TRABAJO DE FIN DE CARRERA

Optimizador de rutas para iOS

Memoria del proyecto

Fecha de entrega: 02/01/2016

Autor: Javier Fernández Cabanas

Javier.Fernandez.C@gmail.com

Consultor: Jordi Ceballos Villach

Índice de contenidos

1 Contexto y justificación

Mi objetivo es realizar una aplicación de cálculo de rutas que permita mostrar una ruta optimizada entre un punto inicial y otro final y con varios puntos intermedios.

La orientación que pretendo dar a la aplicación es de un uso recreativo o de tiempo libre, de modo que se puedan planificar rutas en coche o a pie, y que se puedan guardar en un servidor accesible para que tanto el usuario que crea la ruta como otros usuarios de la aplicación usuarios puedan utilizarlas, comentarlas y puntuarlas.

A continuación, trataré de exponer una estructura y una planificación básicas. Hay que tener en cuenta que se tratan de estimaciones y que ambas pueden estar, y seguramente estarán, sujetas a cambios y desviaciones a lo largo del desarrollo del proyecto.

1.1 Estudio de mercado

He realizado una breve búsqueda de aplicaciones que puedan hacer algo parecido a lo propuesto para este proyecto. En primer lugar, he observado que las aplicaciones de mapas para iPad de Google y Apple no ofrecen el cálculo de rutas con varios puntos intermedios.

Existe una aplicación, llamada "*Routes. Planning your journeys*", de la empresa *The Clash Soft*, disponible en español, orientada a la planificación de rutas durante un viaje. Dispone de una gran variedad de funciones y modo offline.

También existe para iOS la versión móvil de Wikiloc, la conocida aplicación web para subir rutas de todo tipo, tanto urbanas como de montaña.

Por mi parte, como es lógico por el poco tiempo de que dispongo, trataré de crear algo más sencillo pero que ofrezca nuevas funcionalidades y formas de uso, intentando diferenciar un poco mi producto de los ya existentes. En concreto, mi propósito es apoyarme en la sencillez para conseguir una facilidad de uso mayor que en las demás aplicaciones.

2 Objetivo

El proyecto “Optimizador de rutas para iOS” consiste básicamente en realizar una aplicación que permita mostrar una ruta optimizada con varios puntos de paso intermedios.

La orientación que pretendo dar a la aplicación es de un uso recreativo o de tiempo libre, de modo que se puedan planificar rutas en coche o a pie, y que se puedan guardar en un servidor accesible para que tanto el usuario que crea la ruta como otros usuarios de la aplicación usuarios puedan utilizarlas, comentarlas y puntuarlas.

En concreto, el objetivo es ofrecer al usuario una serie de funcionalidades muy concretas, que paso a detallar a continuación.

2.1 Funcionalidades principales

2.1.1 Planificación de rutas individuales en coche

Extracción de cada una de las rutas mediante la API de Google Maps.

2.1.2 Extracción de datos de tráfico

Extracción del estado del tráfico mediante la API de Google Maps.

2.1.3 Cálculo de ruta

Cálculo de la ruta más óptima.

2.1.4 Visualización de resultados

Obtención de los resultados en una vista de mapa de Google Maps con indicaciones.

2.2 Funcionalidades opcionales

2.2.1 Ruta óptima

Desarrollo de un algoritmo de cálculo de ruta óptima (problema del viajante) para un número pequeño de puntos intermedios.

2.2.2 Elección de tipo de transporte

El usuario podrá elegir entre rutas en coche, transporte público o a pie. Para ello también se utilizará la API de Google Maps.

2.2.3 Guardar rutas

El usuario tendrá la posibilidad de guardar las rutas, indicando si son públicas o privadas.

2.2.4 Acceso a rutas púbicas de otros usuarios

Todos los usuarios dispondrán de la posibilidad de ver, utilizar, comentar y puntuar las rutas públicas de otros usuarios, así como el nombre de usuario que creo la ruta.

2.2.5 Más populares

Se crearán dos listas, una de rutas más puntuadas y otra de usuarios ordenados por la suma de votos de sus rutas públicas.

La aplicación se desarrollará para poder ser utilizada en un teléfono iPhone con iOS 8 o superior.

3 Planificación

Para la realización de proyecto dispongo de la última versión de XCode para iOS. Estoy a la espera de que se me conceda una licencia de estudiante para utilizar Xamarin en Visual Studio, ya que me resulta más familiar por ser el IDE que utilizo en mi trabajo.

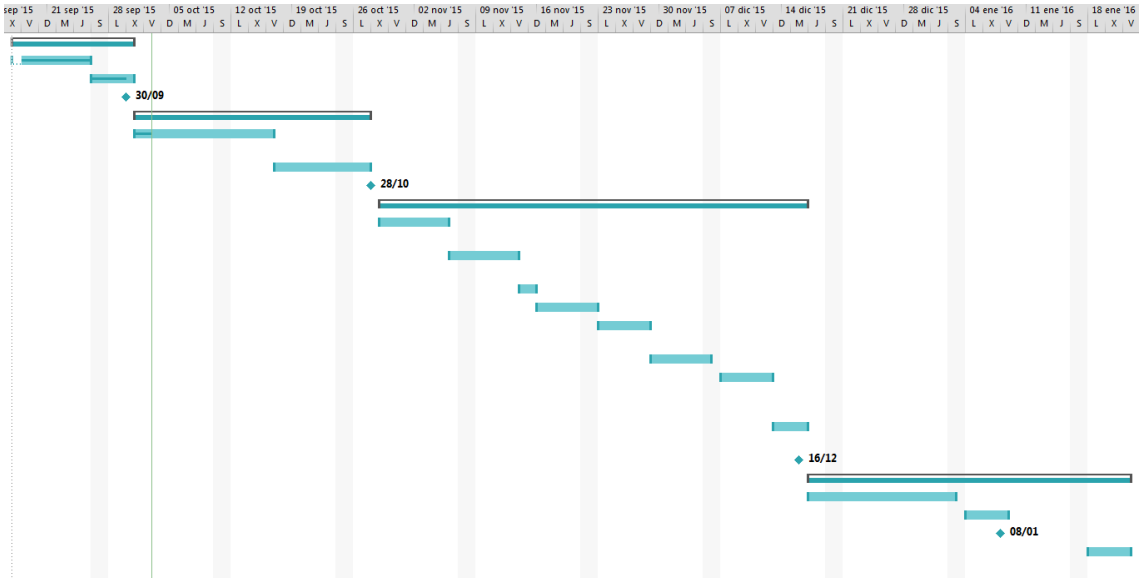
Por tanto, tengo dos opciones, XCode con Swift o Xamarin con F#, dependiendo la elección de una u otra de si recibo una licencia de Xamarin en los próximos días.

Las tareas a realizar son las siguientes:

- Plan de Trabajo (PAC 1)
  - Lectura de material
  - Documentación del Plan de Trabajo
  - Aplicación “Hola mundo”
  - Entrega de PAC 1
- Diseño y arquitectura (PAC 2)
  - Construcción del diseño de la aplicación
  - Documentación del diseño de la aplicación
  - Entrega de PAC 2
- Implementación (PAC 3)
  - Módulo 1. Planificación de ruta individual
  - Módulo 2. Extracción de datos de tráfico
  - Módulo 3. Cálculo de ruta óptima
  - Módulo 4. Visualización de resultados
  - Módulo 5. Elección de tipo de transporte
  - Módulo 6. Guardar rutas
  - Módulo 7. Funcionalidades sociales: rutas públicas con puntuación y comentarios
  - Módulo 8. Popularidad de rutas y usuarios
- Entrega final
  - Redacción de memoria
  - Presentación
  - Entrega TFC final

3.1 Diagrama de Gantt

Nombre de tarea	Comienzo	Fin	Horas
PAC 1. Plan de trabajo	**jue 17/09/15**	**mié 30/09/15**	**44 h**
Lectura de material	jue 17/09/15	vie 25/09/15	26 h
Redacción PAC 1	sáb 26/09/15	mié 30/09/15	18 h
Entrega de PAC 1	mié 30/09/15	mié 30/09/15	
PAC 2. Diseño	**jue 01/10/15**	**mié 28/10/15**	**86 h**
Construcción del diseño de la aplicación	jue 01/10/15	vie 16/10/15	48 h
Redacción PAC 2	sáb 17/10/15	mar 27/10/15	38 h
Entrega de PAC 2	mié 28/10/15	mié 28/10/15	
PAC 3. Implementación	**jue 29/10/15**	**mié 16/12/15**	**154 h**
Módulo 1. Planificación ruta individual	jue 29/10/15	jue 05/11/15	24 h
Módulo 2. Extracción de datos de tráfico	vie 06/11/15	vie 13/11/15	24 h
Módulo 3. Cálculo ruta óptima	sáb 14/11/15	dom 15/11/15	12 h
Módulo 4. Visualización resultados	lun 16/11/15	dom 22/11/15	22 h
Módulo 5. Elección tipo de transporte	lun 23/11/15	sáb 28/11/15	16 h
Módulo 6. Guardar rutas	dom 29/11/15	dom 06/12/15	28 h
Módulo 7. Funcionalidades sociales: rutas públicas con puntuación y comentarios	lun 07/12/15	sáb 12/12/15	16 h
Módulo 8. Popularidad rutas y usuarios	dom 13/12/15	mié 16/12/15	12 h
Entrega PAC 3	mié 16/12/15	mié 16/12/15	
Entrega final	**jue 17/12/15**	**vie 22/01/16**	**80 h**
Redacción memoria	jue 17/12/15	dom 03/01/16	60 h
Presentación	lun 04/01/16	vie 08/01/16	10 h
Entrega TFC final	vie 08/01/16	vie 08/01/16	
Debate virtual	lun 18/01/16	vie 22/01/16	10 h



3.2 Gestión del tiempo

Dispongo de unas veintidós horas semanales, distribuidas como sigue:

- De lunes a viernes, 10 horas (horario de 21 a 23 h)
- Sábados y domingos, 12 horas (horario de 7 a 13 h)

4 Riesgos del proyecto

Durante el transcurso del proyecto, pueden aparecer factores no previstos que afecten negativamente al desarrollo previsto del mismo.

A continuación, enumeramos los posibles riesgos detectados para este proyecto.

Riesgo	Descripción	Probabilidad	Impacto	Acciones mitigantes
[R01] Falta de conocimientos en desarrollo iOS	Nunca he programado con XCode ni para dispositivos móviles.	**100%**	**Alto**	Licencia Xamarin para desarrollo con Visual Studio.
[R02] Licencia Xamarin	Es posible que no reciba la licencia solicitada para programar con Xamarin en Visual Studio	**Alta**	**Medio**	Programar el proyecto con XCode y seguir el curso Udemy de programación iOS por Fernando Rodríguez.
[R03] Planificación incorrecta	Es posible que el tiempo estimado no se ajuste al que realmente voy a necesitar.	**Alta**	**Alto**	Planificación en cascada con los módulos más importantes primero.

[R04] Reducción horas disponibles	Es posible que tenga que realizar horas extras de mi trabajo, lo que impactaría en el tiempo dedicado al proyecto	**Media**	**Alto**	Revisión constante de la evolución del proyecto, adaptándolo a las circunstancias.

5 Usuarios y contextos de uso

Con el objeto de conocer a los potenciales usuarios de nuestra aplicación, así como los diferentes contextos en que la usarán, sus necesidades, objetivos y actitudes, utilizaremos a continuación una serie de métodos de indagación.

5.1 Métodos de indagación

He seleccionado como métodos de indagación más apropiados para este proyecto, el método de diario, la entrevista y el análisis comparativo (*benchmarking*).

5.1.1 Justificación de la elección

Uno de los métodos de indagación cualitativos más utilizados es el de investigación contextual. En el ámbito de los móviles, dado la variedad de entornos de uso, existe la variante del método de seguimiento (*shadowing*) consistente en la observación directa mientras el participante lleva a cabo sus tareas diarias. En mi caso, mi horario de trabajo me ha resultado inapropiado para utilizar esta técnica, por lo que he decidido utilizar en su lugar la técnica de diario, que tiene también la ventaja de ser menos intrusiva.

Para paliar algunas de las limitaciones del método de estudio diario, utilizaré la entrevista de campo como técnica complementaria para recabar mayor información de los participantes acerca del cómo y el porqué de sus comportamientos (Ginsburg, 2010).

