



Escuela
Politécnica
Superior

TripMinder



Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:

Alejandro Jover Morales

Tutor/es:

José García Rodríguez, Sergio Orts Escolano



Universitat d'Alacant
Universidad de Alicante

Justificación y Objetivos

La principal motivación de este Trabajo de Fin de Grado, tal como indica su definición, es la puesta en práctica de los conocimientos adquiridos durante los estudios de grado, en este caso Ingeniería Multimedia. Dado que se trata de una nueva titulación, es posible que no queden claras las competencias y perfiles de los Ingenieros Multimedia, por lo que intentaremos aclarar sus capacidades y atribuciones profesionales.

Según sus perfiles profesionales, disponibles en la memoria de la titulación accesible desde la web de la Universidad de Alicante, un Ingeniero Multimedia es una persona capaz de dirigir proyectos profesionales en los sectores del *ocio digital*, entre los que destaca la creación de videojuegos, realidad virtual y producción, así mismo, en el sector de la *difusión de contenidos digitales*, donde es capaz de crear sistemas de difusión de contenidos multimedia, así como gestores de contenido. Además, un Ingeniero Multimedia debe ser capaz de analizar y captar las necesidades y transformarlas en un sistema o producto.

Sin embargo, por mi propia experiencia adquirida durante la carrera, un Ingeniero Multimedia va mucho más allá. Es un profesional con un perfil muy emprendedor e independiente, capaz de desarrollar habilidades relativas al diseño, desarrollo y programación. Un practicante de las nuevas tecnologías web y el desarrollo en los nuevos dispositivos, que adquiere el concepto de *multiplataforma* como una necesidad. Un *UI y UX designer*.

Por tanto, la principal motivación de este Trabajo de Fin de Grado es demostrar dichos conocimientos adquiridos y ponerlos en práctica para crear un aplicación multiplataforma centrada en proporcionar una gran experiencia de usuario.

Otra motivación es la del uso de los diferentes dispositivos multimedia para aplicar el concepto de multiplataforma como meta personal, ya que parece que el futuro lo están ganando los dispositivos móviles. Éstos no sólo pueden comunicarse a través de

internet y ser de bolsillo, permitiendo que una persona esté conectada a internet en todo momento, sino que también tienen muchas capacidades añadidas, gracias a sus sensores de luz, GPS, giroscopio, acelerómetro, magnetómetro, proximidad y un etcétera que se puede seguir ampliando.

Las posibilidades de uso de un smartphone están más que demostradas, por lo que empieza a encontrarse su aplicación en el E-commerce, no sólo haciendo compras en sistemas como Ebay a través de tu móvil, permitiendo incluso pagar en un restaurante con tu dispositivo móvil mediante tecnología NFC, lo que convierte este sector en Mobile-Commerce.

El objetivo principal de la realización de este TFG es la creación de TripMinder, una aplicación multiplataforma que aprovecha la información multimedia y las conexiones para ofrecer al usuario una completa herramienta de planificación de viajes y guía turística. Durante el desarrollo se espera aplicar y aprender más sobre los conceptos adquiridos en el Grado de Ingeniería Multimedia, así como seguir ampliando mis capacidades y habilidades en el mundo del desarrollo.

Agradecimientos

Debo dedicar un especial agradecimiento a mis tutores José García y Sergio Orts, por todo el tiempo que han dedicado a ayudarme durante todo el proceso, tanto presencialmente como durante mi estancia en Salzburgo (Austria). Por los consejos y sugerencias, las correcciones, las reuniones semanales y el tiempo encontrado de dónde no lo había.

Además agradecerles no sólo por este TFG, sino también por la enseñanza y ayuda prestada durante todo el grado de Ingeniería Multimedia como profesores.

Dedicatoria

Principalmente, debo dedicar este proyecto a mis padres, mi hermana y mis abuelos, presentes y no presentes. Ellos han sido quienes siempre me han apoyado en los estudios y me han proporcionado todos los medios para conseguirlo.

A mi pareja, que siempre me apoya y me guía en las decisiones importantes y está ahí siempre que necesito contar con ella.

Y en general a mis amigos, las personas que te sacan de la rutina de los estudios y proyectos para pasar un buen rato. En especial Daniel Mesa, quien siempre saca un rato para vernos, me apoya y se enorgullece como un padre cuando le enseño mis proyectos.

Índice de contenido

1. Introducción.....	16
1.1. Acrónimos	17
1.2. Definiciones	17
1.3. Relación con asignaturas.....	18
2. Estado del arte.....	20
2.1. Sistemas similares	21
2.1.1. GoEuro.....	21
2.1.2. Google Maps.....	23
2.1.3. DB Navigator (DB Bahn).....	26
2.1.4. RouteRank	27
2.1.5. Planificador Guía Repsol	28
2.1.6. Planificador Way-Away	29
2.2. Modelos de negocio	30
2.2.1. RouteRank	30
2.2.2. GoEuro.....	31
2.3. Comparativa entre sistemas.....	32
2.3.1. GoEuro & RouteRank	32
2.4. Limitaciones en los sistemas.....	34
2.5. Comparativa de tecnologías	35
2.5.1. RAD Studio, de Embarcadero Technologies.....	35
2.5.2. Xamarin.....	37
2.5.3. Apps híbridas	38
2.5.4. Ionic Framework.....	40
2.5.5. Conclusión y tecnología seleccionada.....	42
3. Objetivos.....	44
4. Estudio de las fuentes de datos.....	46
4.1. API's de geolocalización.....	46
4.1.1. Google Places API	47
4.1.2. API de codificación geográfica de Google.....	47
4.2. API's de tráfico y tránsito	48
4.2.1. GTFS	48

4.2.2.	Google Transit.....	49
4.2.3.	Google Maps.....	49
4.2.4.	QPX Express API	51
4.2.5.	Flight Aware API	51
4.2.6.	FlightStats Flex API.....	52
4.2.7.	Taxi Fare Finder API	52
4.2.8.	InKnowledge Taxi Fare Calculator REST Service	52
4.2.9.	Otras API's	53
4.3.	API's de información turística.....	53
4.3.1.	Google Places.....	53
4.3.2.	Yelp API	54
4.3.3.	Minube API.....	54
4.3.4.	Tixik API	54
4.3.5.	Comparativa de API's.....	55
4.4.	API's que utiliza TripMinder	56
5.	Metodología.....	58
5.1.	Metodologías software	59
5.2.	Metodología en TripMinder.....	63
5.2.1.	Planificación y gestión del proyecto	64
5.2.2.	Documentación	64
5.2.3.	Diseño.....	65
5.2.4.	Desarrollo	66
6.	Especificación de requisitos	68
6.1.	Casos de uso	68
6.2.	Requisitos funcionales.....	74
6.3.	Requisitos de interfaz.....	78
6.4.	Requisitos no funcionales.....	79
7.	Estudio de la viabilidad	82
7.1.	Planificación temporal	82
7.2.	Estimación de costes	85
7.3.	Estimación de riesgos	87
7.3.1.	Riesgos tecnológicos	87
7.3.2.	Riesgos de personal.....	87
7.3.3.	Riesgos de herramientas.....	88

7.3.4.	Riesgos de requerimientos.....	88
7.3.5.	Riesgos de estimación	89
7.4.	Línea de futuro.....	89
8.	Diseño y desarrollo del sistema	90
8.1.	Multiplataforma	90
8.2.	Arquitectura	92
8.2.1.	Diagrama	93
8.2.2.	Views y Controllers.....	94
8.2.3.	Services	96
8.2.4.	Cordova plugins.....	97
8.2.5.	Bower packages	98
8.2.6.	Almacenamiento	99
8.3.	Harvesting (recolección de datos)	100
8.3.1.	Coche, bicicleta y a pie.....	101
8.3.2.	Autobús y tren.....	101
8.3.3.	Avión	102
9.	Diseño gráfico	108
9.1.	Conceptos previos.....	108
9.2.	Diseño previo	112
9.2.1.	Guía de estilo	112
9.2.2.	Reglas generales.....	112
9.2.3.	Paleta de colores	113
9.2.4.	Tipografía.....	119
9.2.5.	Cuadrícula.....	119
9.2.6.	Navegación	119
9.2.7.	Componentes UI y UX	120
9.2.8.	Bocetos (Mockups)	125
9.3.	Diseño final y resultado	130
9.3.1.	Logotipo y splash screen	131
9.3.2.	Diseño de pantallas	133
10.	Conclusiones.....	144
10.1.	Conclusiones sobre los objetivos personales	144
10.2.	Conclusiones sobre los objetivos de la aplicación	146
10.3.	Mejoras y ampliaciones.....	148

10.4. Resultado y conclusión final	149
11. Bibliografía	150
12. Anexo	154

Índice de Figuras

Figura 2.1. Web de GoEuro	22
Figura 2.2. App de GoEuro	22
Figura 2.3. Uso de Google Transit en la web de Google Maps	24
Figura 2.4. Google Transit en la App de Google Maps.....	25
Figura 2.5. App de DB Navigator	26
Figura 2.6. Web de RouteRank.....	27
Figura 2.7. Web de Guía Repsol	28
Figura 2.8. Web de Way-Away	29
Figura 2.9. Plan de 5 días a París.....	29
Figura 2.10. Resultado de GoEuro para Alicante – París	32
Figura 2.11. Resultado de RouteRanks para Alicante – París.....	33
Figura 2.12. RAD Studio XE7	35
Figura 2.13. Xamarin	37
Figura 2.14. Apache Cordova	39
Figura 2.15. Titanium Studio	39
Figura 2.16. Ionic Framework	40
Figura 2.17. Comparación entre las apps generadas por Ionic Framework (TripMinder, parte superior) y por Rad Studio (TripMinder01, parte inferior)..	43
Figura 6.1. Diagrama de casos de uso	69
Figura 7.1. Diagrama de Gantt, página 1	83
Figura 7.2. Diagrama de Gantt, página 2	83
Figura 7.3. Informe de costes de TripMinder	86
Figura 8.1. Diagrama MVC de TripMinder	93
Figura 8.2. Cordova plugins que usa TripMinder	97
Figura 8.3. Lista de paquetes Bower	98
Figura 9.1. Comparación entre UX y UI designer	110
Figura 9.2. Paleta análoga	114
Figura 9.3. Paleta monocromática.....	115
Figura 9.4. Paleta complementaria	115

Figura 9.5. Triada	116
Figura 9.6. Paleta compuesta	117
Figura 9.7. Paleta de TripMinder	118
Figura 9.8. Pantallas de navegación de TripMinder.....	120
Figura 9.9. Ejemplo de componentes UI.....	121
Figura 9.10. Ejemplo de dark y light UI	122
Figura 9.11. Summer UI Kit, ejemplo de UI con estilo Flat Design	124
Figura 9.12. Pantalla principal	125
Figura 9.13. Resultados	126
Figura 9.14. Seguimiento	126
Figura 9.15. Crear guía turística	126
Figura 9.16. Detalle de ruta.....	126
Figura 9.17. Detalle de guía	127
Figura 9.18. Pantalla principal en tablet.....	128
Figura 9.19. Resultado de búsqueda en tablet	129
Figura 9.20. Logotipo de TripMinder	131
Figura 9.21. Splash screen de TripMinder	132
Figura 9.22. Inicio	133
Figura 9.23. Menu lateral.....	133
Figura 9.24. Ventana de progreso de búsqueda	134
Figura 9.25. Resultados de búsqueda.....	135
Figura 9.26. Vuelos y pantalla de otras opciones.....	136
Figura 9.27. Detalles de ruta de transporte público	137
Figura 9.28. Selección de Guía	138
Figura 9.29. Mensaje con enlace a preferencias	139
Figura 9.30. Resumen de guía, y mensaje que aparece al guardar.....	140
Figura 9.31. Mis guías.....	141
Figura 9.32. Detalle de guía en Mis guías.....	141
Figura 9.33. Preferencias	142
Figura 9.34. Historial	143

Índice de Tablas

Tabla 2.1. Comparativa de planes de RouteRank.....	31
Tabla 2.2. Pros y contras de RAD Studio XE7	36
Tabla 2.3. Pros y contras de Xamarin.....	38
Tabla 4.1. Tabla de limitaciones de Google Maps	50
Tabla 4.2. Tabla de precios de Flight Aware API.....	51
Tabla 4.3. Comparativa de API's de información turística.....	55
Tabla 4.4. Relación entre API's y funcionalidad	56
Tabla 6.1. Caso de uso 1	70
Tabla 6.2. Caso de uso 2	70
Tabla 6.3. Caso de uso 3	71
Tabla 6.4. Caso de uso 4	71
Tabla 6.5. Caso de uso 5	71
Tabla 6.6. Caso de uso 6	72
Tabla 6.7. Caso de uso 7	72
Tabla 6.8. Caso de uso 8	72
Tabla 6.9. Caso de uso 9	73
Tabla 6.10. Caso de uso 10.....	73
Tabla 6.11. Caso de uso 11.....	73
Tabla 6.12. Caso de uso 12.....	73
Tabla 6.13. Requisito funcional 1.....	74
Tabla 6.14. Requisito funcional 2.....	74
Tabla 6.15. Requisito funcional 3.....	75
Tabla 6.16. Requisito funcional 4.....	75
Tabla 6.17. Requisito funcional 5.....	75
Tabla 6.18. Requisito funcional 6.....	75
Tabla 6.19. Requisito funcional 7.....	76
Tabla 6.20. Requisito funcional 8.....	76
Tabla 6.21. Requisito funcional 10.....	76
Tabla 6.22. Requisito funcional 11.....	77
Tabla 6.23. Requisito funcional 12.....	77

Tabla 6.24. Requisito funcional 13	77
Tabla 6.25. Requisito funcional 14	77
Tabla 6.26. Requisito de interfaz 1	78
Tabla 6.27. Requisito de interfaz 2	78
Tabla 6.28. Requisito de interfaz 3	78
Tabla 6.29. Requisito no funcional 1.....	79
Tabla 6.30. Requisito no funcional 2.....	79
Tabla 6.31. Requisito no funcional 3.....	80
Tabla 6.32. Requisito no funcional 4.....	80
Tabla 7.1. Detalle de tareas.....	85
Tabla 7.2. Riesgo tecnológico 1	87
Tabla 7.3. Riesgo tecnológico 2	87
Tabla 7.4. Riesgo de personal 1.....	87
Tabla 7.5. Riesgo de personal 2.....	88
Tabla 7.6. Riesgo de herramientas 1.....	88
Tabla 7.7. Riesgo de herramientas 2.....	88
Tabla 7.8. Riesgo de requerimientos 1	88
Tabla 7.9. Riesgo de estimación 1	89
Tabla 7.10. Riesgo de estimación 2	89

1. Introducción

En los últimos años el sector tecnológico ha experimentado un gran cambio. En la década de los 80 empezaban a aparecer los primeros teléfonos y dispositivos de comunicación. Sin embargo, parece solo un recuerdo. Actualmente, podemos encontrar vehículos autónomos sin asistencia humana, o casas inteligentes (domótica) controlables desde tu propio smartphone, que a su vez es un teléfono inteligente con el que puedes hacer multitud de cosas.

Haciendo hincapié en este último, el smartphone se ha convertido en el dispositivo más usado en los últimos años, y no para la comunicación hablada, sino para su uso como medio de acceso a internet. Las aplicaciones de mensajería, redes sociales y videojuegos son de las más utilizadas en un smartphone, especialmente en el ámbito universitario y juvenil.

Estas aplicaciones, utilizadas en smartphones y tablets, aprovechan al máximo el concepto de “Información Multimedia” y se han convertido en una potencia en el ámbito de la comunicación. Hoy en día, las mencionadas aplicaciones de mensajería y redes sociales comprenden el mayor uso de la comunicación.

El sector del ocio y del turismo juega un papel muy importante dentro de esta revolución tecnológica multimedia, ya que son la principal motivación de la creación de estas innovadoras aplicaciones. Hablando en particular del turismo, cada vez es más común y más económico viajar. Cada persona tiene su propio propósito y motivo para su viaje, entre los más comunes están los viajes de trabajo, visitas a lugares turísticos, interés por la gastronomía de la zona, o simplemente por el hecho de conocer nuevas culturas.

TripMinder es una aplicación multiplataforma que hace uso de la información multimedia para ser una completa herramienta de planificación de viajes, que provee al usuario información para desplazarse entre diferentes lugares usando diferentes medios alternativos. Para ello, calcularía las mejores rutas de cada medio de

transporte y así el usuario podría decidir cuál es la que más le conviene, en base a criterios de distancia, precio y modalidad de la ruta.

Aparte de herramienta de búsqueda de rutas, TripMinder ofrece la posibilidad de crear guías turísticas, según las preferencias de cada usuario, y personalizarla para ver tantos sitios como se quiera. De esta forma, desde la misma aplicación un usuario podría saber cómo ir a un sitio y los lugares más interesantes para visitar, dónde comer o dónde alojarse. Gracias a estas facilidades, el usuario puede despreocuparse ya que la aplicación encontrará todo lo que busca en el destino al que se dirige.

En definitiva, TripMinder está concebido para ser una herramienta “de bolsillo” imprescindible para ser utilizada antes y durante un viaje, permitiendo hacer un buen uso de la información para proporcionar la mejor experiencia de usuario.

1.1. Acrónimos

NFC: Near Field Communication

UML: Unified Modelling Language

IATA: International Air Transport Association

1.2. Definiciones

Tester: persona que se encarga de comprobar el correcto funcionamiento de un software.

UI designer: diseñador de interfaces de usuario

UX designer: diseñador de experiencias de usuario

Geocodificación: proceso por el que a partir de un lugar dado, por ejemplo una ciudad, se obtiene la información referente a su geolocalización, como las coordenadas. La Geocodificación Inversa es exactamente el proceso inverso.

Routing: concepto general que refiere al cálculo, uso y visualización de rutas de tránsito.

1.3. Relación con asignaturas

Ingeniería Multimedia es un estudio de grado en el que se encuentra principalmente dos itinerarios: Gestión de contenidos y Creación y entretenimiento digital.

El itinerario de **Gestión de contenidos** se centra en la creación, gestión y difusión de contenido multimedia. Aspectos como la programación web, programación para dispositivos móviles, difusión de contenidos o desarrollo de UI son algunos ejemplos de las competencias estudiadas en este itinerario.

Por otro lado, en el itinerario de **Creación y entretenimiento digital**, el enfoque principal reside en la creación de videojuegos, aunque también comprende el ámbito de la industria de producción digital y creación de imagen sintética para cine, televisión y efectos especiales, además de realidad virtual.

TripMinder no tiene relación ninguna con el itinerario de Creación y entretenimiento digital, por tanto la mayor parte está relacionada con el de Gestión de contenidos, además de asignaturas obligatorias comunes. A continuación se comenta la relación con cada asignatura y el aspecto que se abarca:

- **(PM) Proyectos Multimedia:** gestión del proyecto, documentación, análisis de costes y análisis de riesgos.
- **(SDM) Sistemas de Difusión Multimedia:** uso de API's y tratamiento de información.
- **(SMA) Servicios Multimedia Avanzados:** uso de servicios web y móviles, uso de nuevas tecnologías, capacidad de aprendizaje de nuevos frameworks.
- **(PH) Programación hipermédia (1 y 2):** desarrollo de la aplicación utilizando tecnologías web, entre ellas HTML5, CSS3 y javascript.
- **(DSM) Diseño de Sistemas Multimedia:** aplicación de los patrones MVC y Observer, desarrollo de interfaz de usuario UI.

- **(AESM) Análisis y Especificación de Sistemas Multimedia:** planteamiento y desarrollo de un sistema multimedia, análisis de requisitos, seguimiento de metodología de trabajo SCRUM y KANBAN, diagrama de casos de uso.
- **(DISM) Dispositivos e Infraestructuras para Sistemas Multimedia:** creación de aplicación multiplataforma, testeo de la aplicación en diferentes plataformas móviles y de escritorio.

2. Estado del arte

El uso de dispositivos móviles está en pleno auge y ha tenido una gran repercusión en la tecnología. Para empezar, el modelo de desarrollo ha cambiado totalmente. Mientras que antes se necesitaba un equipo de desarrollo de tamaño variable, según las necesidades del software, ahora un único desarrollador puede crear su propio proyecto. Además ha cambiado el modelo de negocio. Han aparecido las “tiendas de apps” donde cada desarrollador publica su aplicación y se lleva su porcentaje de beneficio, centralizando así para los usuarios la búsqueda y descarga de software y aumentando la seguridad en estas aplicaciones que son revisadas por un equipo de testers.

Se podría decir que estamos en período de **revolución tecnológica multimedia**, donde se están creando multitud de aplicaciones para muchos dispositivos. Hoy por hoy, los usuarios esperan aplicaciones que proporcionen una gran experiencia de usuario, en las que buscan funcionalidad, facilidad de uso, comodidad e innovación.

Como cabía esperar, a su vez, también han aparecido nuevas herramientas que permiten un desarrollo más sencillo, rápido, seguro y escalable. El problema es que han aparecido tantas, que no es trivial escoger la adecuada, ya que pueden existir múltiples aplicaciones que solucionen el mismo problema. Además. es habitual el uso de más de una herramienta y/o tecnologías para realizar un desarrollo. Esto es más común en entornos web o cliente-servidor.

Para conocer con mayor nivel de detalle el estado del arte, se van a realizar estudios y comparativas de los diferentes sistemas y tecnologías existentes en el ámbito de esta aplicación.

2.1. Sistemas similares

2.1.1. GoEuro

Es una aplicación desarrollada por una empresa cuya idea surgió a partir de un viaje mochilero que realizó su fundador y SEO Naren Shaam, donde observó las limitaciones en el acceso a información sobre cómo viajar entre ciudades o lugares de interés.

Actualmente sólo funciona en algunas regiones de Europa. La aplicación dispone de una interfaz bastante usable, tanto la app móvil como la web.

Características:

- Interfaz web, app Android e iOS
- Funciona en España, Reino unido, Alemania, Bélgica, Holanda, Luxemburgo
- Interfaz clara y sencilla
- Combina varios medios de transporte
- Permite ordenar y filtrar de muchas formas (en las app no de tantas)
- Muestra “en primer plano” la ruta más barata o más rápida para cada uno de los medios de transporte
- También tienen buscador de alquiler de coches, aunque no se contempla en las app

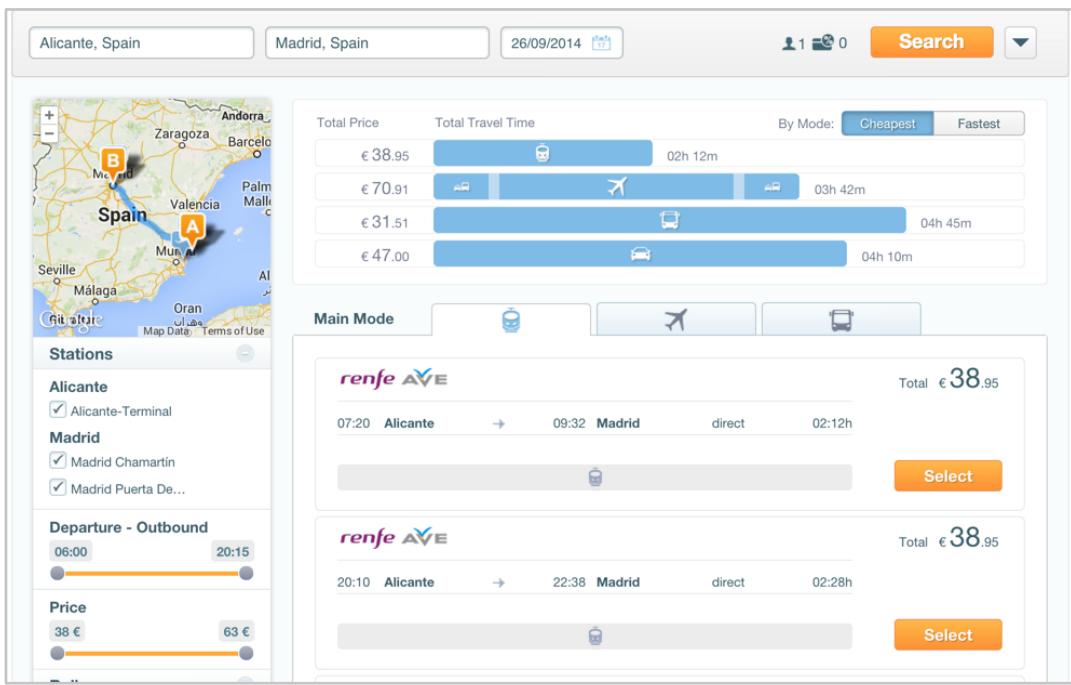


Figura 2.1. Web de GoEuro

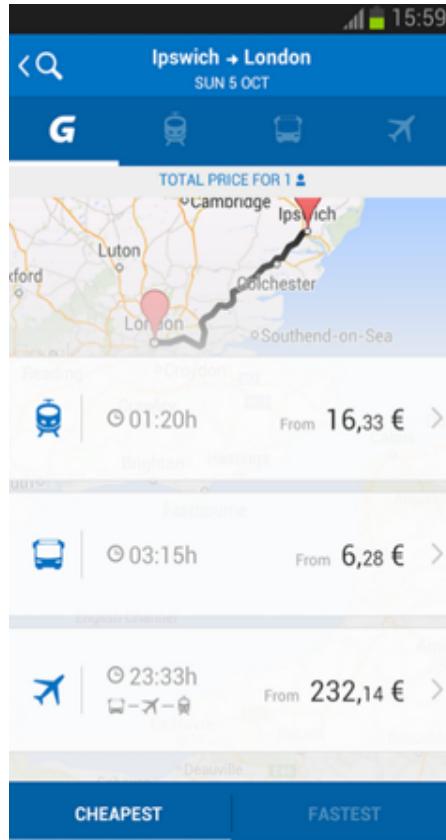


Figura 2.2. App de GoEuro

2.1.2. Google Maps

Combina las API's de *Google Flights*, *Google Transit*, *Google Traffic*, *Google Maps*, *Google Directions*, *Google Street View*, *Google Earth* y *Google Places*, para así proporcionar un sistema similar al de GoEuro.

Aunque éste se centra más en dar más información y navegar por las calles, aún así también presenta las mejores rutas en diferentes medios de transporte, precio, etc.

Características:

- Multiplataforma (Web, Android, iOS, Windows Phone, BlackBerry...)
- Explorar mapa, y poder guardar “vistas” de mapa para uso offline.
- Buscar rutas para ir de un sitio A a uno B
 - Seleccionar el medio de transporte (avión, bus, tren, tranvía, metro, coche, bici, andando)
 - Detalla los horarios y los puntos internos de la ruta. Muestra gráficas en la web.
 - Guía al usuario a través de la ruta
 - Permite filtrar resultados, por preferencias de medios, distancia, precio, etc.
 - Puedes indicarle la hora para que te aconseje las salidas
- Búsqueda de sitios de interés
 - Se pueden filtrar por precio, localización, distancia, horarios de apertura y cierre...
 - También por categorías
 - Puedes ver comentarios y sugerencias de la gente
- Con Google Images y Google Goggles App puedes hacer una foto y mostrar información sobre el sitio
- Aunque es una App separada, está MyMaps, que te permite planear vacaciones o encontrar alojamiento.

Defectos:

- Google Transit (la opción de medios de transportes públicos), no combina países. Por ejemplo, Alicante - París no muestra rutas en tren, tranvía, bus, etc. Sólo avión y coche.

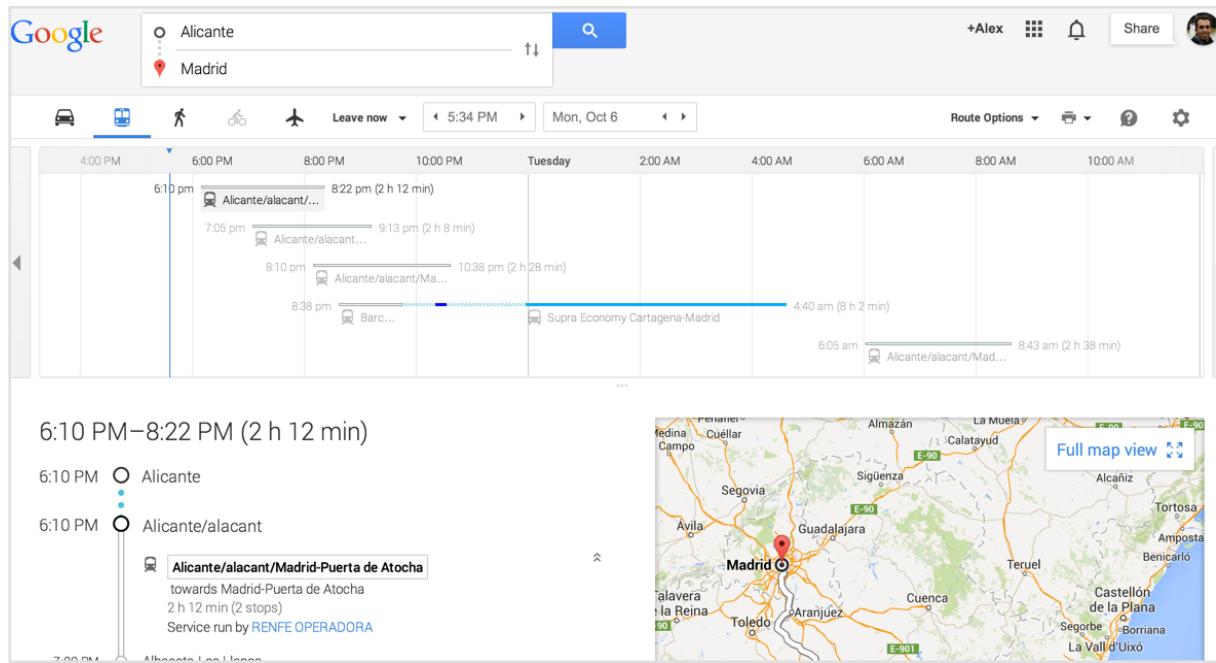


Figura 2.3. Uso de Google Transit en la web de Google Maps

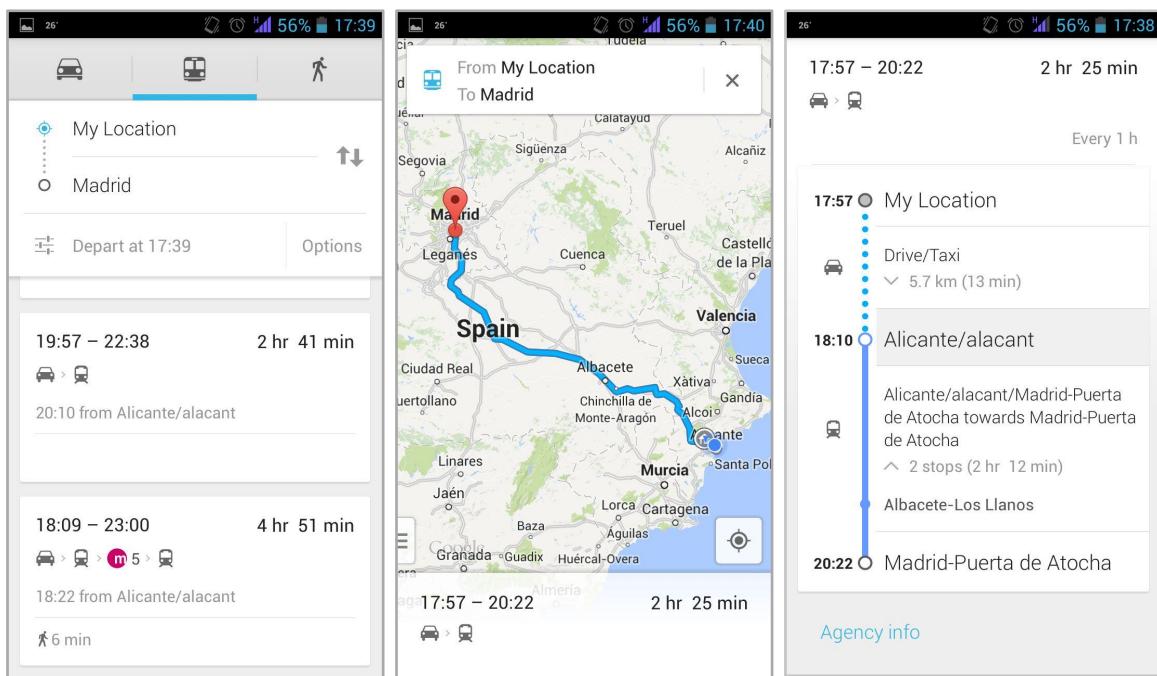


Figura 2.4. Google Transit en la App de Google Maps

2.1.3. DB Navigator (DB Bahn)

Sistema creado inicialmente para Alemania, aunque ahora funciona también para exteriores aunque de forma más limitada.

