ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

"iPark: Aplicación sobre la Zona Azul para dispositivos iOS"

Mikel Elorz Berástegui

Dr. Mikel Izal Azcárate

Pamplona, 1 de julio de 2011

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Agradecimientos

Estos tres años de carrera han sido muy especiales y también duros. Se ha puesto a prueba tanto mi capacidad de aprender como mi capacidad de crecer como persona. Al final de esta primera etapa de mis estudios quería dedicar unas palabras de agradecimiento a las personas que tanto me han ayudado:

- A mis padres, porque sin ellos no sería quien soy ni tendría lo que tengo.
- A mi hermana por su experiencia de valor incalculable y por responder pacientemente a todas mis preguntas.
- A mi pareja, por su forma de ser, su apoyo incondicional, su interés y por soportar largas horas de estudio. Estaré siempre agradecido.
- Al Dr. Mikel Izal, tutor de este proyecto, que me brindó una gran oportunidad al mismo tiempo que me otorgó la libertad de trabajar en algo que realmente me gusta.
- A mis amigos Fernando, Juan, Paula, etc. por demostrarme que la verdadera amistad existe.
- A mis colegas de clase, buenos compañeros y excelentes amigos.
- A Quomai, la empresa que con una oferta de trabajo me ha dado la motivación necesaria para terminar este proyecto y empezar una nueva etapa de mi vida.
- A la persona que una vez me dijo: "Dedícate a lo que te gusta y nunca trabajarás"

A todos ellos, gracias.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Resumen

El título del presente proyecto, *iPark: Aplicación sobre la Zona Azul para dispositivos iOS*, puede dividirse en tres partes que resumen su contenido y propósito: Aplicación, Zona Azul y dispositivos iOS.

Aplicación. El objetivo principal del proyecto es obtener una aplicación multilingüe que se llamará iPark y que tenga un valor real en el mercado. Al ser una aplicación informática se suceden una serie de fases hasta completar su desarrollo: desde la especificación de los requisitos hasta la codificación del producto pasando por el diseño de la interfaz y del programa.

Zona Azul. Las zonas en las que se paga por estacionar en la calle existen en numerosas ciudades españolas. Concretamente, en Pamplona se las conoce como Zona Azul por el color de la señalización en el asfalto. Estas zonas son diariamente utilizadas por muchas personas que desconocen gran parte de su funcionamiento. Se ha desarrollado la aplicación para ayudar estos usuarios a:

- Calcular el tiempo y el dinero para el parquímetro.
- Consultar un mapa de sectores de residentes.
- Obtener información sobre multas, horarios, datos útiles...
- Establecer recordatorios que avisen del fin del periodo de estacionamiento.

Dispositivos iOS. El sistema operativo utilizado por los dispositivos móviles de la marca Apple entre los que se encuentran el iPhone, iPad y iPod Touch. iPark está diseñada para el iPhone pero funciona en todos los demás dispositivos. Ya que la aplicación se utiliza al aparcar en la calle, es imprescindible que la aplicación se ejecute en dispositivos móviles.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Índice

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

0. Introducción al PFC

0.1. Antecedentes del proyecto

El presente proyecto ha sido elaborado como respuesta a una necesidad real que ha sido observada en el día a día de un conductor de Pamplona: la complejidad y poca claridad de las Zonas de Estacionamiento Limitado y Restringido (ZEL y ZER); comúnmente conocidas como *Zona Azul.*

La plataforma elegida para el desarrollo ha sido el sistema operativo *iOS* de *Apple* para dispositivos móviles. Esto permite que los usuarios dispongan de la aplicación cuando realmente la necesitan, a la hora de estacionar en la calle. Además, se ha elegido *iOS* frente a su competidor más directo, *Android,* por tener un ecosistema de desarrollo y distribución de aplicaciones más afianzado y con grandes expectativas de futuro.

0.2. Objeto del proyecto

El objeto del proyecto es el de desarrollar un proyecto informático que dé como resultado una pieza de software que pueda ser distribuida en la *Apple Store* y ofrecida al ayuntamiento de Pamplona (o en su defecto, a la empresa adjudicataria *Dornier, S.A.*).

El otro objetivo principal del proyecto es el de obtener una base de conocimiento en esta plataforma y arquitectura que permita al proyectista acceder a una empresa de desarrollo de software para dispositivos móviles. Este objetivo ha sido logrado consiguiendo un trabajo en la empresa emprendedora *Quomai*[1].

0.3. Descripción y estructura del PFC

Este PFC se compone de dos partes:

- El conjunto de archivos (código fuente, imágenes y otros recursos) que conforman un proyecto de *Xcode 4.* Se entrega en soporte informático junto con esta memoria. La explicación detallada de cada

[1] http://quomai.es/

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

componente, funcionalidad y algoritmo utilizado se encuentra incrustada en el código fuente a modo de comentarios.

- Esta memoria en la que se detalla el desarrollo de la aplicación como proyecto informático, los mecanismos utilizados y las soluciones a los problemas encontrados

Para el correcto desarrollo de un proyecto informático se requiere una planificación previa y la descomposición del proyecto en fases:

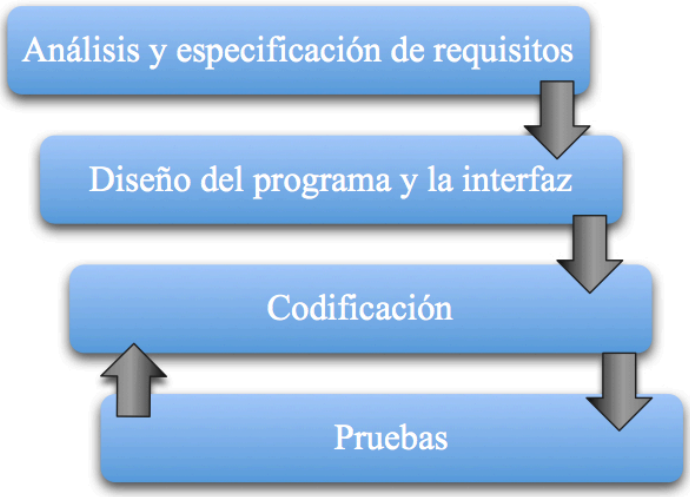


Figura 0-3-1 Descomposición en cascada de tareas del proyecto informático.

El proceso de análisis y especificación de requisitos tratará de establecer cuáles van a ser las funcionalidades que va a tener el programa. Para ello habrá que decidir cuáles son factibles con los recursos disponibles.

El diseño se realizará primero mediante bocetos en papel y luego con herramientas de diseño más complejas. Se tratará de diseñar el aspecto aproximado de la interfaz y de la organización de las diferentes vistas y componentes. Aquí se definirá el flujo básico que se sigue para realizar las diferentes funcionalidades propuestas. Se aplicarán diferentes estándares y principios de la interacción hombre-máquina. Asimismo, se tratarán de seguir las *Human Interface Guidelines* de *Apple* para *iOS*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Además, se realizará un diseño de la arquitectura de la aplicación de forma y manera que se permita que las tareas de codificación se desarrollen de manera independiente unas de otras.

El proceso de codificación y pruebas será el más extenso. Aquí es donde se realizará la aplicación en sí y se verificará su correcto funcionamiento. El desarrollo seguirá una metodología Top-Down, encargándose primero de implementar el funcionamiento general de la aplicación (esqueleto) para luego ir encargándose de funcionalidades más concretas (pestañas de *Calcular, Sectores, Info, Alarmas y Ajustes*).

Aunque no quede plasmado en esta memoria dada su naturaleza, se va a utilizar un mecanismo de control de versiones. Se ha elegido *Subversion* porque es un mecanismo simple que se integra perfectamente con el *IDE Xcode 4* y que no resulta complicado de mantener. También sirve como copia de seguridad ya que todo el contenido de la aplicación está almacenado en el servidor.

Con el propósito de demostrar el funcionamiento de la aplicación, se ha creado un vídeo que se adjunta con esta memoria y que puede verse en el blog de la aplicación (que cuenta con versión para móviles):

http://iparkapp.blogspot.com/

o en *Youtube*:

http://www.youtube.com/watch?v=bm9oQHv6E6c

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1. Análisis y especificación de requisitos

1.1. Objeto básico de la aplicación

Se pretende desarrollar una aplicación que permita realizar de una manera intuitiva y fácil los cálculos de tarifas de la *Zona Azul* de Pamplona. Además, se quiere ofrecer una serie de funcionalidades y herramientas extra que faciliten la experiencia de los usuarios de estas zonas.

1.2. Requisitos funcionales

Se recogen a continuación aquellas funcionalidades que se requerirán del producto final una vez haya sido finalizado. Estos requisitos deberán tomarse en cuenta en la posterior fase de diseño para acabar implementando cada una de ellas en la fase de codificación.

1.2.1. La aplicación deberá permitir el cálculo de la duración de un ticket de la Zona Azul a partir de una cantidad monetaria introducida por el usuario. Se deberán tener en cuenta las particularidades de las tarifas así como los horarios y fechas.

1.2.2. De manera similar, la aplicación deberá poder realizar el cálculo inverso: a partir de una duración deseada introducida por el usuario obtener el importe necesario.

1.2.3. Se podrán consultar de manera gráfica los diferentes sectores en los que actualmente está dividida la Zona Azul.

1.2.4. La aplicación deberá ofrecer información detallada y relevante acerca de la Zona Azul y sus diferentes normativas, exenciones, penalizaciones... Se pretende enfatizar aquella información que se estime que el público general desconoce.