Además, también utilizaré la técnica de análisis comparativo, al existir diversas aplicaciones que permiten calcular rutas, tales como *Rutas*, de la empresa *The Class Soft*. También tendré en cuenta la propia Google Maps, en cuanto a su forma de uso, aunque en el caso de la aplicación para iPhone no dispone de la posibilidad del cálculo de puntos intermedios.

5.1.2 Planteamiento

Solicitaré la colaboración de uno o varios participantes, personas de mi familia y amigos, que utilizan su teléfono con mayor o menor frecuencia para obtener rutas y registrarán su actividad relacionada con rutas y mapas durante una semana. Para mayor dinamismo, utilizaremos un twitter privado para los registros y las entrevistas de campo.

La entrada de diario será de tipo cuestionario, de modo que la regla transmitida a cada participante es que, para cada entrada, debe contestar un número de preguntas predefinidas.

Esta información será complementada con un análisis comparativo de las siguientes aplicaciones:

- Rutas, de The Class Soft.
- actiRuta, de 4GFlota.

5.1.3 Desarrollo

Finalmente, he elegido como colaboradores a dos personas que utilizan el móvil habitualmente como navegador GPS. El primero, José, se trata de una persona que lleva poco tiempo viviendo en Sabadell, está en paro y frecuentemente asiste a entrevistas de trabajo, principalmente en Barcelona.

El segundo caso se trata de David, que trabaja de administrativo en una inmobiliaria en Barcelona y, en ocasiones, debe salir a realizar diversas gestiones, con lo que tiene que desplazarse a varias direcciones en una misma salida.

Acordamos que enviarían vía twitter la respuesta al cuestionario. Yo, por mi parte, podré hacer alguna pregunta complementaria si considero necesario recabar más información, haciendo uso de la entrevista de campo.

5.1.4 Resultados

Las preguntas a contestar en cada entrada serán las siguientes:

- ¿Qué actividad vas a realizar? Indicar hora y descripción
- ¿Cuántos puntos, incluyendo el origen y destino finales, componen el recorrido? Indicar tipo de transporte.
- ¿Cuánto tiempo te ha llevado realizar la ruta completa, contando solo el tiempo empleado en los desplazamientos, aproximadamente?

5.1.4.1 Estudio diario

José (trabajador en paro)

Actividad registrada	*Aclaración mediante entrevista de campo*	*Implicaciones*
7:30 am Ver cómo llegar a primera entrevista desde Sabadell, en coche y a pie. 5 puntos en total. 10:35 am Ver cómo llegar a la segunda entrevista. 12:30 pm Ver cómo llegar al restaurante.		

15:30 pm Ver cómo llegar a la tercera entrevista.	En el restaurante no tenía cobertura 3g, tuvo que salir a la calle para calcular la siguiente ruta.	
17:00 pm Ver cómo regresar a Sabadell.		
Tiempo total aproximado utilizado en los desplazamientos, 90 minutos.	José ha ido a Barcelona a asistir a tres entrevistas de trabajo, dos por la mañana y una por la tarde. Se quedó a comer en un restaurante de la zona. Utiliza la aplicación Google Maps en el móvil para ir calculando cada ruta si va a pie, y utiliza un navegador para ir en coche.	No es necesario conocer la ruta óptima, pero sí sería interesante que pudiera calcular la ruta antes de salir y guardarla.

David (administrativo)

Actividad registrada	*Aclaración mediante entrevista de campo*	*Implicaciones*
9:15 am Salida a realizar gestiones. 7 puntos, todos a pie.	Las gestiones son independientes entre sí. Conoce la zona y sabe cómo llegar, pero se ha hecho una lista en un papel y el orden de visitas, seguramente, no es óptimo.	Podría utilizar los puntos de ruta como recordatorio de los lugares a visitar. Sería interesante calcular la ruta óptima para ahorrar tiempo.
12:30 am Vuelta a la oficina, el tiempo total desplazamientos, 45 minutos.		

5.1.4.2 Análisis comparativo

He escogido dos aplicaciones para el cálculo de rutas y las he analizado en comparación con la aplicación que estoy proyectando.

Aplicación	Puntos positivos	Puntos negativos
Rutas, de The Class Soft	Aplicación orientada al uso turístico, por lo que es muy completa en ese aspecto.	Está organizada jerárquicamente en viajes, itinerarios y rutas. Por tanto, fuera del uso turístico es demasiado compleja.
	Cálculo automático de ruta más rápida.	No tiene la opción de ruta circular (punto inicial igual a punto final).
	Gran cantidad de opciones: tipos de mapa, tipos de transporte, notas, enlace a Wikipedia, publicación de rutas, puntos de interés, etc.	
	Modo offline y exportación de ficheros a formatos CSV y GPX.	
actiRuta, de 4GFlota	Optimización de ruta.	Aplicación web no optimizada para dispositivos móviles.
	Aplicación orientada a transporte en carretera.	

5.1.5 Conclusiones

Para poder distinguir nuestra aplicación del uso de un navegador o de Google Maps, sería necesario añadir como característica principal el guardado de las rutas en el dispositivo.

El guardado en la nube, para compartir socialmente, quizá sea interesante únicamente para el uso de visita turística.

5.2 Perfiles identificados

Como resultado de la indagación, he detectado un abanico de perfiles con diversos intereses y motivaciones. He visto que la utilidad de cálculo de mejor ruta sobre una serie de puntos a visitar es poco reclamada en general, ya sea porque este cálculo lo hace el usuario de forma intuitiva o porque las visitas están determinadas por una agenda. Por tanto, la optimización solo sería de interés para los casos como el de un transportista o un turista que va visitando a pie distintos puntos y quiere realizar el trayecto más corto posible. Este caso es de menor importancia, ya que la ruta a pie está contemplada solo como implementación opcional.

Por tanto, mayoritariamente el usuario decide el orden en que realiza las visitas a los puntos elegidos, en lugar de dejar que la aplicación calcule un orden óptimo en función del tiempo o de la distancia. Por tanto, este cálculo de optimización optativo y debe tenerse en cuenta que será poco utilizado.

En concreto, los perfiles identificados son los siguientes:

5.2.1 Usuario turista

5.2.1.1 Características

- Usuario adulto, que suele estar en compañía de otras personas mientras utiliza la aplicación.
- Usuario acostumbrado al uso diario de *smartphone*.
- En muchas ocasiones, necesita las rutas a pie o en transporte público.
- Puede ser importante el cálculo de la ruta optimizada.
- Sería interesante poder compartir públicamente las rutas, para usar las que otros hayan realizado.

5.2.1.2 Contexto de uso

Hará uso de la aplicación en su tiempo libre, posiblemente en compañía de su pareja o amigos. Buscará rutas en lugares turísticos y que visita por primera vez. El usuario querrá planificar su itinerario turístico durante una jornada, visitando museos, lugares de interés, restaurantes, etc.

5.2.1.3 Análisis de tareas

El usuario conoce qué quiere visitar y necesita saber dónde se encuentran esos lugares y cómo llegar a ellos. Frecuentemente, preferirá que la aplicación elija el orden de visita en función de la distancia o del tiempo.

Por tanto, las tareas serán:

- Elegir punto de salida.
- Elegir punto de llegada (o indicar que la salida y la llegada son idénticas).
- Añadir lugar de visita.

- Si el orden no está determinado, poder optimizar la ruta en función del menor tiempo o distancia.
- Mostrar ruta en el mapa.
- Mostrar ruta con indicaciones.
- Poder elegir ir tipo de transporte (coche, a pie o transporte público).

5.2.1.4 Nuevas características descubiertas

Por un lado, el turista es un tipo de usuario al que le interesa especialmente poder elegir el tipo de transporte, característica ya contemplada como opcional. Por tanto, hay que considerar la posibilidad de que sea una funcionalidad principal.

En cambio, le interesan las siguientes características no contempladas inicialmente:

- Cálculo de mejor ruta en función del tiempo o de la distancia. En un principio, no se había contemplado la posibilidad de elegir entre ambos parámetros.
- Mostrar ruta con indicaciones. Inicialmente, solo se había previsto mostrar la ruta sobre el mapa.
- Poder alternar entre dos vistas, mapa y ruta con indicaciones.

5.2.2 Usuario metropolitano

5.2.2.1 Características

- Usuario adulto.
- Uso medio o bajo de *smartphone*.
- En muchas ocasiones, necesita las rutas a pie o en transporte público.
- Es importante el cálculo de la ruta optimizada por tiempo.
- Necesidades de organización del tiempo, le interesa una gran sencillez y facilidad de uso.

5.2.2.2 Contexto de uso

Dentro de este perfil se incluye a aquellos usuarios que durante su actividad diaria, ya sea personal o de trabajo, tenga que desplazarse a distintos lugares dentro de un área urbana.

Por ejemplo, el caso del administrativo que tiene que realizar de vez en cuando gestiones fuera de la oficina visitando distintos lugares en una misma salida.

Otro contexto de uso sería el de personas que tienen que realizar una serie de visitas con hora determinada, por ejemplo, para entrevistas de trabajo. En este caso, no se haría uso de la función de optimización.

5.2.2.3 Análisis de tareas

El usuario conoce qué quiere visitar y cómo llegar a cada uno de las direcciones, pero necesita saber cómo hacer el recorrido en el menor tiempo posible.

Por tanto, las tareas serán:

- Elegir punto de salida.
- Elegir punto de llegada (o indicar que la salida y la llegada son idénticas).
- Añadir lugar de visita.
- Si el orden no está determinado, poder optimizar la ruta en función del menor tiempo o distancia.
- Mostrar ruta en el mapa.
- Mostrar ruta con indicaciones.
- Poder elegir ir tipo de transporte (coche, a pie o transporte público).
- Poder guardar la ruta para reutilizarla.

5.2.2.4 Nuevas características descubiertas

- Guardado en local de las rutas. Se ha contemplado como opcional el guardado en un servidor externo. Quizá habría que añadir el guardado en local como funcionalidad principal.

5.2.3 Usuario intermetropolitano

5.2.3.1 Características

- Usuario adulto.
- Uso medio o bajo de *smartphone*.
- Necesidades de organización del tiempo, le interesa una gran sencillez y facilidad de uso.
- Uso como herramienta de trabajo.

5.2.3.2 Contexto de uso

Dentro de este perfil se incluye a aquellos usuarios que durante su actividad diaria deben desplazarse a múltiples puntos en una zona más amplia que los casos anteriores.

Por tanto, si la hora de visita no está determinada con precisión, puede ser muy importante el cálculo de la ruta que emplee el menor tiempo posible.

Por ejemplo, un comercial que a diario tiene que visitar distintas empresas para vender su producto.

Otro ejemplo, sería el de un transportista que tiene que entregar la mercancía durante el día con un horario de entrega elástico, lo que le permite utilizar rutas optimizadas.

5.2.3.3 Análisis de tareas

El usuario conoce qué quiere visitar y necesita saber dónde se encuentran esos lugares y cómo llegar a ellos. Par el usuario es muy importante realizar la ruta en el menor tiempo posible.

Por tanto, las tareas serán:

- Elegir punto de salida.
- Elegir punto de llegada (o indicar que la salida y la llegada son idénticas).
- Añadir lugar de visita.

- Optimizar ruta en función del menor tiempo posible.
- Mostrar ruta en el mapa.

6 Diseño conceptual

En este apartado se describen los escenarios de uso el flujo de interacción de la aplicación, a partir de los perfiles de usuario identificados anteriormente.

6.1 Escenarios de uso

A continuación, se describen distintos escenarios de uso según los tipos de usuario.

6.1.1 Usuario turista

- *Contexto*: jornada turística en una ciudad o región desconocida.
- *Objetivos*: visitar una serie de lugares y puntos turísticos.
- *Tareas*: creación de una ruta que incluya todos esos puntos con inicio y llegada en el hotel donde se hospeda.
- *Necesidades de información*: necesita saber cómo llegar a cada uno de los lugares que quiere visitar, pudiendo elegir el tipo de transporte.
- *Funcionalidades que necesita*:
    - Planificación de rutas.
    - Cálculo de ruta óptima.
    - Visualización de resultados.
    - Elección de tipo de transporte.
- *Forma de realizar las tareas*: el usuario indica los puntos de partida, llegada e intermedios y el medio de transporte. A continuación, pide que se muestre la ruta en el mapa. Prefiere elegir el orden de visitas, ya que ha reservado mesa en un restaurante para comer a una hora concreta. Durante el trayecto a pie, cambia a modo de vista de ruta con indicaciones.