Características:

- App para iOS, Android y Blackberry, además de interfaz web.
- Es muy funcional, incluye reserva de ticket y puedes escoger el destino y origen según una estación concreta.
- Bastante personalizable, con muchos parámetros de configuración.

Desventajas:

- Tiene tantos parámetros y opciones que se hace difícil de usar.
- Se necesita saber dónde están ubicadas las estaciones.
- Interfaz con una usabilidad muy pobre y poco organizada.

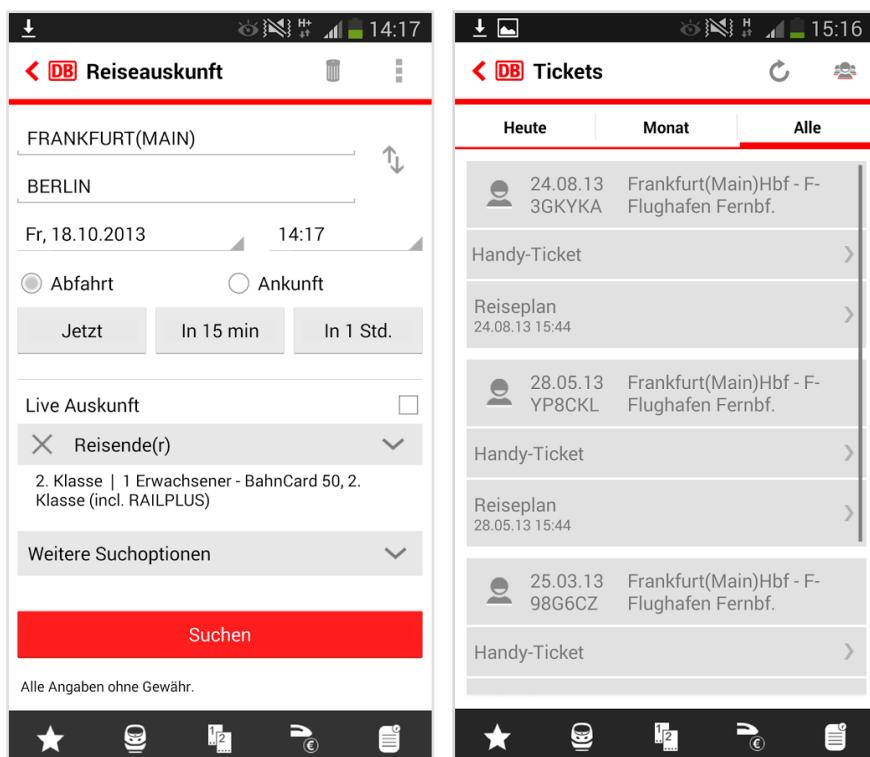


Figura 2.5. App de DB Navigator

2.1.4. RouteRank

Otro sistema de la misma categoría. Lleva más años en el sector que GoEuro, lo que le proporciona cierta experiencia. Además, se preocupan por el medio ambiente, proporcionando información sobre el CO₂ consumido para cada medio de transporte.

Características:

- Funciona también fuera de Europa
- Permite además ordenar por horarios de salida , llegada y emisión de CO₂
- No tienen app
- No permite bus
- Interfaz más anticuada y obsoleta

The screenshot shows the RouteRank website interface. At the top, there are input fields for 'De' (Alicante (Valencia), España) and 'A' (Madrid (Comunidad de Madrid), España ida y vuelta). Below these are date and time inputs ('Fecha' 01-10-2014, 'Hora' 08:00). To the right, there's a 'Blog' section with a link to 'Total Trip Planning video, TCS route planner and meet us at ITB'. Social sharing icons for Twitter, Facebook, and LinkedIn are present, along with a 'Compartir' (Share) button and a 'Newsletter' link. A search button labeled 'Encuentre y reserve' (Find and book) is also visible. The main content area shows a progress bar 'Buscando Rutas en Automóvil: ViaMichelin...' followed by a yellow banner 'Búsqueda de Alicante (Valencia), España a Madrid (Comunidad de Madrid), España el 01-10-2014 a las 08:00'. Below this is a table listing three train routes:

SAL	LLE	VIA	MEDIOS	TIEMPO	CO ₂	PRECIO	Datos reserva	Compensar CO ₂
11:00	14:22	En tren	tren	3h22	20kg	CHF 78.-	Datos reserva	Compensar CO₂
9:20	11:47	En tren	tren	2h27	20kg	CHF 80.-	Datos reserva	Compensar CO₂
12:40	15:07	En tren	tren	2h27	20kg	CHF 80.-	Datos reserva	Compensar CO₂

Figura 2.6. Web de RouteRank

2.1.5. Planificador Guía Repsol

Este sistema no calcula rutas, sino que muestra información y sitios que visitar del lugar al que quieras desplazarte, por lo que pertenece al sector de información turística.

Permite buscar los rincones de más interés de una ciudad o destino seleccionado, filtrado por categorías, tales como museos, restaurantes, lugares para visitar, etc. Además, se puede crear un “Plan de viaje” para así luego poder ver el plan a modo lista o mapa, con opción a imprimir o descargar ese plan.

The screenshot shows a web interface for travel recommendations in Madrid. At the top, it says "Nuestras recomendaciones en Madrid". There is a "Cambiar destino" button and a "Ver todo" button. Below this, there is a "Filtros" section with a filter icon. The main content area displays three cards:

- LOCALIDAD**: Madrid (0 m). Description: Abierta y dinámica, la capital de España despliega una oferta lúdica y cultural capaz de satisfacer a cualquier viajero. Su situación geográfica, en la zona centro peninsular, la convierte... (with a photo of the Royal Palace)
- RESTAURANTE**: Palacio de Cibeles (37 m). Description: Adolfo Muñoz comparte dirección culinaria con la casa madre en Toledo, secundado por su hijo Javier. Cocina saludable a base de vapor y horno en un notable... (with a photo of a restaurant interior)
- PLAZA**: Plaza de Cibeles (89 m). Description: En el centro de este emblemático espacio público madrileño se sitúa la célebre fuente de Cibeles, esculpida en el año 1782, a partir de un diseño de Ventura Rodríguez. Cada una de las... (with a photo of the Fuente de Cibeles)

To the right, there is a sidebar for a restaurant named "Meating" with a photo of its interior, a trash bin icon, and a "RESTAURANTE Meating" section. There is also a small icon at the bottom right.

Figura 2.7. Web de Guía Repsol

2.1.6. Planificador Way-Away

Similar al de la Guía Repsol. La diferencia reside en su forma de uso. Mientras que en Guía Repsol te montas tu propio plan, aquí indicas donde quieres ir y el número de días, y se te sugiere un plan completo diario, que puedes comprar a un precio económico.

The screenshot shows the Way-Away website interface. At the top, there's a green banner with the text "Sin perder tiempo. Sin equivocarte. Disfrutando más." and a "Iniciar sesión" button. Below the banner, there are three buttons: "Selecciona tu viaje", "Test de tu viaje ideal", and "Ver cómo funciona". A dropdown menu shows "Europa" selected under "Días" and "Amsterdam" under "3". A pink button says "Ver Ruta recomendada". The main content area features a photo of a person walking along a canal in Amsterdam. Below the photo, a section titled "Resumen ruta: Fin de semana Ámsterdam" includes two items: "Llegada a Ámsterdam: El Barrio Rojo" (with a photo of traditional Dutch houses) and "Los mercados de Ámsterdam, las 9 Calles y Jordaan" (with a photo of a market). To the right, a pink button says "Comprar Ámsterdam por sólo 4.95 €". Another section titled "Qué incluye el Planificador" lists: "Ruta de 3 días", "Guía diaria de visitas (qué ver en Ámsterdam)", "Cómo ir de un sitio a otro", "Restaurantes auténticos indicados en la ruta", "Hoteles recomendados según tu presupuesto", "Smart Links para reservar cada cosa al mejor precio", and "Plan de viaje con tu Ruta detallada para imprimir".

Figura 2.8. Web de Way-Away

Este es un ejemplo para una guía de 5 días en París, cuyo precio resulta ser de 5€:

The screenshot shows a section titled "Qué incluye el Planificador a París". It lists the following items: "Rutas de 3 a 6 días", "Guía diaria de visitas (qué ver en París)", "Cómo ir de un sitio a otro", "Restaurantes auténticos indicados en la ruta", "Hoteles recomendados según tu presupuesto", "Smart Links para reservar cada cosa al mejor precio", and "Plan de viaje con tu Ruta detallada para imprimir".

Figura 2.9. Plan de 5 días a París

Fuente: Web de Web-Away

2.2. Modelos de negocio

En esta sección se realizará un pequeño estudio sobre los modelos de negocio de los dos sistemas más próximos y cercanos a TripMinder, que son RouteRank y GoEuro. Para ello se investigará sobre sus fuentes de ingresos, acuerdos y vías de comunicación con sus partners.

2.2.1. RouteRank

El modelo de negocio de RouteRank consiste en ofrecer más funcionalidades y en no limitar el uso de su sistema, además de permitir su uso integrado con webs o apps.

El modo de pago es el “pago por uso”, de ese modo se paga por lo que se gasta. Es decir, se contabiliza según el tráfico de datos, el número de consultas a la API que tengan, o el uso de sus claves de API para la recurrencia y frecuencia de esas peticiones.

Para ello, disponen de tres planes:

- **Custom version:** esta versión permite a otras empresas utilizar el sistema de Route Rank para sus objetivos, ya sea para sus webs, apps o necesidades de trabajo. Además, estas empresas podrían añadir sus propias fuentes de datos y de reservas a los de Route Rank.
- **Standard Professional:** es la versión sin límites y con más funcionalidades de Route Rank
- **Logistics version:** versión específica para empresas de transportes. No es muy diferente a la Professional.

En la siguiente **tabla comparativa** podemos ver mejor estos datos:

Version	Custom developed	Standard professional	Public
Layout and usage			
Branding	any branding	routeRANK	routeRANK
Commercial use	yes	yes	no
Unlimited searches	yes	yes	no
Ads-free	yes [1]	yes	no
Account management	yes [1]	yes	no
Search parameters & criteria			
Search scope	custom	expanded	standard
Schedule and fare data	any available interface	standard	standard
Price, duration, CO ₂	yes [1]	yes	yes
Work time/productivity	yes [1]	yes	no
Risks assessment	yes	no	no
Additional locations ^[2]	yes	no	no
Features			
Results filters	all criteria [1]	all criteria	airport and time
Additional car types	taxi & rental [1]	taxi & rental	taxi
Car customization	yes [1]	yes	yes
Train customization (% etc.)	yes [1]	yes	no
Customized results ranking	yes [1]	yes	no
Multiple travelers	yes	yes	no

Tabla 2.1. Comparativa de planes de RouteRank

Fuente: Elaboración propia

2.2.2. GoEuro

En el caso de GoEuro, no se dispone de planes específicos para empresas. Sin embargo, se encargan de interconectar los servicios de las empresas de países en los que trabajan (renfe, iberia, expressbus...), o llamados partners.

De esta manera consiguen más clientes a sus partners de modo indirecto. Por esto, reciben beneficio de sus partners, los cuales son su fuente de ingresos. Este tipo de operativa se podría denominar una “**relación de beneficio mutuo**”.

GoEuro trabaja con los servicios [GeoNames](#), [OpenFlights](#), [OpenStreetMap](#), [RSP \(Atoc\)](#), y [TheMoneyConverter.com](#).

2.3. Comparativa entre sistemas

A continuación se realizará una misma prueba aplicada a varios sistemas para exponer una comparativa de resultados.

2.3.1. GoEuro & RouteRank

Se testean estos dos sistemas para el día 16-10-2014 con trayecto de *Alicante a París*. Estas son los resultados obtenidos:

- Los dos sistemas presentan rutas de horarios similares, aunque precios diferentes.
- GoEuro da más información sobre qué línea de buses coger (cuando se pulsa sobre esta opción).
- RouteRank permite ordenar además por CO₂, y horas de salida o llegada.

Main Mode	Bus	Air	Bus	Total Travel Time	Total
18:30 ALC → 20:35 ORY direct				16:33 - 21:30	04:57h Total € 65.18
18:30 ALC → 20:35 ORY direct		Air		16:33 - 21:30	04:57h Total € 110.56
20:30 ALC → 22:35 ORY direct		Air		18:33 - 23:30	04:57h Total € 116.84
20:30 ALC → 22:35 ORY direct		Air		18:33 - 23:30	04:57h Total € 116.84
12:45 ALC → 15:05 BVA direct		Air		10:48 - 16:50	06:02h Total € 58.11
17:15 ALC → 21:10 ORY 1 change		Air		15:18 - 22:05	06:47h Total € 120.34

Figura 2.10. Resultado de GoEuro para Alicante – París

De Alicante (Valencia), España

A París (Isla de Francia), Francia
ida y vuelta

Blog: Total Trip Planner and me
 Compartir

Fecha 16-10-2014 Hora 08:00

Reserva de Alicante (Valencia), España a París (Isla de Francia), Francia el 16-10-2014 a las 08:00

SAL	LLE	VIA	MEDIOS	TIEMPO	CO ₂	PRECIO
16:41	22:02	Alicante (ALC) → Paris (ORY)		5h21	249kg	CHF 90.-
16:41	22:07	Alicante (ALC) → Paris (ORY)		5h26	253kg	CHF 86.-
6:51	13:02	Alicante (ALC) → Paris (ORY)		6h11	272kg	CHF 264.-
15:50	22:02	Alicante (ALC) → Paris (ORY)		6h12	247kg	CHF 87.-
15:50	22:07	Alicante (ALC) → Paris (ORY)		6h17	250kg	CHF 83.-
6:51	13:17	Alicante (ALC) → Paris (ORY)		6h26	268kg	CHF 268.-
6:30	13:02	Alicante (ALC) → Paris (ORY)		6h32	270kg	CHF 262.-
16:46	23:22	Alicante (ALC) → Paris (ORY)		6h36	303kg	CHF 151.-

Figura 2.11. Resultado de RouteRanks para Alicante – París

2.4. Limitaciones en los sistemas

- **Google Maps**
 - Indica la compañía, pero no hay un enlace para reservar ese viaje directo. En GoEuro si existe.
 - Tiene funcionalidad de búsqueda de lugares, pero no permite crear guía y los filtros y la UI Web podría ser algo mejorable. La UI de la App móvil tiene gran calidad.
 - No liga países por tren, bus y tranvía. Sólo por coche o vuelos.
 - En la App móvil, no incluye vuelos, ni la funcionalidad de MyMaps.
- **GoEuro**
 - No guía al usuario por las rutas
 - No incluye lugares de interés turístico
 - No incluye ruta en coche propio
- **DB Navigator (BAHN)**
 - Presenta demasiada información. Se puede hacer difícil plantear "quiero ir de Madrid a París", ya que presenta todas las paradas de bus, tren y aeropuertos de Madrid y París, en lugar de mostrar únicamente los de la ciudad
 - Interfaz compleja y poco organizada
- **Guía Repsol o similares**
 - No incluye búsqueda de rutas, ni guían al usuario

2.5. Comparativa de tecnologías

El desarrollo de TripMinder, para que un uso multiplataforma requiere un entorno de programación (IDE) que permita desarrollar una aplicación generando un solo código. Además, se busca que funcione tanto sobre SO de sobremesa Windows y Mac OS, como sobre SO de dispositivos móviles tipo iOS, Android o Windows Phone .

A continuación, se realizará un estudio sobre las tecnologías mencionadas y posteriormente se decidirá cuál puede ser el mejor candidato.

2.5.1. RAD Studio, de Embarcadero Technologies



Figura 2.12. RAD Studio XE7

[RAD Studio](#), en su actual versión XE7, se le podría considerar como el líder de los IDE multiplataforma. Cabe de esperar, viniendo de una empresa, que desde su fundación en San Francisco (EEUU) se ha dedicado al desarrollo de herramientas para desarrolladores.

[Embarcadero Technologies](#) no sólo dispone de RAD Studio, también son especialistas en bases de datos y en arquitectura y modelado software, por lo que ofrecen muchos otros productos, como pueden ser [Interbase XE3](#), [DB Power Studio](#) o [ER/Studio Data Architect](#). Su último producto, [All-Access XE](#), combina las funcionalidades de todos los productos.

Por tanto, RAD Studio ofrece la confianza de una empresa con muchos años de dedicación, y un gran soporte actualizado que hemos podido experimentar de antemano.

Respecto al **estudio** de RAD Studio, podemos ver las ventajas y desventajas en la siguiente tabla comparativa:

Pros	Contras
Soporta iOS, Android, MacOS, Windows y “Gadgets & Wearables”	Precios elevados (sobre 1.000 € en 1 lenguaje y 1.900 € en ambos)
Variedad de herramientas adicionales	No soporta Windows Phone.
Permite elegir lenguaje, entre Delphi y C++	Poca flexibilidad y dificultad de programación.
Buena relación, colaboración y soporte por parte de la empresa.	Funcionamiento muy “buggy”, a pesar de estar en una versión ya madura.
Presentan gran madurez	Comunidad pequeña, falta de documentación para ciertas cosas.

Tabla 2.2. Pros y contras de RAD Studio XE7

2.5.2. Xamarin



Figura 2.13. Xamarin

[**Xamarin**](#) es la competencia directa de RAD Studio, ya que es el otro IDE que también soporta Mac, Windows, iOS y Android. Además, Xamarin también soporta Windows Phone. La ventaja que presenta respecto al anterior es su integración con Visual Studio además de poseer su IDE propio, y sobre todo, el lenguaje que utiliza es .NET.

Aunque Xamarin es una empresa que cuenta con 170 empleados y 15.000 clientes, actúa más en forma de comunidad que de modelo de empresa. Se puede apreciar por los servicios que poseen y la forma en la que distribuyen su documentación, por lo que puede apoyar a una documentación mejorada por los mismos usuarios de la comunidad.

Ofrecen [Xamarin University](#), que aunque no es nada barato (1550 € / año), se le puede sacar provecho, ya que tienen clases virtuales privadas, comunidades, tutorías y muchos recursos con los que aprender a usar Xamarin.

En la siguiente tabla se realiza un pequeño estudio resumido de los pros y contras más característicos de este IDE:

Pros	Contras
Soporta iOS, Android, Windows Phone, MacOS y Windows. También Gadgets y Wearables	Comparte código al 75%, no al 100%
Utiliza lenguaje C# y los convenios y estilos de programación de .NET de Visual Studio	Xamarin es menos maduro que RAD Studio
Puedes trabajar en Visual Studio o Xamarin Studio	
Los precios son inferiores a los de RAD Studio (620 €, solo pagas el primer año (sin soporte))	
Promueven eventos y tienen academia virtual	
Recientemente, han publicado una versión gratuita para estudiantes	

Tabla 2.3. Pros y contras de Xamarin

2.5.3. Apps híbridas

Los siguientes IDE's de desarrollo generan aplicaciones multiplataforma para sistemas operativos móviles.

El término de Apps híbrida refiere a que realmente no genera aplicaciones totalmente nativas, sino que la interfaz la dibuja sobre un componente web (WebView). Sin embargo, la funcionalidad de acceso a móvil es nativa.

2.5.3.1. Apache Cordova (también conocido como Phonegap)



Figura 2.14. Apache Cordova

- Totalmente gratuito, tanto la plataforma como todas las extensiones.
- Soporta una gran cantidad de S.O.'s móviles.
- Posee una enorme comunidad y extensibilidad, con cientos de plugins que permiten amplificar sus capacidades.
- Utiliza HTML y CSS para la interfaz + Javascript para la lógica, por lo que separa interfaz de lógica.
- Se adecua a los nuevos modelos de programación.
- Para la creación de la App, es totalmente gratuito.

2.5.3.2. Titanium Studio, de Appcelerator Inc.



Figura 2.15. Titanium Studio

- Soporta iOS, Android, Windows Phone, BlackBerry y WebOS.
- Utiliza Javascript para todo.
- Gratuito para uso personal. Para uso comercial se debe usar Appcelerator Platform, que no es gratuito.

2.5.4. Ionic Framework



Figura 2.16. Ionic Framework

Plataforma para el desarrollo de aplicaciones híbridas. No es una más como Apache Cordova, sino que es una combinación de este con las mejores tecnologías para el desarrollo frontend que lo convierten en un completo entorno de desarrollo.

Ionic Framework se compone de:

- **Apache Cordova:** explicado anteriormente. Además, muchos de los plugins de Cordova han sido adaptados a Ionic.
- **AngularJS:** posiblemente el más completo framework frontend, proveniente de la mano de Google. Provee de el patrón MVVM, binding, services, controllers, componentes para API's rest, vistas, filtros, y una infinidad de características que se resumen en una perfecta y flexible arquitectura de una aplicación.
- **SASS:** preprocesador de CSS. Básicamente extiende css con variables y funciones. Por defecto, Ionic integra SASS, aunque se puede utilizar LESS (bajo configuración propia), o simple CSS.
- **Gulp.js:** junto con Grunt.js, son sistemas de construcción y ejecución de tareas. Gulp es posterior a Grunt, de sintaxis algo simplificada. Permite ejecutar tareas del tipo “Minimiza, comprime y concatena todos los .css y .js, añádeles prefijos, reemplaza X cadenas, cópialos a X carpeta y súbelos a Github”.
- **Framework:** Ionic añade mediante su palabra “Framework” un conjunto de librerías, estilos y componentes desarrollados en SASS y AngularJS listos para usar.
- **Ionic-cli:** interfaz de consola que permite realizar diversas tareas, como crear una aplicación, instalar un plugin, lanzar la app, etc.

Es de destacar que, además de Ionic Framework, el equipo de Ionic ha desarrollado otras novedosas herramientas que se integran con Ionic a la perfección. A día de hoy, estas son:

- **Ionicons** (<http://ionicons.com/>): librería de icon-font.
- **ng-cordova**: adaptación de los plugins de Apache Cordova para AngularJS y Ionic.
- **Ionic.io** (<https://apps.ionic.io/>): conjunto de servicios para Ionic, que incluye un creador de interfaz “drag&drop”, publicación en las stores de Android y Apple, sincronización en dispositivos con Ionic View, y más en desarrollo.

Existen otras competencias de plataformas “todo en uno” como Ionic Framework, que se nombran a continuación mencionando la razón por las que no se han planteado como candidatos:

- **Angular Mobile UI**: se encuentra en una fase de desarrollo aún temprana y su comunidad es menos numerosa.
- **Onsen UI**: muy similar y más maduro que Ionic. Sin embargo, no proporciona las mismas funcionalidades. Tampoco servicios de AngularJS delegados, SASS, ni plugins de terceros como lo hace Ionic.
- **Famo.us + AngularJS**: versión de Famo.us combinada con AngularJS. Muy prometedora, ya que usa su propio sistema de render, capaz de realizar animaciones a 60 fps. Sin embargo, se encuentra en una fase muy temprana y no presenta las mismas funcionalidades y extensibilidad que Ionic.

2.5.5. Conclusión y tecnología seleccionada

La tecnología seleccionada ha sido **Ionic**. A continuación se explican las razones.

En base al estudio realizado, las tecnologías que más se adecuan son RAD Studio, Xamarin y Ionic Framework.

Primero se escogió **Rad Studio**, con el que el alumno dedicó un tiempo de programación de 30 horas, en las que unas 22 fueron de resolución de problemas de la tecnología. Durante ellas se encontraron muchos contratiempos y dificultades, por parte del lenguaje pero sobretodo por el propio Rad Studio.

Los mayores problemas fueron:

- Poca flexibilidad de programación. Un ejemplo fue el lanzamiento de una petición a Google Autocomplete al evento de actualización del campo de entrada, con cierto retraso y de manera asíncrona.
- Mala optimización de las apps generadas (se detallará a continuación).
- Número de librerías externas para C++ casi nulo. Para Delphi si que existen algunas.
- Modelo de programación muy tradicional y lento. En otras tecnologías se podría haber conseguido más en menos tiempo
- El componente de TWebBrowser no da soporte a Windows y Mac Os, por tanto no se pueden insertar mapas multiplataforma. Existe un modo de hacerlo para Delphi, que consiste en integrar una versión adaptada a Delphi de Chromium Embedded Framework (CEF), lo cual supone integrar el motor que usa Chromium en la aplicación, con 60MB de peso. Esto fue decisivo.

Entonces, y todavía a tiempo, se decidió migrar a **Ionic Framework**. Los resultados han sido mejores de lo esperado, todo gracias a las numerosas ventajas que presenta esta tecnología.

El primer resultado notable reside en la **rapidez de desarrollo**, las mismas tareas realizadas con Rad Studio en 30 horas, se han realizado en Ionic Framework en solo 3 horas.

La facilidad y soporte de crear **interfaces modernas** que proporciona Ionic ha sido otro punto importante. Además de todos los servicios y extras que posee, ya mencionados.

Otro punto importante, el **rendimiento** de la app generada. No ha sido necesario medir tiempos, ya que a simple vista se nota la mayor fluidez de la aplicación y la velocidad de ejecución de las animaciones y gestos.

Y por último, el **peso y optimización** de la app. Mientras que la supuesta aplicación nativa de Rad Studio pesa sobre 14MB (de hecho, una app en blanco ya ocupa 6MB), la versión equivalente realizada con Ionic Framework pesa sobre 4MB.

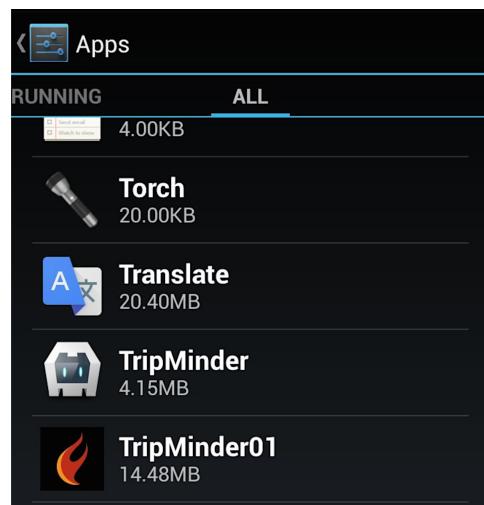


Figura 2.17. Comparación entre las apps generadas por Ionic Framework (TripMinder, parte superior) y por Rad Studio (TripMinder01, parte inferior)

Por tanto, las ventajas que suponía generar una aplicación nativa, se han convertido finalmente en una desventaja.

3. Objetivos

Han sido muchas las ideas y objetivos que han ido surgiendo en el planteamiento del proyecto. Algunas de ellas se han tenido que descartar por limitaciones temporales en un proyecto ya de por sí bastante ambicioso. Por ello, se han tenido que acotar los objetivos a los que describimos a continuación.

Objetivos personales:

- Crear una aplicación multiplataforma que utilice y *exprima* lo máximo posible las capacidades del dispositivo.
- Enfrentarse a la problemática de la recolección de datos y de su organización, uso y distribución.
- Aplicar metodologías de desarrollo ágil, realizando reuniones semanales y llevando al día un Kanban de las tareas.
- Poner a prueba la capacidad de improvisación en lo referente a los riesgos de las tecnologías que se utilizan.
- Adquirir experiencia en tecnologías desconocidas.
- Seguir una planificación y poder cumplir en la medida de lo posible los objetivos marcados.

Objetivos principales de la aplicación:

- Diseñar una UI (interfaz de usuario) de diseño moderno y usable.
- Diseñar una interacción usuario-dispositivo que proporcione una gran experiencia de usuario.
- Recolectar y tratar información referente al tráfico y al tránsito proveniente de medios globales.
- Calcular la mejor ruta de cada medio de transporte.
- Proporcionar capacidad de creación de guías turísticas, dadas unas preferencias.

- Permitir la configuración principal del perfil, donde residen las preferencias de usuario.

Objetivos **opcionales de la aplicación:**

- Incluir medios locales para la recolección de la información.
- Realizar seguimiento, tanto para las rutas de transporte como para las guías turísticas creadas.

4. Estudio de las fuentes de datos

La principal dificultad para el desarrollo de TripMinder reside en la recolección y el tratamiento de datos, que se accederán mediante el uso de API's. Los datos requeridos pueden ser muy variados, de diferentes categorías y estructuras. Principalmente tendremos los que se refieren al "routing" y la geolocalización, y los de información turística.