1.2.5. El sistema deberá implementar un sistema de notificaciones o alarmas configurables. Dichas notificaciones deberán poder establecerse a partir de los cálculos de los puntos 1.2.1. y 1.2.2.

1.2.6. La interfaz de la aplicación deberá soportar, en la mayor medida posible, múltiples idiomas. Se pretende, en un principio, ofrecer la aplicación en Castellano, Euskera e Inglés. Deberá implementarse de manera independiente al sistema operativo y las herramientas de

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

internacionalización disponibles ya que no dan soporte para todos los idiomas propuestos (en concreto, el Euskera).

1.2.7. Deberán poder configurarse diferentes parámetros como el formato de la fecha y hora, el idioma, ... Dichas preferencias podrán ser accesibles tanto desde la aplicación como desde la herramienta de configuraciones por defecto del sistema operativo (*Settings*).

1.3. Requisitos no funcionales

Se detallan a continuación una serie de requisitos que, aun no siendo obligatorios para el funcionamiento de la aplicación, deberán tratar de satisfacerse en la medida de lo posible.

1.3.1. La interfaz del usuario deberá diseñarse siguiendo en la mayor medida de lo posible las *iOS Human Interface Guidelines*[2] de Apple así como otros principios y estándares de la interacción hombre máquina.

1.3.2. Como consecuencia del punto anterior, la interfaz deberá ser intuitiva y de fácil aprendizaje. Asimismo, se intentará ofrecer una interfaz estéticamente agradable y personalizable.

1.3.3. Como extensión del requisito 1.3.1, la aplicación intentará tener un "look Apple" (ver Figura 1-3-1) para ser consistente con el resto de aplicaciones disponibles y asegurar una buena integración.

[2] *iOS Human Interface Guidelines* (link acortado): http://bit.ly/awcxrz

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

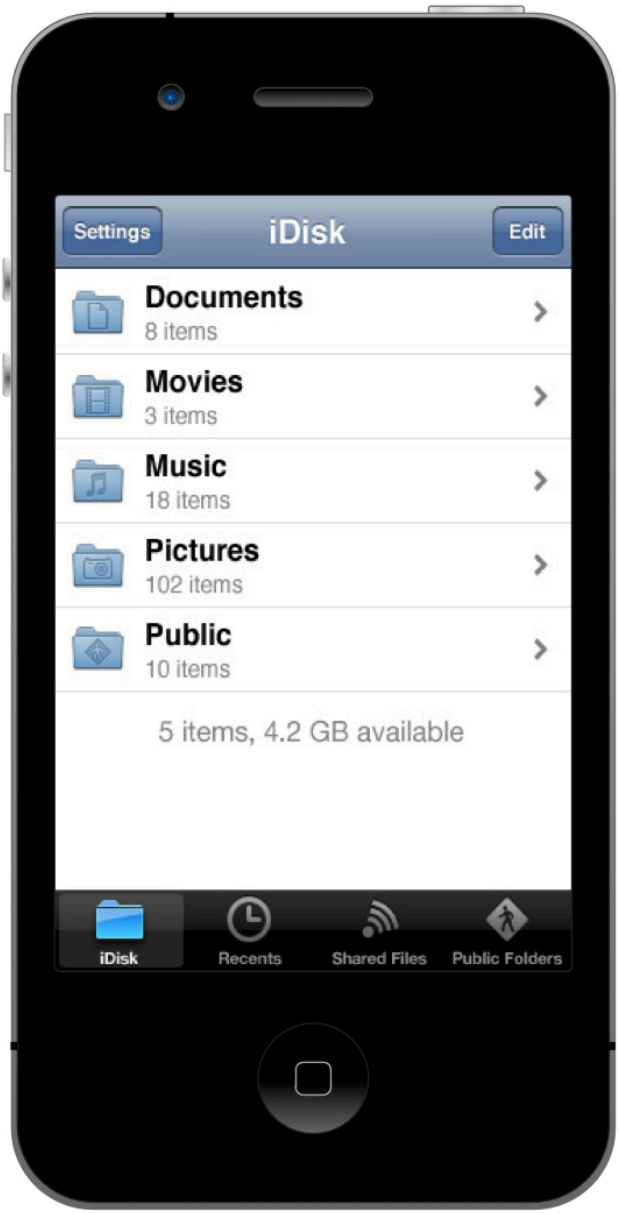


Figura 1-3-1 Ejemplo de aplicación que muestra una estética familiar para los usuarios de iPhone.

1.3.4. La aplicación deberá facilitar que los cálculos más comunes se hagan con facilidad y con poco pasos utilizando para ello las técnicas adecuadas.

1.3.5. El código fuente deberá abogar por la simplicidad, modularidad y claridad. Deberá estar abundante pero relevantemente comentado.

1.3.6. Las funcionalidades básicas de la aplicación deberán presentarse como pestañas. Para ello se pretenden utilizar los denominados *TabBar (ver Figura 1-3-2).*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 1-3-2 Barra de pestañas en la parte inferior de la interfaz en la aplicación del reloj del sistema.

1.3.7. Una vez seleccionada una de las funcionalidades básicas, si el contenido lo permite, la visualización deberá organizarse de manera jerárquica. Para ello se pretenden utilizar las *Navigation Bar (ver Figura 1-3-3).*

Figura 1-3-3 *Navigation Bar* mostrando un botón que permite navegar a la vista anterior.

1.3.8. Se pretende que el mapa de sectores sea interactivo (no estático) y que pueda obtener la localización del usuario.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.3.9. Se intentará evitar que la construcción de la interfaz se haga mediante código. Por ello, se prefiere el uso del *Interface Builder* para una construcción más gráfica y libre de errores.

1.3.10. Se tratarán de aprovechar al máximo las nuevas características de la versión 4 de *iOS* como el *multitasking*

1.4. Requisitos adicionales

Aquellos requisitos que han sido barajados durante el análisis pero que, con los recursos disponibles, se consideran no viables o viables para versiones posteriores del producto.

1.4.1. La aplicación podrá determinar de manera automática el sector en el que uno se encuentra y el tipo de zona en el que se pretende aparcar (azul, naranja, roja, ...).

1.4.2. Obtención de datos actualizados de tarifas, cambios en los sectores y zonas desde un servidor.

1.4.3. La aplicación podrá obtener y recordar la posición del usuario cuando estaciona. También podría vincularse esa información con una foto de la plaza de aparcamiento. Posteriormente, podrían ofrecerse instrucciones para volver al vehículo.

1.4.4. Ofrecer un sistema de información acerca de los aparcamientos subterráneos de la ciudad: su posición, tarifa y nivel de ocupación a tiempo real.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2. Diseño de la interfaz

2.1. Primera iteración

Se presenta a continuación la primera fase de diseño que se ha realizado para la propuesta del proyecto. Se ha diseñado una primera aproximación para mostrar la principal funcionalidad de la aplicación (aún sin definir completamente en aquel momento). La intención de esta primera iteración no es tanto mostrar el aspecto final de la aplicación sino establecer una serie de ideas base para el futuro refinamiento del diseño.

2.1.1. Bocetos manuales

En las Figuras 2-1-1, 2-1-2 y 2-1-3 se muestran los escaneos de algunos de los primeros bocetos hechos a mano.

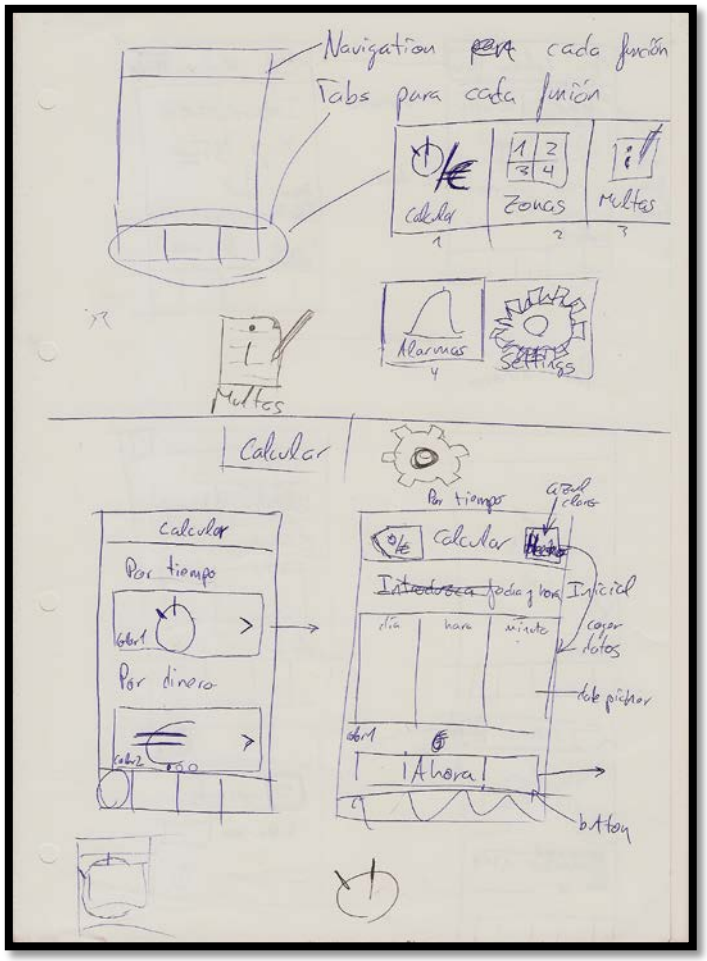


Figura 2-1-1 Diseño de iconos y pestaña *Calcular*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