6.1.2 Usuario metropolitano

- *Contexto*: ruta urbana para realizar gestiones de diverso tipo en horario comercial.
- *Objetivos*: visitar una serie de oficinas y organismos públicos.
- *Tareas*: creación de una ruta que incluya todos esos puntos con inicio y llegada en el mismo punto.
- *Necesidades de información*: necesita saber cómo llegar a cada uno de los lugares que quiere visitar, pudiendo elegir el tipo de transporte.
- *Funcionalidades que necesita*:
    - Planificación de rutas.
    - Cálculo de ruta óptima.
    - Visualización de resultados.
    - Elección de tipo de transporte.
    - Guardado de rutas.
- *Forma de realizar las tareas*: el usuario indica los puntos de partida, llegada e intermedios y el medio de transporte. A continuación, pide que se muestre en el mapa

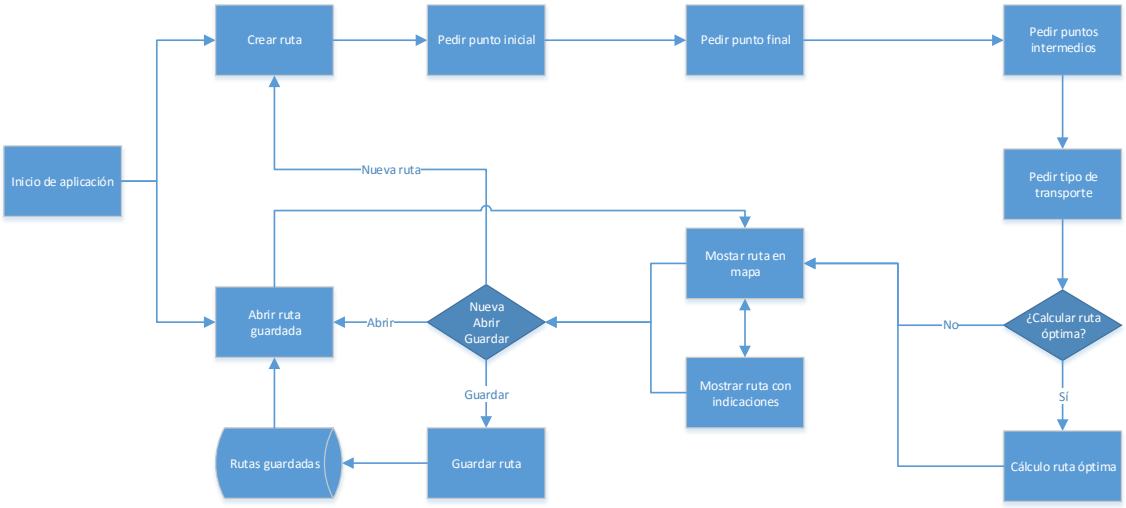
o con indicaciones, la ruta óptima para realizar las visitas con el menor tiempo posible de desplazamiento. Si la ruta es a pie, quiere poder alternar entre las vistas de mapa y de ruta con indicaciones. Es posible que otro día tenga que repetir la ruta, así que la guarda con un nombre descriptivo.

6.1.3 Usuario intermetropolitano

- *Contexto*: ruta interurbana de visitas a empresas, visitas turísticas a diversas localidades o reparto de mercancías.
- *Objetivos*: realizar una serie de viajes de largo recorrido en el menor tiempo posible.
- *Tareas*: creación de una ruta óptima.
- *Necesidades de información*: necesita saber cómo llegar a cada uno de los lugares que quiere visitar en coche.
- *Funcionalidades que necesita*:
    - Planificación de rutas.
    - Cálculo de ruta óptima.
    - Visualización de resultados.
    - Guardado de rutas.
- *Forma de realizar las tareas*: el usuario indica los puntos de partida, llegada e intermedios. A continuación, pide que se muestre en el mapa o con indicaciones, la ruta óptima para realizar las visitas con el menor tiempo posible de desplazamiento. Es posible que otro día tenga que repetir la ruta, así que la guarda con un nombre descriptivo.

6.2 Flujo de interacción

A continuación, se muestra gráficamente el funcionamiento general de la aplicación.



Una vez que se muestra la ruta en cualquiera de sus dos vistas, modo mapa o modo indicaciones, existirá también la posibilidad, como se muestra en el diagrama, de:

- Empezar de nuevo a crear una ruta

- Guardar la ruta actual
- Cargar una ruta guardada.

Además, desde cualquiera de las dos vistas, se podrá:

- Modificar los puntos de paso (dirección y orden de visita).
- Modificar el tipo de transporte.
- Pedir la ruta óptima.

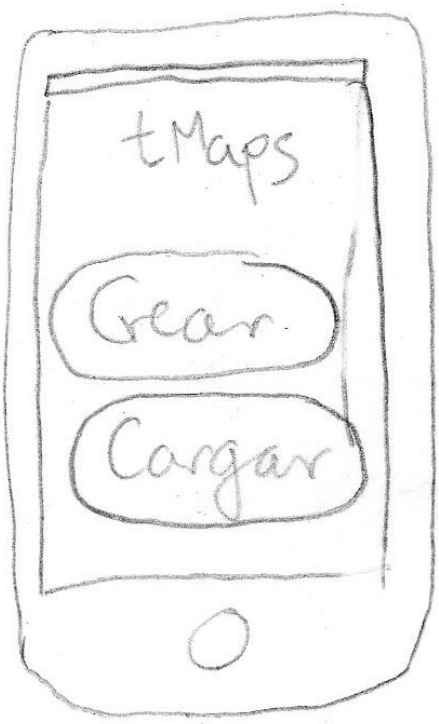
7 Diseño de prototipos

A partir de los flujos de interacción diseñados, se muestran a continuación los esbozos y los prototipos horizontales de alta fidelidad.

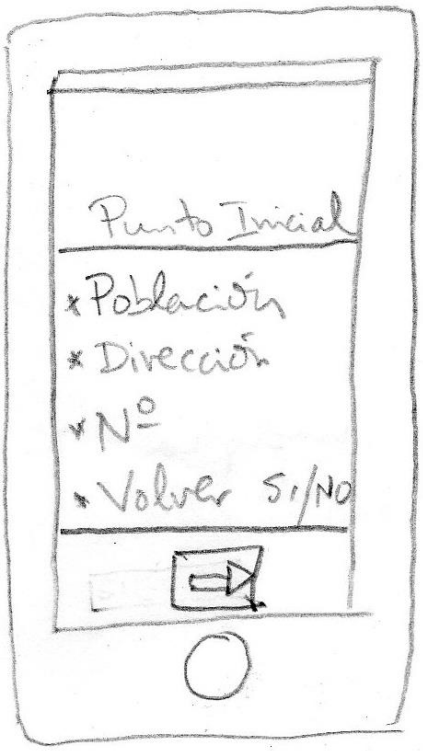
7.1 Sketches

A continuación se presentan los esbozos correspondientes a los distintos pasos dibujados en el flujo de interacción.