ProgramableWeb (<http://www.programmableweb.com/>) es un gran recurso para la búsqueda de API's. Incluye muchísimas (a fecha de hoy, 09/11/2014, posee 12323) y se encuentran organizadas en categorías. En este sitio se han encontrado varias de las API's que se mencionan posteriormente.

En cuanto a los medios de los que se va a extraer la información, también son muy diversos. Cada uno presenta la información con unos atributos diferentes, por lo que será necesario desarrollar un lector de medios por cada uno de los medios utilizados. Para detallar estos medios, se procede a describir qué medios se van a utilizar para cada funcionalidad de las previstas. Es de cierta relevancia mencionar que una misma API puede servir para diferentes propósitos y funcionalidades, como se va a ver a continuación.

4.1. API's de geolocalización

Las API's de geolocalización serán las encargadas de proporcionar información sobre todo lo referente a lugares y puntos de localización.

4.1.1. Google Places API

Google Places es utilizada en las aplicaciones basadas en la ubicación. Numerosas aplicaciones utilizan esta API, entre ellas WhatsApp, RoadTrippers o Cycling The Alps.

Permite llevar a cabo ciertas acciones como las que pueden ser:

- Autocompletado de palabras según términos de búsqueda basados en texto. No sólo sugiere la palabra adecuada, sino que proporciona toda la información detallada sobre las sugerencias presentadas, como el nombre de la ciudad, su población, coordenadas, código internacional, código postal, etc.
- Búsquedas de sitios de interés, los cuáles también se pueden filtrar por categorías. Esto es de especial interés turístico, como se comentará más adelante.
- Información sobre sitios geográficos en general, dadas unas coordenadas o un lugar.

4.1.2. API de codificación geográfica de Google

Esta API tiene una función determinada, que es la de realizar la denominada *Codificación geográfica* y la *Codificación geográfica inversa*. La primera es descrita como el proceso de transformar direcciones (como "1600 Amphitheatre Parkway, Mountain View, CA") en coordenadas geográficas (como 67.423021 de latitud y -104.083739 de longitud). Como es de esperar, el proceso inverso es la Codificación geográfica inversa.

Como se verá, pertenece al conjunto de API's de Google Maps.

4.2. API's de tráfico y tránsito

La principal funcionalidad que ofrecen este tipo de API's es la de proporcionar información relativa a los medios de transporte y las rutas en sí. Mediante su uso, se puede descubrir las diferentes rutas que se pueden tomar para llegar a ciertos sitios, mediante qué medio de transporte, a cuanta distancia, etc.

Las API's más completas en este ámbito son las que contiene Google Maps, pero antes de la explicación de éstas, es necesario saber qué es Google Transit y el formato GTFS.

4.2.1. GTFS

GTFS (*General Transit Feed Specification*) define un formato de intercambio de datos común para información de transporte público, como los horarios o información geográfica asociada. Estos “feeds” permite que las empresas de transporte público publiquen sus datos de transporte.

La web de GTFS Data Exchange (<http://www.gtfs-data-exchange.com/>) (mencionado desde la propia web de Google) recoge ficheros GTFS y utiliza su API para el acceso a ellos, al igual que Google también recoge estos datos, que se pueden proporcionar de la manera que se explica en la web de Google:

<http://maps.google.com/help/maps/mapcontent/transit/participate.html>

Este formato está compuesto por un fichero .zip, que debe incluir una serie de ficheros de texto con una información en específico, y por supuesto, un formato específico. Varios de estos ficheros son opcionales. Se puede encontrar más información en el sitio web:

<https://developers.google.com/transit/gtfs/>

GTFS-Realtime es una extensión de GTFS que proporciona información a tiempo real y añade varias opciones como cambios en rutas, posición del vehículo, etc.

4.2.2. Google Transit

Google Transit, parte de la suite de Google Maps, es una herramienta de planeamiento de uso del transporte público que utiliza los feeds GTFS combinados con otras API's de Google Maps. Proporciona información del transporte público relacionada con tarifas, programas, rutas y paradas.

La herramienta de Google Transit, se puede probar tanto desde la misma web o aplicación de Google Maps, como por separado en su sitio web:

<http://maps.google.es/intl/en/landing/transit/>

4.2.3. Google Maps

Google Maps no es sólo una web o una aplicación móvil, en la que podemos ver mapas y navegar por el mundo. Google Maps es una completa suite de herramientas y API's combinadas entre ellas para proporcionar una gran cantidad de funcionalidades relacionadas, concluyendo en una gran experiencia de usuario. No es el propósito de este documento comentar todo lo que puede hacer, ya que se puede encontrar en la web:

<https://developers.google.com/maps>

Google Maps, como se ha comentado, está formado por varias herramientas y API's. Éstas son:

- **Street View:** a partir de imágenes, crea un entorno por el que se puede navegar por las carreteras y realizar rutas virtuales completas.
- **API de codificación geográfica:** es la que se ha mencionado en el apartado [4.1. API's de geolocalización](#).
- **API de rutas:** también conocida como Google Directions. Proporciona información sobre rutas dados 2 o más puntos. Proporciona una gran variedad de parámetros de personalización para las consultas, pudiendo filtrar por

medio de transporte y muchos otros parámetros. Ésta será la API clave para la aplicación.

- **API de matriz de distancia:** como su nombre indica, su utilidad es la de dar información sobre distancia entre diferentes orígenes y destinos. También proporciona otra información como el tiempo de recorrido según el medio de transporte.
- **API de elevación:** proporciona información de la elevación de cualquier punto del mundo. Es de gran utilidad para el desarrollo de aplicaciones sobre senderismo y ciclismo, aunque para nuestro caso no tiene mucha relevancia.
- **API de Google Places:** se ha descrito anteriormente en el apartado [4.1. API's de geolocalización](#).

Todas las API's comentadas permiten su uso en base a servicio web REST, que será la forma con la que se utilizará en esta aplicación. El uso de estas es totalmente gratuito para aplicaciones gratuitas, con las siguientes limitaciones de uso:

Funciones	API de Google Maps
Street View	Lo contiene
Servicio web de codificación geográfica	2.500 solicitudes diarias
Servicio web de rutas	2.500 solicitudes diarias con 10 hitos por solicitud
Servicio web de matriz de distancia	100 elementos por consulta 100 elementos cada 10 segundos 2.500 elementos diarios
Servicio web de elevación	2.500 solicitudes diarias con 25.000 muestras diarias
Resolución del API de Google Static Maps	640 x 640
Escala máxima de Google Static Maps	2x

Tabla 4.1. Tabla de limitaciones de Google Maps

4.2.4. QPX Express API

Entrando en el ámbito de los vuelos con avión, una vez más, aparece Google en acción. QPX Express API pertenece a Google y es la que internamente utiliza el sistema de Google Flights.

Su funcionamiento es similar al de las otras API's de Google, aunque esta vez relativa a las aerolíneas. Proporciona prácticamente toda la información que se puede esperar para vuelos, como la disponibilidad, precio, horarios, opciones de vuelo, etc.

En cuanto a las limitaciones de uso, QPX Express API está bastante limitado, aunque el precio no es muy alto. Los límites de uso permiten 50 consultas gratuitas diarias, y a partir de ahí a 0.03 € la petición.

4.2.5. Flight Aware API

Dejando de lado las API's de Google, Flight Aware API ofrece información en tiempo real sobre estados de los vuelos, tal como tiempo restante en llegar, localización actual, retrasos en vuelos, etc.

El inconveniente de esta API es su precio. En la siguiente tabla podemos ver los precios que ofrecen:

Consultas al mes	Costes por consulta			
	Clase 1	Clase 2	Clase 3	Clase 4
1 - 9.999	\$0.0120	\$0.0079	\$0.0020	\$0.0008
10.000 - 24.999	\$0.0070	\$0.0046	\$0.0012	\$0.0005
25.000 - 49.999	\$0.0060	\$0.0040	\$0.0010	\$0.0004
50.000 - 99.999	\$0.0050	\$0.0033	\$0.0008	\$0.0003
100.000 - 249.999	\$0.0040	\$0.0026	\$0.0007	\$0.0003
250.000 - 999.999	\$0.0030	\$0.0020	\$0.0005	\$0.0002

Tabla 4.2. Tabla de precios de Flight Aware API

4.2.6. FlightStats Flex API

Muy similar a la anterior, FlightStats Flex API proporciona información sobre el estado de vuelos. Ciertamente, los datos que proporciona son prácticamente los mismos que Flight Aware API.

Aunque también es de pago, el modelo de negocio es diferente: FlightStats Flex API ofrece una cuenta de evaluación, y aunque no especifican el precio, al acabarse se debe abonar para usar el servicio.

4.2.7. Taxi Fare Finder API

Entrando en el ámbito de los taxis, Taxi Fare Finder API ofrece información sobre las tarifas de los taxis en diferentes ciudades. Aunque en los inicios se creó para funcionar en EEUU, actualmente opera en ciudades de diferentes países (España incluido). Aunque está en fase Beta, es totalmente funcional y utilizable.

En este momento, Taxi Fare Finder API es gratuito. Sin embargo, no podemos asegurar que continúe siéndolo después de la fase Beta.

4.2.8. InKnowledge Taxi Fare Calculator REST Service

InKnowledge, empresa ubicada en el Reino Unido, posee una API REST totalmente equivalente a la anteriormente mencionada. Opera en ciudades de 150 países diferentes y, a diferencia de la anterior, no está en fase beta, por lo que se sabe que posee un nivel de madurez más alto.

Las condiciones de uso de esta API no están claros, ya que no mencionan que sea gratuita ni de pago, y para conseguir una clave del API se ha de contactar por email.

4.2.9. Otras API's

Durante el estudio y análisis de API's para el tránsito y el transporte público, se han encontrado diversas opciones para este propósito, pero ninguna era tan completa, aunque se mencionan ya que podrían ser de utilidad para otros casos.

Para el caso de los autobuses, se han encontrado algunas soluciones como [OneBusAway](#), pero están limitadas a un país. Varios países tienen su propia API de transportes en tren y bus, pero no hay ninguna que englobe varios países. En cuanto a los aviones, no se han encontrado mejores productos que los ya mencionados.

En el caso de API's para viajes en barco, no se ha encontrado tampoco ningún producto disponible, aunque hay algunas como [Fleetmon API's](#) y [Vessel Tracker Data](#), pero orientadas a buques de carga, no a viajes en barcos turísticos.

4.3. API's de información turística

Las características esperadas en estas API's incluyen la disponibilidad de información acerca de los intereses turísticos más demandados que se pueden encontrar en una ciudad, o durante un trayecto. Estos intereses turísticos pueden pertenecer a diferentes categorías, como la de restaurantes, hoteles o museos, y cada sitio tiene su horario. Estos son los aspectos esperados en un API de este tipo.

Para la descripción de estas API's, se presentarán las mejores que se han encontrado para este sector y se presentará una tabla comparativa entre ellas.

4.3.1. Google Places

Vuelve a entrar en juego Google Places, que además también posee la funcionalidad que estamos buscando en este tipo de API's.

Permite buscar sitios por categorías, próximos a ciertas coordenadas o indicando cierto rango de lejanía. Esta es la llamada funcionalidad “*Búsqueda de sitio*”. También ofrece la funcionalidad “*Detalle de sitio*”, donde podemos encontrar toda la información referente a ese lugar.

4.3.2. Yelp API

Sitio web, inicialmente ubicado y restringido a EEUU, que se expandió y actualmente posee muchos datos relativos a lugares turísticos, tales como hoteles, tiendas, restaurantes, peluquerías, etc. Es un proyecto maduro, quizás es el que más información posee del sector.

La API de Yelp permite únicamente la adquisición de datos vía Json y se divide en dos sub-API's, lo que hace que el manejo de esta sea algo más complicada de lo que podría parecer.

4.3.3. Minube API

Minube es una aplicación web que ayuda a planificar viajes y recomienda lugares para visitar, comer y hoteles dónde dormir. Además de esto, ofrecen una API en versión beta pero totalmente funcional con acceso a los numerosos lugares de los que dispone Minube.

Está orientado a crear guías turísticas, y presenta una interfaz de acceso sencilla y funcional, con la información que se busca para el caso, incluyendo opiniones de usuarios.

4.3.4. Tixik API

Tixik realmente es un sistema algo más pequeño, nacido en la República Checa, cuyas funcionalidades intentan asemejarse a las de Minube y Yelp.

No ofrece muchas categorías de búsqueda de lugares.

4.3.5. Comparativa de API's

Aquí se muestra una tabla comparativa de las API's mencionadas:

	Minube	Google Places	Yelp	Tixik
Autenticación	Key	Key	OAuth v1.0a	Key
Límite	2.000 / mes (para desarrollo). Luego te lo amplian	1.000 / día 100.000 / día (si verificas) <u>Requerimientos.</u> Se limitan en poner en tu app una sección de "condiciones de uso" y otra de "política de privacidad"	25.000 / día <u>Requerimientos</u>	Desconocido
Llamada	Rest	Rest	Rest	Rest
Datos	Json, XML	Json, XML	Json	XML
Lugares	Muchos tipos	<u>Muchos tipos</u>	<u>Muchos tipos</u>	
Nivel configuración	Muy alto	Muy alto	Muy alto	Muy bajo
Filtros	Muchos filtros	Muchos filtros	Muchos filtros	Límite resultados, idioma, clave
Autenticación	Key	Key	OAuth v1.0a	Key
Límite	2.000 / mes (para desarrollo). Luego te lo amplian	1.000 / día 100.000 / día (si verificas) <u>Requerimientos.</u> Se limitan en poner en tu app una sección de "condiciones de uso" y otra de "política de privacidad"	25.000 / día <u>Requerimientos</u>	Desconocido

Tabla 4.3. Comparativa de API's de información turística

Como conclusión, parece que la mejor opción es la de **Google Places**, por sus 4 API's en 1 y la NO necesidad de añadir branding. A continuación, estaría **Minube** y **Yelp**, que aunque no están mal. Tienen branding, además de muy pocas solicitudes en Minube y cierta complejidad y requisitos de uso en Yelp, aunque no se descarta su uso.

4.4. API's que utiliza TripMinder

Hasta ahora se ha hecho un análisis de API's categorizado y se han estudiado los casos, comparándolos entre ellos y así determinando las características de cada uno.

Sin embargo, no se han seleccionado las que se van a utilizar definitivamente, y para qué funcionalidades. Para ello, se ha creado una tabla donde por cada *requisito funcional* (vienen descritos más adelante) se definen las API's que se tiene planificado utilizar para cada caso, tal como se ve a continuación:

	Nombre	API's
	Presentar lista de ciudades	Google Places
	Implementar Geocodificación [Inversa]	Google Geocoding
	Obtener rutas en coche, tren, metro y bus	Google Directions
	Calcular distancia y tiempo de ruta	Google Distance Matrix
	Obtener vuelos	GPX Express
	Calcular tarifas de taxis en un área	Taxi Fare Finder InKnowledge Taxi Fare
	Obtener lugares de interés turístico	Google Places Minube

Tabla 4.4. Relación entre API's y funcionalidad

5. Metodología

En esta sección se describe el proceso, las herramientas y las tecnologías que se utilizarán para el desarrollo de este TFG. Se detallarán sus usos específicos y se agruparán según el objetivo para el que se utilizarán.

Para el desarrollo de este TFG se realizará el mismo proceso que se utiliza para la creación de un producto en Ingeniería del Software. Se define el proceso de creación de un software como una disciplina que ofrece métodos y técnicas para desarrollar, mantener y documentar software de calidad. Para ello se integran herramientas, métodos y procesos que forman la base para la gestión de proyectos, por lo tanto se define un marco de trabajo.

En las metodologías de desarrollo de software, el proceso de la creación del producto se divide en diferentes fases y subprocessos, que son los siguientes:

- **Análisis:** es la primera fase o fase inicial. En ella, básicamente se realiza un estudio de mercado para analizar la actualidad del sector, y en base a ello realizar investigaciones y estudios para determinar la orientación final del producto. Además se realiza la especificación de requisitos.
- **Diseño:** posteriormente al análisis, comienza esta fase. El objetivo principal es la realización de un diseño conceptual en el que se estructuran de forma gráfica y conceptual los requisitos establecidos.
- **Implementación:** en base a los documentos e información generada en la fase de diseño, en esta fase se desarrollan los contenidos de los mapas y documentos conceptuales para crear el producto en sí. Se incluye también la implementación del diseño gráfico y de UI.
- **Despliegue:** fase final donde se realiza todo el proceso de pruebas y puesta en marcha. Se verifican los módulos y el sistema en conjunto para posteriormente aprobar el producto y ponerlo en producción. Se determinará si es necesario realizar otra iteración del proceso.

Estas son las fases principales que se siguen en todo proceso de metodología de desarrollo de software. Sin embargo, existen diferentes metodologías que aplican estos conceptos de manera diferente. Algunas siguen este proceso de una manera más estricta, sin interceptar e ignorando posibles cambios en los requisitos. Otras, comúnmente menos tradicionales que las anteriores, tienen unas características más dinámicas y se adecuan a los requisitos reales actuales, permitiendo la readaptación en los cambios de requisitos.

A continuación, con el objetivo de que el lector adquiera los conceptos básicos de las diferentes metodologías de software, se explicarán de manera muy resumida las más conocidas.

5.1. Metodologías software

Para describir las diferentes metodologías software, se hará de manera cronológica, así podrá observarse la evolución que han experimentado durante el paso del tiempo.

Las primeras en aparecer fueron las **metodologías secuenciales**, que definen el proceso de software como un seguimiento estricto y sin vuelta atrás de las fases descritas anteriormente.

Las características de estas metodologías son:

- Control del proceso sencillo.
- Enfoque sistemático y secuencial.
- Las fases vienen claramente definidas en la especificación y el desarrollo.
- Filosofía poco realista. En la realidad no sucede así, suele haber cambios de requisitos durante el desarrollo, se necesita capacidad de adaptación a cambios.

Se pueden diferenciar dos modelos principales, que son el modelo *primitivo* y el modelo *en cascada*. El primero, ni siquiera llega a seguir las fases descritas, sino que se basa en realizar iteraciones de “*codificación - depuración*”. Realmente este modelo sólo se sigue en proyectos pequeños por desarrolladores poco experimentados, que al final acaba produciendo el denominado *código espagueti*. El modelo en cascada sí que sigue las fases descritas, de tal manera que la siguiente no comienza hasta que no acaba la actual. Tiene en cuenta ciclos de retroalimentación, aunque no se suelen utilizar muy a menudo.

A partir de la evolución de esta metodología, surgieron las **metodologías evolutivas**. Aunque están influenciadas por las metodologías secuenciales tradicionales, las metodologías evolutivas comienzan a tener en cuenta las deficiencias que presentan las anteriores. Las características principales son:

- No definen un proceso tan secuencial y sistemático. Existen incrementos e iteraciones, que son repeticiones en el ciclo de vida de las fases.
- El tiempo inicial para presentar un sistema funcional al cliente es más corto.
- El control del proceso se hace más difícil.

Esto es común a todas las metodologías evolutivas. No obstante, existen diferentes modelos, cada uno con unas características y usos específicos:

- **Modelo basado en prototipos:** un prototipo es un modelo experimental de un sistema o subsistema que se puede presentar de manera funcional pero no optimizada. Por consiguiente, este modelo se debe considerar un medio para desarrollar posteriormente el producto final, ya que no tiene como resultado un producto de calidad. Sin embargo, es de gran utilidad para presentar resultados rápidos al cliente, y a partir de la retroalimentación que proporciona, hacer los cambios oportunos en la especificación de requisitos para ir refinando el sistema.

- **Modelo en espiral:** este modelo combina el modelo basado en prototipos con el modelo en cascada, de forma que así consigue tener como resultado un producto acabado y de calidad. En cada ciclo se definen los objetivos, se evalúan y reducen los riesgos, se desarrolla la parte y se revisa y decide si se continúa con el siguiente ciclo. Su finalidad, por consiguiente, es la de un modelo orientado a riesgos, su objetivo es prevenir lo que pueda ir mal en el desarrollo y saber resolverlo.
- **Modelo incremental:** define un ciclo de vida dividido en incrementos. En cada uno de ellos se aplican las fases mencionadas sobre el proceso del desarrollo de software. En este caso, se necesita algo de experiencia para saber definir bien esos incrementos. La principal característica es la subdivisión del sistema en un núcleo de sistema, al que se le van incorporando módulos, definidos por incrementos.
- **Modelo iterativo:** modelo similar al anterior, en el que se definen iteraciones. A diferencia del anterior, este presenta un sistema completo desde el principio, al que luego se le modifican las partes o subsistemas para hacerlo funcional. Esta característica hace que desde un principio se pueda apreciar la arquitectura final del sistema, y por tanto definirla desde un comienzo.

Estas metodologías, sin embargo, en la actualidad se utilizan para proyectos muy grandes, ya que requieren de mucho tiempo dedicado a la documentación y a la gestión del proceso.

Para contrarrestar el inconveniente anterior, aparecen las **metodologías ágiles**, que para proyectos de tamaño pequeño y mediano solventan los inconvenientes que presentan todas las anteriores. Para ello, combinan características de las anteriores adaptándolas a un entorno en el que se permite el cambio continuo de requisitos.

Las principales características que las definen son:

- Alto grado de adaptación a diversidad de proyectos, que además cambian rápidamente de requisitos durante el desarrollo.
- La mayor prioridad es satisfacer al cliente mediante la entrega temprana de software funcionando. Para ello se trabaja conjuntamente con él.
- Los responsables de negocio y el equipo trabajan conjuntamente, motivándose unos a otros y utilizando la conversación cara a cara. Se les denomina equipos auto-organizados.
- Se promueve el desarrollo sostenible, permitiendo mantener a los promotores y desarrolladores a un ritmo constante.

Como anteriormente, existen varios modelos de metodologías ágiles. Entre ellos los más conocidos son los modelos **XP** y **Scrum**. No se va a explicar en detalle las características de estos, ya que no tiene cabida en este documento. Solo mencionar que siguen las directrices mencionadas en común.

Las metodologías ágiles fomentan el uso de herramientas que faciliten la especificación del desarrollo de software y la gestión del proyecto y las tareas relacionadas. A partir de ahí nacen herramientas de control de proyectos ágiles, entre ellos **Kanban**, u otras más completas como **Redmine** o **Microsoft Project**.

Kanban es una herramienta específica para el control de tareas, que utiliza una pizarra con listas To-Do, donde se sitúan cartas de tareas. Redmine o Microsoft Project, en cambio, son herramientas completas que cubren todo el proceso de análisis y documentación, y tienen funcionalidades como las de estimar costes, mostrar gráficas, estimación temporal, etc.

5.2. Metodología en TripMinder

Se han descrito las principales metodologías en el desarrollo de software y se han mencionado algunas herramientas para la aplicación de las metodologías ágiles. La pregunta a plantear en este punto es, ¿Qué metodología se debe utilizar para desarrollar TripMinder? Bien, si se plantea las características del proyecto, tenemos las siguientes:

- Proyecto con base tecnológica a realizar individualmente, coordinado por dos tutores.
- A presentar al “cliente” en un plazo de 6/7 meses, además se han de realizar presentaciones iterativas a los clientes con el fin de refinar el sistema. En este caso los clientes se consideran los tutores del TFG.
- Trabajarán diferentes roles en conjunto, realizando reuniones semanales además de reuniones adicionales cuando se requiera.
- Se ha de mantener una documentación actualizada, además de un manual de usuario final. Esa documentación se considera este documento.
- El tiempo a invertir en documentación debe ser el mínimo posible, ya que no hay mucho tiempo para el desarrollo del sistema, y se debe tener en cuenta los posibles riesgos.

Con estas características, no es necesario indagar mucho entre las diferentes metodologías de software para saber que a la que más se asimila es al modelo de **metodología ágil Scrum**, ya que este define reuniones muy a menudo, mucha comunicación entre equipo y cliente.

Además, las metodologías ágiles, inclusive Scrum, definen un desarrollo guiado por iteraciones, que en Scrum se denominan “*sprints*”. En este proyecto se seguirán esos sprints para aplicar las fases que se han descrito anteriormente, de tal modo que por ejemplo, para desarrollar la arquitectura del sistema, el alumno la realiza, los tutores

la revisan y se comenta en las reuniones en las que se puede definir si realizar otra iteración para modificar la arquitectura del sistema.

Dicho esto, se van a comentar las herramientas que se utilizarán para cada parte del desarrollo de TripMinder, aunque algunas están presentes en varias de ellas.

5.2.1. Planificación y gestión del proyecto

En esta fase es donde más presente está la documentación y las gráficas relacionadas. Se utiliza para este fin:

- **Microsoft Project 2013:** herramienta que forma parte de la suite de Office 2013 de Microsoft. Provee de muchas de las funcionalidades para la gestión de un proyecto. En este caso se utilizará para realizar la estimación en la duración de las tareas, la planificación y la generación de diagramas, como el de Gantt. También cubrirá el campo de la estimación de costes y viabilidad del proyecto.
- **Trello:** es un creador de pizarras kanban y listas de tareas. Se utilizará para incluir y poder visualizar de forma gráfica y a simple vista las tareas. Para ello se creará una pizarra con las listas del *Backlog*, *To do*, *Doing* y *Done*.

5.2.2. Documentación

Como documentación, principalmente, se tiene esta memoria de TFG. Existen además dos documentos adicionales que se incluyen como anexos en esta memoria, que son una bitácora de desarrollo y un manual de usuario. Para la creación de los tres se ha utilizado:

- **Google Docs:** ha sido el procesador de textos principal. Se ha escogido por su integración con todos los dispositivos, pero sobretodo, por lo fácil que es la colaboración en la edición y visualización del documento, que se puede hacer

online en tiempo real. Además se integra perfectamente en Google Drive, ya que es de google.

- **Microsoft Word 2013:** para dar los últimos retoques y añadir elementos que no se pueden realizar, o al menos de manera óptima, en Google Docs, se ha utilizado este procesador de textos de Microsoft. Entre esos elementos está la tabla de contenidos, los pies de imagen, etc.
- **Google Drive:** se ha utilizado como sistema de almacenamiento personal donde el alumno guarda todos los documentos, imágenes, kanban y recursos con los que el alumno trabaja.
- **Dropbox:** Aunque ya se ha utilizado Google Drive, también se utiliza Dropbox para almacenar otros documentos, pero esta vez no se usa como medio personal, sino como medio compartido donde se alojan los recursos compartidos con los tutores.

5.2.3. Diseño

En la fase de diseño se abarcará tanto la parte del diseño visual, como puede ser la creación de los bocetos o mockups, como el diseño conceptual del sistema, entre los que se encuentran diagramas como el de clases, el de casos de uso, etc. Para ello se ha utilizado:

- **Just in mind:** es un software muy completo para la creación de mockups y wireframes. Su interfaz es muy amigable y posee una gran variedad de elementos de diseño y paletas donde escoger, por lo que ha sido el mejor candidato para la labor. Con él se han realizado los bocetos de las interfaces.
- **Star UML:** se ha utilizado en su versión 2.0.0 beta. Es un software construido bajo el software del novedoso software Brackets de Adobe. Por consiguiente, su interfaz es muy amigable y ligera, mucho mejor que cualquiera de la competencia, además de ser libre y extensible modularmente. Se ha utilizado para realizar todos los diagramas correspondientes al diseño del sistema.

5.2.4. Desarrollo

En la fase de desarrollo se implementa todo lo establecido en el diseño de la aplicación. Aquí también se incluyen las herramientas para el testeo, ya que son las mismas.

- **WebStorm:** escogido por ser uno de los más populares editores de texto para el desarrollo web, en especial front-end. Otras opciones serían Sublime Text y Brackets de Adobe.
- **Git + Github:** sistema y repositorio donde se almacena el código, llevando un control de versiones y así conservar la seguridad del código ante posibles pérdidas, catástrofes y malas versiones. Se ha preferido Git ante otros como Subversión por su fiabilidad, por ser un sistema distribuido y por la confianza y la experiencia con este sistema.

6. Especificación de requisitos

En el análisis de requisitos se definen explícitamente todos los aspectos funcionales y no funcionales que tendrá la aplicación. Mediante esta especificación, cualquier miembro del personal tiene una referencia común para consultar qué y cómo se debe construir el sistema, incluso para alguien que se acaba de incorporar al equipo.

Esta especificación de requisitos debe ser lo más concreta posible, pues servirá como guía de desarrollo, y durante un desarrollo es probable que se detecten problemas en la especificación de requisitos. Además esa detección, cuando más adelantada está la fase de desarrollo más coste supone solventar un problema de requisitos. Por ello, la fase de análisis de requisitos es una de las más importantes en el desarrollo de software, y productos en general.

Los objetivos principales de una especificación de requisitos es informar sobre los alcances que tiene el sistema sin mencionar la manera de implementación de estos. Dicho de otra manera, se describe lo que el sistema hace pero no cómo lo hace.

En este capítulo se definirá un diagrama de casos de uso en el que se puede ver el sistema de una forma más general. A partir de ahí, se entrará en detalle para definir los requisitos, tanto funcionales como no funcionales, y se relacionarán con los casos de usos correspondientes.

6.1. Casos de uso

Los casos de uso facilitan la visión global del sistema, desde un punto de vista familiar para el cliente o usuario. Además ayudan al desarrollador a identificar cuáles son las necesidades del usuario y para cuales acciones en concreto.

Un caso de uso describe un empleo que un usuario puede dar al sistema. Su principal utilidad es la de analizar y definir de manera rápida y visual el comportamiento que debe tener el sistema según unos actores, que en este caso será el usuario. Además,

un caso de uso debe representar una secuencia de acciones y pasos que el usuario debe seguir para realizar ese uso.

La definición de los casos de uso se realiza mediante la creación de un diagrama. Este diagrama posee un formato en concreto incluido en la especificación de un documento UML, en el que se utilizan elipses, relaciones y ciertas directivas para establecer los límites del sistema. Los actores se encuentran fuera del sistema y los casos de uso están dentro.