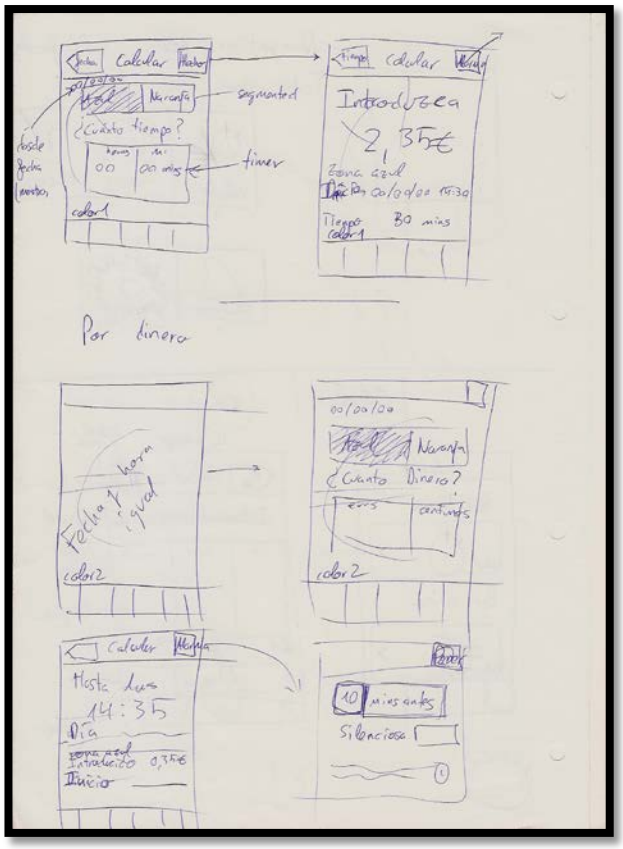


Figura 2-1-2 Distintas pantallas de la pestaña *Calcular*

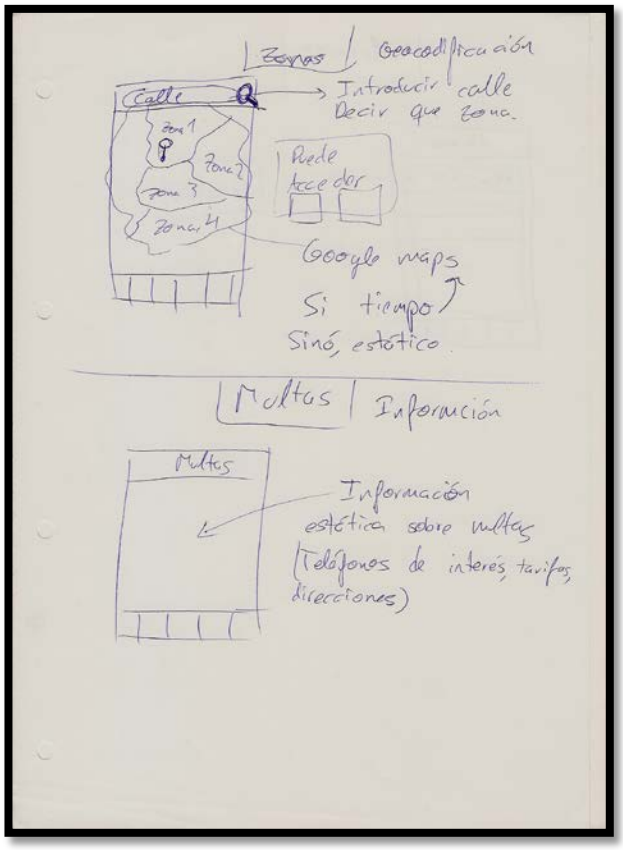


Figura 2-1-3 Pestañas *Zonas* e *Info*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.1.2. Prototipo de la interfaz

Se ha utilizado el *Interface Builder* incluido con *Xcode* para poder usar los elementos de interfaz predeterminados de *iOS*.

En la Figura 2-1-4 pueden apreciarse las diferentes pantallas que se le presentarían al usuario. En la primera pantalla se le ofrecen dos opciones al usuario.

Con el primer botón (verde) pasaría a calcular el importe necesario para el tiempo que introduzca el usuario. Para ello debe introducir primero la fecha en la que desea estacionar. Luego, en la siguiente pantalla, puede elegir el tipo de zona con un *segmented control* y el período de estacionamiento con un *picker*. En la última pantalla se le mostraría el resultado del cálculo junto con un resumen de los datos introducidos anteriormente para evitar que el usuario los retenga en su memoria[3].

Con el segundo botón (azul) se realizaría el cálculo inverso de manera similar, el usuario introduciría fecha, zona y el importe que desee y se le mostraría el tiempo del que dispondría con ese dinero.

Uno de los aspectos mejorables que se puede apreciar en este diseño es que el usuario tiene dos caminos diferentes para realizar dos acciones similares; por lo que no es muy intuitivo. Además el "cálculo por dinero" y "cálculo por tiempo" puede resultar confuso. Para el programador puede resultar obvio separar los procesos por tipos de entrada, pero el usuario común no tiene porqué distinguir las tareas de la misma manera.

[3] Siguiendo el octavo principio de Shneiderman: "Reduce short-term memory load". *SHNEIDERMAN, B. EIGHT GOLDEN RULES FOR INTERFACE DESIGN. EN: DESIGNING THE USER INTERFACE, 3ª EDN. ADDISON WESLEY, U.S.A. (1998).*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 2-1-4 Primer diseño de la interfaz de usuario

2.2. Segunda iteración

Esta segunda iteración se ha realizado una vez el análisis y especificación de los requisitos del proyecto han sido fijados. Esto supone un diseño más realista y similar a lo que se espera que sea la aplicación final. También se ha creado un diseño mucho más detallado que permitirá, al codificar la aplicación, centrarse en aspectos como la documentación y la optimización del código en vez de en la toma de decisiones sobre la interfaz.

2.2.1. Bocetos manuales

Para realizar los bocetos manuales de esta iteración se han utilizado unas plantillas[4] (ver Figura 2-2-1) que facilitan la visualización de los contenidos y la proporción de los diferentes elementos gráficos en relación con el dispositivo.

[4] Plantilla creada por la empresa canadiense *fullofdesign*. Enlace de descarga de la plantilla: http://fullofdesign.ca/uploads/iphone_notepad_horizontal.pdf

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

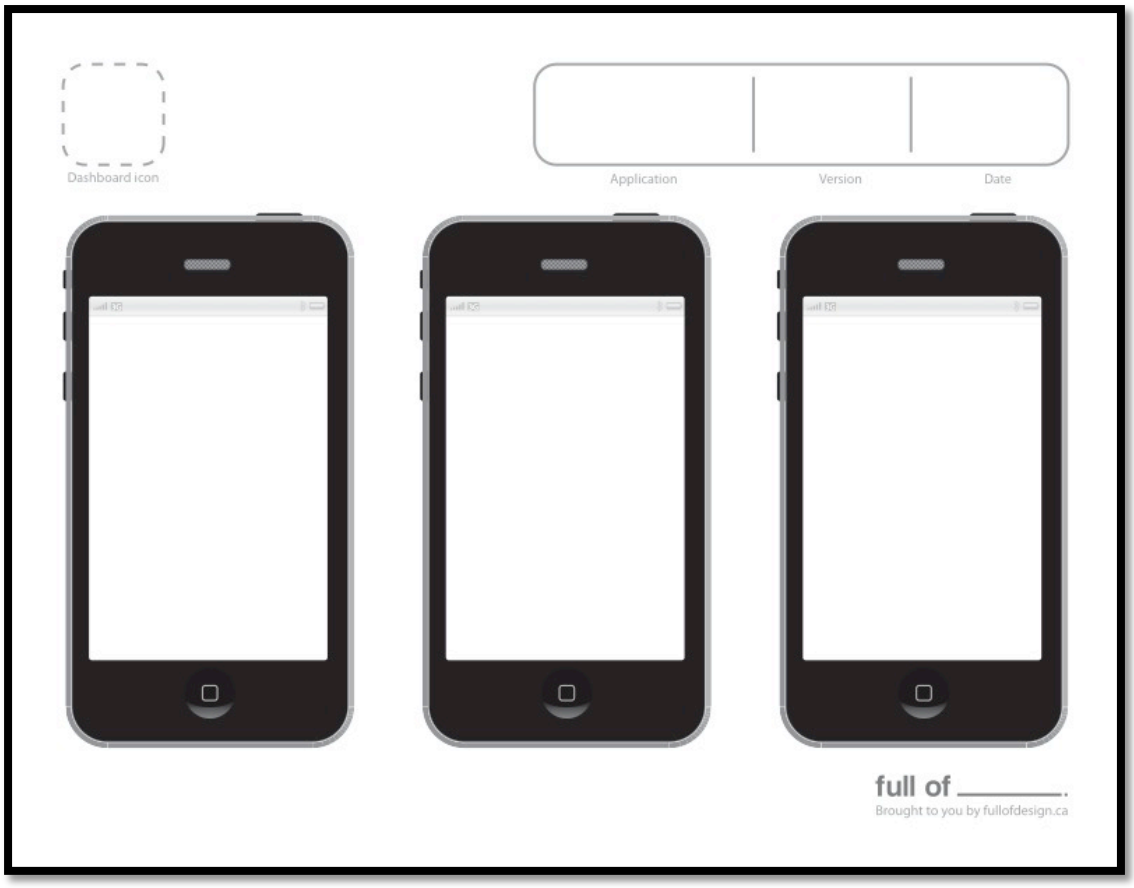


Figura 2-2-1 Plantilla utilizada para realizar los dibujos

A continuación se muestran escaneos de 3 de las 6 hojas originales (Figuras 2-2-2, 2-2-3 y 2-2-4). También se han considerado diferentes opciones para los iconos de las diferentes pestañas de la aplicación. Por último, se han barajado 4 opciones de diseño del icono principal de la aplicación.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