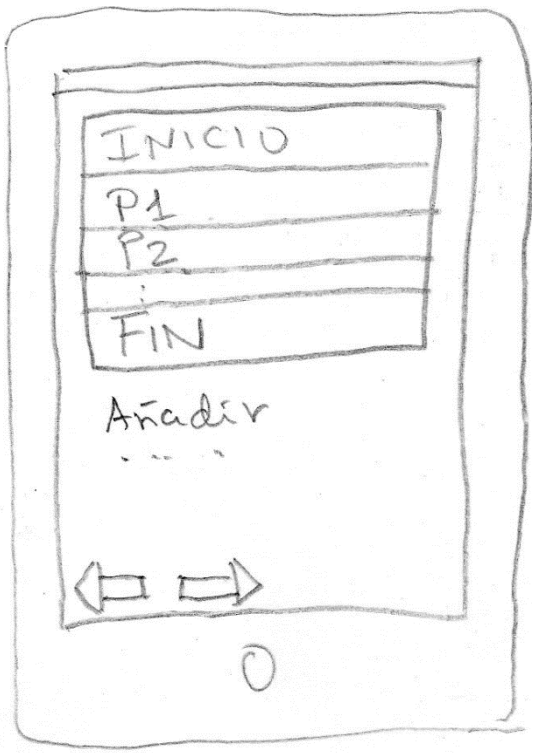
7.1.1 Pantalla de inicio



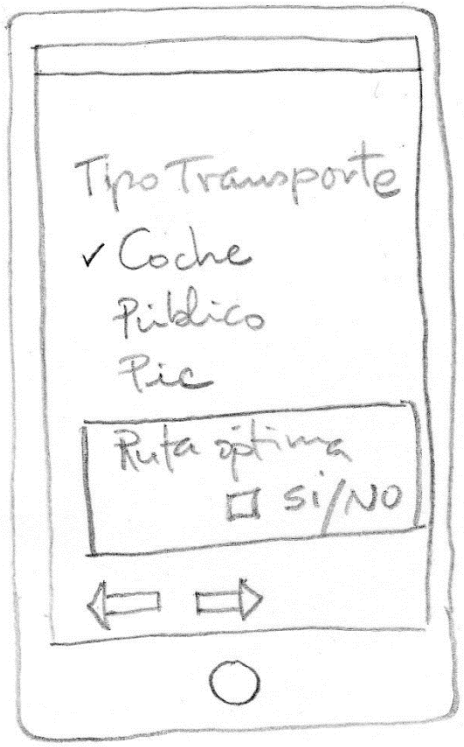
7.1.2 Pedir punto inicial



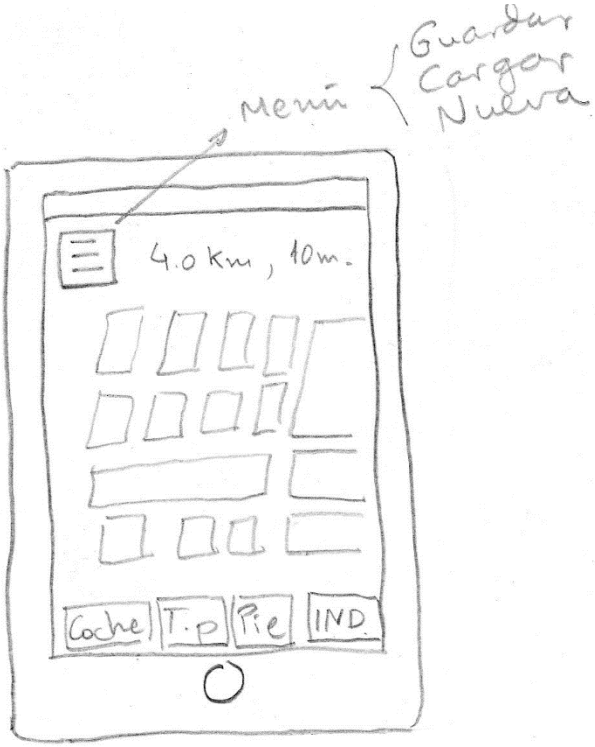
7.1.3 Pedir puntos intermedios



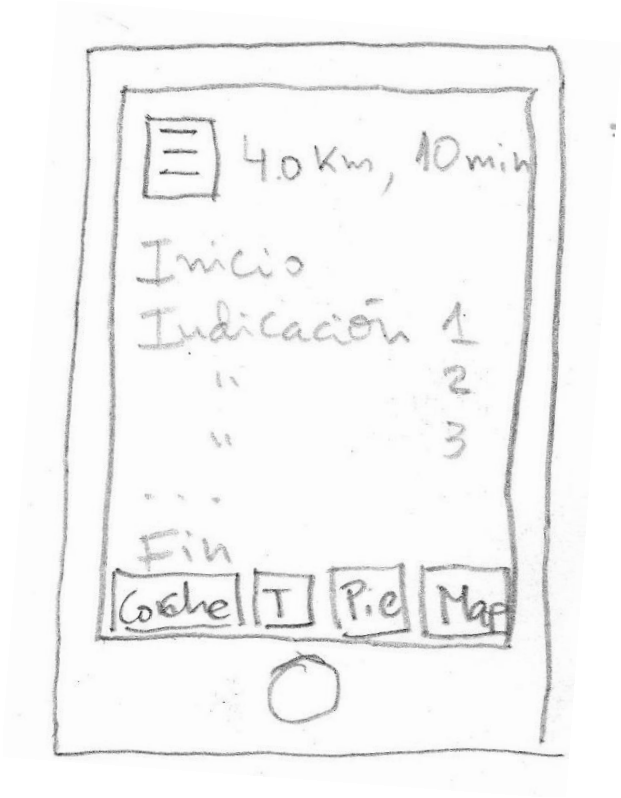
7.1.4 Pedir tipo de transporte



7.1.5 Mostrar ruta en mapa

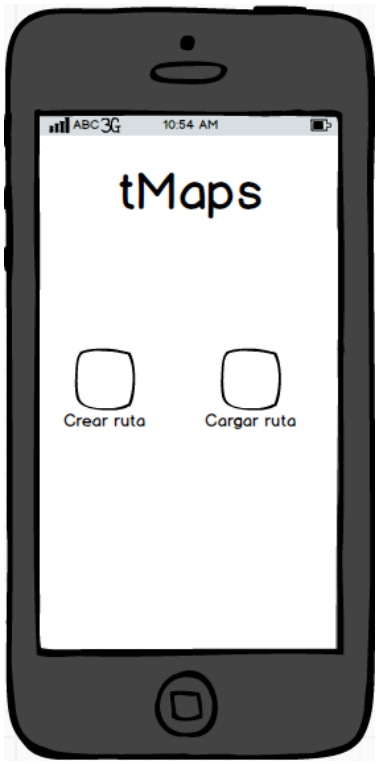


7.1.6 Mostrar ruta con indicaciones



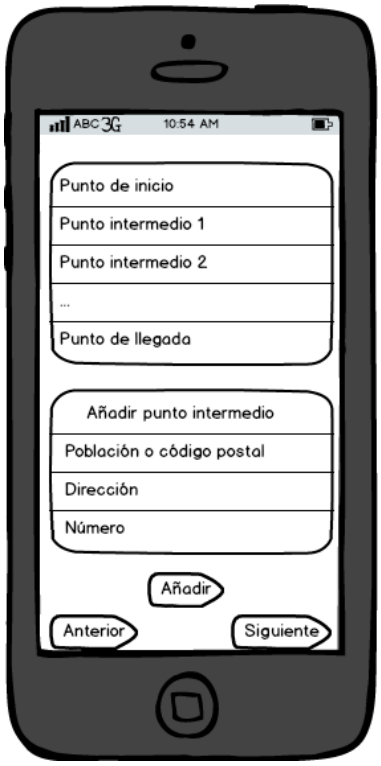
7.2 Prototipo horizontal de alta fidelidad

7.2.1 Pantalla de inicio



7.2.2 Pedir punto inicial

7.2.3 Pedir puntos intermedios



7.2.4 Pedir tipo de transporte

7.2.5 Mostrar ruta en mapa

7.2.6 Mostrar ruta con indicaciones

7.2.7 Cargar ruta guardada

Se podrá eliminar una ruta guardada haciendo el gesto de deslizar el dedo horizontalmente sobre la ruta que se quiera suprimir.

8 Evaluación

Durante el proceso de evaluación, se muestra el prototipo al usuario para que lo analice detenidamente. A continuación, se le pide que simule cómo realizar algunas tareas básicas con la aplicación.

Finalmente, se le pasa un cuestionario con el propósito de recabar información sobre el diseño y sus posibles defectos o problemas.

8.1 Preguntas sobre el usuario

1. ¿Estás interesado o utilizas habitualmente mapas, de cualquier tipo y soporte, para orientarte?
2. ¿Conoces y utilizas la tecnología GPS?
3. ¿Te ocurre con frecuencia que tienes que preguntar para llegar a tu destino?
4. ¿Tienes un teléfono iPhone?
5. Si es así, ¿utilizas con frecuencia alguna de las aplicaciones de mapas disponibles?
6. ¿Utilizas el GPS en el coche?
7. ¿Utilizas el GPS a pie?
8. ¿Utilizas la búsqueda de Google Maps para conocer la ruta a un destino?
9. ¿Has utilizado o utilizas con frecuencia, la búsqueda de Google Maps de ruta en transporte público?

8.2 Tareas a realizar por el usuario

1. Crear una ruta nueva entre tres puntos.
2. Modificar manualmente el orden de visita.
3. Guardar la ruta.
4. Abrir la ruta guardada.
5. Utilizar el cálculo de ruta óptima para reorganizar el orden de visitas.
6. Guardar de nuevo, sobrescribiendo la ruta anterior.

8.3 Preguntas sobre las tareas

1. ¿Has tenido algún problema la tarea?
2. ¿Crees que hay demasiados pasos en la tarea?
3. ¿Falta o sobra algún botón o elemento de menú en la pantalla?
4. ¿Es importante, opcional o prescindible la tarea?

8.4 Resultado del test

He realizado el test a algunos usuarios potenciales, mediante el cual he obtenido información suficiente de los usuarios potenciales como para ver que existe la posibilidad de una mejora sustancial sobre el modelo inicial.

8.4.1 Análisis de resultados

En general, parece que la aplicación es sencilla y clara de utilizar, como principales objetivos de diseño. Los usuarios no han tenido mayores problemas para la realización de las tareas.

He visto, no obstante, que el diseño faltan las pantallas de guardar y cargar mapas. Además, no es posible, tal como se presentan las pantallas correspondientes, realizar el cálculo de mejor ruta sobre rutas ya creadas (falta el botón correspondiente). Tampoco es posible la edición de una ruta existente.

Por tanto, se han detectado una serie de posibles mejoras, que se pasan a enumerar a continuación.

8.4.2 Mejoras propuestas

Se han detectado una serie de mejoras importantes en función de los resultados del test. Estas mejoras suponen una refactorización profunda del diseño inicial de la aplicación, que queda esbozada a continuación.

- Eliminar la pantalla de inicio. Se comenzaría con la vista de Mapa sin ninguna ruta, en la posición actual del usuario. Desde el menú se podrá acceder a las tres opciones de Nueva ruta, Guardar ruta (atenuada) y Cargar ruta.
- Fusionar las pantallas siguientes:
    - Pedir punto inicial
    - Pedir puntos intermedios
    - Pedir tipo de transporte"

  En una sola "Puntos de ruta", donde se contiene toda la información de las tres.
  Después de las simplificaciones propuestas, nos quedarían las siguientes vistas:
    - Mapa
    - Indicaciones
    - Puntos de ruta
    - Guardar / Cargar rutas.
- En las vistas de Mapa y de Indicaciones, sustituir "Nueva ruta" por opción "Editar ruta", que nos lleve a la pantalla de "Puntos de ruta". Aquí podremos añadir, quitar o reordenar los puntos de la ruta, cambiar el tipo de transporte y activar o desactivar el cálculo de la ruta óptima. En puntos de ruta tendremos las siguientes opciones:
    - Tipo de transporte
    - Ruta óptima: ordena automáticamente los puntos existentes.
    - Ruta circular: termina en el punto inicial.
- En la pantalla de Puntos de ruta, poder indicar la ubicación actual como un punto de ruta.
- Si es posible, además de la información de distancia y tiempo estimados, añadir precisión de la ubicación actual en metros.
- Poder acceder a guardar o cargar ruta también desde la vista "Puntos de ruta".

- En las tres vistas (Mapa, Indicaciones o Puntos de ruta), ver el nombre de la ruta cargada (o Ruta sin guardar, si no está guardada).
- Marcar en otro color los tramos ya recorridos.

9 Implementación

9.1 Herramientas utilizadas

Finalmente, aunque he podido conseguir una licencia de Xamarin, he decidido utilizar Xcode. He visto que el tiempo necesario para conocer mínimamente Xamarin era demasiado y no me compensaba lo suficiente.

En principio, la razón principal para intentar utilizar Xamarin era la de experimentar la programación iOS con F#, ya que estoy interesado en la programación funcional. Pero he visto que se lanzaba la versión 2 de Swift con muchas mejoras y es un lenguaje funcional con muchos parecidos con F#.

Por estas razones, he creído que el mejor camino era utilizar las herramientas nativas de Apple.

Por otro lado, al no disponer de ordenador Mac, he utilizado para la ocasión una virtualización de OS X El Capitán sobre VMware Workstation. Como contrapartida, el sistema es lento y el simulador va también muy lento.

Para intentar solventar el bajo rendimiento del simulador, me actualicé a la última versión de Xcode, en concreto la versión 7.1.1 ya que con esta versión es posible conectar un dispositivo de Apple para realizar pruebas en entorno real sin tener licencia de desarrollador. Pero me he encontrado con el problema de que el puerto USB bloquea totalmente el sistema operativo virtual, así que nunca he podido instalar la aplicación en mi iPhone.

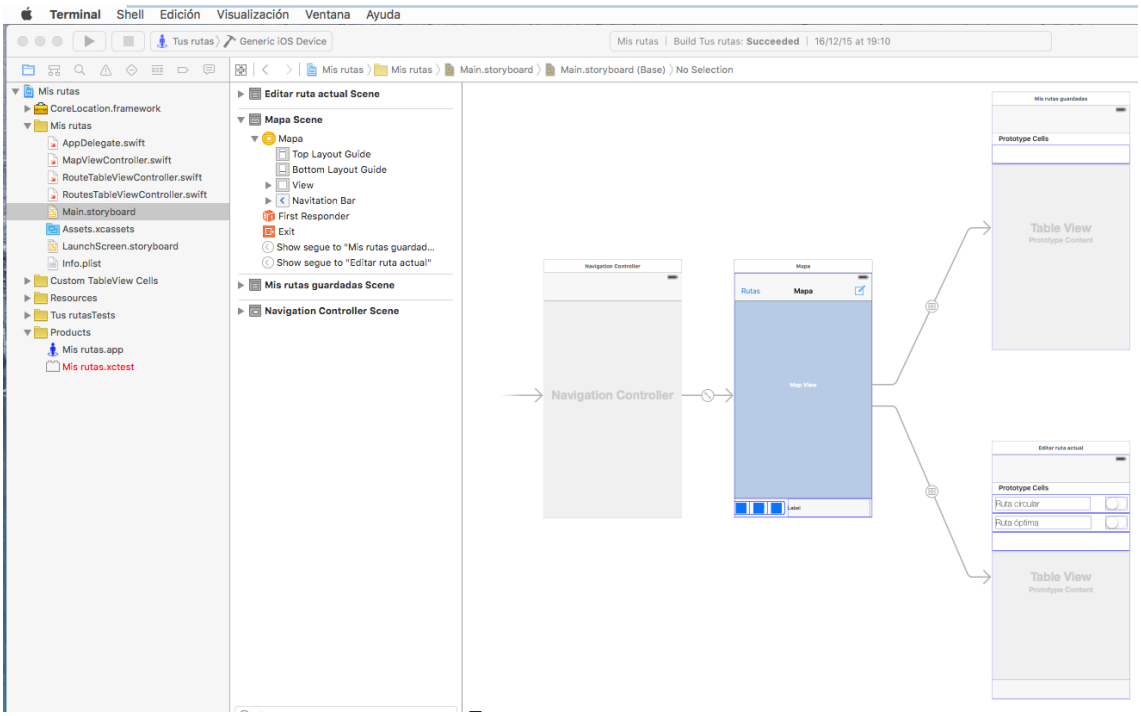
9.1.1 Xcode

Xcode es el entorno de desarrollo integrado de Apple y se suministra gratuitamente junto con Mac OS X. Incluye diversos compiladores de distintos lenguajes, aunque yo he utilizado únicamente Swift.

Xcode facilita el uso del patrón de diseño MVC (Modelo Vista Controlador) mediante el uso de Storyboards. El *Storyboarding* permite implementar la lógica de navegación entre vistas sin tener que escribir código. En su lugar, podemos ver gráficamente cada una de las vistas de nuestra aplicación y conocer cuál es la navegación que se establece entre cada una de ellas.

Por ejemplo, en el caso de la aplicación Mis Rutas, vemos en la imagen que existen líneas de navegación (llamadas *Segue*, pronunciado *Seg-way*), entre La vista de *Mapa* y de *Mis rutas*

guardadas y Entre *Mapa* y *Editar ruta actual*, pero no entre *Mis rutas guardadas* y *Editar ruta actual*.



Por tanto, utilizando *storyboarding* y sin una sola letra de código es posible establecer cuál es la navegación para cada uno de los elementos de la interfaz e, incluso, personalizar la forma en que se produce la transición entre los mismos (con una serie de efectos que vienen predefinidos y pudiendo establecer los nuestros personalizados).

9.1.2 Swift

Dentro de Xcode he tenido que elegir entre utilizar Objective-C o el nuevo lenguaje lanzado en 2014 por Apple.

Me he decantado finalmente por Swift (versión 2) por ser un lenguaje más parecido a C# o F#, lenguajes que ya conozco. También por mi interés por la programación funcional, ya que Swift incorpora muchas características de este paradigma de programación.

