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 — 29 de junio de 2019
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, 29 de junio de 2019

Vº. Bº. del Tutor:

D. nombre tutor

Resumen

Este proyecto se basa en realizar 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 esta diseñada para ser utilizada en sistemas operativos Android.

Para desarrollar este trabajo hemos cooperado mi compañero, Ignacio Aparicio Blanco y yo, pero cada uno realizando dos partes separadas que finalmente cumplen una función juntas, ya que Ignacio ha realizado la parte del backEnd (algoritmos para la generación de las rutas y manejo de la base de datos) y yo me he encargado de la parte frontEnd (interfaz y funcionamiento interno de la aplicación Android).

A continuación explicaré brevemente las mejoras que he realizado en la parte FrontEnd de la aplicación. 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 futuro mucho mas sencillo.

Se ha conseguido actualizar la aplicación para que pueda funcionar en los dispositivos mas modernos, ya que esta desarrollada para la versión de Android 6.0, pero también es compatible hasta la versión Android 9.

También se ha hecho uso de una herramienta estándar para la creación del menú de la aplicación, que se encuentra dentro del paquete de Android. Con esto se consigue que la aplicación tenga una apariencia mucho más parecida a otras aplicaciones muy utilizadas en sistemas Android, como puede ser el propio Google Maps o la aplicación de Gmail, lo cual hace que resulte mas fácil de entender su funcionamiento para el usuario medio.

Otra parte de este proyecto es bastante centrada en la optimización con respecto a los recursos y al uso de ciertos métodos que contiene Android. Se ha cambiado la estructura de .ªctivities"dentro de la aplicación, para pasar a usar una estructura mas óptima.Más adelante se explicará en que consiste.

Por otro lado, como ya he dicho, este trabajo se centra en la estandarización, y por ello se han dejado de utilizar 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, menú, estandarización, frontEnd

...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

En esta memoria se podrán ver las distintas mejoras y actualizaciones que se han realizado para el trabajo de fin de grado. La aplicación sobre la que se desarrolla ya ha sido desarrollada anteriormente en distintos trabajos.

La parte mas computacional se basa en la generación de rutas turísticas optimas respecto a los gustos de un usuario y también añadiendo el factor de horarios de apertura de los puntos de interés. Esto se consigue con el desarrollo de una serie de algoritmos que ha desarrollado mi compañero Ignacio Aparicio Blanco.

Por el otro lado, la parte mas visual seria la correspondiente a la aplicación de Android, que se ha desarrollado de forma que cualquier usuario que tenga costumbre en el manejo de cualquier dispositivo Android le resulte fácil y accesible.

Por ultimo tendríamos la parte que corresponde a los datos propiamente dichos, que es la base de datos donde se almacenara información sobre los usuarios, puntos de interés, tiempos, rutas, etc.

Objetivos del proyecto

En este apartado explicaré 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 de proyecto desde un sistema Android obsoleto hacia la version mas 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 interes, ya que la ya implementada era utilizando el servicio de Panoramio, y habría que valorar si esos servicios siguen estando disponibles o si es necesario valerse de otros servicios, como Flickr o Instagram.
- Diferenciación entre una ruta y otra si se ha realizado en la misma ciudad, para que la aplicación calcule lugares diferentes basándose en los ya visitados.
- Implementación de aplicaciones externas como Twitter o TripAdvisor para poder descubrir las opiniones o valoraciones de otros usuarios.
- Diferenciación entre puntos de interés basándose en la actualización del algoritmo que pasa a utilizar horarios de apertura y cierre de los mismos, para que la aplicación no visualice ningún lugar que no pueda ser visitado por estos factores.

- Estandarización de los métodos que se utilizan y de la interfaz que se utiliza en la aplicación.
- Optimización de las clases y tipos que se utilizan, ya que desde la version de Android en la que se implementó originalmente la aplicación han quedado varios obsoletos y han aparecido otros nuevos mas efectivos.
- Aplicar técnicas de refactorización en el código que ya hay desarrollado en la aplicación.

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.
- Aprender a trabajar en modo 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, para conseguir experiencia sobre un proyecto mas 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 estamos utilizando. En este modelo y en el proyecto hay unas caracteristicas principales y unos actores concretos que son los siguientes:

Cliente

- Este se encarga de lanzar solicitudes o peticiones al servidor. Si lo comparamos con el proyecto de TourPlanner, podemos ver que la parte del cliente correspondería a la aplicación de Android.

- El cliente es capaz de lanzar peticiones a diferentes servidores, como es nuestro caso en el que lanza peticiones tanto al servidor de OpenStreetMaps para poder visualizar el mapa online, a la base de datos que desarrolla mi compañero(lanza la petición a través de Glassfish) o por ejemplo a un servidor que provee las imágenes de los puntos de interés.

- Obviamente este cliente contiene la interfaz que utilizará el usuario final, por lo que la interacción es 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.
- Para este proyecto, quien cumple la funcion de servidor es GlassFish, que es a su vez un intermediario entre la aplicacion de android y la base de datos.

El proceso básico que también se verá en detalle mas adelante es el siguiente: el usuario interactua directamente con la aplicación (cliente), la cual realiza la petición a GlassFish (servidor), que a su vez realiza la petición a la base de datos y se la devuelve al cliente.

Al estar usando este modelo conseguimos una serie de ventajas:

- Centralización: Al contar con un servidor que recibe todas las peticiones que le lance el cliente, conseguimos 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 cierta relación con la primera ventaja, conseguimos seguridad, ya que actualmente contamos con tecnologías que permiten conexiones seguras entre cliente y servidor, y a su vez evitando posibles hackeos, ya que aunque varios clientes estén accediendo al mismo servidor, no pueden conocer las direcciones IP del resto.

3.2. Estándar

Cuando se habla de estándar podemos considerar varias opciones, pero a la que nos referimos en este caso es el hecho de desarrollar software de un

tipo especifico 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 de una misma librería, o por seguir ciertos patrones con los que estructuras los programas de una manera concreta.

En el caso de Android, podemos considerar como estándar la librería en si que ofrece Android 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 las fotos. Todos estos ejemplos utilizan una estructura prácticamente idéntica, y con esto se consigue lo que se llama estandarización, lo cual será explicado en el apartado de técnicas y herramientas.

3.3. Code Smell

Cuando se habla de Code Smell se esta hablando de problemas que puede contener el codigo de un programa, que no causa un error a la hora de ejecutar o compilar el programa, pero es un problema de caracter 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, como su propio nombre indica, provee de una interfaz de programacion de aplicaciones, en concreto, de una aplicación que será la que provee de esa API a otro software para que pueda hacer uso de rutinas, métodos y procedimientos con los cuales incluya la funcionalidad del software proveedor de la API.

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 procesos de llamadas y usando los mismos métodos, así que se benefician de todas las ventajas con las que cuenta la estandarización, así como alguna de las limitaciones.

Llevándolo a nuestro caso, tendríamos por un lado la API que nos permite visualizar imágenes de los puntos de interés, o por ejemplo la API con la que estamos viendo los mapas (OpenStreeMaps).

3.5. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.6. Referencias

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**].

3.7. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.8. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.9. Tablas

Igualmente se pueden usar los comandos específicos de LaTeXo bien usar alguno de los comandos de la plantilla.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

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 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.

- 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.
- 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

hola

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.

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.