A continuación, se presenta el diagrama de casos de uso para TripMinder:

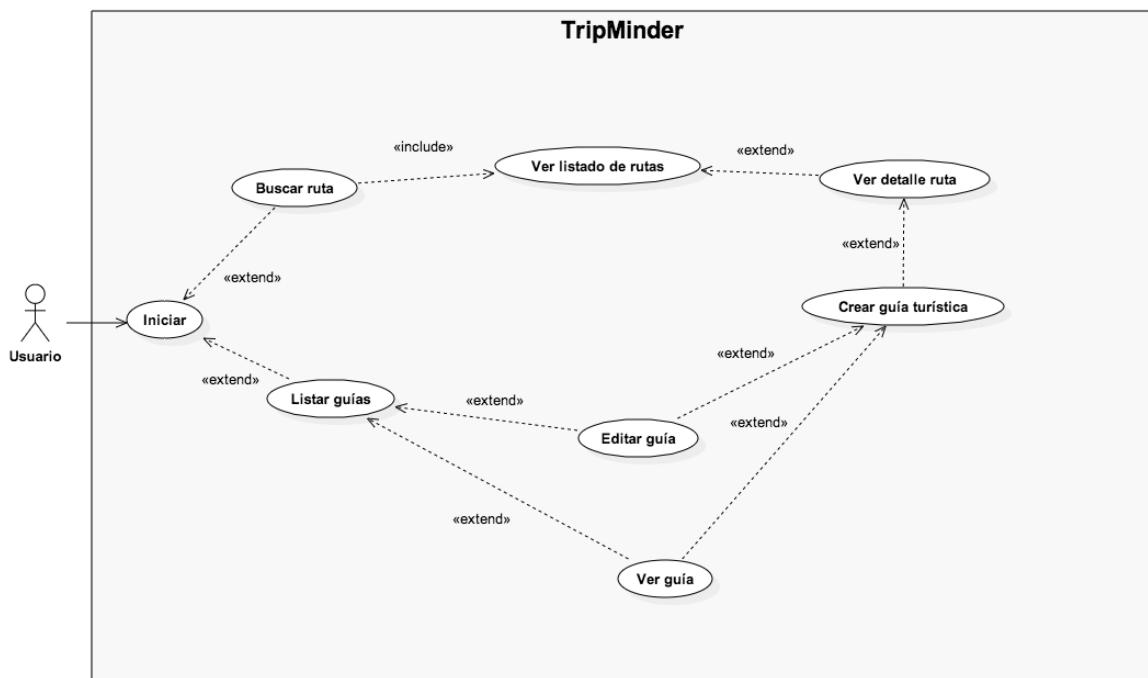


Figura 6.1. Diagrama de casos de uso

Un diagrama de casos de uso se suele acompañar con el detalle de los casos de uso, que son “tarjetas” en la que en cada una se describen en detalle un caso de uso en concreto.

En estas tarjetas, se utilizan los siguientes campos:

- Identificador de Caso de uso

- Actores involucrados
- Nombre
- Pre-condiciones
- Secuencia de eventos de flujo
- Post-condiciones

Para este caso, el único actor es el usuario, por lo que se omitirá en la inclusión de las tarjetas. Dicho esto, a continuación se muestra la lista de tarjetas de especificación de Casos de uso:

Identificador	CU-01
Nombre	Iniciar
Pre-condición	El usuario abre la aplicación, con conexión a Internet.
Secuencia	La aplicación se abre
Post-condición	Al abrirse, se muestra la pantalla de inicio.

Tabla 6.1. Caso de uso 1

Identificador	CU-02
Nombre	Buscar ruta
Pre-condición	El usuario inserta campos de origen y/o destino.
Secuencia	<ul style="list-style-type: none"> • El usuario puede insertar dos puntos de origen-destino en el mapa • Se realizan las peticiones a las API's • Se devuelven los resultados
Post-condición	Si hay resultados, se parsean.

Tabla 6.2. Caso de uso 2

Identificador	CU-03
----------------------	--------------

Nombre	Ver destinos sugeridos
Pre-condición	El usuario realiza una búsqueda sin destino
Secuencia	<ul style="list-style-type: none"> • Se abre pantalla con los destinos recomendados, según interés turístico y distancia • El usuario selecciona un destino
Post-condición	Se procede a la búsqueda de forma normal

Tabla 6.3. Caso de uso 3

Identificador	CU-04
Nombre	Ver listado de rutas
Pre-condición	El usuario ha realizado una búsqueda
Secuencia	<ul style="list-style-type: none"> • Se muestra una lista de los resultados de la búsqueda • Se permite reordenar los resultados
Post-condición	Los resultados de la búsqueda son mostrados y organizados en pestañas

Tabla 6.4. Caso de uso 4

Identificador	CU-05
Nombre	Ver detalle ruta
Pre-condición	Se listan las rutas y se pulsa sobre una ruta
Secuencia	Se muestran los detalles de las rutas, como los transbordos, costes, distancia, tiempo, etc. También un mensaje para crear guía turística, y un botón para seguir la ruta.
Post-condición	Se muestra el nombre del lugar en el campo de texto

Tabla 6.5. Caso de uso 5

Identificador	CU-06
----------------------	--------------

Nombre	Seguir ruta
Pre-condición	El usuario ha entrado en el detalle de la ruta
Secuencia	Se abrirá una nueva ventana donde aparecerá un mapa e instrucciones de seguimiento de ruta.
Post-condición	Se iniciará el seguimiento de la ruta

Tabla 6.6. Caso de uso 6

Identificador	CU-07
Nombre	Crear guía turística
Pre-condición	El usuario ha pulsado en crear guía turística
Secuencia	Se mostrarán las opciones de crear guía turística
Post-condición	El usuario selecciona si crearla manual o automáticamente

Tabla 6.7. Caso de uso 7

Identificador	CU-08
Nombre	Editar guía manual
Pre-condición	El usuario ha seleccionado crear guía manualmente, o quiere editar una guía existente
Secuencia	<ul style="list-style-type: none"> • Se mostrarán lugares turísticos, separados en pestañas por las categorías correspondientes • El usuario añadirá lugares a la guía
Post-condición	Cuando acabe de crear la guía, se almacena

Tabla 6.8. Caso de uso 8

Identificador	CU-09
Nombre	Crear guía automáticamente
Pre-condición	El usuario ha pulsado en crear guía automáticamente

Secuencia	Se le requerirán al usuario los días que va a estar y los sitios que quiere visitar, y en base a ello creará una guía.
Post-condición	Cuando acabe de crear la guía, se almacena

Tabla 6.9. Caso de uso 9

Identificador	CU-10
Nombre	Listar guías
Pre-condición	El usuario ha pulsado sobre listar guías
Secuencia	Se cargarán todas las guías almacenadas y se mostrarán
Post-condición	Se mostrarán las guías

Tabla 6.10. Caso de uso 10

Identificador	CU-11
Nombre	Ver guía
Pre-condición	El usuario ha pulsado sobre una guía
Secuencia	Se mostrarán los detalles de la guía y todos los sitios relacionados con ella
Post-condición	Se mostrará la guía

Tabla 6.11. Caso de uso 11

Identificador	CU-12
Nombre	Seguir guía
Pre-condición	El usuario ha pulsado sobre seguir guía
Secuencia	Al igual que en CU-08, se abrirá una ventana con mapa e instrucciones de seguimiento de la guía
Post-condición	Se iniciará el seguimiento de la guía

Tabla 6.12. Caso de uso 12

6.2. Requisitos funcionales

Los requisitos funcionales describen los aspectos funcionales que debe poseer un sistema, es decir, qué debe de hacer y cómo tiene que hacerlo. Para ello, se aplicará de nuevo un sistema de “tarjetas” donde aparecerán los siguientes campos:

- Identificado
- Nombre
- Descripción
- Prioridad

A continuación se presenta la lista de requisitos funcionales:

Identificador	RF-01
Nombre	Calcular ruta
Descripción	En base a un punto A y otro B, se debe calcular las rutas posibles que conecten los dos puntos. Estas pueden incluir transbordos y varios medios, entre los que se encuentran el avión, tren, bus, coche, bicicleta y a pie.
Prioridad	<input checked="" type="checkbox"/> Alta/Essencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.13. Requisito funcional 1

Identificador	RF-02
Nombre	Listar posibles destinos
Descripción	Cuando no se selecciona un destino, se debe realizar una consulta sobre una lista de posibles destinos, determinados por su interés turístico y distancia.
Prioridad	<input checked="" type="checkbox"/> Alta/Essencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.14. Requisito funcional 2

Identificador	RF-03
Nombre	Aplicar Geocodificación [Inversa]
Descripción	Implementar geocodificación y geocodificación inversa para a partir de una coordenada, poder saber el lugar que es con datos humanos, como ciudad, código postal, país, etc.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.15. Requisito funcional 3

Identificador	RF-04
Nombre	Calcular distancia y tiempo de ruta
Descripción	Se debe calcular y posteriormente mostrar la distancia y el tiempo de la ruta marcada entre un origen y un destino.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.16. Requisito funcional 4

Identificador	RF-05
Nombre	Ordenar resultados
Descripción	Los resultados serán ordenados por distancia, precio y tiempo.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.17. Requisito funcional 5

Identificador	RF-06
Nombre	Crear guía turística manual
Descripción	Se mostrarán, separados por pestañas, lugares de las categorías indicadas en las preferencias de usuario.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.18. Requisito funcional 6

Identificador	RF-07
Nombre	Obtener lugares turísticos
Descripción	Es utilizado por los requisitos RF-06 y RF-07. Se realizan las peticiones necesarias a las API's de información turística para obtener los lugares de interés turístico correspondientes.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.19. Requisito funcional 7

Identificador	RF-08
Nombre	Configurar perfil de usuario
Descripción	Se podrán cambiar preferencias de sitios (museos, bares...), preferencias de rutas (autovías, autopistas...) y otros datos de utilidad.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.20. Requisito funcional 8

Identificador	RF-09
Nombre	Incluir medios globales
Descripción	Los datos de rutas se obtendrán de medios globales. Estos son las API's, como la de Google Directions, que permiten obtener rutas de muchas partes del mundo.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.21. Requisito funcional 10

Identificador	RF-11		
Nombre	Calcular coste de taxis		
Descripción	Se obtendrá el coste sobre un trayecto en taxi para una ciudad determinada.		
Prioridad	<input type="checkbox"/> Alta/Esencial <input checked="" type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional		

Tabla 6.22. Requisito funcional 11

Identificador	RF-12		
Nombre	Reconocer lugares		
Descripción	Mediante una foto y el gps, utilizar las API's correspondientes para reconocer y proporcionar información sobre el lugar.		
Prioridad	<input type="checkbox"/> Alta/Esencial <input checked="" type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional		

Tabla 6.23. Requisito funcional 12

Identificador	RF-13		
Nombre	Historial de búsquedas		
Descripción	Se creará un historial simple de búsquedas. Se podrá utilizar para sugerir destinos en base a la zona de destinos previos		
Prioridad	<input type="checkbox"/> Alta/Esencial <input checked="" type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional		

Tabla 6.24. Requisito funcional 13

Identificador	RF-14		
Nombre	Calcular coste de aduanas y fronteras		
Descripción	Se intentará obtener datos sobre precios de aduanas y fronteras, útiles para las rutas en coche, bicicleta o andando.		
Prioridad	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input checked="" type="checkbox"/> Baja/Opcional		

Tabla 6.25. Requisito funcional 14

6.3. Requisitos de interfaz

Los requisitos de interfaz, como su nombre indica, corresponden a los que tienen relación con la interfaz, y por tanto con la distribución del contenido y del diseño de la aplicación.

Los detalles sobre la interfaz y el diseño gráfico en general se detallan en otro punto más adelante, por lo que en este punto se mencionarán los aspectos esperados de la interfaz de modo general.

Identificador	RI-01
Nombre	Diseño Plano
Descripción	El diseño de toda la aplicación tendrá un estilo plano
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.26. Requisito de interfaz 1

Identificador	RI-02
Nombre	Multiplataforma
Descripción	Se deben crear y organizar las pantallas de la aplicación teniendo en cuenta los múltiples tamaños de pantallas de dispositivos.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.27. Requisito de interfaz 2

Identificador	RI-03
Nombre	Diseñar UI usable
Descripción	Realizar un diseño personalizado de UI moderno y usable que proporcione una gran experiencia de usuario
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.28. Requisito de interfaz 3

6.4. Requisitos no funcionales

Los requisitos no funcionales, en cambio, no describen el modo de actuar del sistema, sino que son definiciones sobre aspectos que se esperan del sistema pero no describen su funcionalidad. Entre ellos, aspectos de preferencias o de seguridad.

Los campos que se utilizarán para este caso son los mismos que los anteriores.

Identificador	RNF-01
Nombre	Optimización
Descripción	Las consultas a las API's deben ser mínimas para así no gastar puntos de API y evitar tiempos de espera largos. Además habrá que guardar ciertos resultados.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.29. Requisito no funcional 1

Identificador	RNF-02
Nombre	Seguridad
Descripción	Aunque no son muchos, los datos que maneja la aplicación deben cumplir con las normas mínimas de seguridad y protección de datos.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.30. Requisito no funcional 2

Identificador	RNF-03
Nombre	Multiplataforma
Descripción	El sistema debe ser multiplataforma, y por tanto funcionar, al menos en los siguientes sistemas operativos: Android, iOS, Windows y Mac OS X
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.31. Requisito no funcional 3

Identificador	RNF-04
Nombre	Mantenibilidad
Descripción	La aplicación se desarrollará con una arquitectura y documentación específicas, que tienen como principio la estructuración del código para así facilitar modificaciones y actualizaciones, además ser lo más entendible posible.
Prioridad	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

Tabla 6.32. Requisito no funcional 4

7. Estudio de la viabilidad

En el estudio de la viabilidad se comentan varios aspectos que tratan sobre la gestión del proyecto. Para ello, se explica en cada punto lo relacionado con él. Además se utilizarán gráficas y material visual donde se puede ver con mayor claridad.

7.1. Planificación temporal

En la ejecución del proyecto de TripMinder se ha distribuido temporalmente la carga de trabajo mediante la división por hitos. Exactamente, se han distribuido en los siguientes 3 hitos:

- **Hito 1, Previo:** Se realiza toda la documentación posible de la memoria del TFG, se diseñan los logotipos y otros aspectos gráficos, y se comienza una iniciación en el software de desarrollo, en principio RAD Studio.
- **Hito 2, Ejecutables:** Se comienza el desarrollo, siguiendo una sistemática de desarrollo por componentes que posteriormente se unificarán y combinarán para la aplicación final.
- **Hito 3, App final:** Se combinan los ejecutables y se crean las pantallas para dar forma a la aplicación final. En el tiempo restante se intentará incluir alguna mejora y finalizar la documentación restante de la memoria.

A continuación se muestra el diagrama de Gantt creado con la herramienta Microsoft Project 2013:

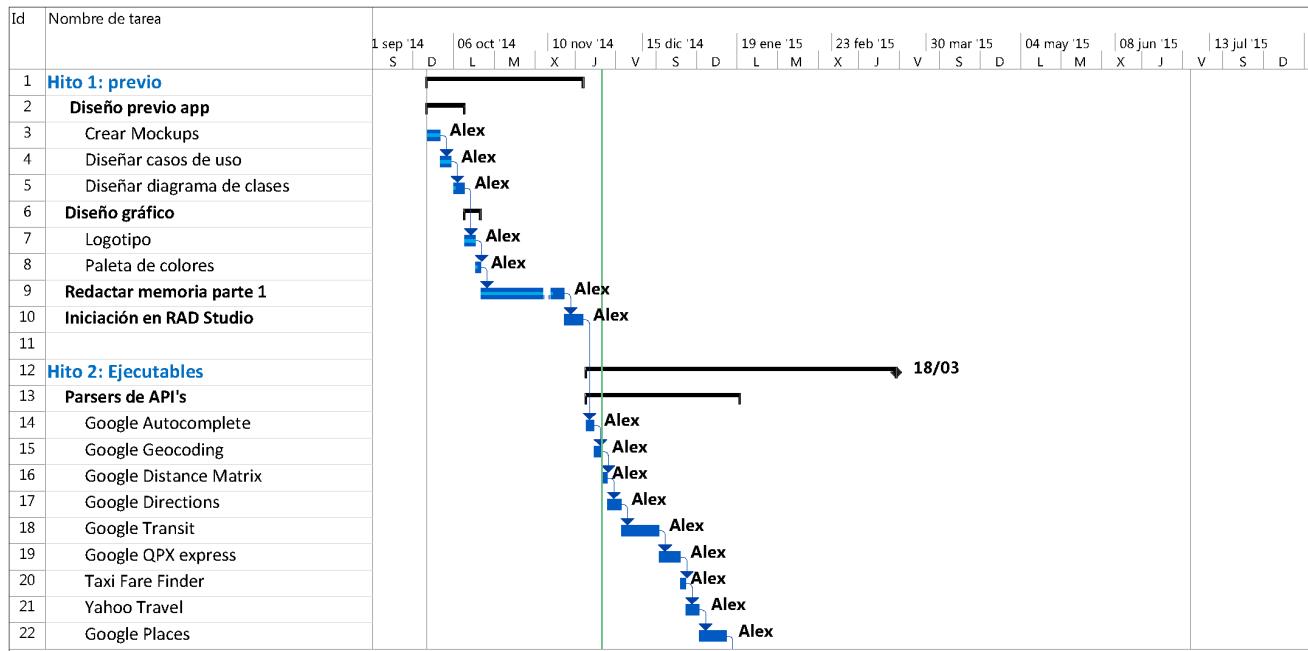


Figura 7.1. Diagrama de Gantt, página 1

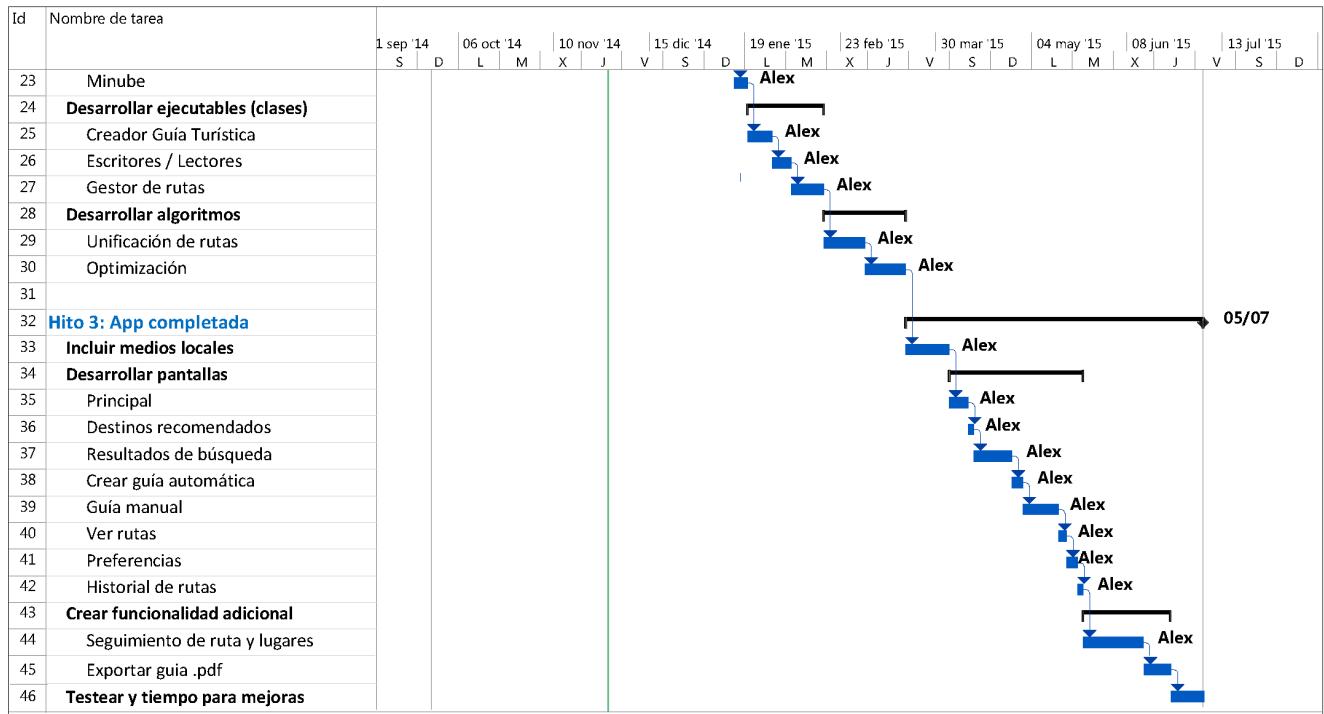


Figura 7.2. Diagrama de Gantt, página 2

En estos diagramas se puede observar la distribución temporal respecto a cada tarea, de forma gráfica, aunque no se pueden apreciar las fechas exactas y otros detalles.

Para ello, se adjunta la tabla del detalle de las tareas, donde se puede apreciar otros datos como las horas dedicadas, coste, etc.

Nombre de tarea	Trabajo	Duración	Comienzo	Fin	Coste
Hito 1: previo	114 horas	58 días	vie 26/09/14	sáb 22/11/14	1.710,00 €
Diseño previo app	26 horas	14 días	vie 26/09/14	jue 09/10/14	390,00 €
Crear Mockups	10 horas	5 días	vie 26/09/14	mar 30/09/14	150,00 €
Diseñar casos de uso	8 horas	4 días	mié 01/10/14	sáb 04/10/14	120,00 €
Diseñar diagrama de clases	8 horas	4 días	lun 06/10/14	jue 09/10/14	120,00 €
Diseño gráfico	12 horas	6 días	vie 10/10/14	mié 15/10/14	180,00 €
Logotipo	8 horas	4 días	vie 10/10/14	lun 13/10/14	120,00 €
Paleta de colores	4 horas	2 días	mar 14/10/14	mié 15/10/14	60,00 €
Redactar memoria parte 1	62 horas	31 días	jue 16/10/14	sáb 15/11/14	930,00 €
Iniciación en RAD Studio	14 horas	7 días	dom 16/11/14	sáb 22/11/14	210,00 €
Hito 2: Ejecutables	230 horas	115 días	lun 24/11/14	mié 18/03/15	3.450,00 €
Parsers de API's	114 horas	57 días	lun 24/11/14	lun 19/01/15	1.710,00 €
Google Autocomplete	6 horas	3 días	lun 24/11/14	mié 26/11/14	90,00 €
Google Geocoding	6 horas	3 días	jue 27/11/14	sáb 29/11/14	90,00 €
Google Distance Matrix	4 horas	2 días	dom 30/11/14	lun 01/12/14	60,00 €
Google Directions	10 horas	5 días	mar 02/12/14	sáb 06/12/14	150,00 €
Google Transit	28 horas	14 días	dom 07/12/14	sáb 20/12/14	420,00 €
Google QPX express	16 horas	8 días	dom 21/12/14	dom 28/12/14	240,00 €
Taxi Fare Finder	4 horas	2 días	lun 29/12/14	mar 30/12/14	60,00 €
Yahoo Travel	10 horas	5 días	mié 31/12/14	dom 04/01/15	150,00 €
Google Places	20 horas	10 días	lun 05/01/15	mié 14/01/15	300,00 €
Minube	10 horas	5 días	jue 15/01/15	lun 19/01/15	150,00 €
Desarrollar ejecutables (clases)	56 horas	28 días	mar 20/01/15	lun 16/02/15	840,00 €
Creador Guía Turística	18 horas	9 días	mar 20/01/15	mié 28/01/15	270,00 €
Escritores / Lectores	14 horas	7 días	jue 29/01/15	mié 04/02/15	210,00 €
Gestor de rutas	24 horas	12 días	jue 05/02/15	lun 16/02/15	360,00 €
Desarrollar algoritmos	60 horas	30 días	mar 17/02/15	mié 18/03/15	900,00 €
Unificación de rutas	30 horas	15 días	mar 17/02/15	mar 03/03/15	450,00 €
Optimización	30 horas	15 días	mié 04/03/15	mié 18/03/15	450,00 €
Hito 3: App completada	218 horas	109 días	jue 19/03/15	dom 05/07/15	3.270,00 €
Incluir medios locales	32 horas	16 días	jue 19/03/15	vie 03/04/15	480,00 €

Desarrollar pantallas	98 horas	49 días	sáb 04/04/15	vie 22/05/15	1.470,00 €
Principal	14 horas	7 días	sáb 04/04/15	vie 10/04/15	210,00 €
Destinos recomendados	4 horas	2 días	sáb 11/04/15	dom 12/04/15	60,00 €
Resultados de búsqueda	28 horas	14 días	lun 13/04/15	dom 26/04/15	420,00 €
Crear guía automática	8 horas	4 días	lun 27/04/15	jue 30/04/15	120,00 €
Guía manual	26 horas	13 días	vie 01/05/15	mié 13/05/15	390,00 €
Ver rutas	6 horas	3 días	jue 14/05/15	sáb 16/05/15	90,00 €
Preferencias	8 horas	4 días	dom 17/05/15	mié 20/05/15	120,00 €
Historial de rutas	4 horas	2 días	jue 21/05/15	vie 22/05/15	60,00 €
Crear funcionalidad adicional	64 horas	32 días	sáb 23/05/15	mar 23/06/15	960,00 €
Seguimiento de ruta y lugares	44 horas	22 días	sáb 23/05/15	sáb 13/06/15	660,00 €
Exportar guia .pdf	20 horas	10 días	dom 14/06/15	mar 23/06/15	300,00 €
Testear y tiempo para mejoras	24 horas	12 días	mié 24/06/15	dom 05/07/15	360,00 €

Tabla 7.1. Detalle de tareas

7.2. Estimación de costes

En la tabla anterior se ha podido comprobar que incluye una estimación de costes, puesto que incluye una columna de costes. Sin embargo no se ha explicado de donde proceden esos costes.

Existen varios métodos de estimación de costes, tales como la técnica iterativa, el modelado algorítmico de costes, estimación por analogía o pricing to win. En este caso se ha escogido lo más parecido al modelo de “Juicio experto” dónde por valoración propia, se tiene en cuenta varias variables para el cálculo del coste como son:

- Peso en horas de una tarea
- Asignación de recursos
- Costes y calendario de los recursos

El único recurso sería el alumno, que tendría un calendario muy diverso, por lo que se ha simplificado acordando una media de trabajo diario de 2 horas, a un precio de 15 € la hora, lo que puede considerarse un sueldo de un desarrollador junior.

A continuación se adjunta una gráfica, creada mediante Microsoft Project 2013, donde se puede ver los costes del proyecto:

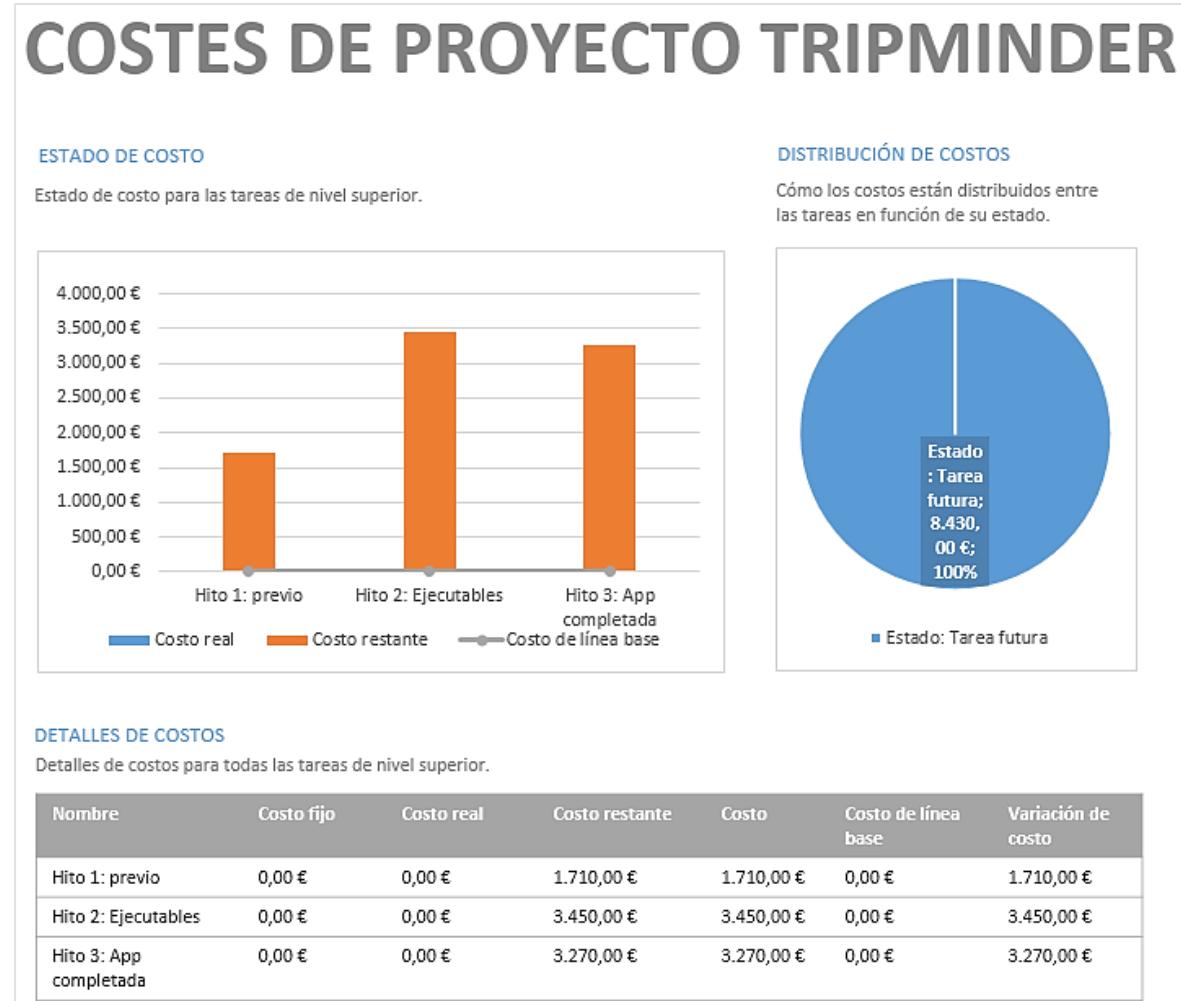


Figura 7.3. Informe de costes de TripMinder

7.3. Estimación de riesgos

Al igual que para los costes, para la estimación de riesgos también se dispone de varios métodos y formas para hacerlo. En este caso, se presentarán separados según el tipo de riesgo.

7.3.1. Riesgos tecnológicos

Identificador	RIT-01		
Descripción	Problemas de ejecución y “deploy” en alguna de las plataformas		
Probabilidad	Alta	Efecto	Tolerable
Estrategia	Se estudiaría qué puede causar el problema y se intentará solventarlo. Si no se pudiera, no sería tan importante, siempre que en alguna funcione.		

Tabla 7.2. Riesgo tecnológico 1

Identificador	RIT-02		
Descripción	El rendimiento de la aplicación no es el esperado		
Probabilidad	Media	Efecto	Tolerable
Estrategia	Es de esperar una aplicación de numerosas peticiones por internet. Además, no influye en el desarrollo de la aplicación.		