Swift es un lenguaje fuertemente tipado, aunque la declaración de tipo no siempre es necesaria gracias a su capacidad de inferencia de los mismos.

Las razones que me han llevado a elegir Swift y dejar de lado Objective-C son las siguientes. Primero, es más robusto y seguro y, por tanto, su uso reduce el número de errores de programación.

Además, es un lenguaje más moderno y potente y con más proyección de futuro. Aunque su sintaxis puede ser más compleja y su curva de aprendizaje es mayor, el esfuerzo extra se ve compensado por las características mencionadas.

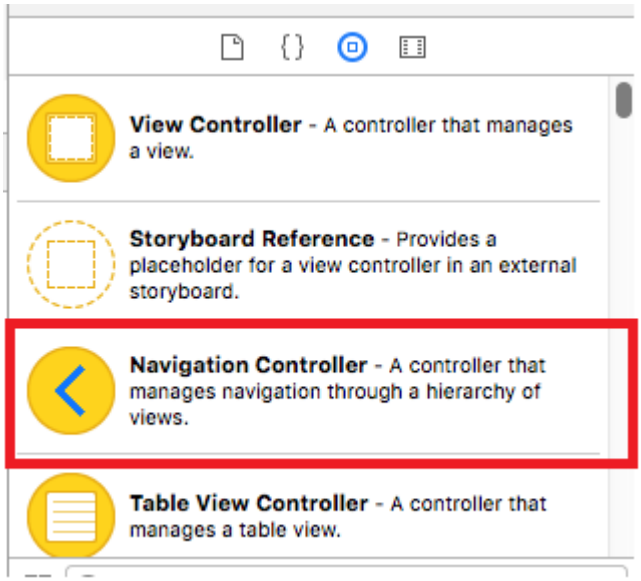
9.1.3 SDK iOS 9

El SDK (*Software Development Kit*) de iOS consiste en un conjunto de librerías y frameworks a disposición de los desarrolladores que permiten desarrollar una gran variedad de aplicaciones de distintos tipos.

En concreto, para una aplicación de mapas existen dos frameworks fundamentales, Core Location y Map Kit y que he utilizado en la aplicación desarrollada para obtener la posición y mostrar el mapa y las rutas y puntos en el mismo.

9.2 Proceso de desarrollo

Para no tener que empezar totalmente desde cero a diseñar las vistas y la navegación entre ellas, he utilizado un elemento de Xcode incluido en la Biblioteca de objetos de Xcode llamado *Navigation Controller*.

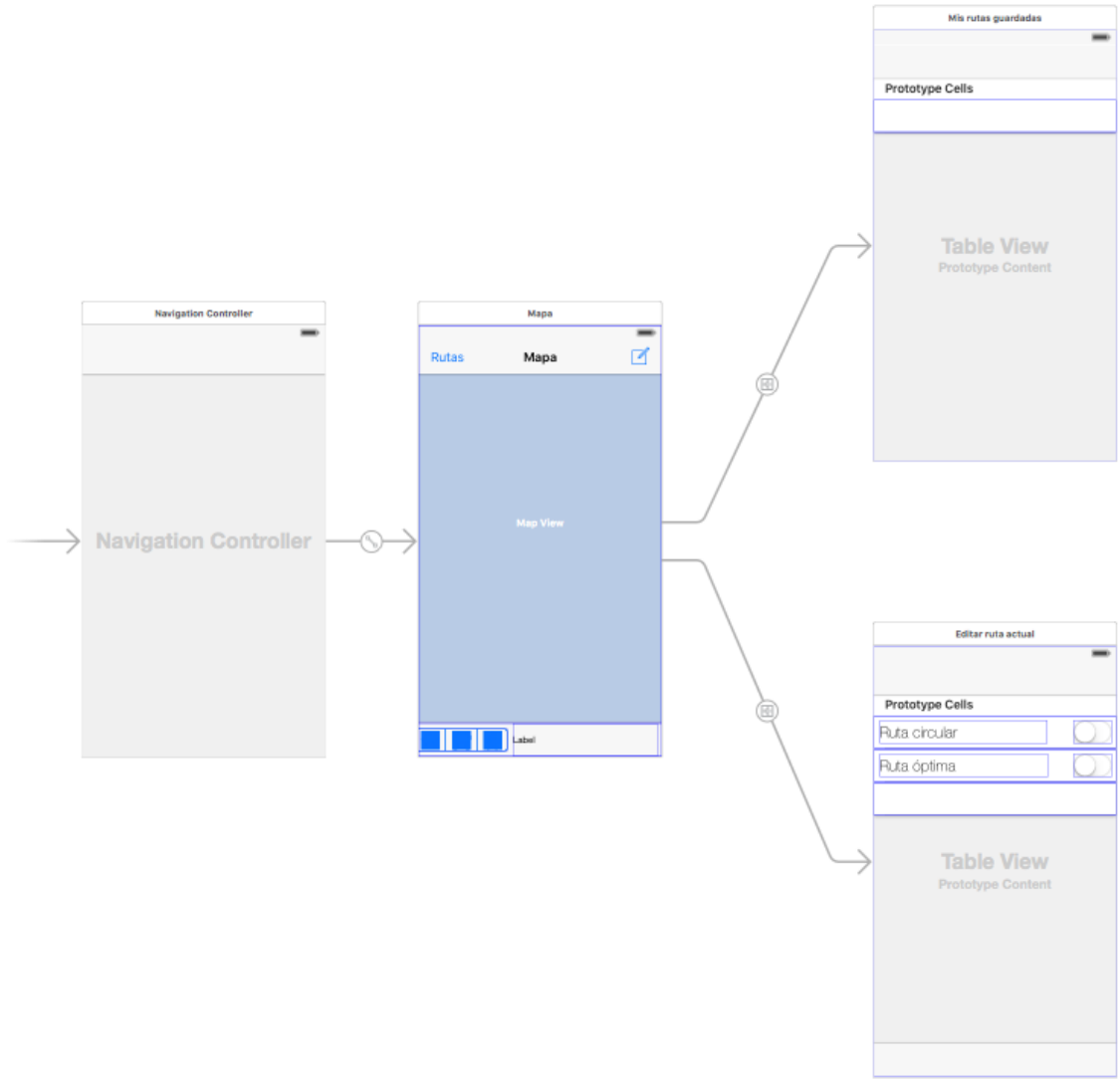


Un *Navigation Controller* gestiona las transiciones hacia adelante y hacia atrás entre una serie de *View Controllers*. El conjunto de *View Controllers* gestionados por un *navigation controller* concreto se denomina una pila de navegación (*navigation stack*).

Por tanto, este es el elemento base y contenedor de las otras vistas que utilizo en la aplicación.

En mi aplicación la pila de navegación incluida en el *Navigation Controller* se compone de tres *View Controllers*, que son:

- Mapa
- Mis rutas guardadas
- Editar ruta actual



Cada una de estas vistas se corresponde con una clase controladora:

- Mapa -> MapViewController.swift
- Mis rutas guardadas -> RoutesTableViewController
- Editar ruta actual -> RouteTableViewController

A continuación, explicaré con más detalle la responsabilidad de cada una de las tres clases controladoras.

9.2.1 MapViewController

Esta clase deriva de UIViewController, que es la clase base controladora proporcionada por la librería UIKit de iOS.

Como ya he mencionado, esta es la clase controladora de la vista principal Mapa, que se muestra al inicio de la aplicación y que es la que contiene el mapa con la posición actual o la ruta cargada, además de una barra de navegación en la parte superior y una *Toolbar* en la parte inferior con los botones cambiar de tipo de ruta y un elemento *Label* que muestra información contextualizada.

Esta es la clase principal de la aplicación y se ocupa principalmente de:

- Mostrar el mapa con la información adecuada sobre posición y ruta.
- Contiene en memoria la estructura de datos que utilizarán las otras vistas para mostrar información de rutas y puntos de paso.
- Se encarga del almacenamiento de forma permanente de las rutas guardadas.
- Derivado del primer punto, esta clase implementa funciones de delegado de las librerías Core Location y MapKit, necesarias para poder gestionar las rutas y el GPS del dispositivo.

Core Location es una de las frameworks más pequeñas de las disponibles en el SDK (*Software Development Kit*). Sirve para detectar la posición del dispositivo, pero la posición en sentido geográfico: latitud, longitud y altura. No es la posición espacial del dispositivo (si está horizontal o vertical, por ejemplo) que, en este caso, sería la librería Core Motion la que se encargaría de esta proporcionar información.

Las principales clases que he utilizado de esta librería son CLLocation y CLLocationManager.

CLLocation encapsula la posición del dispositivo y CLLocationManager obtiene la posición del dispositivo y nos la envía a través de un delegado. Es por esta razón que MapViewController se declara a sí misma como delegado de CLLocationManager y así poder recibir periódicamente instancias de CLLocation con la información de la posición geográfica del dispositivo.

Cuando utilizamos esta librería es muy importante tener en cuenta el consumo de batería, ya que la obtención de la posición y, por tanto, el uso del GPS del dispositivo que está asociado a un enorme consumo de batería.

La precisión es directamente proporcional al consumo de batería. Esos son los tres métodos para obtener la posición:

- GPS: más preciso, mayor consumo de batería.
- BD de nodos wifi: memos preciso, consume menos batería.
- Triangulación de antenas móviles: menor precisión, menor consumo.

En concreto, los métodos de geolocalización ofrecidos por Core Location son los siguientes:

1. GPS - kCLLocationAccuracyBestForNavigation
2. GPS - kCLLocationAccuracyBest
3. GPS - kCLLocationAccuracyNearestTenMeters
4. WIFI (o GPS en zonas rurales) - kCLLocationAccuracyHundredMeters
5. Antena de celda - kCLLocationAccuracyKilometer
6. Antena de celda – kCLLocationAccuracyThreeKilometers

Nosotros pedimos la precisión y Core Location busca la manera de conseguir esa precisión con el menor consumo de batería posible.

Por tanto, el mecanismo que he utilizado para gestionar esta librería es el siguiente:

1. Al cargar la vista de mapa, creo un CLLocationManager.
2. Me declaro a mí mismo (como clase controladora) delegado del manager.
3. Indico la precisión deseada.
4. Cuando he obtenido la posición (un objeto CLLocation) dejo de recibir más datos para ahorrar batería.

A continuación, muestro el código que implementa el proceso descrito:

```
// MARK: - Main View Controller

class MapViewController: UIViewController, CLLocationManagerDelegate, MKMapViewDelegate {
    var defaultTitle = "Mapa"
    var manager : CLLocationManager!
    let latKey = "lat"
    let lonKey = "lon"

    @IBOutlet weak var map: MKMapView!
    @IBOutlet weak var info: UILabel!
    @IBOutlet weak var navitationBar: UINavigationItem!
    @IBOutlet weak var transport: UISegmentedControl!

    // MARK: - Overrides

    override func viewDidLoad() {
        super.viewDidLoad()

        navitationBar.title = defaultTitle

        map.delegate = self
        manager = CLLocationManager()
        manager.delegate = self
        manager.desiredAccuracy = kCLLocationAccuracyBest
```

Como vemos, la clase MapViewController realiza las acciones 1, 2 y 3 en la función sobrescrita de UIViewController `viewDidLoad()`, que se invoca después de cargar la vista.

Cuando la vista va a desaparecer o cuando ya hemos obtenido la información de nuestra posición, paramos el envío de información para ahorrar batería (acción 4):

```
    override func viewWillDisappear(animated: Bool) {
        manager.stopUpdatingLocation()
    }
```

```
func locationManager(manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {

    // Obtiene la ubicación actual del usuario
    let userLocation: CLLocation = locations[0]
    let latitude = userLocation.coordinate.latitude
    let longitude = userLocation.coordinate.longitude
    let latDelta: CLLocationDegrees = 0.005
    let lonDelta: CLLocationDegrees = 0.005
    let location: CLLocationCoordinate2D = CLLocationCoordinate2DMake(latitude, longitude)
    let span:MKCoordinateSpan = MKCoordinateSpanMake(latDelta, lonDelta)
    let region:MKCoordinateRegion = MKCoordinateRegionMake(location, span)
    map.setRegion(region, animated: true)

    var userPositionName = "Posición del usuario"

    CLGeocoder().reverseGeocodeLocation(userLocation) { (placemarks, error) -> Void in
        if (error == nil) {
            if let placemark = placemarks?[0] {
                var formattedAddress : String = ""

                // Calle y número
                if placemark.name != nil {
                    formattedAddress = placemark.name! + "\n"
                    userPositionName = placemark.name!
                }
                if placemark.postalCode != nil {
                    formattedAddress = formattedAddress + placemark.postalCode! + " "
                }
                if placemark.locality != nil {
                    formattedAddress = formattedAddress + placemark.locality! + ", "
                }

                // Provincia
                if placemark.administrativeArea != nil {
                    formattedAddress = formattedAddress + placemark.administrativeArea! + ", "
                }
                if placemark.country != nil {
                    formattedAddress = formattedAddress + placemark.country!
                }
                self.info.text = formattedAddress

                // Update current location
                userPosition =  [routeNameKey:userPositionName, self.latKey:"\(latitude)", self.lonKey:"\(longitude)"]
            }
        }
    }
    manager.stopUpdatingLocation()
}
```

MapKit es otra framework que va de la mano de Core Location y que tendemos a confundir pensando que son lo mismo. Pero no es así, ya que Core Location pertenece a la capa del modelo (recordemos que iOS nos fuerza a utilizar el patrón MVC, modelo-vista-controlador) y no muestra nada al usuario (capa de la vista).