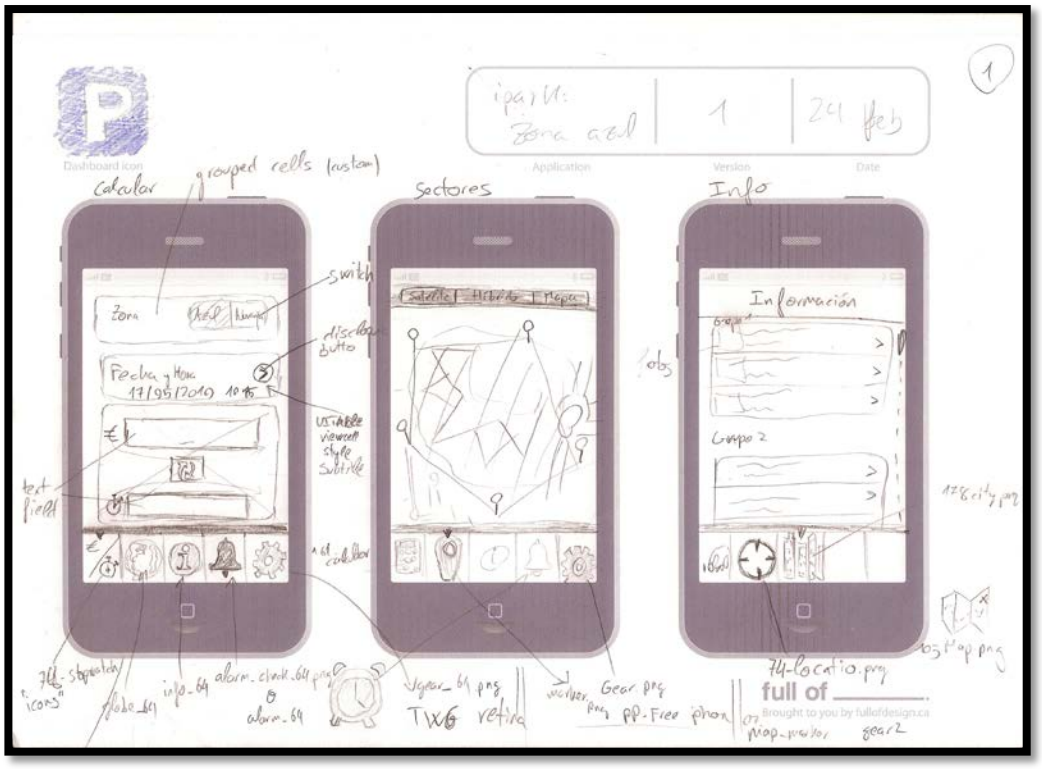


Figura 2-2-2 Diseños de las pestañas *Calcular*, *Sectores* e *Info,* varios iconos de pestañas y una propuesta del icono principal en la esquina superior izquierda

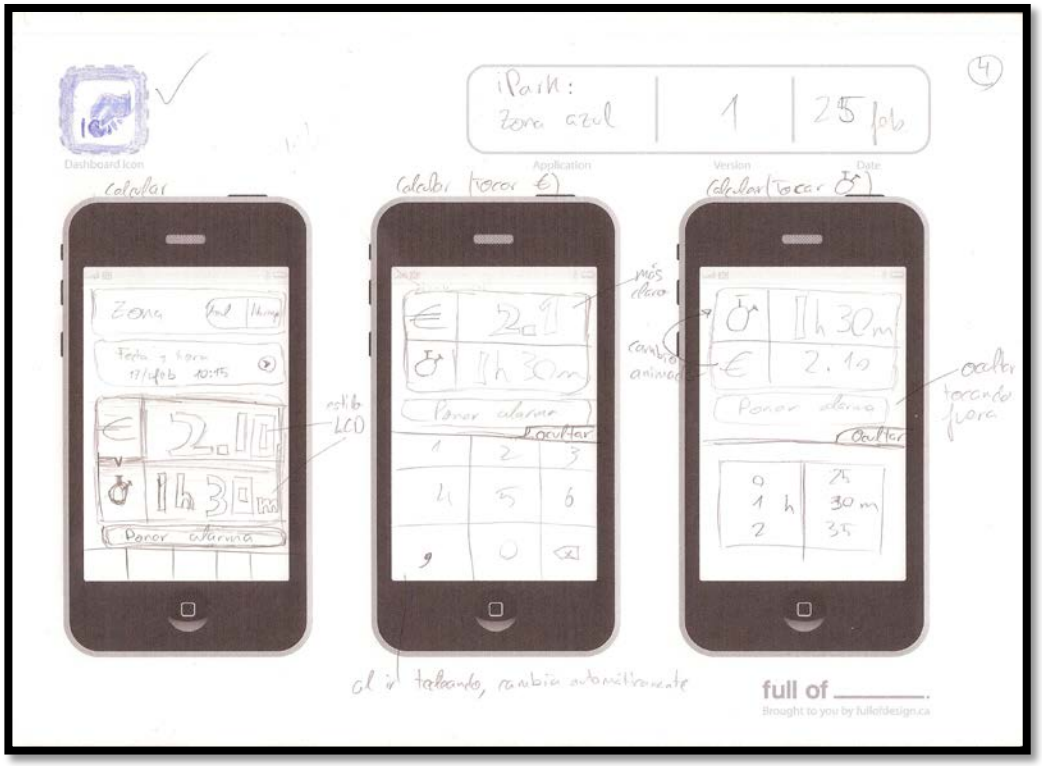


Figura 2-2-3 De izquierda a derecha: rediseño de la vista *Calcular*, teclado desplegado al editar el campo con € y panel desplegado al editar el tiempo de estacionamiento. Esquina superior izquierda: otra propuesta del icono principal.

upna Universidad Pública de Navarro Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

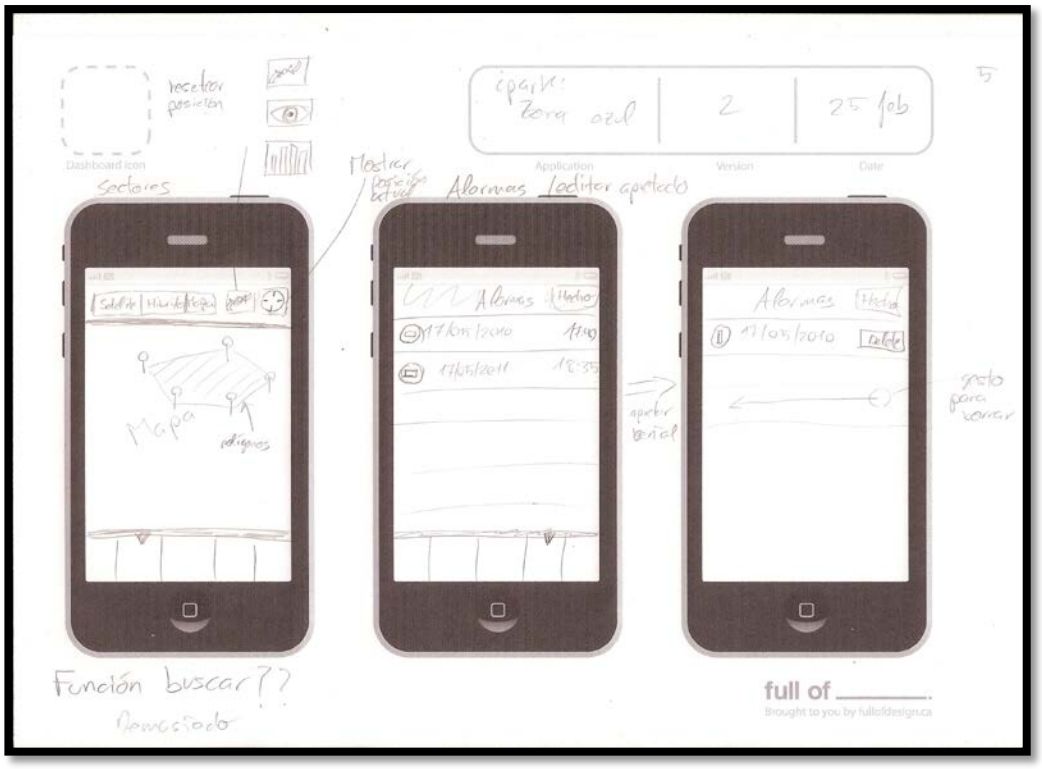


Figura 2-2-4 Refinamiento del diseño de la vista *Sectores* y vistas de la pestaña *Alarmas* cuando el usuario presiona *Borrar* para cancelar las alarmas.

2.2.2. Prototipo de la interfaz

Utilizando de nuevo la herramienta *Interface Builder* se ha creado cada una de las pantallas dibujadas en las plantillas. Luego se han refinado esos pantallazos utilizando la herramienta de dibujo *Omnigraffle Pro*[5].