Tabla 7.3. Riesgo tecnológico 2

7.3.2. Riesgos de personal

Identificador	RIP-01		
Descripción	Enfermedad del alumno		
Probabilidad	Baja	Efecto	Tolerable
Estrategia	Se podría trabajar desde casa, aún con rendimiento más bajo. En la estimación se cuenta con tiempo para posibles retrasos.		

Tabla 7.4. Riesgo de personal 1

Identificador	RIP-02		
Descripción	Desmotivación del alumno en el proyecto		
Probabilidad	Baja	Efecto	Serio
Estrategia	Tomarse un descanso en momentos de frustración es una buena manera de volver a motivarse.		

Tabla 7.5. Riesgo de personal 2

7.3.3. Riesgos de herramientas

Identificador	RIT-01		
Descripción	Mal funcionamiento en la herramienta de desarrollo RAD Studio		
Probabilidad	Alta	Efecto	Serio
Estrategia	Depende de la fase de desarrollo, supondría más o menos problema. Se procedería a migrar y utilizar Xamarin Studio.		

Tabla 7.6. Riesgo de herramientas 1

Identificador	RIT-02		
Descripción	Mal funcionamiento en la beta de Star UML 2.0		
Probabilidad	Alta	Efecto	Tolerable
Estrategia	Existen muchas otras herramientas en el mercado para diagramas UML. Se probaría a utilizar cualquier otra.		

Tabla 7.7. Riesgo de herramientas 2

7.3.4. Riesgos de requerimientos

Identificador	RIR-01		
Descripción	Posibles cambios en los requisitos		
Probabilidad	Baja	Efecto	Serio
Estrategia	Se aplicarían los cambios en la medida de lo posible		

Tabla 7.8. Riesgo de requerimientos 1

7.3.5. Riesgos de estimación

Identificador	RIE-01		
Descripción	Infraestimación del proyecto		
Probabilidad	Media	Efecto	Tolerable
Estrategia	Realizar lo que de tiempo del proyecto		

Tabla 7.9. Riesgo de estimación 1

Identificador	RIE-02		
Descripción	Mala estimación en una tarea		
Probabilidad	Media	Efecto	Tolerable
Estrategia	Se modificarían la estimación de otras tareas para equilibrar		

Tabla 7.10. Riesgo de estimación 2

7.4. Línea de futuro

TripMinder es un proyecto pensado para que sea útil para el usuario, por lo que se intentará acabar la parte esencial de este proyecto. Si el proyecto consiguiera tener cierto éxito y un buen funcionamiento, se estudiaría el caso de ampliarlo para poder distribuirlo, estableciendo un modelo de negocio (posiblemente gratuito), en las stores de las correspondientes plataformas.

Otra línea de futuro se conseguiría mediante la inclusión de obtención de datos de medios locales, de tal forma que podría llegar a convertirse en una aplicación para el uso interurbano de Alicante, o incluso de la Comunidad Valenciana.

8. Diseño y desarrollo del sistema

En esta sección se hablará sobre cómo se ha desarrollado el sistema, sus características más importantes y se comentarán algunos aspectos del diseño y la estructura.

8.1. Multiplataforma

Una de las características más importantes en el desarrollo de este proyecto, TripMinder, es que la aplicación es multiplataforma. Durante su desarrollo se ha soportado y trabajado en las principales plataformas: versiones para iOS >= 7, Android >= 4 y navegadores, accesibles desde cualquier dispositivo o PC. Aunque debería ser también compatible con otras plataformas, como Windows 8, Windows Phone, BlackBerry, Firefox OS, etc.

Ya que Ionic trabaja por debajo con Apache Cordova, aquí se pueden ver una lista de las plataformas que soporta, detallando los plugins compatibles para cada una de ellas:

https://cordova.apache.org/docs/en/4.0.0/guide_support_index.md.html

Tanto la creación de un proyecto en Ionic, como el desarrollo, testeo y el despliegue de la aplicación en un dispositivo o emulador es un proceso relativamente sencillo. Básicamente, se deben instalar y configurar las librerías y SDK's de las plataformas a las que se va a desarrollar.

Una vez hecho, desde la misma línea de comandos, suponiendo que tenemos *node.js* y *npm* instalados, procedemos a instalar el *cli-tool* de Ionic, Apache Cordova y Bower (gestor de paquetes que usa Ionic), mediante el comando:

```
npm install -g cordova ionic bower
```

A partir de este punto, se puede utilizar el *cli-tool* de Ionic. Los comandos más importantes que disponemos son:

- `ionic start APP_NAME sidemenu`: crea una aplicación con un menú lateral incluido.
- `ionic setup sass`: añade SASS como preprocesador de estilos.
- `ionic platform add PLATFORM`: añade una plataforma a la configuración del proyecto.
- `ionic emulate PLATFORM`: ejecuta la aplicación en el emulador de la plataforma seleccionada.
- `ionic run PLATFORM`: ejecuta la aplicación en un dispositivo conectado, o en emulador en su defecto.
- `ionic build PLATFORM`: compila la aplicación para la plataforma seleccionada.
- `ionic serve`: ejecuta la aplicación en el navegador y la actualiza a tiempo real.
- `ionic [add|remove] PACKAGE_NAME`: añade/elimina un paquete de bower.
- `ionic plugin [add|remove] PLUGIN_NAME`: añade/elimina un plugin de Cordova.
- `ionic resources`: genera el icono y splash screen para las diferentes plataformas.
- `ionic upload`: sube la aplicación a la cuenta asociada en ionic.io.

* PLATFORM: pueden ser los valores “ios” o “android”. Ej: `ionic run android`

Se puede encontrar más información sobre esto en la bitácora de desarrollo en Ionic de este proyecto, alojada en Github:

https://github.com/alexjoverm/TripMinder/blob/master/Ionic_documentation.md

8.2. Arquitectura

La arquitectura de la aplicación está acotada a la que utiliza Ionic, que a su vez es la que utiliza AngularJS, y sigue el patrón MVC. Este patrón describe:

- **Model:** El modelo es el objeto que contiene la lógica de negocio, o acceso a la BBDD. En AngularJS 1.x no existe el modelo en sí, aunque se pueden implementar aproximaciones, o utilizar los \$resources de AngularJS, que son lo más parecido a modelos CRUD para servicios REST.
- **View:** Las vistas son la presentación de los datos, es decir, la interfaz en sí. En AngularJS, estas son los archivos *.html* principalmente, además de las vistas embebidas en *.js*.
- **Controller:** Un controlador se encarga de ser intermediario entre la vista y el modelo, además de procesar datos y atender a los eventos generados por la vista.

Además, AngularJS introduce el concepto de **Service**, también conocido como Factory, encargado de llevar procesamientos más pesados y de separar la lógica no relativa a la vista del controlador. Ejemplos donde estos son utilizados: llamadas a servicios REST, intercambio de información entre controladores o encapsular lógica de componentes UI.

También se puede encontrar en el fichero *directives.js* algunas Directivas de AngularJS que se han creado para ciertas funcionalidades.

Existen otros tipos de componentes en AngularJS, de los que no se va a entrar en detalle ya que no es relevante y no se utilizan en la arquitectura de la aplicación. Algunos de ellos son Interceptor, Module, Constant, Provider....

8.2.1. Diagrama

A continuación se detallará la estructuración y funcionamiento del sistema mediante el uso de un diagrama y su posterior explicación.

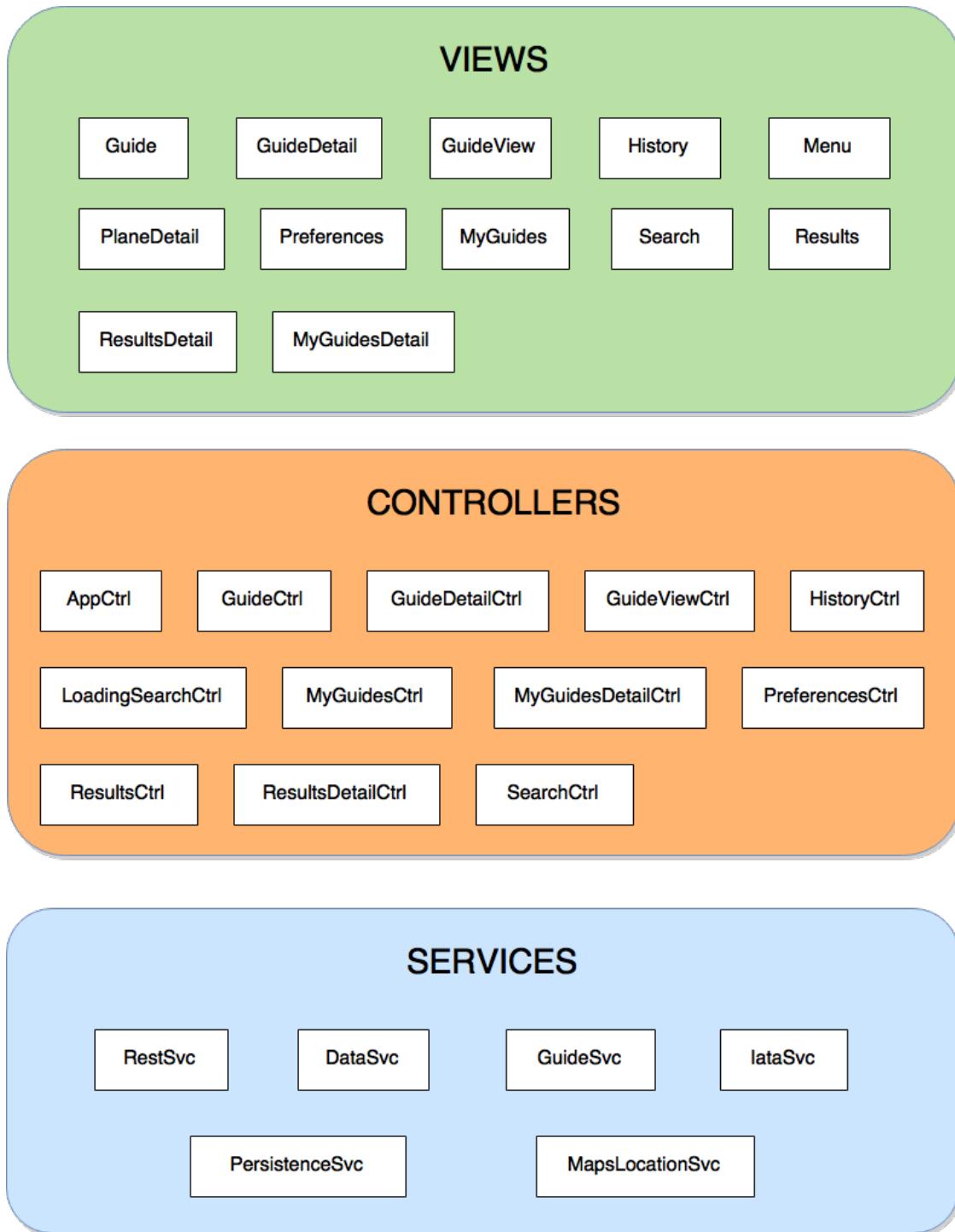


Figura 8.1. Diagrama MVC de TripMinder

8.2.2. Views y Controllers

No se va a entrar en la explicación detallada de cada una de las vistas y controladores. Simplemente decir que las vistas están enlazadas a un controlador, que realiza la lógica relativa a la vista, conecta datos y procesa eventos.

En el siguiente código (*app.js*) se puede ver dicho enlace:

```
.state('app', { url: '',
    abstract: true,
    templateUrl: "templates/menu.html",
    controller: 'AppCtrl'
})
.state('app.search', { url: "/search",
    views: {
        'menuContent': {
            templateUrl: "templates/search.html",
            controller: 'SearchCtrl'
        }
    }
})
.state('app.results', { url: "/results",
    views: {
        'menuContent': {
            templateUrl: "templates/results.html",
            controller: 'ResultsCtrl'
        }
    }
})
.state('app.resultsDetail', { url: "/results/:id/:num",
    views: {
        'menuContent': {
            templateUrl: "templates/resultsDetail.html",
            controller: 'ResultsDetailCtrl'
        }
    }
})
.state('app.planeDetail', { url: "/plane-results/:id/:num",
    views: {
        'menuContent': {
            templateUrl: "templates/planeDetail.html",
            controller: 'ResultsDetailCtrl'
        }
    }
})
.state('app.history', { url: "/history",
    views: {
        'menuContent': {
            templateUrl: "templates/history.html",
            controller: 'HistoryCtrl'
        }
    }
})
```

```

        }
    }
})
.state('app.preferences', {
    url: "/preferences",
    views: {
        'menuContent': {
            templateUrl: "templates/preferences.html",
            controller: 'PreferencesCtrl'
        }
    }
})
.state('app.guide', { url: "/guide",
    views: {
        'menuContent': {
            templateUrl: "templates/guide.html",
            controller: 'GuideCtrl'
        }
    }
})
.state('app.guideDetail', { url: "/guide/:id",
    views: {
        'menuContent': {
            templateUrl: "templates/guideDetail.html",
            controller: 'GuideDetailCtrl'
        }
    }
})
.state('app.guideView', { url: "/guideView",
    views: {
        'menuContent': {
            templateUrl: "templates/guideView.html",
            controller: 'GuideViewCtrl'
        }
    }
})
.state('app.myGuides', { url: "/myGuides",
    views: {
        'menuContent': {
            templateUrl: "templates/myGuides.html",
            controller: 'MyGuidesCtrl'
        }
    }
})
.state('app.myGuidesDetail', { url: "/myGuidesDetail/:id",
    views: {
        'menuContent': {
            templateUrl: "templates/myGuidesDetail.html",
            controller: 'MyGuidesDetailCtrl'
        }
    }
})
})

```

8.2.3. Services

En el diagrama encontramos los siguientes services:

- **RestSvc:** Se encarga de llevar a cabo todas las llamadas a los servicios REST y librerías relativas a la búsqueda y recolección de datos de rutas, además de presentar las ventanas modales que muestran información sobre el progreso de estas y, en general, llevar el control de dichas peticiones.
- **DataSvc:** Este servicio se puede considerar el pilar de la aplicación. DataSvc procesa, guarda, mantiene y comparte los datos que desde RestSvc, GuideSvc y algunos controladores insertan en él. Además procesa la información para llevarla a un formato común, especialmente en los resultados de rutas de transporte público (avión, tren, autobús).
- **GuideSvc:** Servicio utilizado por los controladores relativos a las guías. Realiza las peticiones relativas a las guías, conecta con DataSvc para almacenar datos y además realiza procesamiento de datos. Este servicio resta parte del trabajo que DataSvc y RestSvc llevaría a cabo, además de separar el ámbito de los datos.
- **IataSvc:** Carga y procesa los datos de los aeropuertos, los mantiene en memoria y posteriormente, cuando RestSvc se lo indica, se encarga de buscar los aeropuertos más cercanos respecto a una coordenada.
- **PersistenceSvc:** Como su nombre indica, PersistenceSvc tiene como labor almacenar y recuperar información para preservarla persistentemente. Esto lo aplica al historial, a las guías y a las preferencias. Para ello utiliza el LocalStorage del API de HTML5.
- **MapsLocationSvc:** Contiene lógica relacionada con la creación, edición y mantenimiento de mapas y marcadores. También realiza parte del proceso de localización, para el cual utiliza el plugin de localización de Cordova.

8.2.4. Cordova plugins

Los plugins de Cordova nos permiten aumentar la funcionalidad de nuestra aplicación utilizando las funcionalidades de nuestro smart-device. Estos plugins proveen una API, accesible desde el código javascript, que accede a los sensores y componentes, como pueden ser la cámara, sensor de movimiento, almacenamiento, etc.

Ionic utiliza ng-Cordova, que es una adaptación de los plugins de Cordova para su utilización en AngularJS. Aunque también se podrían utilizar los propios de Cordova, los de ng-Cordova hacen que su uso sea más simple y sencillo para el programador cuando se usa AngularJS.

TripMinder utiliza los siguientes plugins:

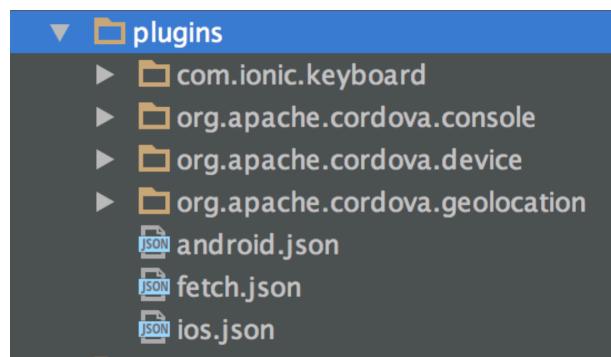


Figura 8.2. Cordova plugins que usa TripMinder

- **Keyboard:** permite acceder al teclado virtual de los dispositivos, como el de Android o iOS. En la app se utiliza para cerrarlo en ciertos momentos.
- **Console:** permite utilizar `console.log()` para hacer debug en los dispositivos, además de añadir más funcionalidad al método.
- **Device:** permite acceder a información del dispositivo. Se utiliza para saber si la plataforma en la que se está ejecutando es iOS.
- **Geolocation:** permite acceder al sensor de localización. Se utiliza para centrar el mapa de búsqueda a la posición donde el usuario se encuentra.

8.2.5. Bower packages

Los paquetes de bower son simplemente librerías en javascript que añaden funcionalidad a la parte front-end, encapsuladas en un paquete. La mayoría de las librerías más populares, como jQuery o Bootstrap, están disponibles para instalar con bower.

No es relevante explicar todos los paquetes que se utilizan. De manera resumida, algunos de ellos son: AngularJS en sí, otros que facilitan el trabajo con API's de Google, y otros que añaden elementos UI. Cabe mencionar que algunos se han modificado, ya sea por mal funcionamiento, falta de funcionalidad o adaptación a la aplicación.

La siguiente imagen muestra la lista completa de paquetes:

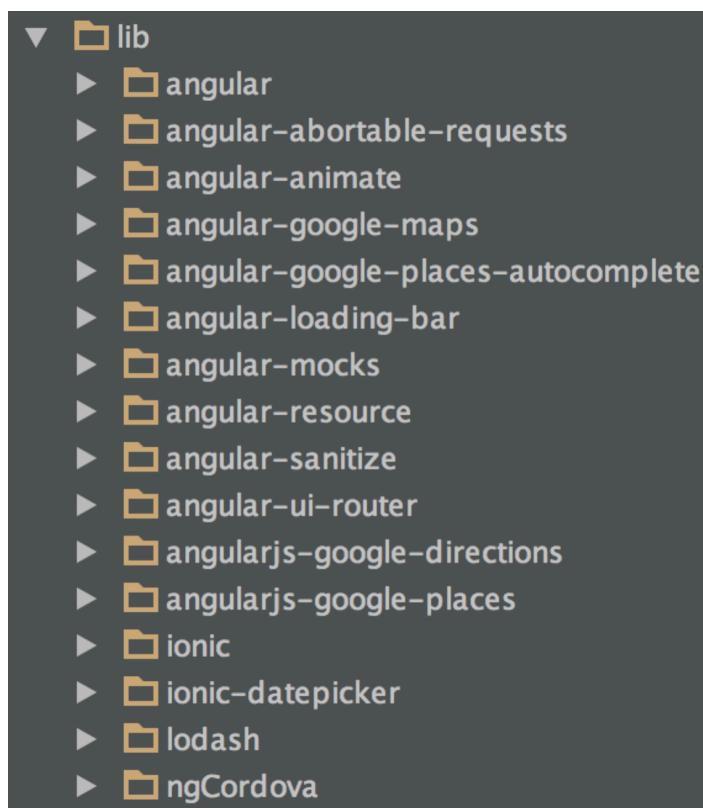


Figura 8.3. Lista de paquetes Bower

8.2.6. Almacenamiento

Como se ha comentado anteriormente, TripMinder utiliza el servicio PersistenceSvc para almacenar datos referentes a guías, rutas y preferencias. Para ello, hay dos opciones:

- **Plugin SQLite de Cordova** (<https://github.com/litehelpers/Cordova-sqlite-storage>): permite utilizar SQLite mediante el uso de un API común, prácticamente válido para todas las plataformas, aunque con posibles problemas.
- **HTML5 Web Storage:** almacenamiento simple del tipo “clave-valor”. Funcionaría en todas las plataformas.

El escogido para su uso en TripMinder es HTML5 Web Storage. Para ello, no ha sido necesario realizar un estudio a fondo. Las razones por las que se ha escogido son:

- No existen relaciones entre los datos, por tanto no hay necesidad de uso de una base de datos relacional.
- Totalmente multiplataforma.
- Evitamos problemas de utilizar un plugin adicional.
- Sólo se necesita un almacenamiento simple.

8.3. Harvesting (recolección de datos)

Quizá la parte de la recolección de datos ha sido la más difícil y costosa de todo el proyecto. Para el usuario es simple: introduce un origen y destino, y se muestra la información organizada. Pero esto lleva un proceso detrás algo tedioso, compuesto por diferentes fases.

El siguiente esquema define el proceso mencionado:

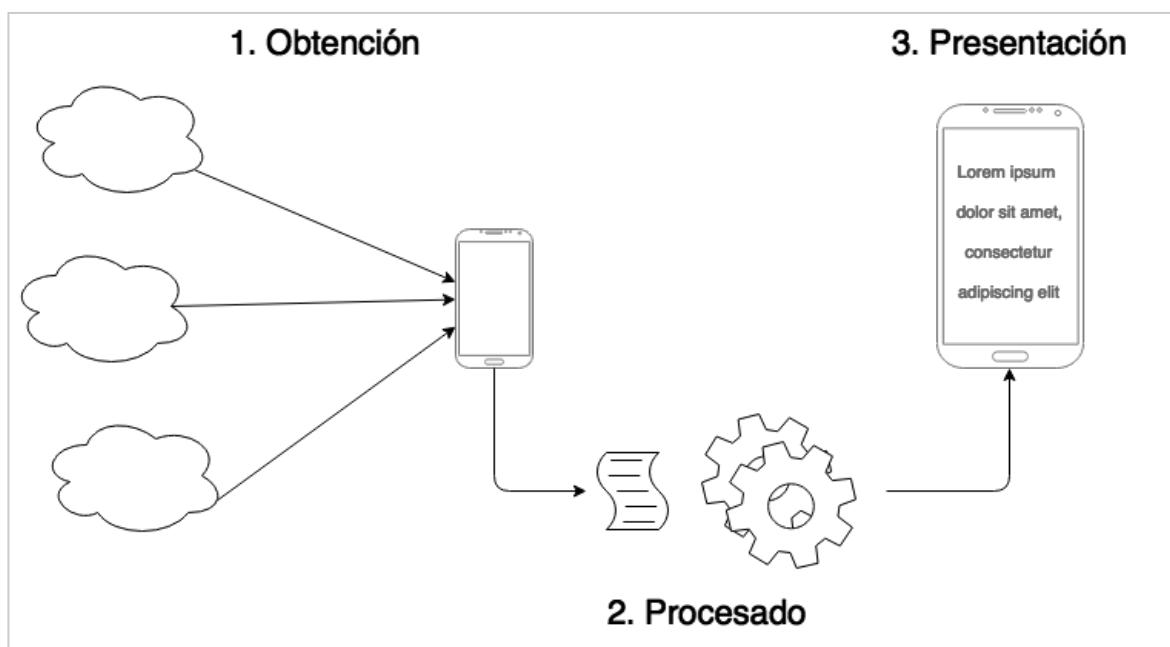


Ilustración 8.1. Esquema del proceso de Harvesting

- 1. Obtención:** se realizan las peticiones a los servicios y fuentes de datos correspondientes para obtener toda la información deseada.
- 2. Procesado:** cuando se procesa esa información, se eliminan datos no deseados, se crean nuevos y se reestructura parte de la información. El objetivo es llevarlo a un formato común.
- 3. Presentación:** una vez se obtiene el formato común, se escriben las vistas correspondientes para la presentación de la información.

8.3.1. Coche, bicicleta y a pie

La obtención de rutas para coche, bicicleta y a pie ha sido el caso más fácil, debido a la flexibilidad del API de Google Directions. Éste permite separar las peticiones por cada uno de los 3 medios de transporte. Además, acepta tanto cadenas como coordenadas como parámetro de origen y destino, aunque en TripMinder se ha preferido utilizar coordenadas, ya que es más exacto.

El procesamiento de este tipo de rutas es mínimo, puesto que la información es parecida al formato común utilizado. Los datos de detalle de ruta también tienen un procesamiento mínimo.

8.3.2. Autobús y tren

Para el caso del autobús y el tren, se complica algo más. La obtención se mantiene simple, puesto que GFTS y Google Transit mantienen una API similar a la de directions. El problema reside en el procesamiento. Puesto que la API no permite separar por medio de transporte, los datos obtenidos están todos los medios de transporte mezclados.

El procesamiento, además de en ciertas modificaciones y reestructuraciones, reside en separar los medios de transporte. Además una ruta puede contener diferentes fases, las cuales también se procesan, en las que en cada una se tiene un medio de transporte diferente. La condición de separación establecida es: el medio de transporte de la ruta seleccionado será el primero, de entre tren o autobús, que se encuentre en sus fases. El detalle de ruta en este caso también incluye la inserción de los tiempos de salida y llegada en el formato requerido para la vista.

8.3.3. Avión

La obtención de rutas para avión ha sido el caso más difícil, con diferencia. Con el objetivo de explicar mejor el problema, se ha dividido en las siguientes subsecciones que se explican a continuación.

8.3.3.1. Problema

La API de QPX Express, además de ser limitada a 50 peticiones por día, es la única gratuita, y es la base en la que se basan otras de pago, como la de FlightAware. Pero ese no es el problema.

QPX Express posee muy poca flexibilidad y la forma de tratamiento de datos es muy compleja (sin necesidad) y diferente a todas las demás API's que posee Google, por lo que se puede notar fácilmente que desde que Google compró esta API, no se ha modificado mucho.

El principal problema es que la única forma de pasarle el origen y destino a la API es mediante códigos IATA del aeropuerto. TripMinder utiliza Google Autocomplete para la obtención de un origen y un destino, incluyendo bastantes datos, pero el código IATA del aeropuerto no se encuentra entre ellos. Por tanto, había que obtener los códigos IATA de algún modo.

Para ello, existe alguna fuente de datos, aunque no muchas. Se escogieron las dos más completas de las encontradas. Aun siendo lo mejor encontrado, ambas son algo “sucias”, es decir, tienen cierta cantidad de datos erróneos, incompletos o no referentes a aeropuertos:

- Global Airport Database:
<http://www.partow.net/miscellaneous/airportdatabase/>
- OpenFlights Airport Database:
<http://openflights.org/data.html>

8.3.3.2. Intento 1: búsqueda por coincidencia de cadenas

Empezando con la base de datos de Global Airport Database, la primera idea fue buscar por coincidencia de cadena, es decir, buscar por el nombre de la ciudad exacto o más parecido posible para obtener el código IATA.

Veamos un caso específico para explicar el algoritmo:

Para la ciudad “*San Vicente del Raspeig, Alicante, España*”:

1. Separar cadena por “,” y buscar coincidencia con el primer elemento (en este caso “San Vicente del Raspeig”). Si se encuentra el resultado exacto, ya acaba. Sino se itera al paso 2.
2. En el caso de no haberse encontrado un resultado concreto, se separan todos los elementos por “ ” y se busca la coincidencia de cada palabra por separado (de más de 2 o 3 letras, para evitar “... de ... los ...”). Entonces se añade a una lista la posición de la coincidencia + el número de coincidencias. También se añadirá a la lista el número de letras que tiene la palabra. Para casos como “East London” y “London”. Ambos tienen 1 coincidencia, pero “London” es más exacto porque tiene menos letras.
3. Se selecciona el elemento con el mayor número de coincidencias. En caso de igualdad, se analizará la que menos letras tenga.

Este método conlleva varios **problemas**:

- Puede que la localización no tenga aeropuerto (ej: Agost).
- Problemas relativos al trabajo con cadenas. Diferente idioma, palabras diferentes. Por ejemplo, para “Londres” no habría resultados. Se podría utilizar Bing Translator, aunque ya sería otra llamada más a servicios REST, y persistirían el resto de problemas de cadenas.

- En algunos casos podría fallar. Por ejemplo, para el caso “London”, se tienen varios resultados coincidentes, entre ellos 2 con la palabra exacta “London”, uno de Canadá y otro de Reino Unido. ¿Cuál es el deseado?

```
Object {city: "london", country: "canada", code: "yxu"}
Object {city: "london", country: "united kingdom", code: "lon"}
Object {city: "east london", country: "south africa", code: "els"}
Object {city: "london - luton", country: "united kingdom", code: "ltn"}
```

Ilustración 8.2. Resultados para búsqueda de "London"

- Tiempo de ejecución > 1200ms en PC y > 4000ms en SmartPhone para una búsqueda exhaustiva, además de latencia adicional en el caso de uso de Bing Translate para traducción al inglés.

Debido a todos los problemas anteriores, este algoritmo se descartó, y se empezaron a estudiar otras soluciones.

8.3.3.3. Solución: búsqueda por coordenadas y OpenFlights

La segunda idea fue, ya que se poseen las coordenadas del lugar que se quiere realizar la búsqueda en sí, realizar una búsqueda de aeropuertos más próximos. Esto solventaría todos los problemas anteriores, además de que presentaría varias ventajas:

- Fiabilidad. Siempre se devolverá un aeropuerto. De hecho, se devuelven varios de ellos para así dar a elegir al usuario.
- Optimización. El cálculo con números es mucho más rápido que la comparación de cadenas.

Aun así, existía el problema de que la base de datos Global Airport Database tiene muchas coordenadas erróneas. Entonces se probó con OpenFlights Airport Database. Los resultados eran mejores, aunque seguía habiendo algunos datos erróneos.

Se realizó cierta limpieza en la base de datos, como eliminar lugares sin coordenadas, datos repetidos con códigos IATA erróneos, o incluso estaban incluidas algunas estaciones de autobús y tren en algunos países como Canadá, Alemania o Suecia. Después de esto la base de datos, no ha quedado perfecta, pero aceptable.

Una vez solventado esto, llega el momento de calcular los aeropuertos más cercanos a una coordenada concreta. Para ello existen varios métodos:

Para todos ellos: $R = 6371$ (radio de la tierra en km)

- **Harversine fórmula:** la más exacta, a la vez que más costosa. Utiliza geometría esférica y es la más apropiada para distancias basadas en latitud y longitud terrestre. Propuesta por el U.S. Census Bureau.

