque Zanon

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Bruno Baruque Zanon, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Jesús Manuel Calvo Ruiz de Temiño, con DNI 13173674X, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Ampliación, actualización y mantenimiento de la aplicación *TourPlanner*. Parte FrontEnd.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 9 de febrero de 2020

Vº. Bº. del Tutor: Vº. Bº. del co-tutor:

Author and supervisor

D. Bruno Baruque Zanón . Santiago Porras Alfonso

Resumen

Este proyecto se basa en la ampliación, actualización y mantenimiento de la aplicación TourPlanner, la cual se encarga de la generación de rutas turísticas personalizadas a través de una serie de algoritmos de búsqueda de caminos óptimos y que está diseñada para ser utilizada en sistemas operativos Android.

Para desarrollar este trabajo se ha trabajado en dos proyectos paralelos, por un lado mi compañero [5], más centrado en el desarrollo del servidor y los algoritmos para la generación de rutas y por el otro, este proyecto que está más centrado en el desarrollo del FrontEnd (interfaz y funcionamiento de la aplicación Android).

A continuación se explicará brevemente las mejoras que se han realizado en el FrontEnd. Cabe destacar que este trabajo esta centrado en la aplicación de mejoras estructurales y en la actualización y estandarización de las librerías y métodos utilizados, para así conseguir un mantenimiento más sencillo y sostenible.

Se ha conseguido actualizar la aplicación para que pueda funcionar en los dispositivos más modernos, ya que está desarrollada bajo la versión de Android 6.0 (API 23), pero también es compatible hasta la última versión de Android (Android 9.0, API 29 en este momento).

Por otro lado, como ya se ha señalado, este trabajo se centra en la estandarización, y por ello se ha dejado de integrar una serie de librerías que, ya sea por no tener continuidad o por no tener compatibilidad con las nuevas versiones de Android, no eran capaces de funcionar en este sistema y por tanto, se ha decidido buscar la opción mas estándar o sostenible para sustituirlas.

Descriptores

Android, actualización, mantenimiento, optimización, estandarización, FrontEnd

...

Abstract

This project is based on the extension, upgrade and maintenance of TourPlanner application. This app is used to generate custom turistic routes through some *pathfinding* algoritms and it's designed to run on Android mobile devices.

The development of this project has been done in two parts, the part from my team mate [5], focused on the server and algoritms development, and this project part which is focused on the development of FrontEnd (interface and functioning from the Android application)

On this project will be explained all the upgrades and improvements done about FrontEnd. It's important to mention that this project is focused on doing structural improvements, upgrades and standarization of the libraries and methods used on it. This will bring an easier and more sustainable maintenance.

The app has been upgraded to be used on the latest devices, developed under Android 6.0 (API 23) version and with compatibility with last Android version (Android 9.0, API 23 at this moment).

On the other hand, as mentioned before, this project works around standarization, so it was necessary to change the integration of some libraries, due to deprecation or incompatibility with newer Android versions. Now there are new standarized and native Android libraries integrated.

Keywords

Android, upgrade, maintenance, optimization, standarization, FrontEnd

Índice general

Índice de figuras

Índice de tablas

Introducción

La aplicación sobre la que se desarrollará este proyecto de fin de grado comenzó con el proyecto *Generación de rutas Turísticas personalizadas* [3] y un año después fue ampliado en el trabajo *Ampliación sobre la aplicación para la generación de rutas turísticas personalizadas* [4]. En este caso se trabajará sobre la última versión de la que se dispone, donde se realizarán todas las mejoras y actualizaciones necesarias, ya que se trata de un proyecto realizado sobre tecnologías con varios años de antigüedad.

Una parte del proyecto se basa en la generación de rutas turísticas óptimas respecto a los gustos del usuario, añadiendo el factor de horarios de apertura y cierre de los puntos de interés. Esto se consigue con el desarrollo de una serie de algoritmos que se han desarrollado en el trabajo de BackEnd [5].

Por el otro lado, la interfaz completa de la aplicación, donde el usuario podrá realizar toda la interacción con el sistema sería la correspondiente a la aplicación de Android. Se ha desarrollado de forma que cualquier usuario que tenga costumbre en el manejo de cualquier dispositivo Android le resulte fácil y accesible.