2.2.2.1. Pestaña Calcular

En la vista principal de esta primera pestaña (ver Figura 2-2-5) se ha intentado que toda la información necesaria para el cálculo de las tarifas se presente en una sola vista sin que el usuario deba navegar. Esto se debe al antes comentado principio de Shneiderman: se pretende que el usuario pueda tanto visualizar como editar toda la información necesaria en un mismo lugar, sin tener que recordar información.

[5] Web oficial: http://www.omnigroup.com/products/omnigraffle/.
La aplicación admite con plantillas creadas por los usuarios, se han usado algunas de ellas con elementos de la interfaz del *iPhone* disponibles en http://graffletopia.com/

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Se muestra una celda dentro de una *Grouped Table View* con una etiqueta y un *Segmented Control* que permita al usuario escoger entre dos o más opciones de la zona en la que aparcará. Este tipo de control es ampliamente conocido por los usuarios de *iOS* por lo que cumple que sea intuitivo, fácil y consistente con otras aplicaciones.

Más abajo, aparece otra celda del tipo *Subtitle* que por defecto mostrará la fecha actual pero podrá editarse al tocar el *Disclosure Button* de la derecha.

Después, se presentan dos campos de texto editables que tendrán apariencia de pantallas de calculadora. El primer campo mostrará la cantidad que se desea pagar y estará precedido por un icono que muestre el símbolo del euro (para que los usuarios sepan que tocándolo puede editarse el campo). El segundo campo mostrará el periodo de tiempo que le corresponde al importe en el campo superior. Tendrá un icono de un cronómetro.

La idea es que tanto el campo del dinero como el de la duración sean, al mismo tiempo, entradas y salidas. Es decir, si se modifica cualquiera uno de los campos, el cambio se verá reflejado automáticamente en el otro campo.

Esto evita que el usuario tenga que elegir si quiere realizar el cálculo por tiempo o por dinero: podrá editar el dato que decida en cualquier momento. Este tipo de interacción es común en las aplicaciones de conversión de unidades (divisa, longitud, peso, ...).

Por último, habrá un botón que permita ajustar una alarma a partir de la fecha introducida y la duración del periodo de estacionamiento.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

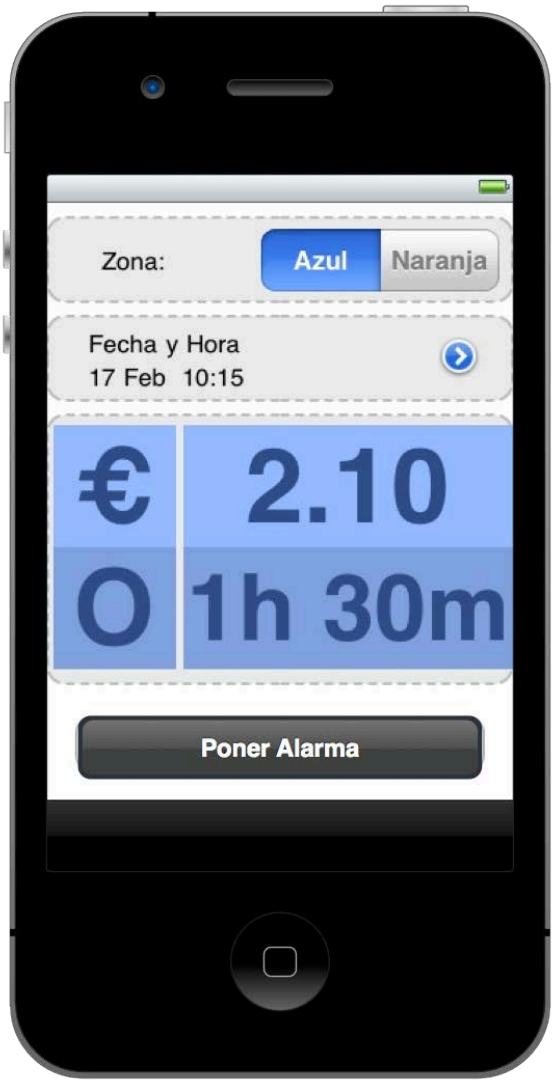


Figura 2-2-5 Vista principal de la pestaña *Calcular*

Un alto porcentaje de usuarios no editarán la fecha y hora por defecto ya que la mayoría realizarán el cálculo en el momento en el que aparcan. Sin embargo, el que decida hacerlo lo podrá realizar con un *Date Picker* que aparecerá en un panel desplegable (ver Figura 2-2-6) al tocar el *Disclosure Button*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

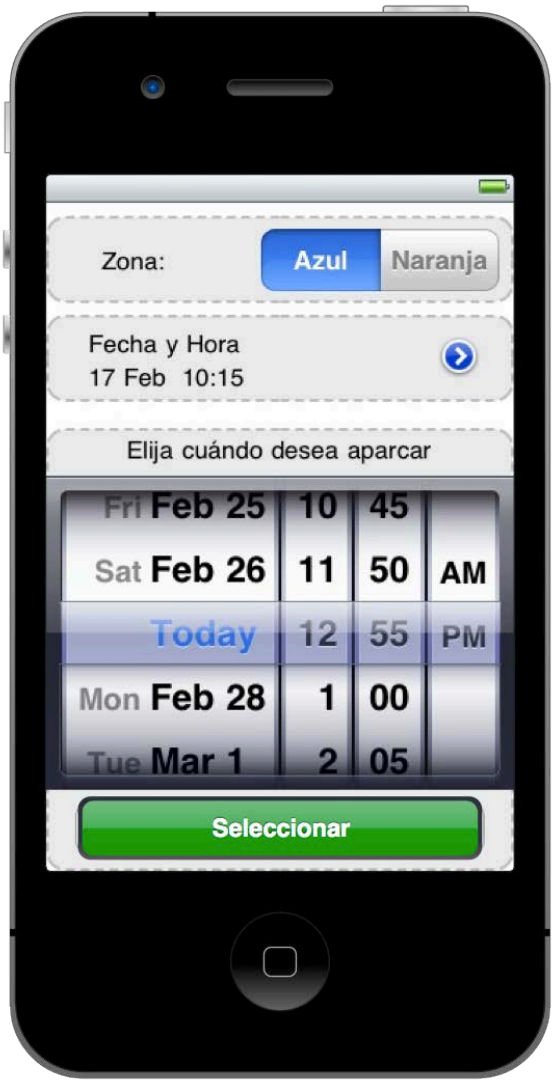


Figura 2-2-6 Editando la fecha

Cuando se seleccione el campo del dinero, se desplegará un teclado como el de la Figura 2-2-7. Nótese que dicho teclado deberá ser reprogramado ya que el campo requiere la introducción de números con dos decimales y el teclado numérico estándar del *iPhone* no tiene la coma decimal. Al desplegarse este teclado se desplazará la vista hacia arriba para dejar que el campo que está siendo editado no quede oculto. Se ofrecerá también un mecanismo para esconder el teclado ya que el usuario puede querer cambiar los datos que ahora quedan ocultos (fecha y zona).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 2-2-7 Editando el valor monetario, se puede apreciar que el teclado mostrado aquí (el que está disponible por defecto) no sirve para introducir números decimales.

Sin embargo, al tocar el campo del período de tiempo, deberá desplegarse un panel con un *Picker* que permita elegir de manera sencilla y sin errores la duración en formato X horas e Y minutos (ver Figura 2-8). En todo momento, el campo siendo editado deberá mostrarse encima del campo que muestra el resultado.

El botón que permite poner la Alarma será visible en todo momento ya que es el paso lógico a seguir después de editar cualquiera de los campos y se quiere que los usuarios no tengan que ocultar el teclado para poder ajustar la alarma.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 2-2-8 Editando el período de estacionamiento.

Por último, presionando el botón *Poner Alarma* se mostrará un *Alert View* con opciones de poner la alarma 5, 10 y 15 minutos antes de que finalice el período de estacionamiento calculado (ver Figura 2-2-9). Estos valores podrán cambiarse en los ajustes pero en principio se ha estimado que 5, 10 y 15 serán los valores más adecuados.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 2-2-9 Ajustando una alarma

2.2.2.2. Pestaña Sectores

La pestaña de Sectores mostrará un mapa dinámico (ver Figura 2-2-10) por el que el usuario podrá desplazarse. Este mapa se servirá del *MKMapKit,* el API de *iOS* para gestionar los mapas interactivos que, a su vez, utiliza los datos de *Google Maps.*

Se mostrarán unos polígonos superpuestos en el mapa que delimitarán los diferentes sectores de la Zona Azul de una manera gráfica

En la parte superior se mostrará una barra de herramientas con un *segmented control* más estrecho y oscuro (típico de los mapas en el *iPhone*) para cambiar entre las tres vistas disponibles del mapa: Satélite, Híbrido y Mapa. Por último, se añadirán dos botones extra en la esquina superior derecha. El primero

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

se utilizará para centrar la vista y el zoom del mapa con los valores por defecto de la aplicación. El segundo servirá para mostrar al usuario su ubicación actual con una chincheta e informarle del sector en el que se encuentra.