MapKit es la vista de este modelo representado por Core Location. Es un framework que nos permite mostrar estos datos de geolocalización en un mapa.

Las principales clases que he utilizado de esta librería son:

- MKMapView: muestra el mapa.
- MKAnnotation: muestra un punto en el mapa.
- MKDirections y MKDirectionsRequest: utilizadas para calcular la ruta y dibujarla en el mapa.

9.2.2 Estructura de datos y almacenamiento permanente

La estructura de datos necesaria para mostrar los datos relativos a las rutas es muy sencilla, por lo que he considerado suficiente utilizar User Defaults para guardar dicha estructura. Por tanto, no he necesitado utilizar el framework Core Data de base de datos.

En un principio, he visto que solo era necesario almacenar permanentemente las rutas diferentes rutas creadas por el usuario. Comencé a desarrollar la aplicación con un sencillo array de tuplas. Cada tupla representa una ruta y contiene dos elementos:

- Un String, con el nombre de la ruta que aparecerá en la lista de rutas guardadas.
- Un Array de diccionarios con clave String y valor String. Cada diccionario representa un punto, que tendrá siempre tres valores: nombre, latitud y longitud.

Así aparece en el código, dentro del fichero MapViewController.swift, antes de la declaración de la clase, para poder acceder a la estructura desde el resto de la aplicación (variable global):

```
// La estructura de datos se compone de:
// - Punto. Es representado por un diccionario: [String: String]. Por ejemplo, ["name":"Taj Mahal","lat":"27.175277","lon":"78.042128"]
// - Ruta (conjunto de puntos). Es representada con un array de diccionarios: [Dictionary<String, String>]
// - Conjunto de rutas guardas. Se represnta con un array de tuplas: [(String : [Dictionary<String, String>])]
var routes = Array<(routeName : String, route: Array<Dictionary<String, String>>)>()
```

En este momento, después de crear esta estructura no implementé aún el guardado en User Defaults y comencé con el desarrollo de la aplicación. Más adelante, cuando ya había avanzado bastante, quise probar el almacenamiento permanente y traté de guardar el array de tuplas en User Defaults, pero me encontré con el escollo de que la tupla es una estructura que no es posible guardar en User Defaults.

Los únicos tipos de datos que se pueden almacenar en User Defaults son:

- NSString
- NSNumber
- NSData
- NSDictionary
- NSArray
- NSDate

En este punto, tuve que decidir entre tres posibilidades:

- Usar Core Data
- Cambiar el array de tuplas por un array de diccionarios
- Crear métodos de serialización y deserialización que tradujera de array de tuplas a array de diccionarios y viceversa.

Finalmente, escogí la tercera opción, que creo que es la mejor. Primero, por evitar usar la complejidad de Core Data para una estructura tan sencilla.

La segunda opción me obligaba a realizar muchos cambios en código ya escrito y disminuir la legibilidad del mismo, ya que el acceso y escritura a un array de diccionarios es más compleja que en el caso de las tuplas.

Por tanto, lo que hice para solventar el problema es guardar la lista de rutas como un array de diccionarios y, al arrancar la aplicación e ir a leer los datos guardados, traducir dicho array de diccionarios al array de tuplas que se utiliza en las distintas clases de la aplicación. Este es el código que realiza esta traducción:

```
// No es posible guardar tuples con StandardUserDefaults.
// Workaround: convertir a diccionario y guardar.
// http://stackoverflow.com/questions/27896642/save-a-tuple-in-nsuserdefaults
private typealias AccessTuple = (routeName: String, route: Array<Dictionary<String, String>>)
private typealias AccessDictionary = [String: Array<Dictionary<String, String>>]

// MARK: - Global functions

private func serializeTuples(tuples: [AccessTuple]) -> [AccessDictionary] {
    var result = [AccessDictionary]()

    if tuples.count > 0 {
        for index in 0...tuples.count - 1 {
            let dict = [tuples[index].routeName : tuples[index].route]
            result.append(dict)
        }
    }

    return result
}

private func deserializeDictionaries(dictionaries: [AccessDictionary]) -> [AccessTuple] {
    var result = [AccessTuple]()

    if dictionaries.count > 0 {
        for index in 0...dictionaries.count - 1 {
            let dictionary = dictionaries[index]
            let key = Array(dictionary.keys)[0]
            let tuple = (key, dictionary[key]!)
            result.append(tuple)
        }
    }

    return result
}

func loadRoutes() {
    let savedRoutes = NSUserDefaults.standardUserDefaults().objectForKey(routesKey) as! [AccessDictionary]?
    if (savedRoutes != nil) {
        routes = deserializeDictionaries(savedRoutes!)
    }
}

func saveRoutes() {
    let routesToSave = serializeTuples(routes)
    NSUserDefaults.standardUserDefaults().setObject(routesToSave, forKey: routesKey)
}
```

Para cargar las rutas guardadas en implementado la función `application` de la clase AppDelegate. Esta función se ejecuta después de que se ejecuta la aplicación.

```
//
//  AppDelegate.swift
//  Tus rutas
//
//  Created by Javier Fernández Cabanas on 5/12/15.
//  Copyright © 2015 Javier Fernández Cabanas. All rights reserved.
//

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        // Override point for customization after application launch.
        loadRoutes()
        return true
    }
```

La función saveRoutes() se invoca cada vez que se realiza un cambio en las rutas.

9.2.3 RoutesTableViewController

Esta clase deriva de UITableViewController que, a su vez, hereda de UIViewController. Por tanto, es una vista con estructura de tabla, adecuada para mostrar listas de elementos. Xcode facilita la implementación de este tipo de vistas, ya que UIViewController contiene una serie de funciones a implementar y que permiten mostrar la tabla de forma adecuada a nuestras necesidades.

Las vistas consumen datos a mostrar. En este caso, la estructura de datos fuente está en la clase principal, como ya hemos indicado en el apartado anterior.

Las funciones que definen la tabla "Rutas guardadas" son las siguientes:

```
// MARK: - Table view data source

override func numberOfSectionsInTableView(tableView: UITableView) -> Int {
    return 1
}

override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return routes.count
}

override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCellWithIdentifier("routeCell", forIndexPath: indexPath)

    cell.textLabel?.text = routes[indexPath.row].routeName + " (\(routes[indexPath.row].route.count))"

    // Selecciona la ruta activa
    if (activeRoute == indexPath.row) {
        cell.accessoryType = UITableViewCellAccessoryType.Checkmark
    } else {
        cell.accessoryType = UITableViewCellAccessoryType.None
    }

    return cell
}
```

En esta tabla solo utilizamos una sección, por tanto la implementación de la primera función es trivial y se devuelve un 1.

La segunda función es la que indica el número de filas de la tabla y devuelve el número de elementos de `routes`, que es la estructura que guarda las rutas, como ya vimos antes.

La tercera función es la que rellena con datos cada fila y marca la ruta activa con una palomita (si existe), que es la ruta cargada en el mapa.

Como se ve en este método, he utilizado una variable global llamada `activeRoute` de tipo `int` para guardar la ruta activa. El valor de esta variable se corresponde con la el número de fila de la tabla donde se encuentra la ruta activa.

En caso de que solo se muestre la posición del usuario y no exista ninguna ruta activa guardada, el valor de `activeRoute` es -1.

9.2.4 RouteTableViewController

Esta clase también hereda de UITableViewController. Por tanto, como en el caso anterior, es una vista con estructura de tabla, adecuada para mostrar listas de elementos.

Esta tabla solo es accesible si existe una ruta activa y muestra los puntos de dicha ruta y opciones de visualización.

Para ello, he utilizado dos secciones, por lo que su definición es un poco más compleja.

La primera sección muestra dos switches para indicar si queremos mostrar una ruta circular y si queremos que se calcule la mejor ruta automáticamente.

La segunda sección muestra los puntos de la ruta y permite eliminarlos o cambiar el orden manualmente con una pulsación larga y arrastrando.

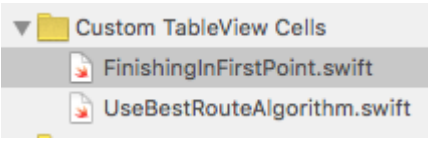
Se muestran a continuación las funciones que he sobrescrito para definir las secciones de la tabla:

```
// MARK: - Table view data source

override func numberOfSectionsInTableView(tableView: UITableView) -> Int {
    return 2
}

override func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
    var sectionTitle = ""

    if (activeRoute > -1 && section == 1) {
        sectionTitle = "Recorrido"
    }

    return sectionTitle
}

override func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath: NSIndexPath) -> CGFloat {
    if (indexPath.section == 0) {
        return 50
    }

    return 44
}
```

En este caso, con la primera función se indica que el número de secciones es 2. La segunda función, `tableView` con el parámetro `titleForHeaderInSection` nos permite indicar un título para cada sección. En este caso, solo se indica un título para la segunda sección, “Recorrido”.

La tercera función, `tableView` con parámetro `heightForRowAtIndexPath` ha sido sobrescrita para configurar el alto de las filas de la cada sección de la tabla.

En la primera sección se colocan los switches. Cada uno de ellos está asociado con un controlador contenido en una clase derivada de UITableViewCell.




```
//
//  FinishingInFirstPoint.swift
//  Mis rutas
//
//  Created by Javier Fernández Cabanas on 12/12/15.
//  Copyright © 2015 Javier Fernández Cabanas. All rights reserved.
//

import UIKit

class FinishingInFirstPoint: UITableViewCell {

    @IBOutlet weak var circularLabel: UILabel!

    @IBOutlet weak var switchButton: UISwitch!

    @IBAction func circularButtonChanged(sender: UISwitch) {
        isCircular = sender.on
    }

    override func awakeFromNib() {
        super.awakeFromNib()
        switchButton.on = isCircular
    }

    override func setSelected(selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

}
```

```
//
//  UseBestRouteAlgorithm.swift
//  Mis rutas
//
//  Created by Javier Fernández Cabanas on 12/12/15.
//  Copyright © 2015 Javier Fernández Cabanas. All rights reserved.
//

import UIKit

class UseBestRouteAlgorithm: UITableViewCell {

    @IBOutlet weak var autoOrderLabel: UILabel!
    @IBOutlet weak var autoOrderButton: UISwitch!

    override func awakeFromNib() {
        super.awakeFromNib()
        autoOrderLabel.enabled = false
        autoOrderButton.enabled = false
    }

    override func setSelected(selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

}
```

Por último, es necesario definir también las funciones necesarias para indicar el número de filas de cada sección y el contenido de las mismas.

```
override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

    var result = 0

    if (section == 0) {
        result = rowsInFirstSection
    } else if (activeRoute > -1) {
        if (section == 0) {
            result = rowsInFirstSection
        } else if (section == 1) {
            result = routes[activeRoute].route.count
        }
    }

    return result
}

override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
    var cell = UITableViewCell()

    if (indexPath.section == 0) {
        switch indexPath.row {
        case 0:
            cell = tableView.dequeueReusableCellWithIdentifier("circular", forIndexPath: indexPath) as! FinishingInFirstPoint
        case 1:
            cell = tableView.dequeueReusableCellWithIdentifier("autoOrder", forIndexPath: indexPath) as! UseBestRouteAlgorithm
        default:
            cell = UITableViewCell()
        }
    }

    if (indexPath.section == 1) {
        cell = tableView.dequeueReusableCellWithIdentifier("pointCell", forIndexPath: indexPath)
        var text = ""
        if (activeRoute > -1) {
            text = routes[activeRoute].route[indexPath.row][routeNameKey]!
        }

        cell.textLabel?.text = text
    }

    return cell
}
```

Como vemos, el número de filas de la primera sección está definido por una constante cuyo valor es 2. El número de filas de la segunda sección es igual al número de puntos de la ruta actual.