Para conseguir esto mencionado, ha sido necesario modificar varias librerías que se encontraban obsoletas, tratando de utilizar otras más estándar y así obtener una interfaz más accesible. Un ejemplo de esto es el menú deslizante, que ha sido modificado por completo para obtener uno muy similar a los que nos podemos encontrar en aplicaciones como Google Maps o Gmail.

También se ha tratado de centrar el trabajo hacia la optimización en el uso de recursos y métodos dentro de la aplicación. Esto se verá más desarrollado en el anexo 4 REFERENCIA.

Para que el proyecto adquiera sentido al completo , ha sido necesario cooperar entre las dos partes durante todo el desarrollo del mismo. Se trata de una aplicación que envía peticiones y datos a un servidor, donde se procesa la información y se realizan los cálculos necesarios, por lo que la conexión entre ambos es importante.

Objetivos del proyecto

En este apartado se explicarán los distintos objetivos que se van a tener en cuenta en este proyecto, diferenciando entre las posibles mejoras a realizar en la aplicación y los objetivos mas personales de aprendizaje y desarrollo como programador en base al trabajo realizado.

Objetivos de mejora:

- Actualización de la interfaz del proyecto desde un sistema Android obsoleto hacia la versión más actualizada posible del mismo, pudiendo implementar mejoras tanto visuales como funcionales.
- Actualización sobre la utilización de una galería de imágenes relacionadas con los puntos de interés, ya que la ya implementada utilizaba el servicio de Panoramio, y habría que valorar si esos servicios siguen estando disponibles o si es necesario valerse de otros, como Flickr o Instagram.
- Búsqueda de librerías nuevas para sustituir las que se encuentran obsoletas.
- Implementación de estas nuevas librerías, ajustando el código necesario.
- Estandarización de los métodos y la interfaz desarrollados en la aplicación.
- Optimización de las clases y tipos que se utilizan, ya que desde la versión de Android en la que se implementó originalmente la aplicación, han quedado varios obsoletos y han aparecido otros nuevos más efectivos.

- Aplicar técnicas de refactorización en el código de la aplicación.
- Implementación de aplicaciones externas como Twitter o TripAdvisor para poder descubrir las opiniones o valoraciones de otros usuarios.

Objetivos personales:

- Aprender a utilizar el entorno y lenguaje de programación de Android, ya que es un sistema que puede resultar muy útil para la vida laboral de un programador.
- Mejorar en el proceso de la gestión de tareas, con aplicaciones como GitHub.
- Aprender a utilizar el entorno de desarrollo de documentación LaTeX, ayudándome de la herramienta Texmaker.
- Mejorar en el trabajo con metodología SCRUM, utilizando Issues y Sprints para la realización del proyecto.
- Aprender a realizar trabajo en equipo, manteniendo comunicación con el compañero encargado de desarrollar el algoritmo del proyecto [5], para conseguir experiencia sobre un proyecto más realista que si se realizara solo.
- Conseguir mayor conocimiento sobre técnicas de refactorización y el software que se utiliza para detectar defectos de código.

Conceptos teóricos

En este apartado se describirán ciertos conceptos teóricos que resultan necesarios para la correcta comprensión del proyecto.

3.1. Modelo Cliente-Servidor

Una parte importante de este proyecto es el concepto del modelo cliente-servidor, ya que cumple con la estructura que se está utilizando. [10] [15]

Un esquema gráfico muy utilizado para definir este modelo es el siguiente:

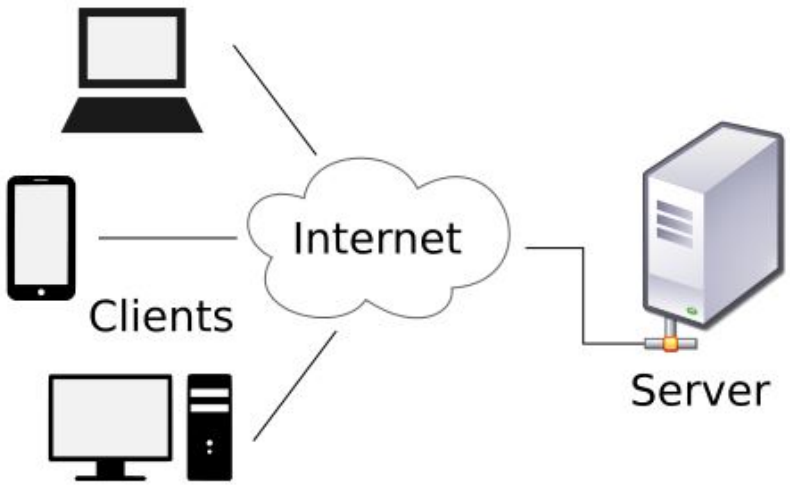

Figura 3.1: Definición gráfica del modelo cliente/servidor [9]