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos \varphi_1 \times \cos \varphi_2 \times \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \times \text{atan}(\sqrt{a}, \sqrt{1-a})$$

$$d = R \times c$$

- **Ley esférica del coseno:** más simple y menos costosa que la anterior. La falta de precisión no es muy perceptible, perfecta cuando no es importante una falta de precision de pocos metros.

$$d = \text{acos}(\sin \varphi_1 \times \sin \varphi_2 + \cos \varphi_1 \times \cos \varphi_2 \times \cos \Delta\lambda) \times R$$

- **Aproximación equirectangular:** mucho más simple y menos costosa, pero carece de precisión. Útil para cortas distancias.

$$x = \Delta\lambda \times \cos \varphi_m$$

$$y = \Delta\varphi$$

$$d = R \times (x^2 + y^2)$$

Se ha realizado un estudio para la comparativa entre estos métodos. Para el ejemplo, se ha realizado la búsqueda de aeropuertos para el input “Agost, España”. Los resultados son un promedio de múltiples pruebas.

PC = Macbook Pro, 2.4 GHz Intel Core i5 2 cores , 8 GB RAM 1600 MHz DDR3, Chrome 42.0.2311.90 (64-bit)

SP (SmartPhone) = Jiayu S3, 1.5GHz ARMv7 4 cores, 1GB RAM

	Tiempo (ms)	Precisión (0-100)	Feedback
Harversine	130ms PC 1800ms SP	100	Resultados correctos.
Ley esférica	120ms PC 1680ms SP	97	Resultados correctos. Distancias difieren en los últimos decimales de un número con 16 decimales, por lo que es casi igual de preciso que Harversine.
Equirectangular	100ms PC 1200ms SP	40	Los 2 aeropuertos más cercanos son correctos. A partir del 3º, eran aeropuertos de EEUU. Conclusión: funciona bien sólo para distancias cortas. No fiable.

Finalmente, se ha decidido por utilizar la Ley esférica del coseno, puesto que la diferencia de precision es inapreciable, y es algo más rápida en cuanto a tiempo de computación.

9. Diseño gráfico

En esta parte se estudia la manera en que se diseña y estructura la aplicación con el objetivo de proporcionar una interfaz clara y usable. En ella se desarrollarán los conceptos de UI y UX design, se aplicará el principio de "*Interfaz auto-descriptiva*" y se planteará un diseño basado en el uso de las fuentes de iconos y el "*Diseño plano*".

Posteriormente, se especificará la paleta de colores a utilizar, se propondrá una guía de estilo y se adjuntarán los Mockups o bocetos referentes a las diversas pantallas de la aplicación, visibles para diferentes dispositivos.

9.1. Conceptos previos

9.1.1.1. UI y UX design

Estos conceptos tienen cosas en común, por lo que se suelen confundir. Ambos están relacionados con el proceso de diseño, incluyendo todo lo que éste significa. Los conocimientos de los roles de un diseñador UI y un diseñador UX tienen una base común, en la que el objetivo es realizar un producto final que sea agradable para el usuario. Sin embargo, cada uno trabaja en ese proceso de una forma diferente, y sobre todo con un punto de vista diferente. Para mostrar esto, se van a describir los conceptos de UI y UX design.

UX (User eXperience) design es el proceso que tiene como objetivo conseguir la mayor satisfacción de un cliente para un producto específico. Durante el proceso, un diseñador UX trabaja en las partes relativas a la interfaz gráfica, la interacción hombre-máquina y la usabilidad. Su función y preocupación principal es la de comprobar que la navegación en un producto fluye de forma que un usuario no tendría ningún problema en utilizarlo.

Por tanto, el rol de diseñador UX comprende un amplio rango de actividades y conocimientos utilizados en el diseño de un producto, en el que su punto de vista está

enfocado al uso de un producto por un usuario. Se podría resumir en que un diseñador UX realiza algunas funcionalidades de un diseñador UI, un diseñador de Interacción y las suyas propias, como puede ser la de realizar pruebas en diferentes usuarios.

UI (User Interface) design es el proceso de diseño de interfaces, ya sean para webs, aplicaciones móviles o para un panel de una máquina industrial, en el que se tiene como objetivo conseguir una gran experiencia de usuario e interacción.

Hasta aquí, no parece muy diferente de un diseñador UX. Sin embargo, la gran diferencia se encuentra en que un diseñador UX está presente en todo el proceso de diseño de un producto, mientras que el diseñador UI sólo para la parte de la interfaz del producto. Por ese mismo motivo, el punto de vista de un diseñador de UI es el de crear una interfaz usable y de buen diseño, pero no se preocupa por el modo en el que un usuario lo utilizará cuando se realice. Se puede decir que un diseñador UI sigue las pautas establecidas por el diseñador UX.

En la siguiente infografía se puede percibir de una manera más conceptual las funciones y características que tienen estos roles:

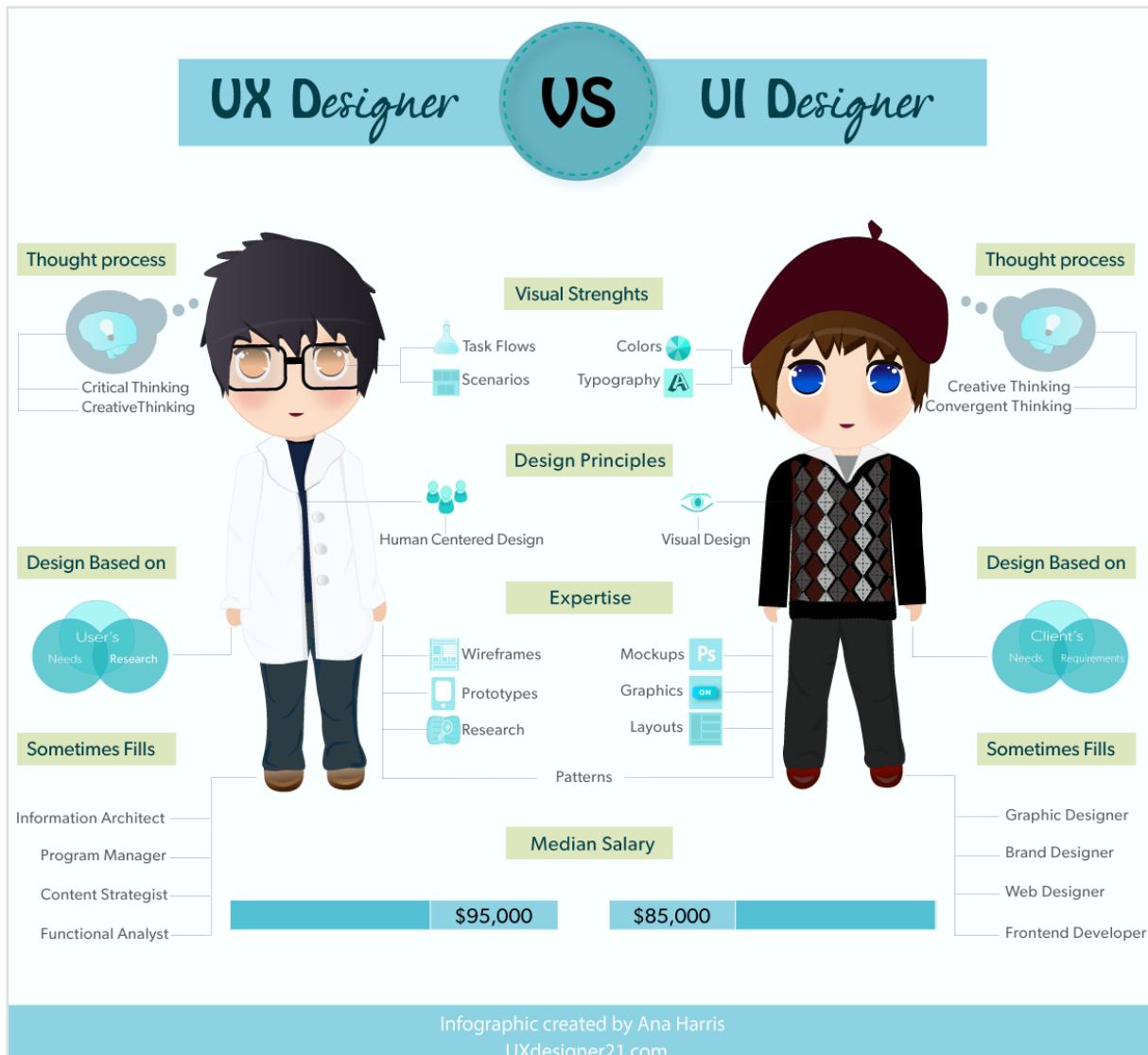


Figura 9.1. Comparación entre UX y UI designer

Fuente: uxdesigner21.com

9.1.1.2. Interfaz auto-descriptiva

Interfaz donde los elementos que las componen y sus partes se describen mediante el diseño y la interacción. Si el concepto de auto-descriptivo significa que se describe a si mismo, que se dé a conocer, una interfaz auto-descriptiva define que los componentes

deben tener expresividad, de tal forma que el usuario debe entender para qué sirve y de qué manera se utiliza.

Los elementos que componen estas interfaces suelen describirse mediante el uso de:

- **Iconos:** mediante ellos podemos describir acciones que no requieren del conocimiento de un idioma en específico. No todos son entendibles, ya que depende de la experiencia de cada usuario, pero por ejemplo, casi todo el mundo sabe que el icono “+” significa sumar, añadir o incrementar.
- **Colores:** al igual que los iconos, los colores aportan información independiente del idioma, pero interpretable de diferentes maneras. Por ejemplo, un botón rojo en una web da la sensación de peligro y se utiliza para cerrar, eliminar o borrar un elemento.
- **Animaciones:** usadas de un modo correcto, pueden provocar insinuaciones al usuario. Por ejemplo, una animación de un elemento intermitente llama la atención de un usuario, o un elemento que se está agitando puede insinuar que se puede mover.
- **Presentar elementos ocultos:** a partir de una determinada acción, se muestran elementos que aportan significado adicional al elemento accionado. En el ámbito web o de aplicaciones, ejemplos de estos son los *tooltips*, *popovers* y ventanas *modales*.

9.2. Diseño previo

9.2.1. Guía de estilo

Una guía de estilo es un documento que especifica el estilo o formato estándar de una aplicación. En ella se establecen las directrices comunes para el diseño de elementos, fuentes a utilizar, formato de las imágenes, etc. Esto se considera una actividad clave en el ámbito empresarial para la creación de aplicaciones de cualquier tamaño, ya que permite:

- Presentar al cliente, superior o supervisor un documento (la guía de estilo), donde podrá ver cómo se van a diseñar los elementos. Así, se puede especificar mediante varias iteraciones el diseño en una fase previa, sin tener que recurrir a la modificación continua de código.
- Cambiar el personal sin preocupación de dedicar tiempos al estudio de la plataforma y su diseño. Además, es sencillo para los nuevos programadores/diseñadores asimilar el modo de trabajar

Una guía de estilos completa, debe incluir los siguientes apartados:

- Reglas generales de estructuración y diseño
- Paleta de colores
- Tipografía
- Cuadrícula
- Navegación
- Componentes UI y UX

9.2.2. Reglas generales

En la guía de estilo de esta aplicación, se aplicarán los principales patrones referentes a los dispositivos en los que se va a utilizar.

Para el diseño móvil suele referirse a evitar elementos en varias columnas, a menos que sea a modo de grid o galería de imágenes donde no aparece mucho texto. También define utilizar elementos de un tamaño mínimo, ya que la interacción táctil no es tan precisa como la del puntero de un ordenador. El modo de acción de las pantallas “*Vista->Detalle*”, en el móvil se ha de separar, de tal modo que cuando se pulsa en un item de una vista, se abre otra con el detalle. En tablets o PC, se divide en un sidebar en el lateral izquierdo, más o menos del 25% del tamaño del ancho, y el resto de espacio se utiliza para presentar el detalle.

9.2.3. Paleta de colores

En el mundo del diseño en general, especialmente en el de interfaces de usuario, una paleta de colores no es algo obvio, que se pueda utilizar sin un estudio previo, ya que el diseño tiene relación directa con el significado semántico de una aplicación y de lo que se quiere transmitir.

Además, no puede formarse una paleta de colores con cualquier combinación de colores. Hay una ciencia y teoría de colores directamente detrás de una paleta de colores. Las paletas de colores pueden ser combinaciones de tonos del mismo color, llamadas “paletas tonales”, o colores diferentes dados un color origen, los cuáles son calculados mediante operaciones matemáticas de colores. Estos serían las *tríadas* y *tétradas* de colores, o también los *complementarios*. Si se siguen estas especificaciones a la hora de crear una paleta, se asegura una buena combinación de colores.

Para demostrar esta teoría, existen herramientas web llamadas “*Ruedas Cromáticas*”. A continuación se detallarán los tipos más importantes de paletas de colores, utilizando la Rueda Cromática que nos ofrece Adobe, llamada Adobe Color CC (anteriormente conocida como Adobe Kuler).

<https://color.adobe.com/es/create/color-wheel/>

9.2.3.1. Análoga

Los colores análogos son aquellos que, tomando un color base, son calculados variando un pequeño porcentaje su tonalidad, y aumentando mínimamente los valores de la saturación. Ese cambio de tonalidad, se realiza a modo de “*abanico abierto*” usando el modelo de color HSV.

Su uso puede ser de utilidad cuando se persigue distinguir algunos elementos de otros (por ejemplo, un bloque de un botón) de forma que no se comparta exactamente el mismo color.

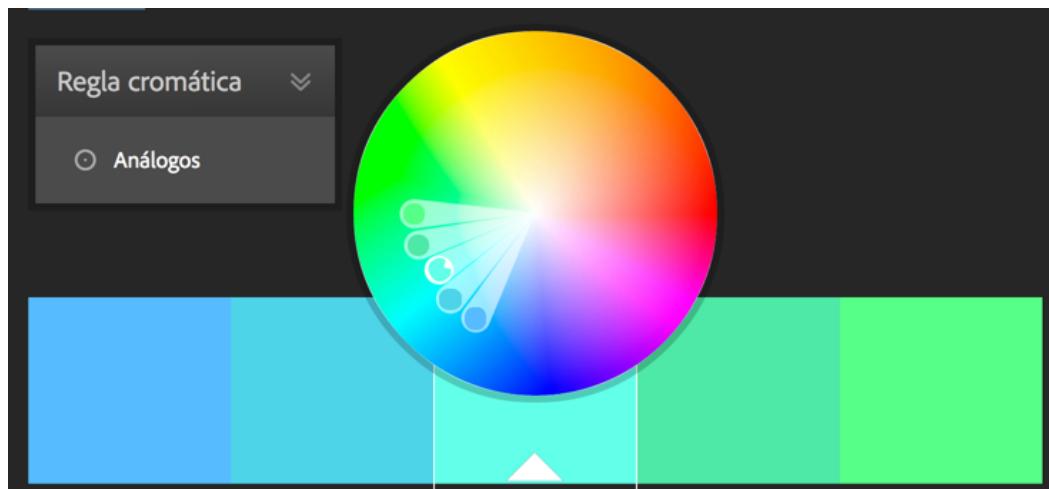


Figura 9.2. Paleta análoga

9.2.3.2. Monocromática

Los colores monocromáticos mantienen su tonalidad intacta, modificando únicamente los valores de brillo y saturación.

Este tipo es muy utilizado para grupos de elementos, donde se quiere destacar uno o varios de ellos oscureciendo o aclarando su color. Un ejemplo de esto, son las barras de navegación web, donde se marca la página dónde se encuentra el usuario.

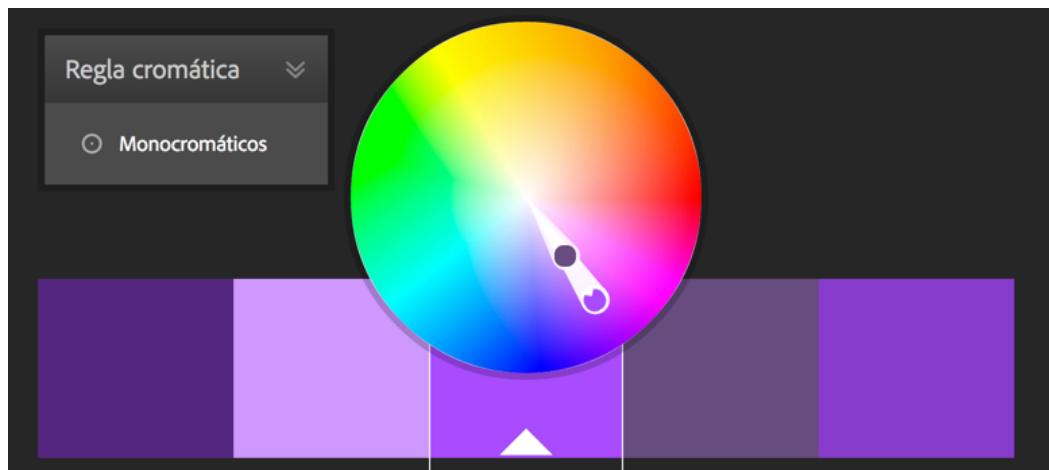


Figura 9.3. Paleta monocromática

9.2.3.3. Complementaria

Las paletas de colores complementarios se componen de un color y su complementario. Ésta se suele combinar con la paleta monocroma, y así se obtienen variaciones de esos dos colores.

Posiblemente, la complementaria es la más utilizada en interfaces, donde se le asigna a los elementos principales un color, y a los secundarios o “helpers” el complementario.

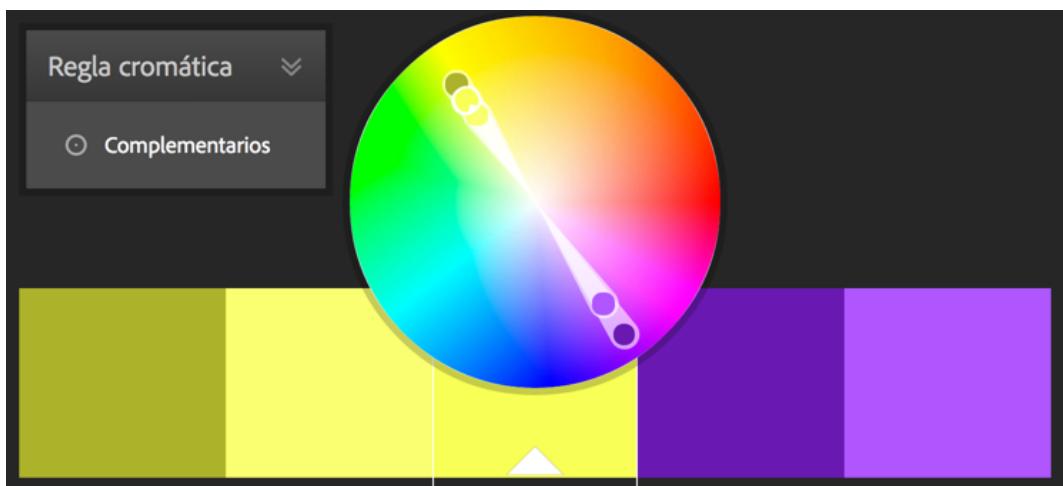


Figura 9.4. Paleta complementaria

9.2.3.4. Tríada

Se basa en la misma idea que la complementaria. De hecho, se genera mediante la misma teoría, pero aplicada a 3 colores.

Una tríada es menos usada, ya que el encaje de más colores es más costoso. No obstante, en aplicaciones que requieran multi-color, por ejemplo para expresar un estado de diversión, se puede utilizar sin problema.

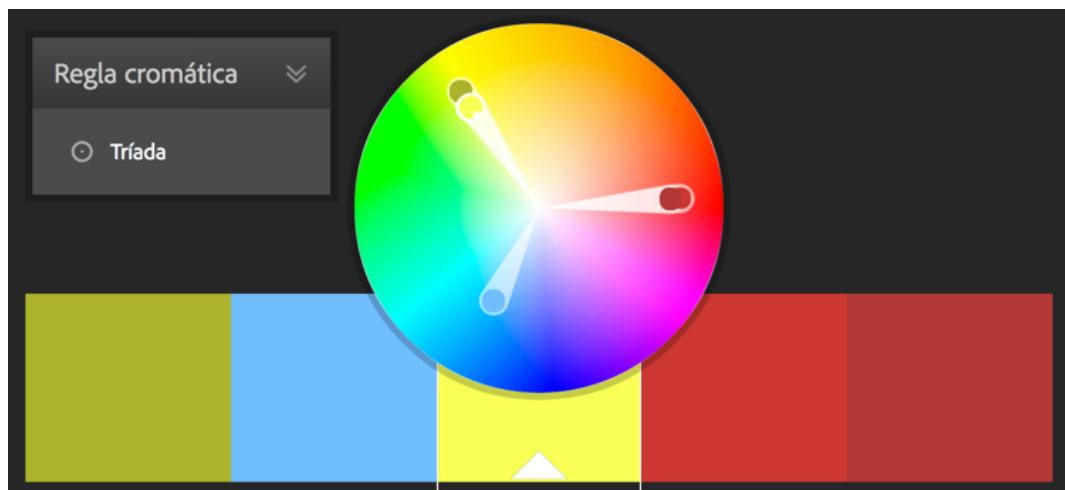


Figura 9.5. Triada

9.2.3.5. Compuesta

Una paleta compuesta combina las características de todas las anteriores, creando una paleta que varía en tono, saturación y brillo. El uso de esta paleta, aunque parezca más complicado que el de la tríada, no lo es, ya que los colores no son tan diversos (realmente, es más parecida a la complementaria).

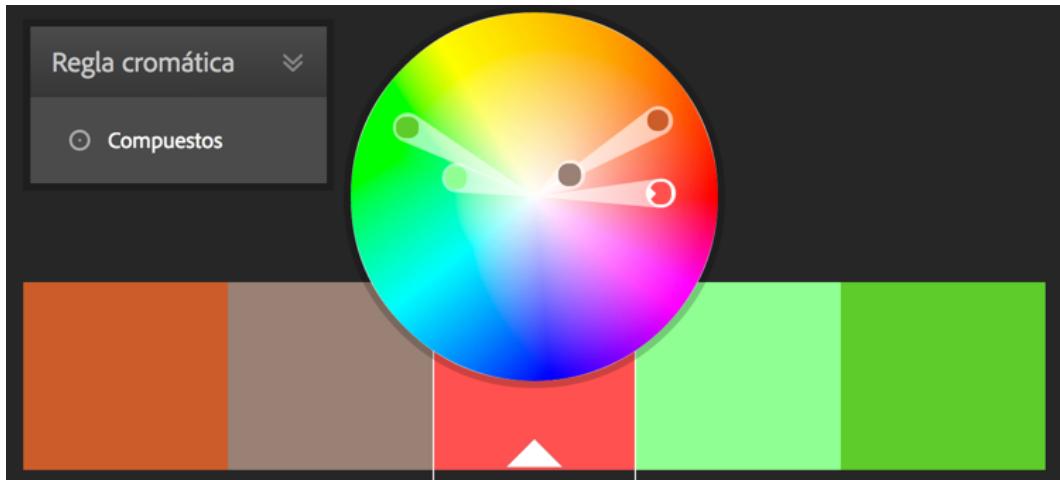


Figura 9.6. Paleta compuesta

9.2.3.6. Conclusión

El objetivo que se persigue es crear una aplicación con una interfaz limpia, y por tanto no se busca la combinación de muchos colores. Aunque una paleta análoga o monocroma puede dar un buen resultado, se apostará por una complementaria, donde el color complementario se aplicará a elementos secundarios, como botones o mensajes flotantes.

La paleta será la siguiente:

<https://color.adobe.com/es/create/color-wheel/?base=2&rule=Complementary&selected=2&name=Mi%20tema%20de%20Color&mode=rgb&rgbvalues=0.03672694865846518,0.2881461639010501,0.7,0.22527374340206374,0.5189402312349627,1,0.12527374340206376,0.45684607027577173,1,0.7,0.5369378991077769,0.20081313514725788,1,0.7142659169356833,0.12527374340206376&swatchOrder=0,1,2,3,4>

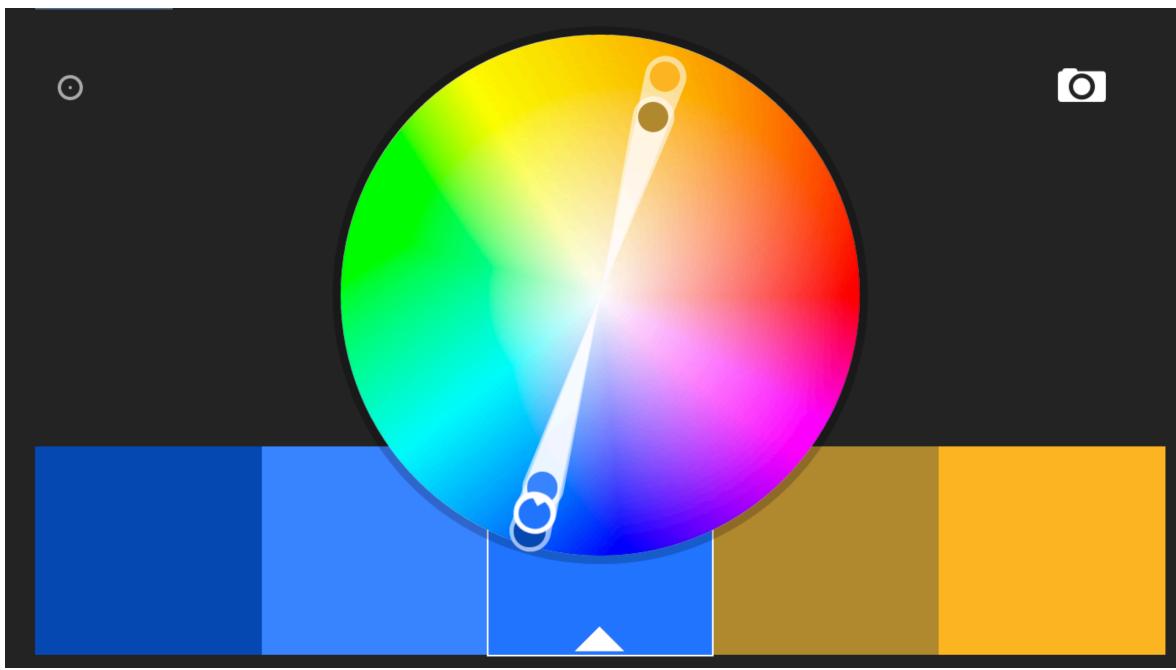


Figura 9.7. Paleta de TripMinder

9.2.4. Tipografía

En un principio, se utilizará la tipografía que el sistema operativo provee. Es la que más usabilidad da, ya que es la que los usuarios de cada sistema están acostumbrados a usarla.

9.2.5. Cuadrícula

La cuadrícula de la aplicación irá acorde a lo descrito anteriormente sobre el modo de presentación de las pantallas “*Vista->Detalle*”. Ésta se puede apreciar de un modo muy notable posteriormente en la sección de Mockups.

9.2.6. Navegación

En el caso de la navegación se ha aprovechado la utilidad que nos ofrece la herramienta utilizada para la creación de Mockups, que es la de JustInMind.

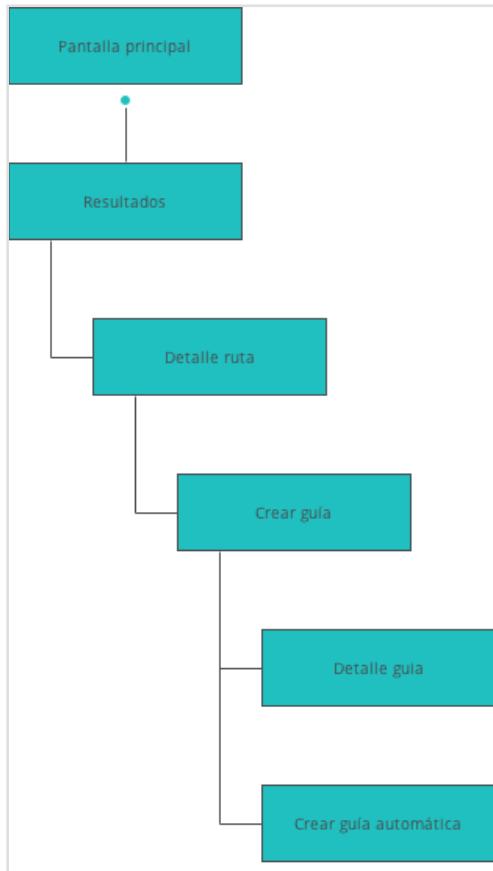


Figura 9.8. Pantallas de navegación de TripMinder

9.2.7. Componentes UI y UX

Los componentes UI y UX refieren al uso de elementos de diseño específicos que se utilizan para componer la experiencia de usuario establecida y la interfaz como parte de ella. Estos componentes suelen ser elementos físicos o virtuales que podrán tener múltiples formas dependiendo del tipo de UX y UI que se esté desarrollando.

En el ámbito del desarrollo de aplicaciones, éstos suelen ser botones, paneles, alertas, listas, etc. En la siguiente imagen, extraída de un creador de UI webs basadas en el framework Bootstrap llamado *Jetstrap*, podemos ver en el panel de la izquierda una muestra de componentes UI y UX.