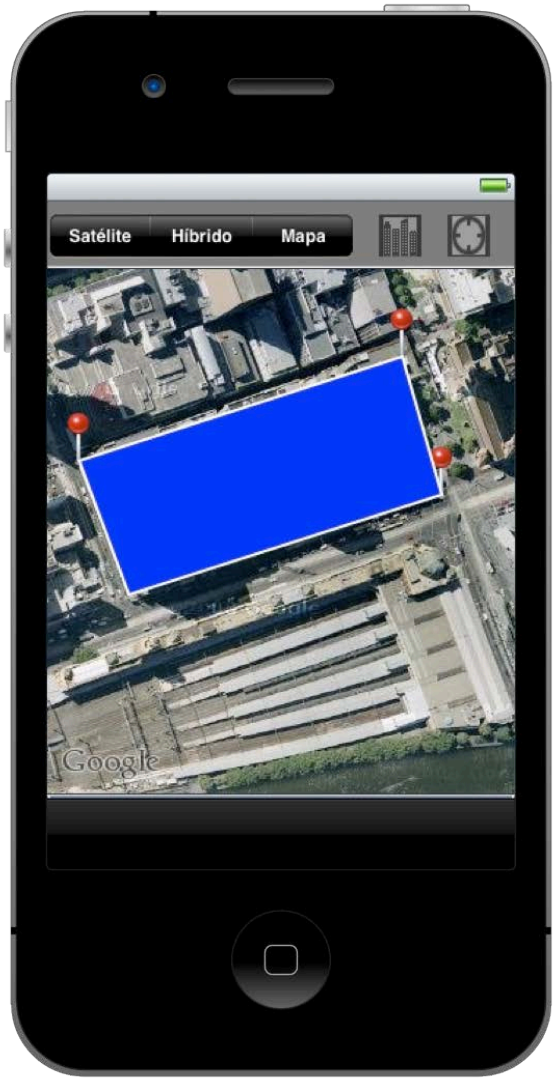


Figura 2-2-10 Mapa con un polígono dibujado y en la parte superior los controles del mapa

2.2.2.3. Pestaña Información

El propósito de esta pestaña es, principalmente, ofrecer información en forma de texto e imágenes. Para ello se organizará dicha información en un esquema jerárquico en el que se podrá navegar gracias a una *Navigation Bar* y unas *Table View.* Este esquema es ampliamente utilizado en el mundo del *iPhone*, las distintas celdas de la *Table View* contienen un título y opcionalmente un subtítulo y una pequeña imagen. Cuando se toca una fila determinada, se avanza un nivel en la jerarquía de la información y se muestra otra vista con otra tabla.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Además, permite navegar hacia atrás con al botón que se encuentra en la esquina superior izquierda. Todo esto es muy intuitivo y los usuarios de iPhone están, en general, acostumbrados a esta forma de navegar.

En la Figura 2-2-11 se muestra una *Table View* de estilo *grouped* ya que se la información mostrada se agrupará en diferentes categorías. También se muestra como es el botón de *Atrás.* Aunque la pantalla inicial (la raíz) no tendrá ese botón por ser el nivel jerárquico más alto, se ha incluido en el diagrama con el propósito de demostrar su ubicación y funcionamiento.

Figura 2-2-11 Diferentes celdas que contienen información. Nótese la pequeña flecha gris a la derecha de cada fila, se utiliza para indicar que tocando esa fila nos desplazaremos a la siguiente vista en la jerarquía.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.2.2.4. Pestaña Alarmas

Con esta pestaña (ver Figura 2-2-12) los usuarios podrán consultar y borrar las alarmas creadas en la pestaña *Calcular.* Para borrarlas se utilizará un mecanismo común en las aplicaciones del *iPhone* que se muestra en la Figura 2-2-13.

Se ha considerado también añadir un botón que permitiera borrar todas las alarmas a la vez. Se cree que con un uso normal de la aplicación el usuario no tendrá un número elevado de alarmas pero aun así se quiere ofrecer al usuario la posibilidad de eficientemente silenciar la aplicación completamente.

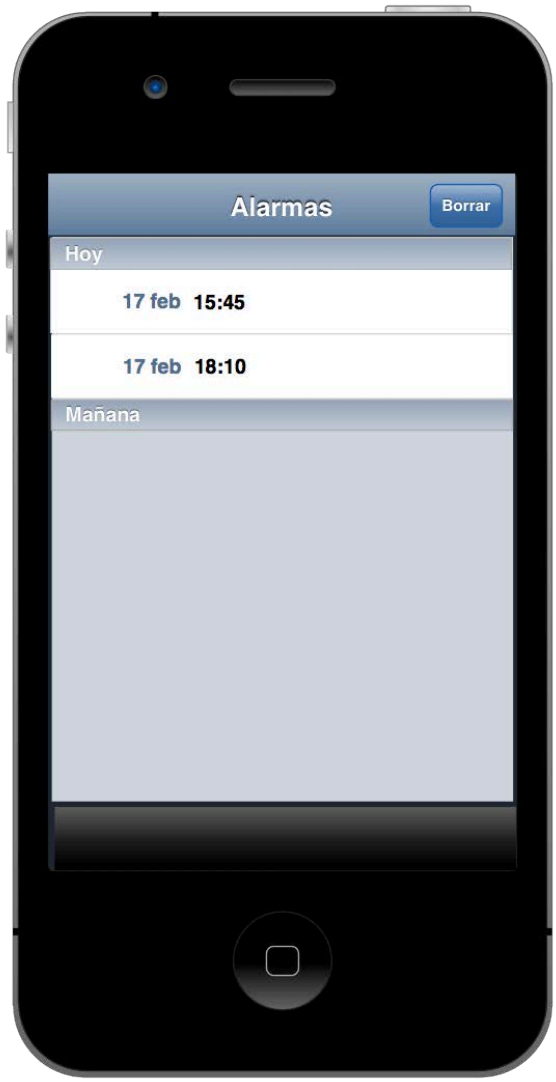


Figura 2-2-12 Pestaña *Alarmas* con dos alarmas programadas para hoy. El botón *Borrar* se mostrará en la esquina superior derecha.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

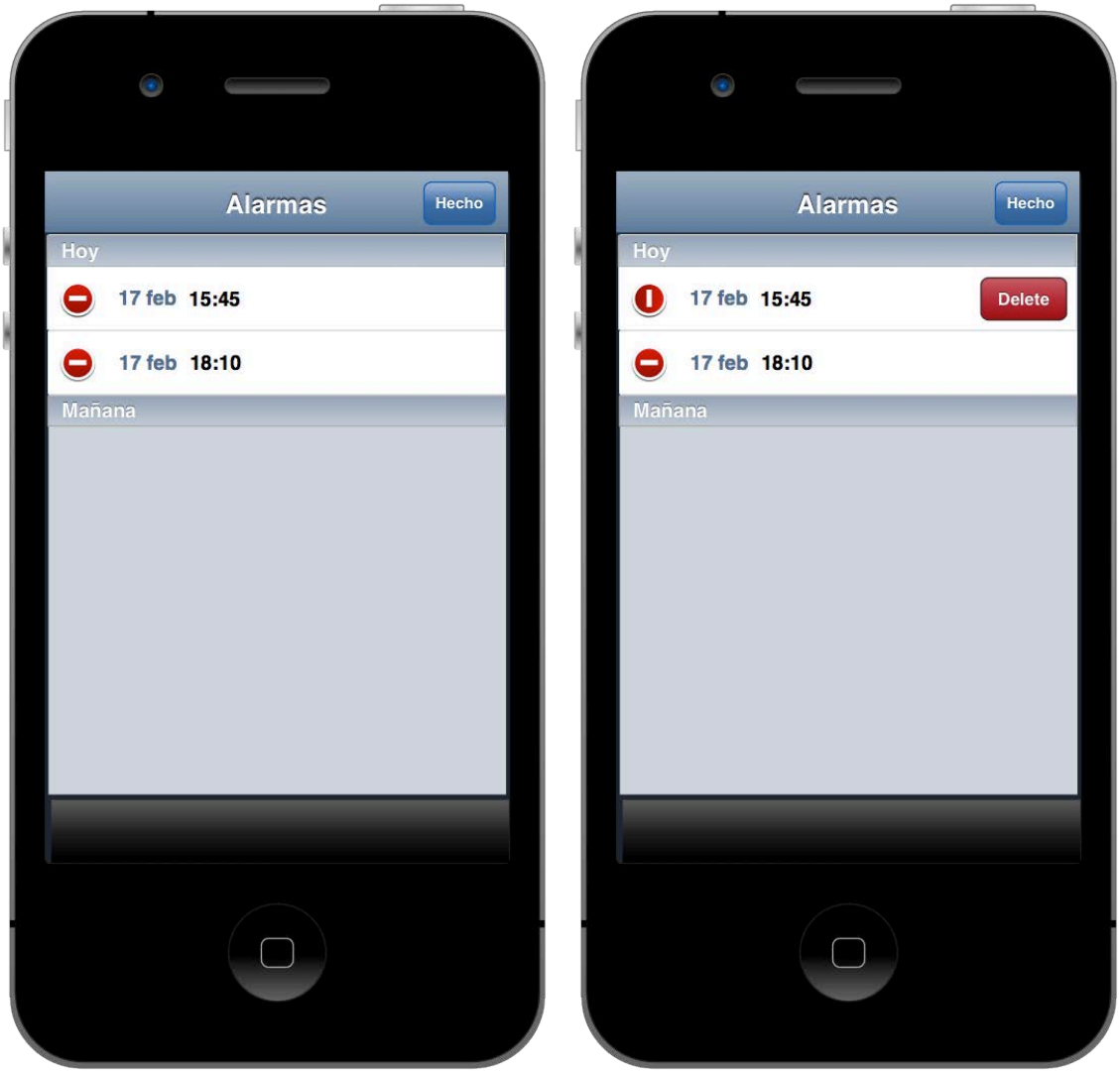


Figura 2-2-13 Proceso de borrado, después de tocar el botón *Borrar* este cambia a *Hecho* y se muestran los iconos de borrado en el margen izquierdo. Al tocar cualquiera de ellos, este girará 90 grados y aparecerá el botón para confirmar el borrado de esa fila.