En este modelo y en el proyecto hay unas características principales y unos actores concretos que son los siguientes:

Cliente

- Este se encarga de lanzar solicitudes o peticiones al servidor. Cuando el cliente requiere cierta información, lanza una petición al servidor, quedando a la espera de recibir una respuesta. Normalmente en esta espera se encuentra definido un tiempo de espera máximo, para evitar desperdicio de recursos.
- El cliente es capaz de lanzar peticiones a diferentes servidores, ya que pueden ser tipos de datos muy variados y sería impensable que un mismo origen de datos le ofreciera todo lo que necesite.
- Este cliente contiene la interfaz que utilizará el usuario final, por lo que se consigue una interacción directa.

Servidor

- El servidor es la otra parte indispensable de este modelo, ya que en el momento en que esta iniciado se encarga de escuchar a la espera de peticiones del cliente.
- Al igual que un cliente puede realizar distintas peticiones a distintos servidores, el servidor puede mantener conexión con un gran numero de clientes y procesar las peticiones.

Al estar usando este modelo se consigue una serie de ventajas:

- **Centralización:** Al contar con un servidor que recibe todas las peticiones que le lance el cliente, se consigue un control de todos los accesos que se realizan y los recursos que se utilizan, pudiendo restringir o aumentar donde sea necesario.
- Es un sistema muy fácilmente escalable, pudiendo aumentar las capacidades tanto de un lado como del otro independientemente, sin que al otro le afecte, o en todo caso teniendo que hacer modificaciones mínimas para que sea funcional. Esto mismo se traduce igualmente a la facilidad de mantenimiento, por las mismas razones.

- Teniendo relación con la primera ventaja, se obtiene seguridad, ya que actualmente existen tecnologías que permiten conexiones seguras entre cliente y servidor, y a su vez evita posibles accesos fraudulentos que tengan intenciones no deseadas, ya que aunque varios clientes estén accediendo al mismo servidor, no pueden conocer las direcciones IP del resto.

3.2. Librería

Cuando se habla de librerías [16] [14] podemos considerar varias opciones, pero a la que se refiere este caso es el hecho de desarrollar software de un tipo específico siguiendo ciertas reglas establecidas para así conseguir tener una cantidad indeterminada de programas y aplicaciones que contengan una estructura y una programación parecida, ya sea por el uso de los mismos métodos, o por seguir ciertos patrones con los que estructuras los programas de una manera concreta.

Este modelo estándar es algo que se consigue a través de la integración de librerías de código. Con ellas obtenemos métodos ya desarrollados para realizar determinados propósitos de programación. Al ser un origen de datos que siempre tendrá estos mismos métodos, se alcanza una estandarización a un gran nivel.

Al utilizar las librerías, también se obtiene un nivel de jerarquía que permite una organización más clara y sencilla. Una serie de clases que dependan de una librería acabarán teniendo varias similitudes, incluso compartirán código, lo cual servirá para agruparlas de mejor manera.

En el caso de Android, se puede considerar como estándar la librería que ofrecen para programar, ya que todas las aplicaciones que están desarrolladas por Google para este tipo de dispositivos mantienen una estructura bastante parecida en el desarrollo de ciertas técnicas.

Una forma visual de demostrar esto se encuentra en el menú que utilizan la mayoría de aplicaciones, como el de Google Maps, Gmail o la aplicación para visualizar imágenes. Todos estos ejemplos utilizan una estructura prácticamente idéntica, y con esto se consigue lo ya mencionado, estandarización, que será explicado en el apartado de técnicas y herramientas.

3.3. Métricas de calidad de software

[1]

Un *Code Smell* se refiere a una serie de problemas que puede contener el código de un programa, que no causa un error a la hora de ejecutar o compilar el programa, pero es un problema de carácter mas profundo, que acarrea otra serie de fallos, como puede ser la lentitud en un programa.