En la segunda función podemos observar cómo se define el contenido de cada fila de forma totalmente distinta para cada sección.

En la primera sección con dos filas, se rellena cada una con el *switch* correspondiente.

En la segunda se rellena con los valores de los puntos de la ruta activa.

9.3 Producto final

En primer lugar, tengo que decir que no he implementado todas las funcionalidades previstas, por falta de tiempo. He dedicado una gran parte del tiempo a aprender desarrollo en iOS y Xcode. He tomado tres cursos online de forma parcial, ya que no tenía tiempo para completarlos. Los cursos son los siguientes, los tres de udemy:

1. Programación iOS para iPhone y iPad, impartido por Fernando Rodríguez B
2. Programación iOS Avanzada ¡De Padawan a Jedi!, impartido por Fernando Rodríguez B
3. The Complete iOS8 and Swift Course: Learn by Building 15 Real World Apps, impartido por Rob Percival.

Estos cursos han sido mi principal guía para seguir buenas prácticas. También tienen multitud de código de ejemplo que me ha sido muy útil.

Por tanto, no he podido llegar a cumplir todos los objetivos de implementación. Por ejemplo, solo he desarrollado tres vistas y no cuatro como había planificado inicialmente. En concreto, no existe la vista de indicaciones. Paso a describir brevemente las tres vistas.

9.3.1 Pantalla de inicio en vista de mapa

- El usuario puede cargar y guardar rutas. Finalmente, he utilizado una estructura sencilla que he podido guardar en User Defaults y no ha sido necesario utilizar CoreData.
- El usuario crea los puntos con una pulsación larga, no es necesario escribir la dirección.
- La aplicación guarda la ruta con un nombre por defecto y calcula el camino entre los puntos, pudiendo elegir el usuario entre tipo de transporte.
- Si no hay ninguna ruta cargada, el usuario puede ver en la parte inferior la dirección en la que se encuentra actualmente.
- Si la ruta contiene solo un punto de salida y uno de llegada, se presenta información sobre el tiempo de llegada estimado y la distancia a recorrer.

9.3.2 Pantalla de rutas guardadas

En esta pantalla está la lista de rutas que se van guardando automáticamente. Aquí se ve marcada con una palomita la ruta cargada actualmente. Podemos cargar o descargar cualquier ruta pulsando sobre ella. También podemos borrar con un gesto de deslizamiento horizontal hacia la izquierda (*swipe*) cualquiera de los elementos de la lista.

9.3.3 Pantalla de ruta actual

En esta pantalla podemos ver los puntos de paso del recorrido cargado y podemos también eliminar puntos o cambiar el orden manualmente, manteniendo pulsado sobre el ítem (*press*) y desplazándolo a la posición deseada.

También hay dos opciones:

- Ruta circular: desactivada por defecto, calcula la ruta con punto de llegada en origen.
- Ruta óptima: opción desactivada. Con esta opción el usuario invoca el algoritmo que calcula la ruta más rápida entre todos los puntos de la lista. No he tenido tiempo de implementar el algoritmo. Quizá sería la parte más interesante del proyecto, así que no descarto terminar la implementación en el futuro.

9.3.4 Pendiente de desarrollar en el futuro

- Cálculo de ruta óptima.
- Vista de indicaciones de ruta.
- Edición del texto de puntos y nombres de ruta.
- Introducción textual de direcciones.
- Botón para mostrar posición actual.
- Numeración de puntos del recorrido.
- Mostrar previsión de llegada y distancia del recorrido para cualquier número de puntos en la ruta.

10 Pruebas

Las pruebas se realizarán de tres formas:

1. Sobre el simulador o sobre el dispositivo. Es la forma principal de prueba y de uso constante durante la implementación.
2. Pruebas unitarias. Se diseñará una prueba por función, tratando de cubrir las más importantes.
3. Tests de Interfaz de Usuario automáticos. Xcode permite escribir tests de IU capturando nuestra interacción con la aplicación y guardando esa información como código de test para que después podamos introducir "*assertions*".

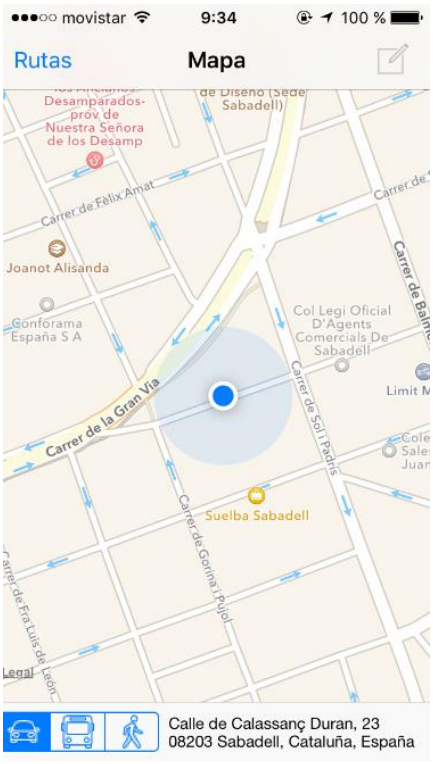
11 Funcionamiento de la aplicación

La aplicación final entregada se denomina “Mis rutas” y se compone de tres pantallas.

- La pantalla principal mostrando el mapa con una ruta o simplemente mostrando la posición del dispositivo.
- La pantalla de rutas guardadas en el dispositivo, donde se muestra una lista de rutas editable. Se accede con el botón “Rutas” situado en la parte superior izquierda de la pantalla principal.
- La pantalla de puntos de la ruta activa, accesible mediante el botón situado en la parte superior derecha de la pantalla y solo disponible si existe una ruta cargada sobre el mapa. En esta pantalla podemos eliminar puntos de la ruta activa, así como activar o desactivar dos opciones: ruta circular y ruta óptima, que explicaremos más adelante.

A continuación, se explica el funcionamiento de la aplicación mediante capturas de pantalla y una breve explicación.

La aplicación se abre como cualquier otra aplicación de un iPhone, pulsando sobre el icono de la aplicación “Mis rutas”.



La pantalla inicial de la aplicación es siempre un mapa mostrando la posición actual del usuario. Si es la primera ocasión que se utiliza, el sistema pedirá permiso al usuario para utilizar el GPS y así poder obtener información precisa de la ubicación del dispositivo.

Inicialmente también se muestra en la barra situada en la parte inferior de la pantalla, la dirección postal de la posición actual.

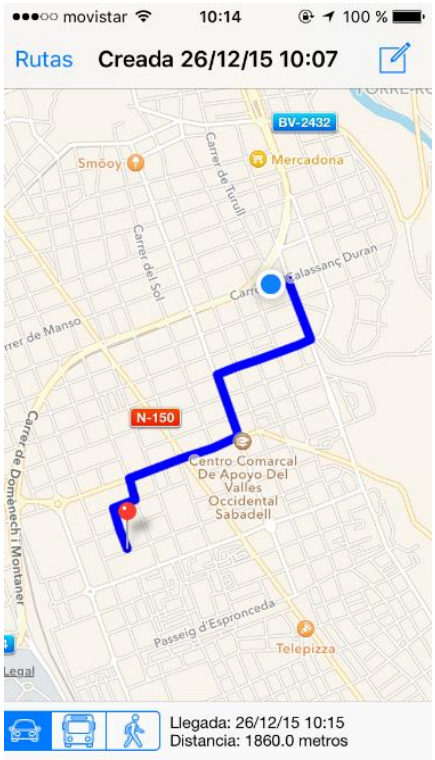
Con el gesto de pellizcar (*pinch*, en inglés), podemos aumentar (*zoom in*) o reducir (*zoom out*) la escala del mapa según nuestras necesidades.

También podemos mover el mapa con un gesto de arrastrar (*drag*, en inglés).

11.1 Creación de una ruta nueva

Con una pulsación larga en un lugar del mapa, añadimos un punto de ruta y la aplicación calcula el camino entre nuestra posición actual y el punto marcado.

En el lugar donde antes se mostraba la dirección postal, ahora aparece información acerca de la distancia entre ambos puntos y la hora estimada de llegada.



Podemos añadir tantos puntos como queramos, aunque la información de la distancia y la hora de llegada solo se muestran en el caso de que la ruta conste de dos puntos.

11.2 Selección del modo de transporte

Podemos elegir entre dos modos de transporte, en coche o andando. Pulsando sobre el botón correspondiente se recalcula la ruta para el modo elegido. También se recalculan la hora de llegada y la distancia.

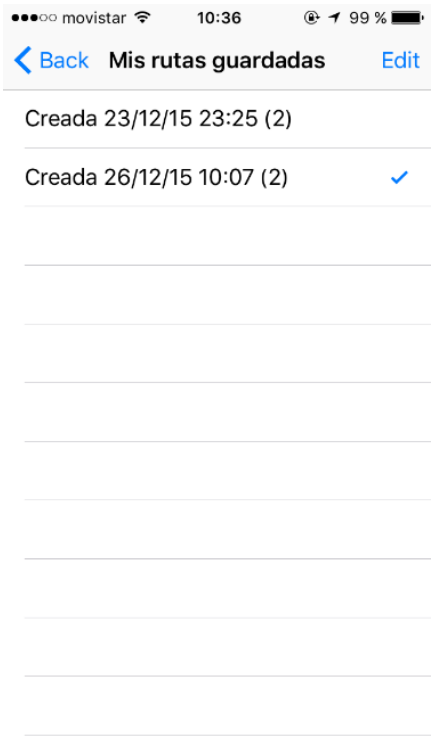
El modo de transporte público no está operativo, quedando en la lista de posibles mejoras futuras.

11.3 Guardado de rutas

Al añadir un punto o más en el mapa, automáticamente queda guardada la ruta creada en la memoria permanente del iPhone. El nombre por defecto se compone de la palabra “Creada” y de la fecha y hora de creación. Este nombre aparece en el centro de la barra de navegación situada en la parte superior de la pantalla.

Podemos acceder a todas las rutas que tenemos guardadas pulsando en el botón “Rutas” en la parte superior izquierda de la pantalla.

La aplicación nos muestra aquí todas las rutas guardadas y la que está cargada actualmente sobre el mapa (ruta activa) se muestra resaltada con una palomita (✓). Entre paréntesis se muestra el número de puntos del que está compuesto cada ruta.

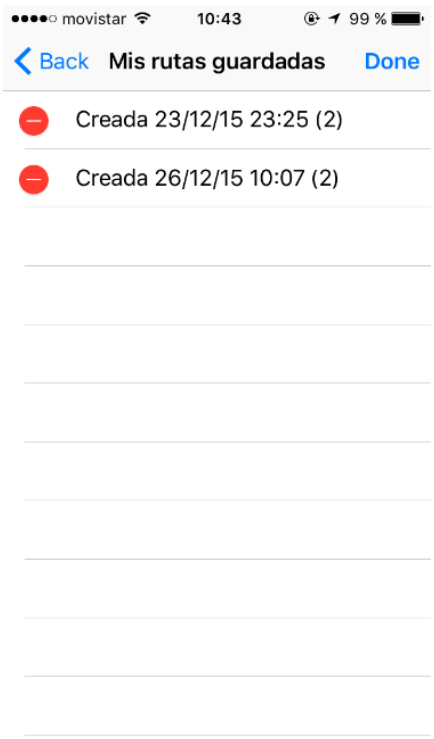


11.4 Edición de rutas guardadas

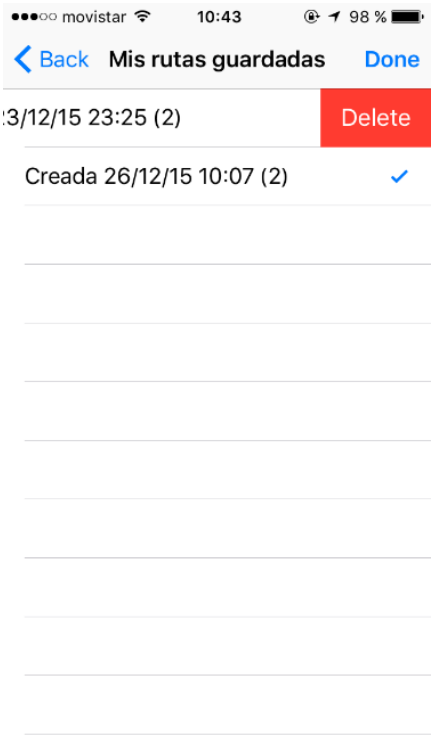
En la pantalla de rutas guardadas, podemos eliminar rutas de la lista así como cargar o borrar del mapa la ruta activa.