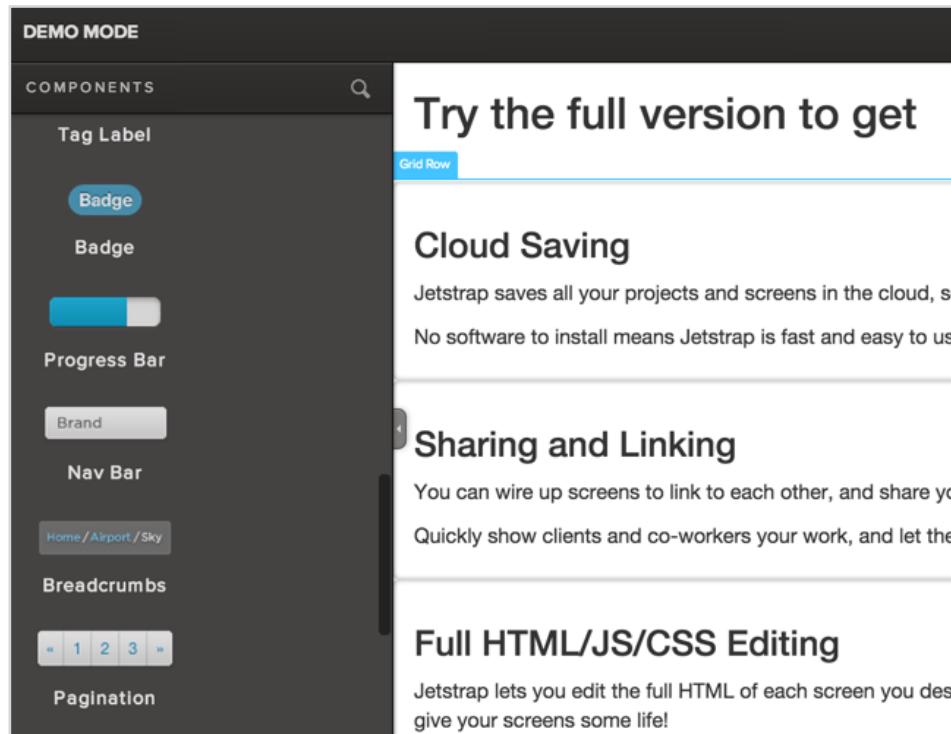


Figura 9.9. Ejemplo de componentes UI

Fuente: Demo en la web de Jetstrap

A la hora de crear los componentes UI y UX de una aplicación, se deben de definir diversas características que deben cumplir estos componentes, llamadas *reglas de estilo*. Mediante estas establecemos directivas y directrices que se deben seguir para crear un *estilo o tema*, que no es más que un conjunto de reglas de estilo agrupadas con el objetivo de transmitir una sensación gráfica común.

Según la cantidad de blancos que tiene un estilo, se pueden dividir en:

- **Dark UI:** caracterizada por tener contenedores y grandes elementos coloreados en grises oscuros y negro, con fuentes y elementos pequeños blancos o grises claros. Se suele utilizar en programas de diseño y desarrollo, como las últimas versiones de Adobe CC. En el ámbito web, es más común verlas en webs de moda y de diseñadores gráficos.

- **Light UI:** las características son opuestas a las de *Dark UI*. Es utilizada las demás webs y aplicaciones en general, como landing pages y webs corporativas.

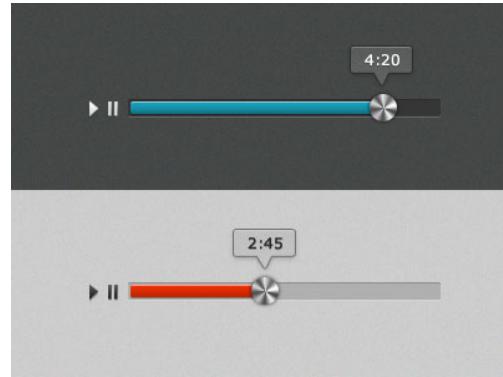


Figura 9.10. Ejemplo de dark y light UI

Fuente: UI cloud

En cuanto al modo de representación del color, existen 2 estilos principales: **Gradient Design** y **Flat Design**. Actualmente, la tendencia en las aplicaciones, tanto de escritorio y móvil como web, el Flat Design es el que está siendo más utilizado.

Normalmente, el uso de uno u otro dependerá del tipo de aplicación que se vaya a desarrollar, pero por norma general está demostrado que el Flat Design proporciona una sensación más limpia y minimalista a la interfaz. Con el diseño gradiente hay que tener cuidado, ya que el sobreuso de colores gradientes puede dar lugar a diseños sobrecargados y diseños anticuados. Esto se puede percibir si se compara el estilo de las nuevas interfaces con las antiguas.

A continuación se puede apreciar un ejemplo de dos botones en los dos estilos mencionados:

Gradient Design

Flat Design



Cuando se establece un estilo determinado para una aplicación, o cualquier otra cosa en general, se crean los componentes que se van a utilizar. Esta agrupación de componentes se denomina un “*UI kit*”. Un ejemplo de un UI kit del tipo *Light Flat Design*, es el siguiente:

Summer UI Kit

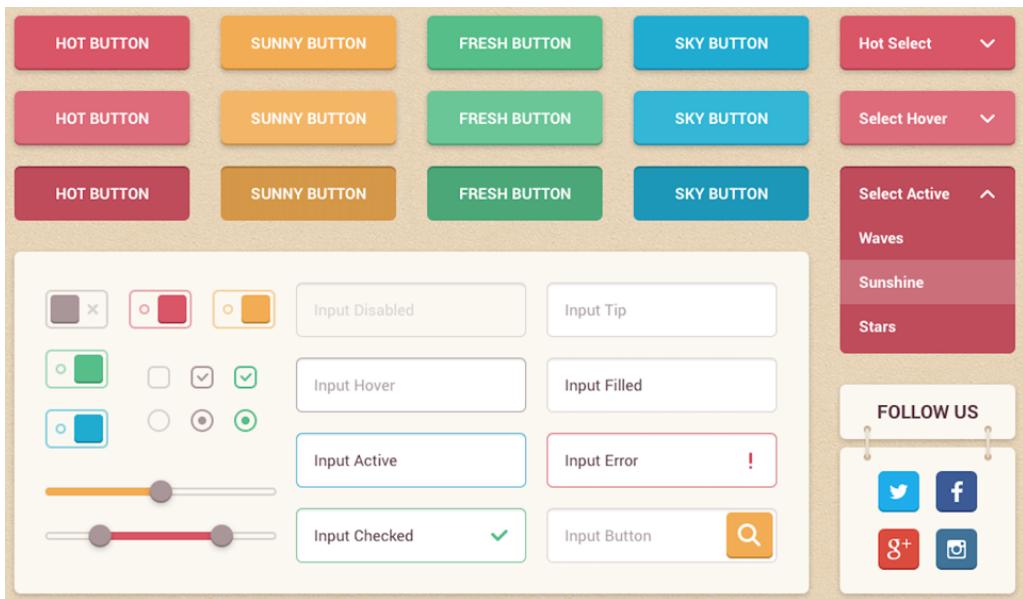


Figura 9.11. Summer UI Kit, ejemplo de UI con estilo Flat Design

Fuente: UI cloud

Para concluir, en el caso de TripMinder se utilizará un UI Kit similar al anterior, donde se utilizará el Flat Design y el Light UI como conceptos de diseño de la aplicación. Se ha escogido estos por la adecuación a una interfaz móvil simple y clara que se busca en la creación de esta aplicación moderna multiplataforma.

9.2.8. Bocetos (Mockups)

Los mockups o bocetos son un elemento clave para el desarrollo de una aplicación. Mediante ellos, se puede definir de una forma más rápida y directa la presentación y estructuración aproximada del sistema en cuestión. Estos no suponen un diseño final, es un diseño aproximado en el que se pondrá la principal atención a la estructura, pero atributos como los colores de los elementos o variaciones en tamaño, podrían cambiar.

A continuación, se presentan por separado las pantallas en las dimensiones de los dispositivos móviles y tablets y ordenadores.

9.2.8.1. Móvil

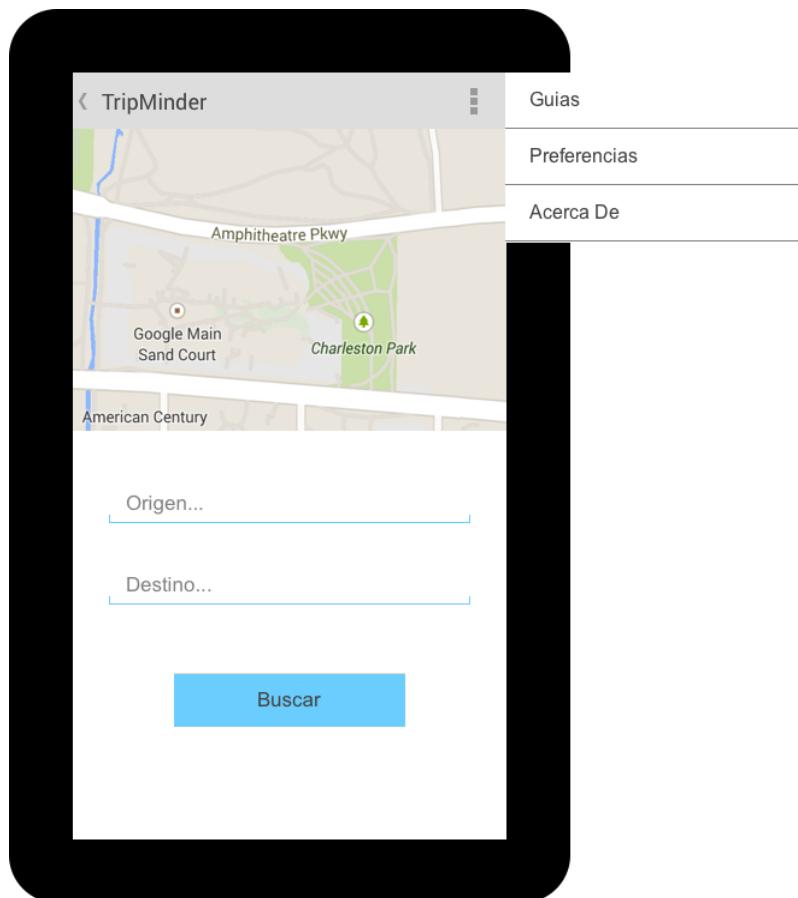


Figura 9.12. Pantalla principal

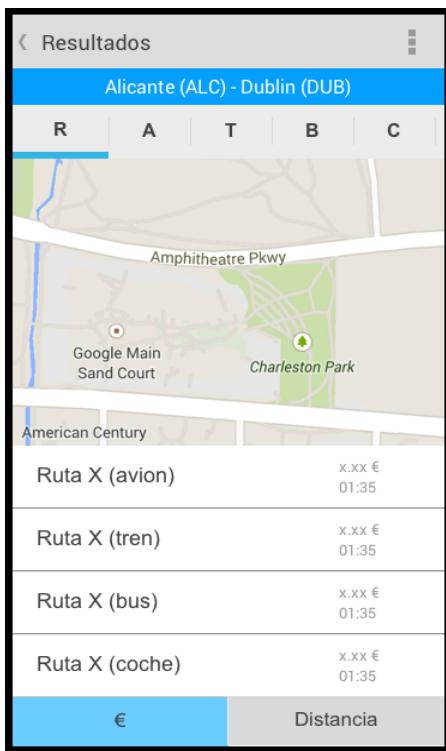


Figura 9.13. Resultados



Figura 9.15. Crear guía turística

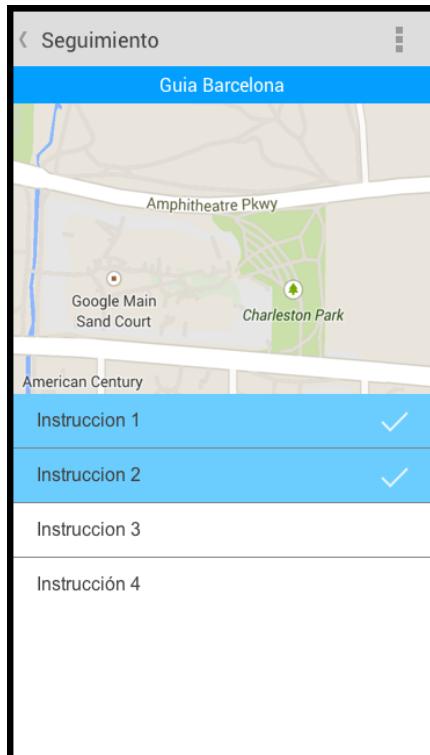


Figura 9.14. Seguimiento

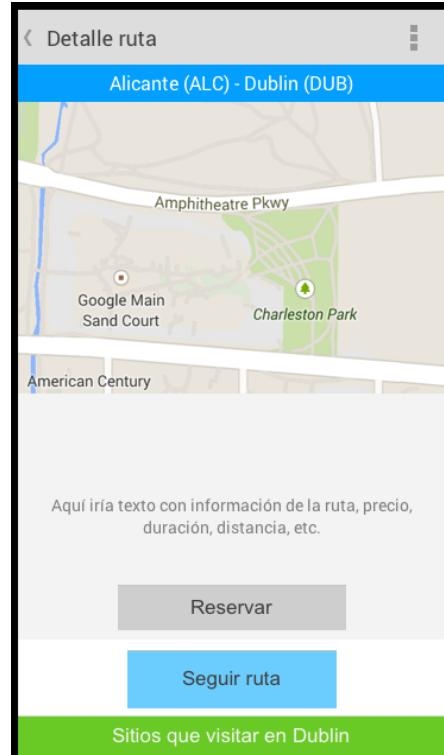


Figura 9.16. Detalle de ruta

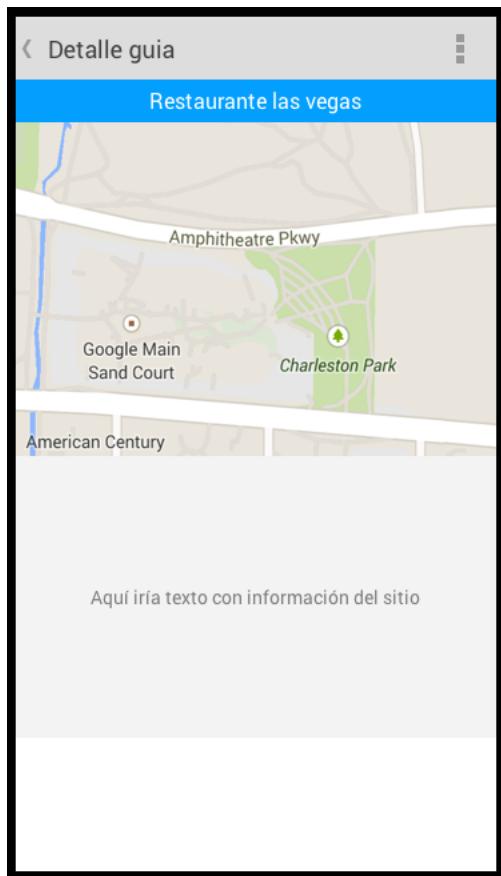


Figura 9.17. Detalle de guía

9.2.8.2. Tablet & PC

En este caso, se hará al modo Vista -> Detalle, en el que a la izquierda aparece la vista y a la derecha el detalle. En las siguientes imágenes se ve una muestra de cómo será en 2 pantallas:

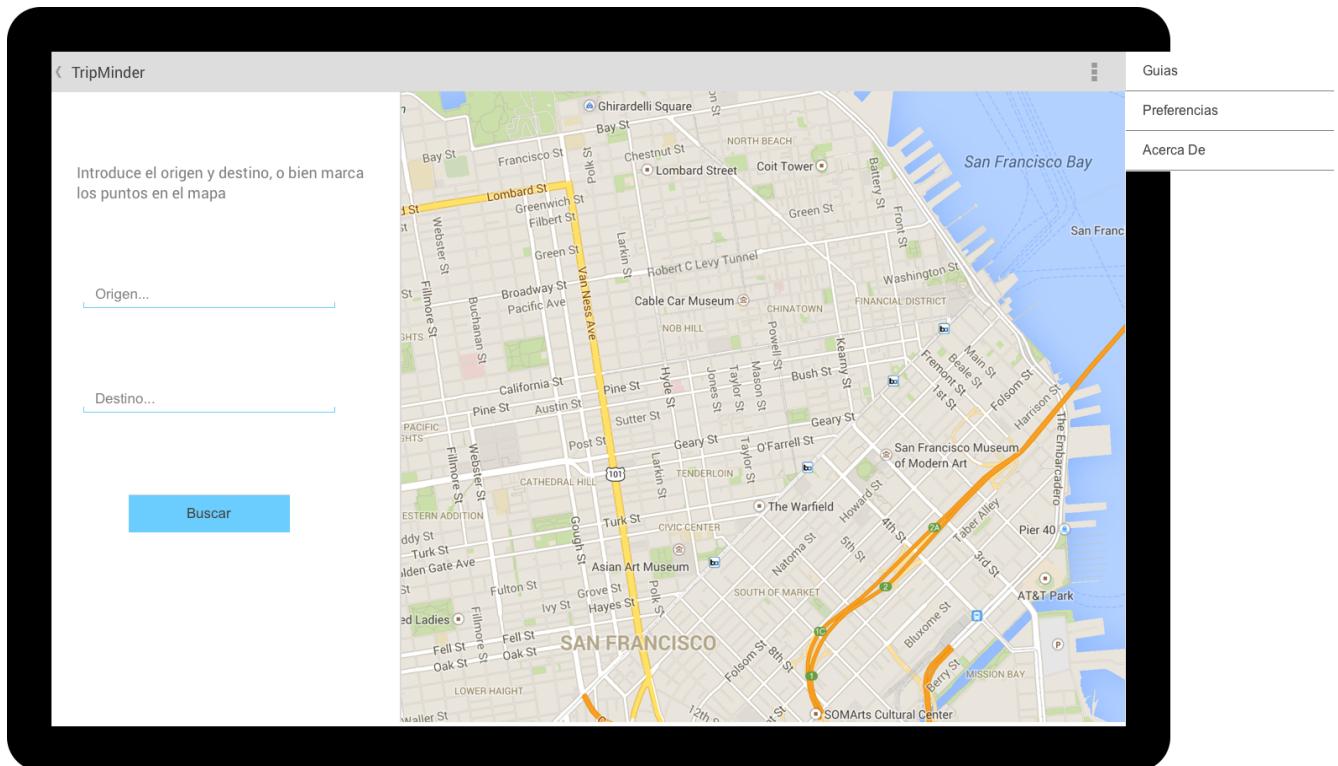


Figura 9.18. Pantalla principal en tablet

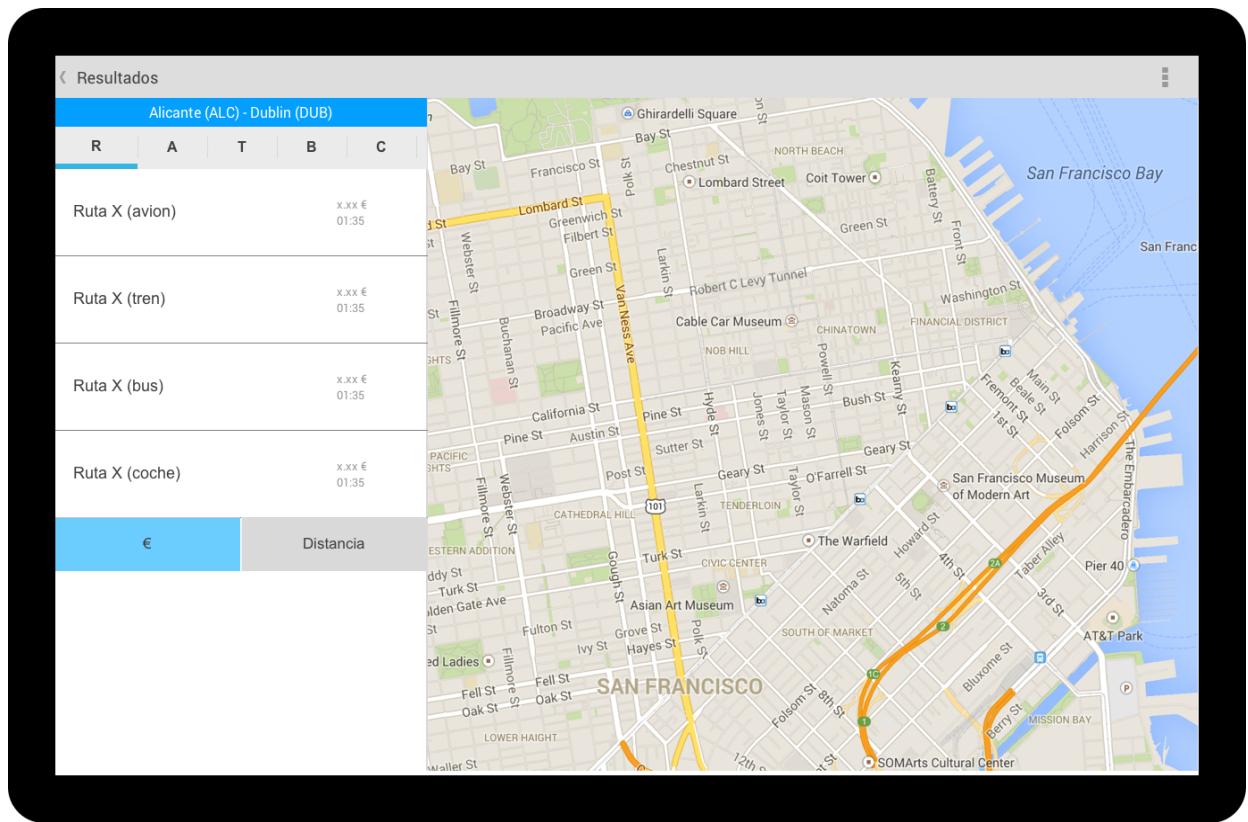


Figura 9.19. Resultado de búsqueda en tablet

9.3. Diseño final y resultado

Durante y tras al desarrollo de la aplicación, algunos aspectos han cambiado, aunque algunos se han mantenido. Esto es principalmente debido al cambio de tecnología producido.

Afortunadamente, la mayoría de estos cambios han sucedido para mejor, siendo Ionic la tecnología finalmente utilizada, facilita el diseño de interfaces modernas y de gran experiencia de usuario.

Se ha preservado:

- Paleta de colores.
- Utilización de mapas y pestañas en presentación de resultados.
- Pantallas básicas planificadas a desarrollar.

Los siguientes aspectos han cambiado:

- Estructuración de las pantallas.
- Diseño y maquetación de estas. Ahora es totalmente responsive, por tanto, está adaptado a todo tipo de pantallas.
- Uso de icon-font para aumentar la experiencia de usuario mediante mejor localización de zonas.
- Elementos UI personalizados.

9.3.1. Logotipo y splash screen

Se han diseñado ambos logotipo y “splash screen” para el proyecto. Splash screen es la pantalla que se muestra cuando se abre una aplicación mientras se cargan los contenidos.

Su diseño está sujeto al color de la paleta de colores y diseño plano. Con este logotipo se quiere plasmar el sentimiento de viajar, con el resultado de una combinación de iconos que incluyen una maleta y una bola del mundo.



Figura 9.20. Logotipo de TripMinder

Como se puede ver, el ícono no presenta bordes redondeados ni ningún tipo de acabado en los bordes. Esto es así porque el cliente de comandos de Ionic tiene la habilidad de generar automáticamente los íconos para todas las plataformas, simplemente con situar en la carpeta *resources* los ficheros “*icon.png*” y “*splash.png*” y ejecutar el comando “*ionic resources*”.

Además, posteriormente cada plataforma aplica sus propios estilos. Por ejemplo, iOS 7 y 8 redondean los íconos.

Respecto al splash screen, se añade el nombre de la aplicación, autor y versión.



Figura 9.21. Splash screen de TripMinder

Se puede apreciar que el diseño es totalmente cuadrado. Se debe hacer así, ya que la herramienta de generación de recursos de Ionic se encargará automáticamente de recortar la imagen para adecuarla a las diferentes pantallas y plataformas. Para ello además se debe dejar un 15% de margen en todos los lados.

9.3.2. Diseño de pantallas

A continuación se expondrán las pantallas o páginas de las que se compone la aplicación.

9.3.2.1. Inicio

Desde el inicio se puede buscar los orígenes y destinos, tanto por texto como haciendo doble click en el mapa. Además se puede acceder al menú principal de la aplicación.

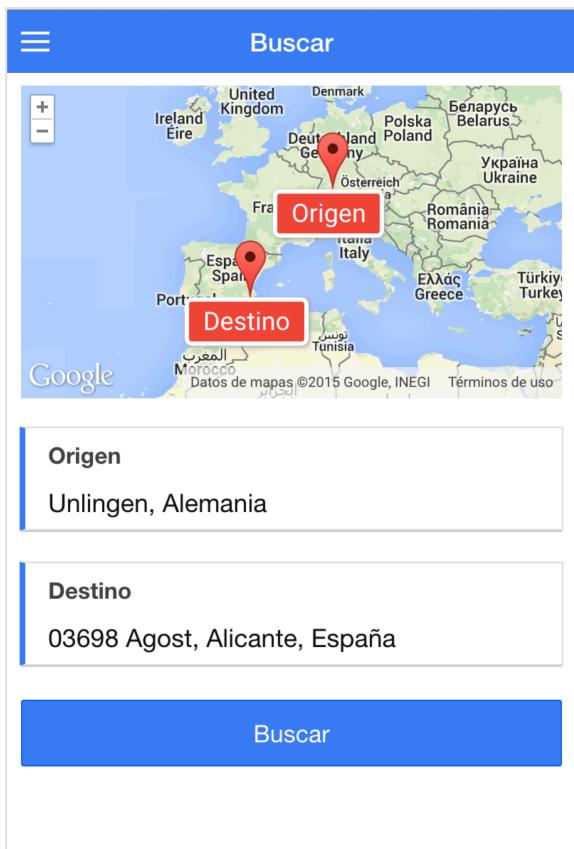


Figura 9.22. Inicio

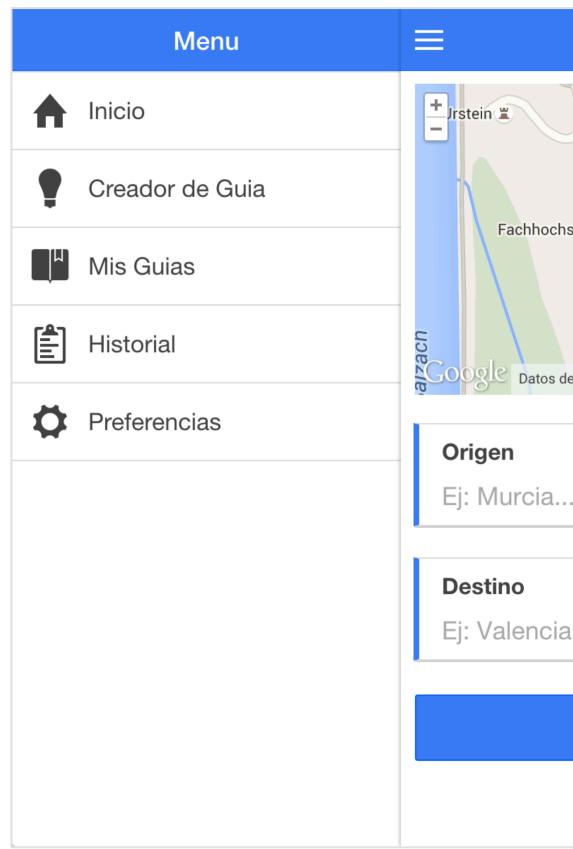


Figura 9.23. Menú lateral

Además, cuando se pulsa el botón buscar, se muestra una ventana modal interactiva donde el usuario puede ver el progreso en la búsqueda de rutas.

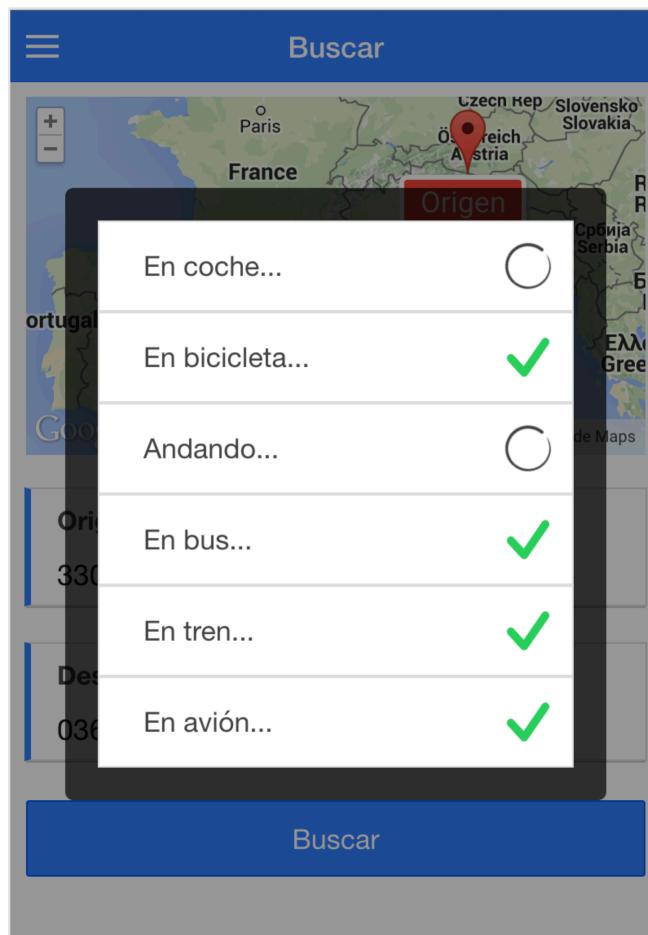


Figura 9.24. Ventana de progreso de búsqueda

9.3.2.2. Resultados de búsqueda

Aquí se muestran los resultados de búsqueda, separados por pestañas.

Para el caso del tren y autobús, la API de GTFS no distingue entre dichos transportes, dado que las rutas pueden combinar diferentes tipos de transporte. Para su clasificación en el procesado de los resultados, se clasifican según el primer transporte público que se encuentre sobre esa ruta. Por ejemplo, si una ruta se compone de un trayecto partido en 3 partes “Tren-Coche-Autobús”, el resultado se clasificará en la pestaña de Tren.