Estos problemas también dificultan el mantenimiento del código a futuro, ya que puede ser un error que en la actualidad no cause fallos, pero en el momento que se realicen modificaciones un programa que contenga estos Code Smells será mas propenso a fallos críticos, que, dependiendo del tipo de Code Smell que lo haya provocado, puede ser mucho mas difícil detectarlo en ese momento que cuando se cometió.

Gracias a la identificación de los Code Smells, se puede saber cuándo es necesario refactorizar código, que es una técnica que se explicará mas adelante. Con respecto a la identificación también es importante mencionar que tiene un carácter muy subjetivo, dependiendo del programador o incluso del lenguaje de programación, aunque existen herramientas que son capaces de detectar bastantes tipos de defecto, ya que algunos son mas subjetivos que otros (en el caso de código duplicado, aunque también pueden existir falsos negativos).

3.4. API

Una API [12] o interfaz de programación de aplicaciones [18] se integra en otro software para que pueda hacer uso de rutinas, métodos y procedimientos con los cuales incluya su funcionalidad.

Visto de otra forma, se basa en el uso de bibliotecas, y con esto se consigue que el programador no tenga que desarrollar todo el código que se encuentra en ellas, ya que con hacer uso de sus métodos y llamadas ya consigue esas funcionalidades.

Por otro lado, al hacer esto también se consigue estandarización, debido a que todos los programas que usen esas librerías para implementar sus funcionalidades estarán haciendo los mismos procesos de llamadas y usando los mismos métodos, así que se benefician de todas las ventajas con las que cuenta la estandarización, pero a su vez también corremos el riesgo de que esa estandarización se rompa, bien porque la API deje de dar soporte a la biblioteca que estamos utilizando, o porque en las nuevas versiones de una aplicación ya no sea compatible nuestra API implementada.

Podemos ver cómo se estructura la utilización de las APIs a través de este esquema:

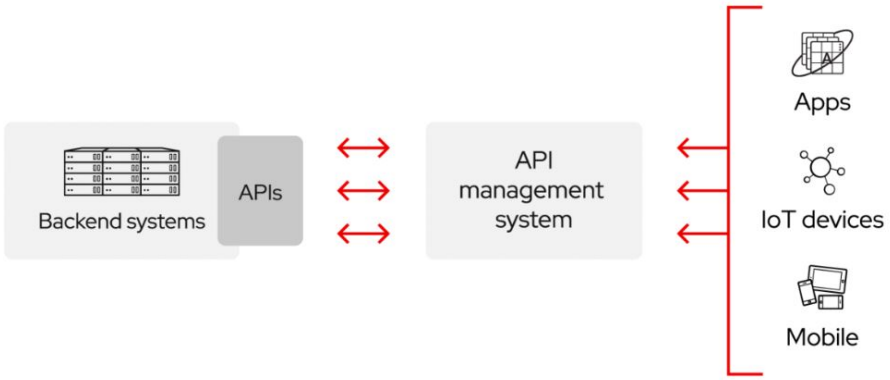

Figura 3.2: Esquema explicativo del uso de las APIs, fuente: [12]

3.5. Mantenimiento

El concepto de mantenimiento en informática posee mucho valor, ya que el desarrollo de cualquier aplicación requiere mantenimiento posterior, y lo podemos dividir en dos aspectos:

Soporte

El soporte [6] [13] es una parte del mantenimiento, ya que después de cualquier desarrollo siempre habrá errores, *bugs* o fallos, y éstos pueden ser problemas que no afecten al funcionamiento normal de un aplicación o por el contrario que causen su parada total, quedando inutilizable.

El soporte es un servicio a través del cual un especialista informático brinda asistencia, de forma física o remota a un individuo o una organización que hacen uso de la herramienta.

En el soporte remoto,se realiza a través de una aplicación web o un foro, donde además de personas especializadas en la aplicación, también hay otros usuarios que aportan su experiencia sobre ciertos problemas y en muchos casos estos problemas pueden ser resueltos sin necesidad de modificar el código, porque la falta de conocimiento del usuario es una de las mayores causas de incidencias que se suelen resolver en entornos de soporte.

Actualización

La otra parte importante del mantenimiento de una aplicación, sobre la que haremos mención en muchos aspectos de este trabajo es la actualización

[20].