2.2.2.5. Pestaña Ajustes

La pestaña de ajustes (ver Figura 2-2-14) será visible en la propia aplicación pero también deberá poder verse en la aplicación *Ajustes* del sistema, donde se centralizan todas las pantallas de configuración de las diferentes aplicaciones instaladas en el dispositivo. Se podrán configurar los siguientes parámetros de la aplicación al gusto del usuario:

- La opción de no permitir a la aplicación localizar el dispositivo. Es importante para los usuarios saber que pueden apagar esta característica ya que la posición de uno es información sensible. Además, evitará que el dispositivo gaste más batería al entrar en la pestaña *Sectores* y que empiece a funcionar el GPS.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- Los tiempos en los que la aplicación da la opción de poner una alarma. En principio los usuarios no tendrían por qué tocar esta configuración ya que los valores por defecto están pensados para satisfacer a la mayoría. Pero se desea dar la opción de personalización.
- El idioma de la aplicación. Esto contradice en cierta medida la filosofía de *Apple*[6] que establece que la que la aplicación se debe adaptar automáticamente al idioma configurado en las preferencias generales del aparato. Pero es necesario si queremos ofrecer la interfaz en Euskera ya que este idioma no se encuentra disponible en dichas preferencias (en el momento de desarrollo de esta aplicación).

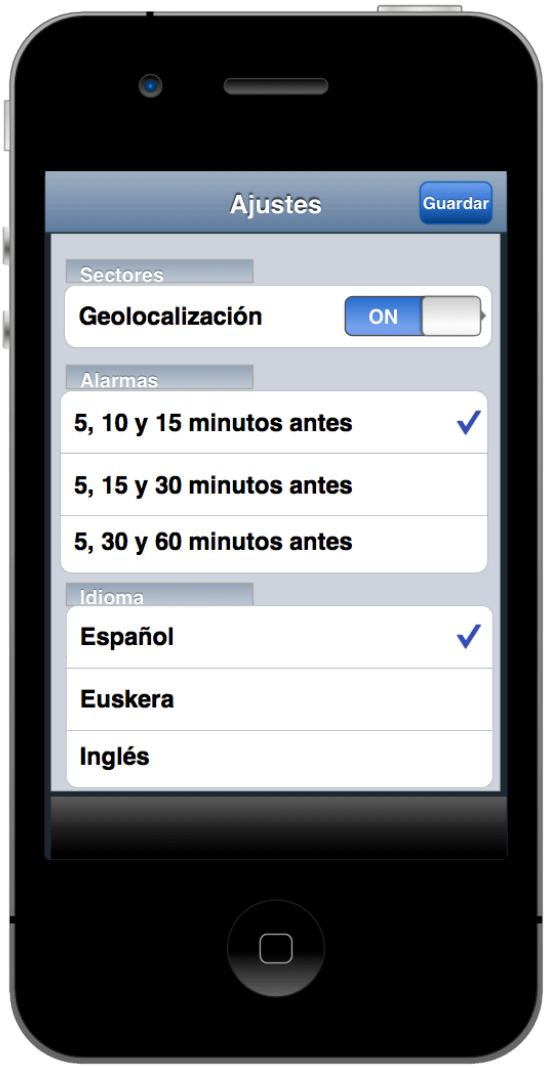


Figura 2-2-14 Ajustes de la aplicación.

[6] (links acortados) http://bit.ly/evdhFi, http://bit.ly/eDgBPi y http://bit.ly/eoXXRG.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.2.3. Diseño del icono de la aplicación.

Uno de los aspectos de la interfaz gráfica que más debe atraer al usuario es el icono principal de la aplicación. Este es el icono que deberá buscar cada vez que desee acceder a las características que le ofrece y por ello hay que diseñarlo con especial atención.

Para ello se han seguido unas *Best Practices*[7] proporcionadas por la web pixelresort.com que detalla algunos de los aspectos a tener en cuenta al realizar esta tarea. Pueden resumirse en 4 reglas

- **No usar palabras**. El icono debe ser una representación pictórica y deben utilizarse otras técnicas: similitud, metáforas, ...

Figura 2-2-15 Ejemplos de iconos mal diseñados: cuando el icono es pequeño apenas podemos leer el texto

- **No utilizar el brillo por defecto** que aplica el *iPhone* a los iconos. Supuestamente esto da uniformidad en la *app store* pero ni siquiera algunas de las aplicaciones que vienen preinstaladas traen ese brillo. Se puede deshabilitar este brillo en el *plist* principal de la aplicación indicando que nuestra imagen ya trae el brillo incrustado en la imagen.

[7] Consejos útiles seguidos para diseñar un buen icono de aplicación de *iPhone*: http://pixelresort.com/blog/iphone-app-icon-design-best-practises/

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 2-2-16 El brillo por defecto en cuatro iconos, el resultado no es muy bueno.

- **Simple es bueno, simple es claro.** Un icono es un espacio muy reducido para representar información y por ello debe reducirse el icono a una analogía simple del propósito de la aplicación. No significa que el icono no deba contener detalles.

Figura 2-2-17 Aplicaciones con iconos simples pero con gran significado

- **Destacar.** Muchas veces existen aplicaciones que cumplen propósitos similares y es bueno que el icono destaque frente a los demás. Por ejemplo, en aplicaciones de parking habrá multitud de iconos con una P blanca sobre fondo azul.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Siguiendo estas recomendaciones, se han barajado diferentes iconos pero se ha decidido utilizar como base del icono esta imagen por su simplicidad y significado:

Figura 2-2-18 Icono original

Para ser adecuada como icono de aplicación, se ha creado una versión de la imagen de 114x114 píxeles para pantallas de alta resolución (iPhone 4) y otra de 57x57 píxeles para las pantallas del iPhone 3GS y anteriores.

La imagen es universal de las *Zonas Azules* de diferentes ciudades. El problema es que en sí es demasiado simple y el borde azul no encaja con el borde redondeado de los iconos del *iPhone* (ver Figura 2-2-21). Por ello se ha editado el icono para obtener una imagen con los bordes más gruesos y más redondeados, como si tuviera un marco:

Figura 2-2-19 Borde grueso y redondeado.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Por último se le ha añadido un brillo personalizado[8], se ha cambiado la tonalidad del azul y se han suavizado ligeramente los bordes:

Figura 2-2-20 Icono final

Se comprueba con el simulador del *iPhone* (Figura 2-2-21) que el icono tiene una resolución aceptable, que el color es adecuado y que el brillo destaca frente al brillo estándar pero sin diferenciarse demasiado.

Figura 2-2-21 Icono de la aplicación en la pantalla principal del *iPhone* junto a otros iconos sin imagen y con el brillo predeterminado.

[8] Añadiendo dos capas con gradientes: una de blanco hacia transparente y otra de negro hacia transparente. Se ha utilizado Photoshop CS3.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

3. Diseño del programa

3.1. Primer diseño

El primer diseño del programa se realizó antes de empezar con la programación de la aplicación. Ha cambiado respondiendo a las necesidades de implementación y las dificultades afrontadas al codificar (ver apartado 3.1.2. Figuras 3-1-2 y 3-1-2b).

En todo el diseño se ha seguido el paradigma de Modelo-Vista-Controlador, como sugieren las *guidelines* de Apple[9].

3.1.1. Diagrama de contexto

A continuación se muestra el diagrama de contexto. Ha servido para hacerse una idea general de las interacciones que hay entre los principales componentes que forman la aplicación.

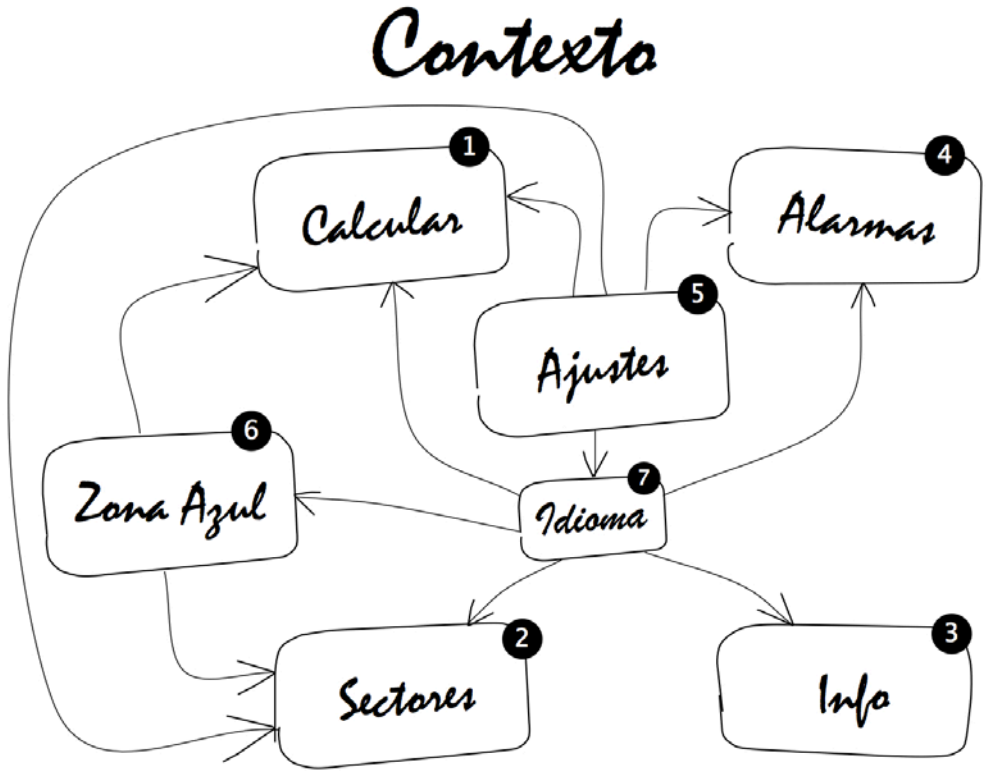


Figura 3-1-1 Diagrama de contexto.