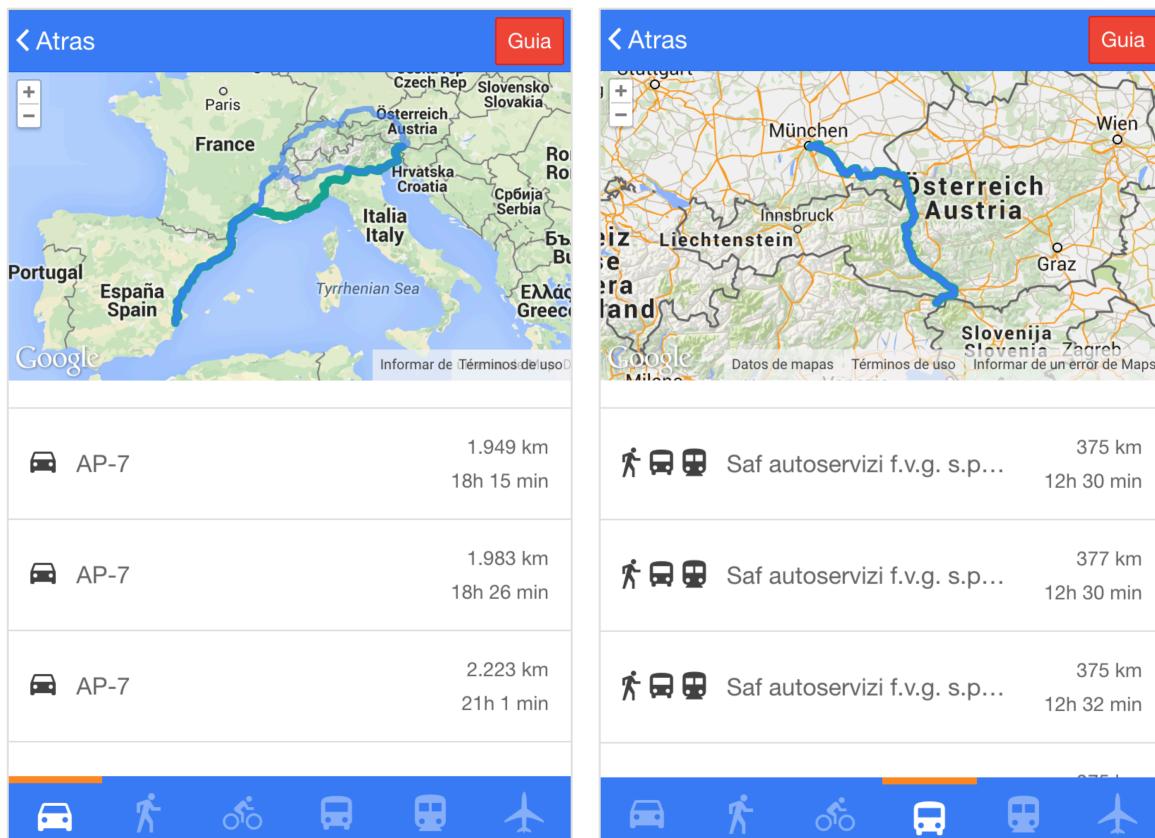


Figura 9.25. Resultados de búsqueda

Para el caso de vuelo, ha sido algo más complicado debido a los problemas relativos a la obtención de los datos. Se puede encontrar más información sobre esto en la sección 8.3.3. La mejor solución encontrada ha sido proporcionar otras opciones al usuario, como seleccionar otro aeropuerto o cambiar fechas.

Screenshot 1: Map and Flight Options

A map showing routes from Alicante (Spain) to Klagenfurt (Austria). The departure point 'Alicante - ALC' and arrival point 'Klagenfurt - KLU' are highlighted with red boxes. The map also shows France, Italy, and Greece.

Vuelo	Precio	Tiempo
Klagenfurt - Alicante	1124.74 EUR	14 h 0 min
Klagenfurt - Alicante	1408.61 EUR	18 h 20 min

Screenshot 2: Itinerary Creation Form

A form titled 'Crea tu itinerario' (Create your itinerary) with the following fields:

- Origen: KLU - Klagenfurt
- Destino: ALC - Alicante
- Fecha: 14/06/2015

A blue 'Buscar' (Search) button is located below the form.

Figura 9.26. Vuelos y pantalla de otras opciones

9.3.2.3. Detalles de ruta

Cuando el usuario pulsa sobre una ruta en la pantalla de resultados, se accede a la vista de detalle de ruta, donde se presenta más información sobre la ruta, además de instrucciones de seguimiento.

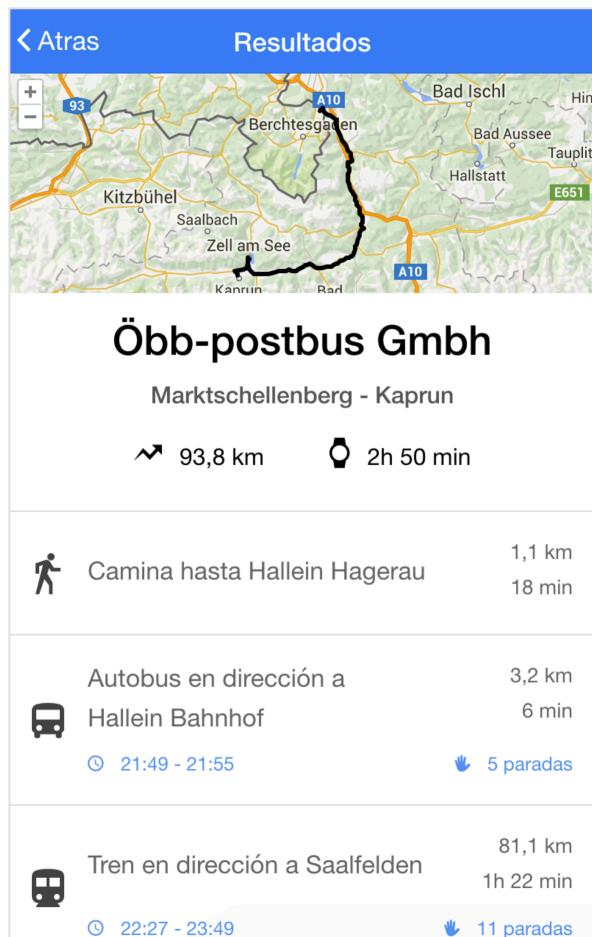


Figura 9.27. Detalles de ruta de transporte público

9.3.2.4. Selección de Guía

Accesible desde el botón rojo que aparece a la derecha de la barra de navegación en las imágenes de resultado de ruta, además de desde el creador de ruta. Se mostrará un catálogo de lugares según las preferencias seleccionadas por el usuario. Para un fácil acceso a estas se añade un botón a la derecha de la barra de navegación.

Cuando se selecciona algún lugar, aparece un botón de crear guía como se puede ver en la imagen de la derecha:

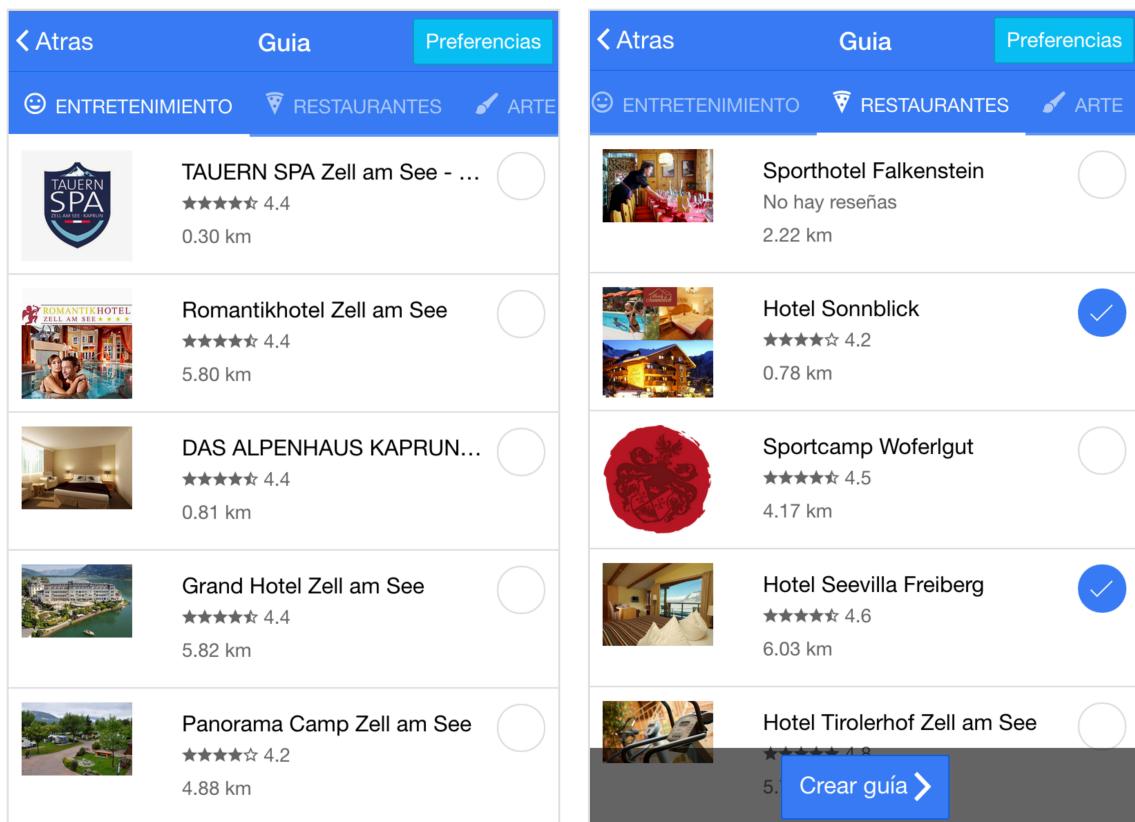


Figura 9.28. Selección de Guía

En caso de que no se tengan ninguna categoría de intereses en las preferencias, un mensaje con un enlace a las preferencias aparecerá.

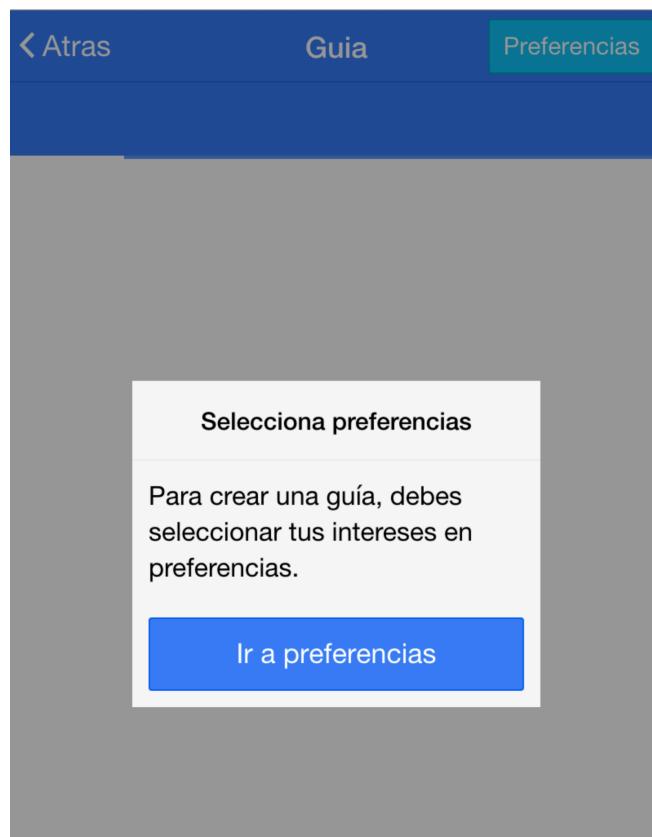


Figura 9.29. Mensaje con enlace a preferencias

9.3.2.5. Creación y guardado de guía

Al pulsar el botón de crear guía, se accede a esta pantalla, dónde se pueden ver los lugares que se han seleccionado además de un mapa adaptado a los marcadores de las coordenadas de esos lugares. Además, al pulsar el botón “Guardar”, se guardará la guía para que esté disponible de modo offline (aunque las imágenes o mapa no aparecerán) y se redirigirá a la pantalla “Mis guías”.

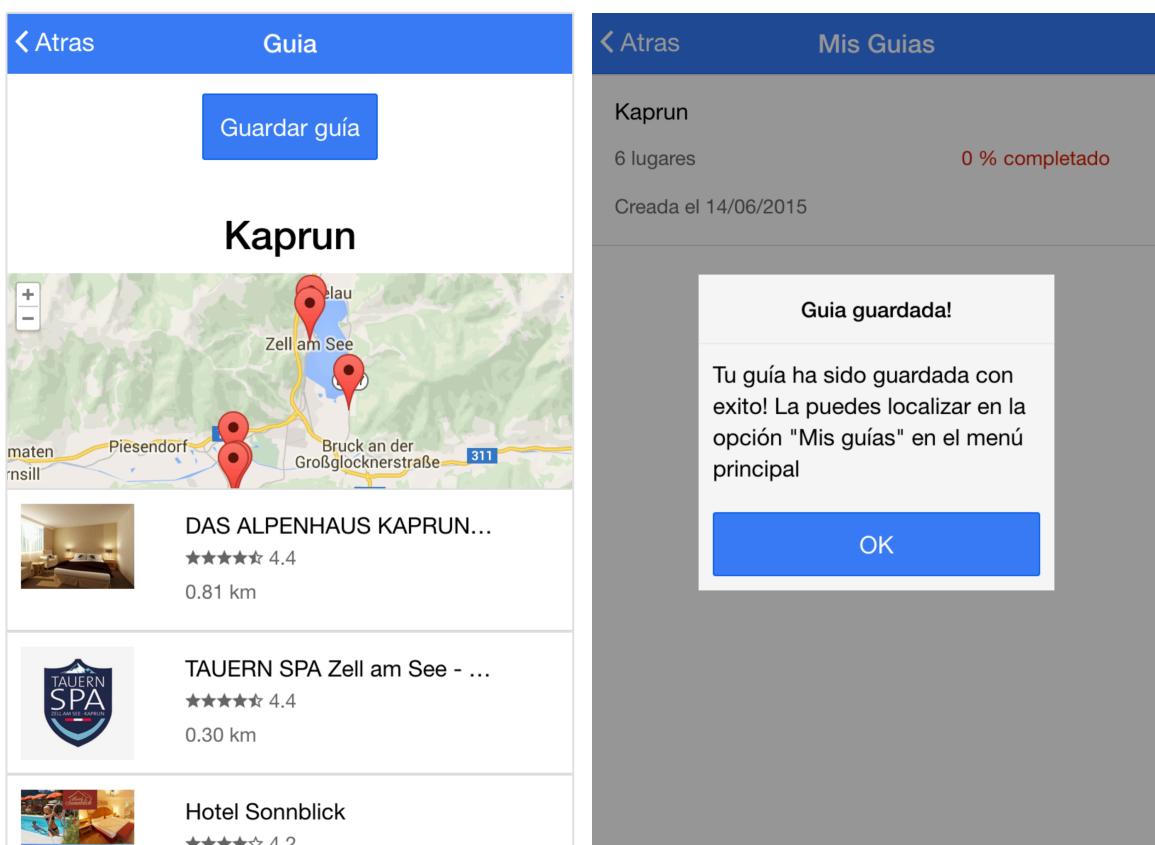


Figura 9.30. Resumen de guía, y mensaje que aparece al guardar

9.3.2.6. Mis guías

Accesible desde el menú principal y tras crear una guía, esta pantalla muestra las guías guardadas y permite llevar un seguimiento de ellas. Para ello, al pulsar sobre una guía se va al detalle de ella y se puede marcar los lugares ya visitados.

Mis Guias	
Mónaco	10 lugares 100 % completado
03698 Agost	7 lugares 57.14 % completado
Kaprun	6 lugares 33.33 % completado

Figura 9.31. Mis guías

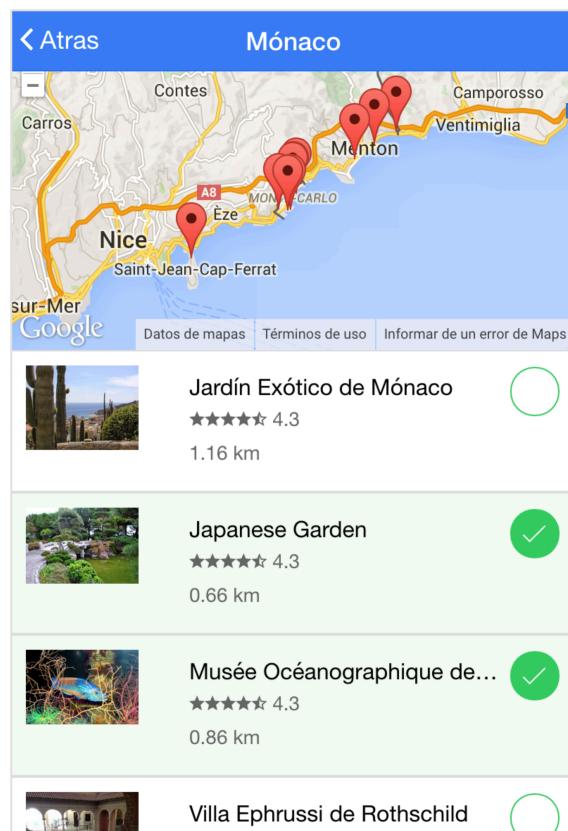


Figura 9.32. Detalle de guía en Mis guías

9.3.2.7. Preferencias

Divididas en dos secciones, Rutas y Guías turísticas, las preferencias son accesibles desde el menú principal y desde la selección de guía.

Rutas

Unidades

Kilómetros

Millas

Si es posible, evitar...

Autopistas

Peajes

Guías turísticas

Ordenar por

Popularidad

Distancia

Radio: 10 km

1km ————— [Slider] ————— 40km

Intereses

Arte

Entretenimiento

Restaurantes

Vida nocturna

Figura 9.33. Preferencias

9.3.2.8. Historial

Se puede acceder al historial desde el menú lateral. En él se encuentra el listado de rutas buscadas.

Historial		
Borrar historial		
5412, Austria - Mónaco	14-06-2015	19:26
5412, Austria - Daillecourt, Francia	14-06-2015	19:18
5412, Austria - 03698 Agost, Alicante, España	14-06-2015	19:17
Marktschellenberg, Alemania - 03698 Agost, Alicante, España	14-06-2015	19:06
Marktschellenberg, Alemania - Kaprun, Austria	14-06-2015	17:32
5412, Austria - Krottendorf, Austria	14-06-2015	17:32

Figura 9.34. Historial

10. Conclusiones

Recapitulando conceptos anteriores, TripMinder está pensado como una aplicación multiplataforma que permite al usuario planificar un viaje, convirtiéndose en una completa aplicación “*de bolsillo*”.

Las principales funcionalidades que debe proveer son:

- Calcular y presentar al usuario las mejores rutas para cada modalidad de transporte de manera ordenada
- Permitir crear guías turísticas para el lugar deseado. Este lugar podría ser el destino o cualquier otro que el usuario desee.

Tras el desarrollo de TripMinder, se puede afirmar que estas funcionalidades han sido desarrolladas completamente, abordando así el desarrollo de un proyecto completo, desde su planificación y diseño hasta su desarrollo y despliegue.

A continuación se evalúan en detalle todos los aspectos a tener en cuenta, se plantean mejoras y se analizan todos los sucesos que han tenido presencia durante el desarrollo de TripMinder para así poder llegar a una conclusión final.

10.1. Conclusiones sobre los objetivos personales

Los objetivos personales han sido cubiertos en su totalidad. A continuación se recapitulan estos objetivos y se expone el resultado de los mismos:

- ***Crear una aplicación multiplataforma que utilice y exprima lo máximo posible las capacidades del dispositivo:*** se ha desarrollado con éxito la aplicación y se ha testeado en los sistemas operativos Android e iOS, además de en el navegador web Chrome y Firefox. En el caso de TripMinder, se han utilizado plugins de localización y de información del dispositivo, por tanto se puede afirmar que se han aprovechado las capacidades que se han necesitado.

- ***Enfrentarse a la problemática de la recolección de datos y de su organización, uso y distribución:*** pese a ser el objetivo más costoso de afrontar, se ha llevado a cabo satisfactoriamente. Se obtienen datos de 4 diferentes API's, se procesan y se utilizan para su posterior presentación al usuario.
- ***Aplicar metodologías de desarrollo ágil, realizando reuniones semanales y llevando al día un Kanban de las tareas:*** se han realizado reuniones semanales donde se ha puesto al día sobre el desarrollo realizado, el desarrollo a realizar en la siguiente semana y el progreso general del proyecto. Además se ha mantenido una pizarra Kanban de tareas en el sitio web Trello. En el Anexo de este documento se puede ver el enlace a la pizarra Kanban.
- ***Poner a prueba la capacidad de improvisación en lo referente a los riesgos de las tecnologías que se utilizan:*** este ha sido otro de los objetivos más difíciles de afrontar. Se tuvo dificultades en el uso y resolución de problemas en la primera tecnología utilizada. Posteriormente, se abandonó esa tecnología para migrar a otra totalmente nueva y diferente, que además estaba en fase beta, lo que provocó que se tuvieran que solucionar ciertos defectos y fallos que se encontraron durante el desarrollo. A pesar de ello, se ha superado el objetivo tomando las decisiones oportunas en el momento oportuno.
- ***Adquirir experiencia en tecnologías desconocidas:*** como se ha comentado en el objetivo anterior, sucedió un cambio de tecnología. Por ello, se puede afirmar que se ha aprendido sobre ambas. En especial, se ha adquirido un gran nivel de experiencia en Ionic, la tecnología definitiva mediante la cual se ha desarrollado la aplicación.
- ***Seguir una planificación y poder cumplir en la medida de lo posible los objetivos marcados:*** en la medida de lo posible se ha seguido una planificación marcada, aunque no ha sido la misma que se planteó en un principio. Debido al cambio de tecnología, se ha tenido que improvisar y replantear, de manera rápida, una planificación. Esta se ha seguido finalmente mediante la pizarra Kanban en el sitio web Trello.

10.2. Conclusiones sobre los objetivos de la aplicación

Los **objetivos** de la aplicación han sido abarcados con éxito. Alguno de ellos ha sido modificado, ya que durante el desarrollo se han producido cambios en tecnologías y requerimientos, y esto ha hecho que alguno de los objetivos haya cambiado ligeramente.

No se han realizado ninguno de los objetivos opcionales de la aplicación, aunque no estaban pensados para ello, sino para trabajar en ellos en el caso de que los objetivos principales fueran realizados en un tiempo menor que el esperado. Sin embargo, se ha añadido funcionalidades adicionales, como permitir seleccionar aeropuertos y fechas para los aviones, o introducir los orígenes y destinos usando el mapa o los campos de texto y sincronizarlos.

A continuación se detallan los objetivos principales y sus resultados:

- **Diseñar una UI (interfaz de usuario) de diseño moderno y usable:** objetivo afrontado con éxito, obteniendo un resultado mejor de lo esperado. Esto es debido a que la tecnología finalmente utilizada es de nueva generación y facilita el diseño y desarrollo de nuevas UI. En el apartado 9.3 “Diseño final y resultado” se puede encontrar más información relativa al diseño final.
- **Diseñar una interacción usuario-dispositivo que proporcione una gran experiencia de usuario:** la interfaz de usuario está desarrollada para proporcionar una gran experiencia de usuario. Para ello se hace uso de fuentes de iconos, mensajes de información y barras de progreso, lo que permite al usuario en todo momento saber que algo está ejecutándose en ese momento.
- **Recolectar y tratar información referente al tráfico y al tránsito proveniente de medios globales:** este objetivo, referente a los medios de transportes públicos, en este caso tren, autobús y avión, también ha sido realizado con éxito. Para ello se utiliza el API de Google Transit y QPX Express, y se trata la información para separar medios de transporte y presentar los resultados al usuario.

- ***Calcular la mejor ruta de cada medio de transporte:*** el cálculo de las rutas es un objetivo crucial para la aplicación. Este cálculo es realizado por las propias API's de las cuales se obtienen las rutas, de las que se obtienen las mejores rutas disponibles, según criterios de distancia y duración, además de ciertas preferencias que el usuario puede seleccionar, como las de evitar autopistas y peajes. Por tanto, el objetivo se ha cumplido.
- ***Proporcionar capacidad de creación de guías turísticas, dadas unas preferencias:*** abordado con éxito. Se permite crear guías del destino de manera cómoda y guardarlas para poder ser visualizadas posteriormente. Además, se puede crear guías turísticas de cualquier lugar desde el creador de guías, disponible en el menú lateral.
- ***Permitir la configuración principal del perfil, donde residen las preferencias de usuario:*** objetivo implementado con éxito, haciendo uso de HTML5 Web Storage para su persistencia. Las preferencias están divididas entre preferencias de ruta y preferencias de guía.

10.3. Mejoras y ampliaciones

Como la mayoría de los proyectos, siempre hay espacio para mejoras. Para el caso de TripMinder, se podría decir que las mejores principales serían los propios objetivos opcionales propuestos para este proyecto:

- **Medios locales:** se puede considerar el siguiente paso y mejora principal. Con ello las rutas locales serían mucho más precisas y se ampliaría el rango. Esto supondría un aumento considerable en el desarrollo de la aplicación, ya que habría que añadir una capa con un servidor propio, donde mediante robots se extraerían los datos de páginas o bases de datos de las empresas de transporte y se insertarían en una base de datos propia común, que además llevaría cierto trabajo de mantenimiento.
- **Seguimiento de ruta y guía:** sería una interesante funcionalidad que puede ser propuesta de dos maneras diferentes: implementada en la propia aplicación o haciendo uso de la aplicación de Google Maps. El primer caso supondría un gran volumen de trabajo, puesto que toda la funcionalidad tiene que ser implementada. El segundo, no debería suponer demasiado. Se podría decir que, en caso de que fuera posible, el caso óptimo sería el segundo ya que TripMinder no está enfocado a reemplazar o copiar la funcionalidad de Google Maps, sino que tiene un enfoque más turístico.

10.4. Resultado y conclusión final

El resultado final ha sido una aplicación multiplataforma acabada y testeada, que podría ser utilizada por los usuarios sin problema alguno. De hecho, si es posible, la aplicación se publicará en las stores de Android y iOS y estará online para su uso con el navegador. Eso sí, con simple objetivo académico. La aplicación podría ser puesta actualmente en producción, debido a que no está preparada para ello. Por ejemplo, la limitación de peticiones de rutas de vuelo es de 50 por día, la que se alcanzaría demasiado pronto. Sin embargo, podría ser preparada para producción si se invirtiera en el pago de los accesos a la API's en el caso de que se sobrepase su límite.

Durante su desarrollo se ha aprendido sobre diferentes aspectos de desarrollo y planificación, obteniendo un resultado más que satisfactorio.

11. Bibliografía

[Apache Cordova, n.d.] Platform support.

https://cordova.apache.org/docs/en/4.0.0/guide_support_index.md.html ,

consultado en fecha 13/02/2015

[Christophe Coenraets, 2014] Ionic Framework Tutorial.

<http://ccoenraets.github.io/ionic-tutorial/> , consultado en fecha 22/01/2015

[FastCodesign, 2014] UX, UI: Who does what?.

<http://www.fastcodesign.com/3032719/ui-ux-who-does-what-a-designers-guide-to-the-tech-industry> , consultado en fecha 14/11/2014

[Flight Aware, n.d.] Flight Aware API.

<http://es.flightrightaware.com/commercial/flightxml/> , consultado en fecha 09/11/2014

[Flight Stats, n.d.] FlightStats Flex API. <https://developer.flightstats.com/products> , consultado en fecha 09/11/2014

[Google Code, n.d.] Chromium Embedded Framework,

<https://code.google.com/p/chromiumembedded/> , consultado en fecha 03/01/2015

[Google Maps, n.d.] Google Maps API. <https://developers.google.com/maps/> , consultado en fecha 09/11/2014

[Google Transit, n.d.] Google Transit API. <https://developers.google.com/transit/> , consultado en fecha 08/11/2014

[Google Places, n.d.] Google Places API. <https://developers.google.com/places/> , consultado en fecha 08/11/2014

[Google QPX Express, n.d.] Google QPX Express API.

<https://developers.google.com/qpx-express/> , consultado en fecha 09/11/2014

[Ian Sommerville, 2011] Ingeniería del Software, 9^a edición.

[InKnowledge, n.d.] Taxi Fare Rest Service.

<http://inknowledge.co.uk/Products/TaxiFareWebServices.aspx> , consultado en fecha 09/11/2014

[Ionic, 2014] Ionic Framework. <http://ionicframework.com>

, consultado en fecha 19/01/2015

[Ionic, 2014] Learn Ionic Framework. <http://learn.ionicframework.com/>

, consultado en fecha 20/01/2015

[Minube, n.d.] Minube Developers. <http://www.minube.com/developers/api> ,

consultado en fecha 08/11/2014

[Mobile life, 2013] Mobile life. <http://www.tnsglobal.com/2013/mobile-life> ,

consultado en fecha 04/11/2014

[Movable Type Scripts, n.d.] Calculate distance, bearing and more between Latitude/Longitude points, <http://www.movable-type.co.uk/scripts/latlong.html> , consultado en fecha 12/03/2015

[Movable Type Scripts, n.d.] GIS FAQ Q5.1: Great circle distance between 2 points, <http://www.movable-type.co.uk/scripts/gis-faq-5.1.html> , consultado en fecha 17/03/2015

[OpenFlights, n.d.] Airport, airline and route data. <http://openflights.org/data.html> , consultado en fecha 22/04/2015

[Pew Research Center, 2013] Smartphone ownership.

<http://www.pewresearch.org/data-trend/media-and-technology/device-ownership/> , consultado en fecha 04/11/2014

[Taxi Fare Finder, n.d.] Taxi Fare Finder API (beta).

<http://www.taxifarefinder.com/api.php> , consultado en fecha 09/11/2014

[Tixik, n.d.] Tixik API. <http://www.tixik.com/info/api/> , consultado en fecha

08/11/2014

[Universidad de Alicante, n.d.] Perfiles profesionales en Ingeniería Multimedia.

<http://cvnet.cpd.ua.es/webcvnet/planetstudio/planestudiond.aspx?plan=C205#> , consultado en fecha 05/11/2014

[Wikipedia, 2014] Near field communication.

http://en.wikipedia.org/wiki/Near_field_communication , consultado en fecha 05/11/2014

[WixBlog, 2011] Cómo preparar una guía de estilo para tu web.

<http://es.wix.com/blog/2011/01/como-crear-una-guia-de-estilo/> , consultado en fecha 10/11/2014

[Yelp, n.d.] Yelp API documentation.

<http://www.yelp.com/developers/documentation/v2/overview> , consultado en fecha 08/11/2014

12. Anexo

Se utilizará este anexo para exponer recursos y enlaces adicionales relativos al proyecto TripMinder.

Para el seguimiento y la planificación de tareas, se ha utilizado Trello. En el siguiente enlace se puede acceder la pizarra kanban:

<https://trello.com/b/qaREQX2y/tripminder-tfg-hito-3>

El proyecto está disponible en Github:

<https://github.com/alexjoverm/TripMinder>

Además, se han realizado manuales de desarrollo (no completos) a modo de bitácora, tanto para la antigua como para la finalmente usada tecnología. En ellos se puede ver como configurar el entorno y empezar a trabajar con él. Además se comentan los problemas encontrados relativos a la tecnología. Ambos están en inglés:

Rad Studio C++6 (antiguo desarrollo):

https://github.com/alexjoverm/TripMinder/blob/master/RadStudio_documentation.md

Ionic (desarrollo final):

https://github.com/alexjoverm/TripMinder/blob/master/Ionic_documentation.md