Se basa en el cambio de código, APIs y librerías para obtener métodos nuevos, que están más optimizados para realizar la misma funcionalidad o también para solucionar errores que existieran en versiones anteriores. Por último, también se aplica a la implantación de nuevas funcionalidades de una misma aplicación, para que cada vez cumpla mejor la labor para la que haya sido desarrollada. En conjunto, supone una parte fundamental y necesaria para la aplicación. Cuando la actualización no se realiza con cierta periodicidad la aplicación acaba sufriendo pequeños errores o errores críticos. Cuanto más largo sea el periodo entre una actualización y la siguiente más complicado será conseguir que todo lo que se haya ido cambiando o mejorando sea aplicado de manera fácil.

Técnicas y herramientas

En este apartado se destacarán las principales técnicas y herramientas utilizadas para el proyecto. Cabe destacar que habrá herramientas también utilizadas que no serán explicadas, bien por pertenecer a la parte del servidor o por ser herramientas que fueron utilizadas en la anterior versión de este trabajo, pero que para esta versión sus funcionalidades no se han modificado y por tanto no se ha aportado nada adicional a lo que ya se implementó.

Técnicas

Las técnicas utilizadas durante el desarrollo de un proyecto pueden ayudar a comprender mejor la metodología que se ha llevado a cabo, así como la robustez que tiene el código. También nos permiten observar ligeramente el rumbo que ha tomado el proyecto, ya que puede ser un proyecto mas enfocado a la implementación de métodos desde cero, o para complementar un proyecto con un ciclo de vida mas largo. Por otro lado, también se puede estar buscando el mantenimiento, optimización y actualización del código que ya esta desarrollado en un proyecto, como es nuestro caso.

Refactorización

La refactorización [19] [1] es una técnica utilizada en programación para realizar modificaciones sobre el código de un programa en cuanto a la estructura interna, pero sin que el comportamiento externo se vea afectado. Son cambios que se pueden apreciar estructuralmente y que a simple vista podrían no tener importancia, pero a medida que un programa avanza, se actualiza o se dejan de usar ciertas técnicas para pasar a usar otras, haber hecho esta refactorización cobra mas importancia.

Si nos paramos a pensar, el termino de refactorización tiene el origen en las matemáticas, ya que la "factorización.[en] las matemáticas lo que puede hacer es convertir un polinomio compuesto por una variable y un numero en 2 polinomio mas simples y con una estructura mas clara de ver. El funcionamiento externo es en sí el mismo, pero se consigue una limpieza o una robustez estructural. Esto es lo que se busca al hacerlo en la programación.

La refactorización abarca desde cambiar el nombre de una variable para que vaya mas acorde con la función que llevará a cabo, hasta mover un método de una clase a otra para así conseguir un acceso mas optimizado y una estructura mas lógica.

Hay una serie de refactorizaciones sencillas pero que son de las que mas se realizan, debido a que solucionan los code smells que más se cometen:

- EXTRACT METHOD: Es el proceso por el cual, tras detectar que ciertas porciones de código se repiten en distintos métodos o incluso dentro de un mismo método en partes distintas del mismo, pero que tienen la misma estructura y en definitiva cumplen la mismo función, por lo que esas porciones de código repetido pueden ser extraídas para crear un método nuevo, de tal manera que cuando se necesite hacer el proceso que exija usar ese mismo proceso, bastará con llamar al método.

  Así se consigue una estructura más robusta en el código, así como mejor mantenibilidad del mismo, ya que si en un momento dado es necesario modificar esas lineas, en vez de modificarlas en cada una de las localizaciones donde estuvieran, solo se hará en el método nuevo.

  Un ejemplo gráfico donde se ve lo que se acaba de explicar:

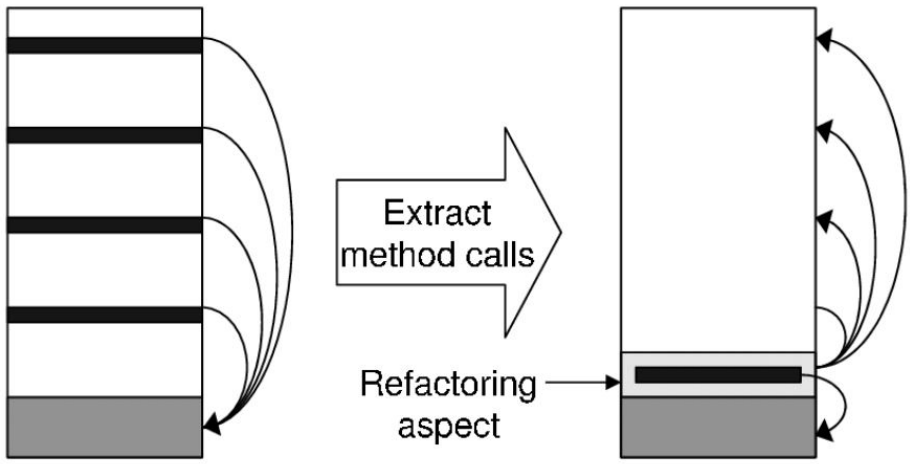

Figura 4.3: Ejemplo sobre la técnica de refactorización *extract method* [2]