El borrado de una ruta existente se puede hacer de dos formas:

- Con el botón "*Edit*" en la parte superior izquierda de la pantalla.



- Con un gesto de arrastrar horizontal hacia la izquierda (*swipe*) sobre el elemento de la lista que queramos eliminar.



También es posible cambiar de ruta activa, eligiendo otra de la lista con una pulsación del dedo sobre ella.

También es posible borrar del mapa la ruta actual mostrando solo el mapa la posición del dispositivo, pulsando sobre la ruta activa marcada con la palomita.

11.5 Edición de los puntos de la ruta activa

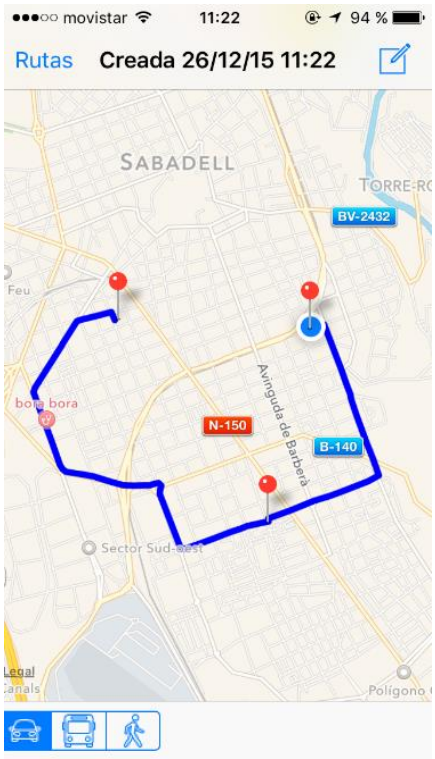
Esta pantalla está dividida en dos secciones. Una sección situada en la parte superior, que agrupa dos opciones:

- Ruta circular. Si está activa, la ruta se completa con la indicación para llegar al punto inicial desde el último punto de la ruta.
- Ruta óptima. Para el caso de que existan más de dos puntos en la ruta, la aplicación calcula la mejor ruta para visitar todos los puntos en el menor tiempo posible. Está desactivada porque no he tenido tiempo de implementar correctamente el algoritmo que realiza el cálculo. Sería la principal tarea en la lista de futuras mejoras.

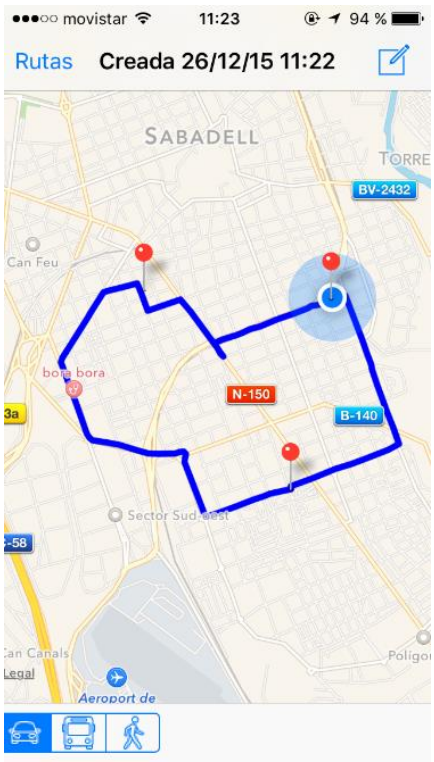
La segunda sección de esta pantalla muestra la lista de puntos de la ruta activa, pudiendo realizar dos operaciones sobre esta lista:

- Eliminar puntos. El usuario puede realizar esta operación de las dos mismas formas disponibles en la pantalla de rutas guardadas.
- Reordenar ruta. El usuario puede, mediante una pulsación larga (*touch and hold*, pulsar y mantener pulsado, en inglés) sobre un punto de la lista y moverlo para modificar el orden. Lo podríamos considerar como la versión manual de la opción “Ruta óptima”. De hecho, ambas operaciones son excluyentes, de modo que si se activase esta opción no debería ser posible reordenar manualmente.

Ruta con tres puntos no circular



Selección de ruta circular



12 Conclusiones

Con este proyecto he intentado simular todo el proceso que se llevaría a cabo en un proyecto profesional real. Esta simulación es de un gran valor para mí, porque me ha podido mostrar todo el esquema teórico a seguir y puede servir de punto de partida para realizar proyectos más serios.

Pero la gran diferencia, en mi opinión, con un verdadero proyecto profesional es la gran incertidumbre inicial en el caso de este proyecto, debido a que partía de cero en lo que se refiere a experiencia y conocimientos en programación de aplicaciones para móviles. Esto hace que las previsiones de tiempo y de las funcionalidades a implementar se hayan hecho "a ciegas" sin saber si realmente sería posible para mí llegar a hacerlas con el tiempo disponible.

Ante esta incertidumbre, he tratado de realizar un mínimo de funcionalidades bien desarrolladas y probadas, sobre una aplicación mínimamente sólida y libre de errores graves. Creo que este objetivo lo he logrado y he dejado una línea de desarrollo futura claramente definida y potencialmente realizable en un período de tiempo razonable.

A continuación, paso a detallar los objetivos realizados y la lista de funcionalidades pendientes.

12.1 Objetivos alcanzados

Creo que he conseguido una aplicación de extrema simplicidad pero también útil en la vida real, con una forma de uso distinta de Google Maps o Mapas de Apple, incluso con algunas funcionalidades que no veo que tengan estas aplicaciones en iOS, como añadir más puntos a la ruta o ruta circular.

Por tanto, creo que puede decirse que es sencilla y con cierta diferenciación de las grandes aplicaciones de mapas utilizadas por la mayoría de los usuarios. Para llegar a este objetivo me ha sido de gran utilidad el diseño centrado en el usuario y, especialmente, la evaluación desde el punto de vista del usuario. A través de la evaluación he cambiado sustancialmente el diseño inicial de la aplicación con un resultado mucho más sencillo, con menos pantallas y con una forma de uso mucho más intuitiva.

Por tanto, aunque no he podido desarrollar la pantalla de indicaciones, o una guía paso a paso centrada en la posición en tiempo real del dispositivo, sí he podido mostrar una visión general de la ruta con un gasto de batería muy adecuado, ya que una vez calculada la ruta o la posición la aplicación deja de utilizar el GPS.

Además, he tratado de seguir las técnicas de diseño y programación recomendadas por Apple y he conseguido que la aplicación se adapte a cualquier tamaño y orientación (vertical u horizontal) de pantalla, incluido la de un iPad.

12.2 Futuras mejoras

Como ya he indicado anteriormente, me ha quedado en el tintero una serie de funcionalidades por añadir. Algunas de ellas estaban previstas en el diseño inicial y no he podido alcanzarlas por

falta de tiempo. He tenido que dedicar mucho esfuerzo a aprender en poco tiempo las bases del desarrollo con Xcode y Swift, lo que me ha consumido gran parte del tiempo disponible para la implementación.

Por otro lado, durante el proceso de implementación y pruebas he visto también otras mejoras, en algunos casos relativamente fáciles de implementar, pero con valor añadido para el usuario como, por ejemplo, la previsión de llegada y distancia de recorrido para cualquier número de puntos. Esta funcionalidad no estaba prevista inicialmente y vi que se podía añadir gracias al MapKit de Apple con relativa sencillez para dos puntos, como finalmente ha sido implementado en la aplicación entregada. Queda pendiente para más de dos puntos.

Paso a detallar todas las futuras mejoras que he detectado.

12.2.1 Cálculo de ruta óptima

Esta es quizá la primera funcionalidad a añadir, ya que se trata de una de las funcionalidades principales inicialmente proyectadas. La parte gráfica está realizada y solo sería necesario acabar de crear un algoritmo que realice el cálculo de reordenación eligiendo la ruta menos costosa en tiempo o en distancia. Para un número elevado de puntos es un algoritmo de complejidad NP y por tanto, no trivial.

12.2.2 Vista de indicaciones de ruta

Se trata de la pantalla que muestra las indicaciones textuales paso a paso. Es una de las funcionalidades principales incluida en el diseño inicial y, por tanto, también muy importante. Sería necesario añadir una nueva pantalla y estudiar dónde situar el botón para acceder a ella.

12.2.3 Edición del texto de puntos y nombres de ruta

También creo que sería importante añadir la posibilidad de cambiar los nombres de las rutas y que el usuario pudiera poner un nombre más descriptivo.

12.2.4 Introducción textual de direcciones

En este caso se trata de una funcionalidad que ha cambiado a partir de la evaluación del usuario y del consejo del tutor, Antonio Rodríguez Gutiérrez en la evaluación de la PAC 2. Se trata de la forma de introducir puntos en la ruta. Inicialmente había pensado hacerlo de una forma muy estructurada indicando la dirección en varios pasos, como se haría en un formulario.

Finalmente, la única forma de hacerlo es marcando los puntos de paso con pulsaciones sobre el mapa. También se pueden reordenar posteriormente los puntos.

Pero también se podría añadir la posibilidad de introducir una dirección escrita.

12.2.5 Botón para mostrar posición actual

Se trata de añadir el botón que existe en cualquier aplicación de mapas que centra el mapa en la posición actual. Esto se realiza de forma automática al entrar en la aplicación o al borrar del

mapa la ruta activa, pero si el usuario desplaza manualmente el mapa no tiene una forma rápida de volver a su posición actual.

12.2.6 Numeración de puntos del recorrido

Esta opción no estaba entre los requisitos iniciales. Durante las pruebas he visto que es difícil en ocasiones ver el sentido del camino, especialmente cuando hay muchos puntos en un área no muy grande o con rutas circulares. Sería de gran ayuda numerar cada punto para ver todo el camino de forma ordenada.

12.2.7 Mostrar previsión de llegada y distancia del recorrido para cualquier número de puntos en la ruta.

Como ya comenté al principio de esta sección, esta funcionalidad no estaba prevista inicialmente y la he incluido para dos puntos por el poco esfuerzo que suponía y que creo que puede resultar muy útil.

Quedaría pendiente hacer el mismo cálculo para cualquier número de puntos.

12.2.8 Extracción de datos de tráfico

Inicialmente la había incluido entre las funcionalidades principales. Debido a la complejidad técnica percibida, la he tenido que descartar.

12.2.9 Acceso a rutas públicas de otros usuarios y rutas más populares

Del mismo modo que el punto anterior, he dejado esta funcionalidad sin implementar por la complejidad técnica y porque implica utilizar almacenamiento en la nube, teniendo que buscar un servicio que puede suponer un coste o investigar servicios gratuitos. Todo esto suponía un tiempo extra del que no disponía.

13 Bibliografía

Ginsburg, S. (2010). *Designing the iPhone User Experience.* Kendallville: Addison-Wesley.

14 Otras fuentes de información

14.1 Cursos online

- *Programación iOS para iPhone y iPad*, impartido por Fernando Rodríguez B. https://www.udemy.com/curso-de-programacion-ios-para-iphone-y-ipad/
- *Programación iOS Avanzada ¡De Padawan a Jedi!*, impartido por Fernando Rodríguez B. https://www.udemy.com/programacion-ios-avanzado-fernando-rodriguez/
- *The Complete iOS8 and Swift Course: Learn by Building 15 Real World Apps*, impartido por Rob Percival. https://www.udemy.com/complete-ios-developer-course/

14.2 Webs

- *Desarrollo Ios 7: Novedades en MapKit*. http://www.migueldiazrubio.com/2013/10/24/desarrollo-ios-7-novedades-en-mapkit/
- *Stackoverflow*. http://stackoverflow.com/questions/tagged/ios
- *Invasivecode. MapKit for iOS9.* https://www.invasivecode.com/weblog/mapkit-flyover-transit?doing_wp_cron=1449605197.7102599143981933593750
- *Maps for Developers*. https://developer.apple.com/maps/
- *Raywenderlick. MapKit Tutorial.* http://www.raywenderlich.com/90971/introduction-mapkit-swift-tutorial
- *Drag and Drop Cells in Swift using an UILongPressGestureRecognizer*. http://www.freshconsulting.com/create-drag-and-drop-uitableview-swift/