- MOVE METHOD: En este caso, durante la detección de los posibles problemas que pueda tener el código, se observa que un método que se encuentra en una clase, no esta correctamente implementado en la misma, ya que a lo mejor esta haciendo mas uso de recursos de otra clase distinta, y así se realiza, se decide mover ese método a la clase en la que debería estar implementado, ya que esto hace que la estructura sea más lógica, y por consiguiente mas óptima.

  Un ejemplo gráfico donde se ve lo que se acaba de explicar:

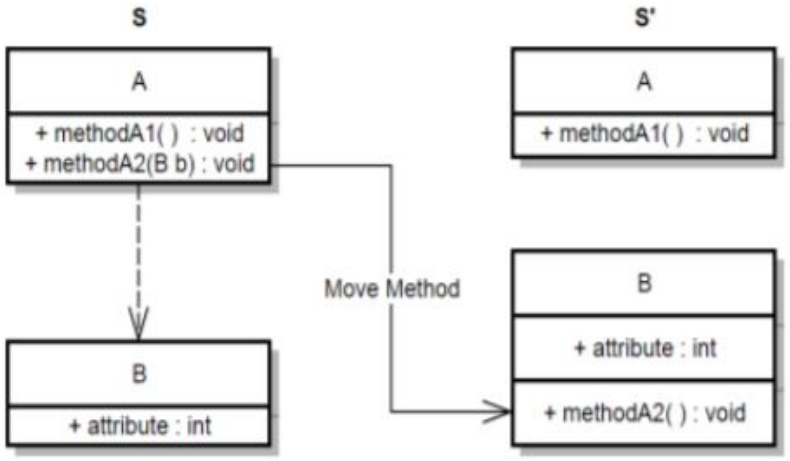

Figura 4.4: Ejemplo sobre la técnica de refactorización *move method* [7]

- RENAME: Este tipo de refactorización es el más sencillo y a la vez es el que más necesario resulta en muchas ocasiones, ya que se basa en renombrar una variable, método o clase para que tenga más sentido con respecto a la labor que desempeñe. El hecho de que existan nombres excesivamente genéricos o que directamente no cumplan con lo que es realmente es un error muy frecuente, por tanto, esta refactorización es muy frecuente.

Migración

El concepto de migración puede ser muy extenso, pero en este caso trataremos acerca de la migración en el software.

Este tipo de migración se realiza cuando se quiere cambiar por completo un software para pasar a utilizar otro que este mas actualizado, o cumpla mejor con las necesidades que exija el proyecto en cuestión.

Para relacionar la técnica con este proyecto, hay que mencionar Android, ya que esta técnica se ha aplicado para realizar la migración desde Android hacia AndroidX, ya que éste último es la version mejorada del anterior. Pero no es una simple actualización, ya que se encuentran en librerías diferentes, pero esto se explicará mas adelante en el apartado de herramientas.

Herramientas

Como ya se ha comentado antes sobre las técnicas y herramientas utilizadas, solo se destacara las que han tenido una verdadera relevancia para este proyecto, ya que hay algunas que ya se mencionan en la version anterior de TourPlanner.

Android

En esta seccion se hablará de Android como herramienta principal, pero también de dos herramientas directamente relacionadas, que son Android Studio y AndroidX.

Android Studio es un

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

En este apartado se van a destacar los aspectos que más relevancia han adquirido a lo largo del desarrollo del proyecto, bien por haber realizado cambios importantes o por haber supuesto un reto más en el trabajo.

Se verán ciertos aspectos que estarán más desarrollados en los anexos, sobre todo en el anexo 4. También se tratará de justificar las modificaciones que se hayan realizado sobre el proyecto anterior [4], para que quede claramente diferenciada la aportación que ha supuesto este proyecto.

5.1. Trabajar sobre un proyecto ya desarrollado

En esta sección se quiere mostrar los aspectos más importantes a la hora de trabajar sobre otro proyecto que ya ha sido desarrollado, ya que pueden suponer ciertas dificultades si no se tienen en cuenta.

Para este proyecto, se ha comenzado con dos preámbulos ya mencionados, el primer desarrollo [3] donde se comenzaba el desarrollo de la aplicación y el segundo [4], donde se mejoraba este desarrollo anterior. A la hora de plantear esta tercera versión del proyecto, ha sido necesario realizar un trabajo inicial de documentación para poder entrar en contexto sobre lo que se había desarrollado y lo que quedaba por desarrollar. Para ello, los apartados sobre objetivos y lineas de trabajo futuras de ambos proyectos han sido bastante útiles, ya que a través de estos brindaban la visión que tenían sobre sus proyectos y las mejoras que se podían desarrollar.

También ha sido necesario investigar la antigüedad de estos proyectos, concretamente de la segunda versión [4], ya que era la última, aunque más adelante se vió que ésta había sido desarrollada sobre la misma versión tanto para el cliente como para el servidor. Por tanto, una parte crucial es la de actualizar, sobre todo en lo referente al cliente, ya que la versión de la que se habla es Android 4.4 (API 14), lo cual se sitúa 7 años atrás. Además durante estos últimos años, los dispositivos móviles han sufrido muchos avances tecnológicos y esto se ha podido ver incluso durante el desarrollo del proyecto, ya que cuando se comenzó la última version de Android era la de Android 9.0 y en la actualidad ya se ha lanzado Android 10. En este artículo [8] se puede ver de una manera sencilla la evolución que ha sufrido Android a lo largo de los años.

Uno de los cambios que ha supuesto toda esta evolución de Android surgió un año después del lanzamiento de Android 4.4, ya que en Android 5.0 se incluye el concepto de *Material Design* que sirve desde entonces como base de diseño de aplicaciones de Android para unificar todos los dispositivos que se encontraran por encima de esta versión. Es un paso más hacia la estandarización que en este proyecto también se ha incluido.

Otro aspecto importante a la hora de trabajar sobre un proyecto anterior, es la necesidad de comprender cómo se ha desarrollado el mismo y para ello toman mucha relevancia los comentarios que se realizaron sobre el código, para evitar confusiones o simplemente para conseguir que la experiencia de mejorar esta aplicación sea más llevadera.

```
/**
 * Clase que se corresponde con la pantalla de planifica tu viaje.
 *
 * @author Inigo Vázquez - Roberto Villuela
 * @author ivg0007@alu.ubu.es - rvu0003@alu.ubu.es
 */
public class PlannerActivity extends androidx.fragment.app.Fragment implements
        IWebServiceTaskResult {
```

Figura 5.5: Ejemplo de comentarios sobre el código que ayudan a la comprensión del mismo

5.2. Actualización y cambio de librerías

Uno de los aspectos más importantes para el desarrollo de este proyecto ha sido la actualización de varias librerías, ya que se encontraban obsoletas por haber dejado de recibir soporte o por estar desactualizadas. Tras haber visto el apartado sobre las librerías en los conceptos teóricos, se puede ver la importancia que poseen en un proyecto, por lo que sin éstas funcionando, no se puede desarrollar el resto de la aplicación.

Ha sido necesario realizar un trabajo de investigación sobre cada una de estas librerías obsoletas con el fin de encontrar una alternativa más estándar y actual, tratando de conseguir más accesibilidad y soporte en un futuro. Las librerías más importantes que han tenido que ser sustituidas son:

- **SlidingMenu:** Librería que permitía implementar un menú deslizante para poder navegar entre ventanas.
- **SherlockActivity:** Librería utilizada para la implementación de cada una de las ventanas (Activities) de la aplicación.
- **Org.Apache:** Librería que se utilizaba para realizar todas las peticiones HTTP y HTTPS hacia el servidor.

Toda la explicación sobre el cambio de estas librerías se encuentra en el anexo 4: Manual del programador.

La sustitución de librerías al completo dentro de una aplicación supone un cambio en todos los métodos que se aprovechen de ellas, ya sea a través de llamadas o variables de las mismas, para pasar a utilizar las nuevas. El problema principal es que en muchas ocasiones, aunque se consiga una nueva librería, puede que hayan cambiado la forma de hacer las llamadas a métodos o los parámetros dentro de esas llamadas y es necesario tenerlo en cuenta para poder hacer un cambio que realmente tenga un efecto positivo.

5.3. Cambio de la herramienta de desarrollo

Para este proyecto, también se ha modificado la herramienta de desarrollo que se estaba utilizando, ya que anteriormente se utilizó Eclipse, por ser el entorno oficial de desarrollo de aplicaciones Android, pero en 2015 se lanzó la primera versión de Android Studio, que lo sustituyó como herramienta oficial.

Al iniciar este proyecto se pudo ver que se había utilizado Eclipse al ser 2014 y no existir Android Studio, pero ahora ya es necesario utulizar esta nueva herramienta, que también ofrece nuevas funcionalidades, y un entorno de desarrollo totalmente especializado para las aplicaciones de Android.

Uno de los grandes cambios que supone esta herramienta, es *Gradle* [11] que es un sistema de automatización de construcción de código abierto, cuyos plugins iniciales están principalmente centrado en desarrollo y despliegue en Java, Groovy y Scala [17]. En concreto, Android Studio es una de las aplicaciones que lo integran para el desarrollo de aplicaciones, ya que permite automatizar la gestión de dependencias sobre librerías, APIs y repositorios.

Se puede ver con más detalle el cambio que ha supuesto la utilización de Android Studio en el anexo 4.

5.4. Programación en Android

5.5. Cambio de interfaz

5.6. Trabajo colaborativo

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Podemos concluir este proyecto indicando una serie de aspectos.

Código con múltiples autores

En esta sección, me gustaría mencionar lo complicado que puede resultar la reutilización de un proyecto en el que han participado varios autores. Esto se debe, entre otras cosas a las diferencias en la manera de estructurar, pensar o desarrollar el código de la aplicación. Enfrentarse a un reto como este puede aportar mucha experiencia de cara al mundo laboral, ya que cuando se trata de desarrollar proyectos fuera del ámbito universitario será lo más parecido que nos encontremos.

En ese sentido esta experiencia ha hecho que maduremos bastante como programadores y también ha aportado una visión diferente en la forma de programar, aunque inicialmente resulta bastante más dificultoso de lo que pudiera parecer.

Bibliografía

[1] Apuntes de la asignatura *Diseño y análisis de sistemas software*.

[2] Extract method. Gráfico que explica la técnica de refactoricación *extract method*.

[3] Trabajo de fin de grado - generación de rutas turísticas personalizadas.

[4] Trabajo de fin de master - ampliación de la aplicación para la generación de rutas turísticas personalizadas.

[5] Ignacio Aparicio Blanco. Trabajo de fin de grado - desarrollo de algoritmos para la generación de rutas turísticas. 2020.

[6] apser blog. Soporte informático: definición y tipos. Definición sobre el concepto de soporte informático.

[7] Ricardo Terra Christian Marlon Souza Couto, Henrique Rocha. Quality-oriented move method refactoring. Estudio sobre métodos de refactorización para mejorar la calidad de código, en concreto la técnica *move method*.

[8] Christian Collado. Versiones de android: un repaso a la historia del sistema operativo. Repaso por toda la evolución que ha sufrido Android a lo largo de los años.

[9] Autor desconocido. cliente servidor. Blog que define el modelo cliente/servidor, fuente de la imágen.

[10] Autor desconocido. Cliente/servidor, 2015. Trabajo sobre el concepto de Cliente/Servidor.

[11] Gradle. Gradle. Página oficial de Gradle.

[12] Red Hat. Qué son las api y para qué sirven. Definición de API.

[13] mikogo. Soporte informático. Qué es el soporte informático.

[14] Regalut. ¿que es una librería o biblioteca en informática? Definición de librerías.

[15] Juan José Gil Ríos. Introducción a los sistemas de información: El modelo cliente/servidor. Transparencias sobre el modelo Cliente/Servidor.

[16] Wikipedia. Definición de librerías (informática). Se define el término de biblioteca o librería informática.

[17] Wikipedia. Gradle. Definición de Gradle.

[18] Wikipedia. Interfaz de programación de aplicaciones. Definición de API.

[19] Wikipedia. Refactorización. Definición de refactorización.

[20] Wikipedia. Software upgrade. Definición del concepto de actualizar.