[9] http://bit.ly/jeZqvh Documento de *Apple* explicando el Modelo-Vista-Controlador.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

3.1.2. Modelo

El modelo de la aplicación es la parte de los datos. Aquí se definen las características particulares de las diferentes clases que representan objetos de la realidad.

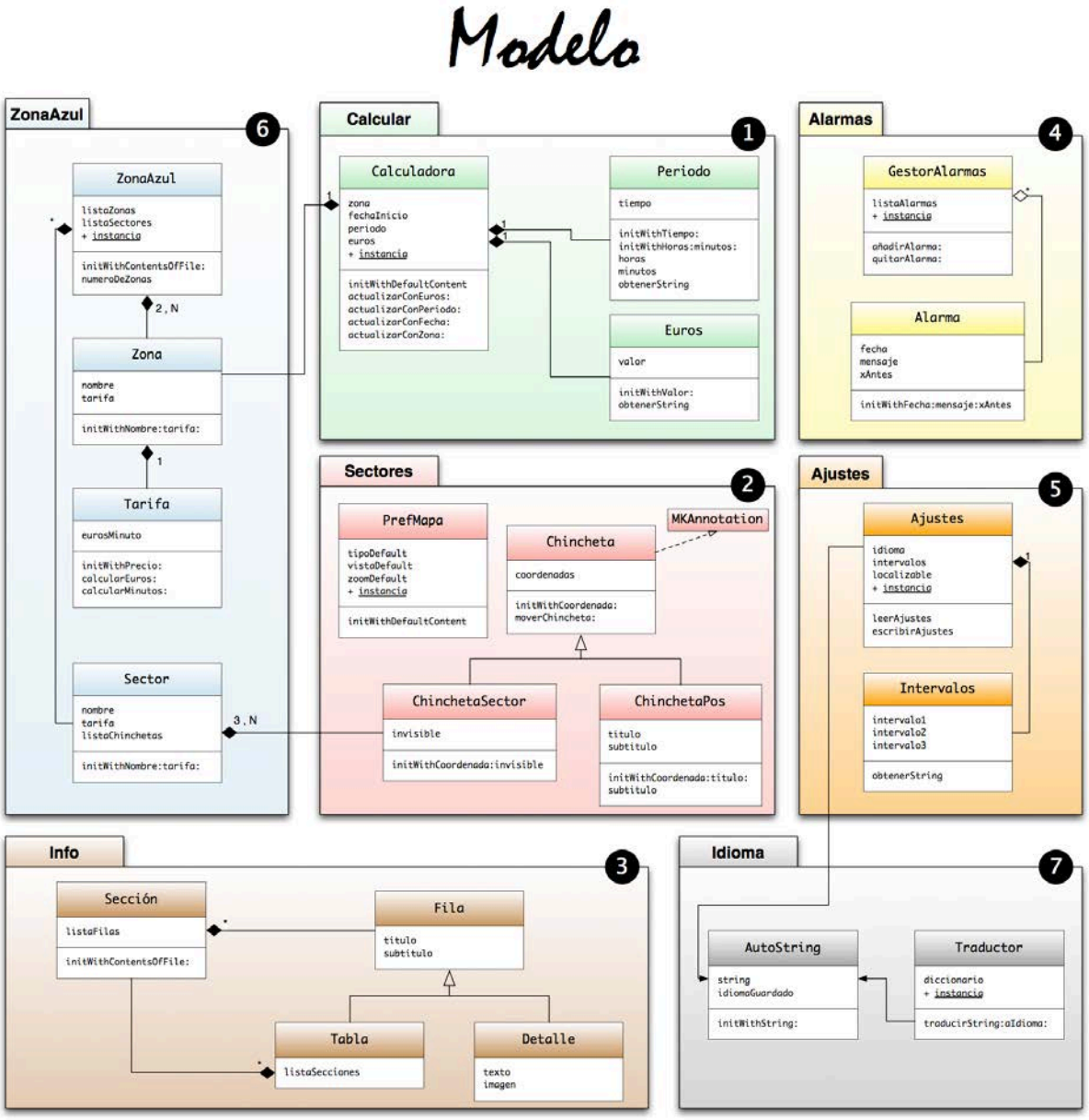


Figura 3-1-2 Diagrama de clases del Modelo inicial.

A modo de comparación, se ha realizado el mismo diagrama de clases una vez finalizada la aplicación.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

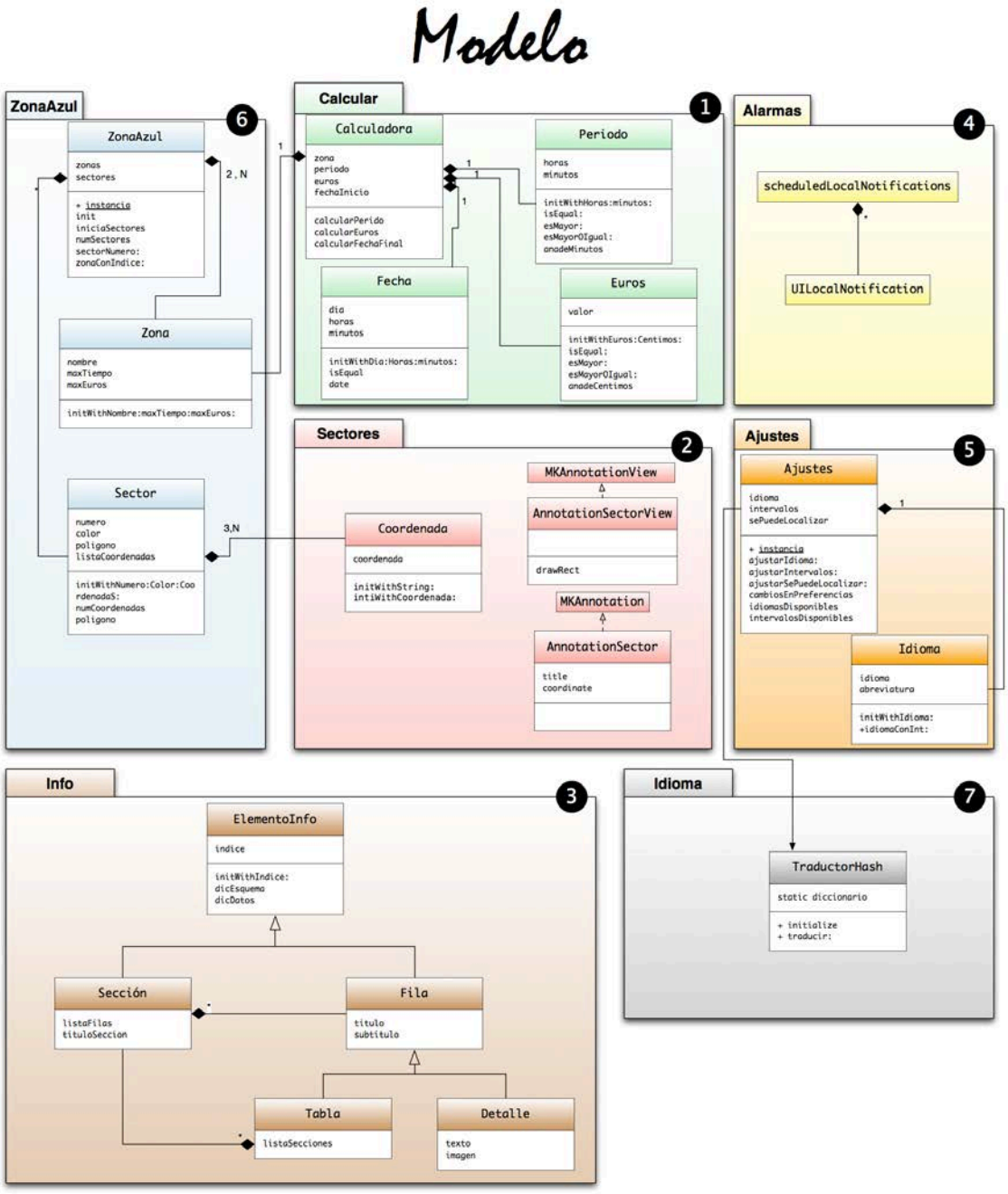


Figura 3-1-2b Diagrama de clases del Modelo después de finalizar la aplicación.

Como conclusión, se puede decir que en líneas generales el diseño realizado en esta primera etapa se ha mantenido a lo largo de la aplicación.

3.1.3. Vista

En el diagrama de vista se definen los diferentes elementos que van a componer la interfaz de usuario.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

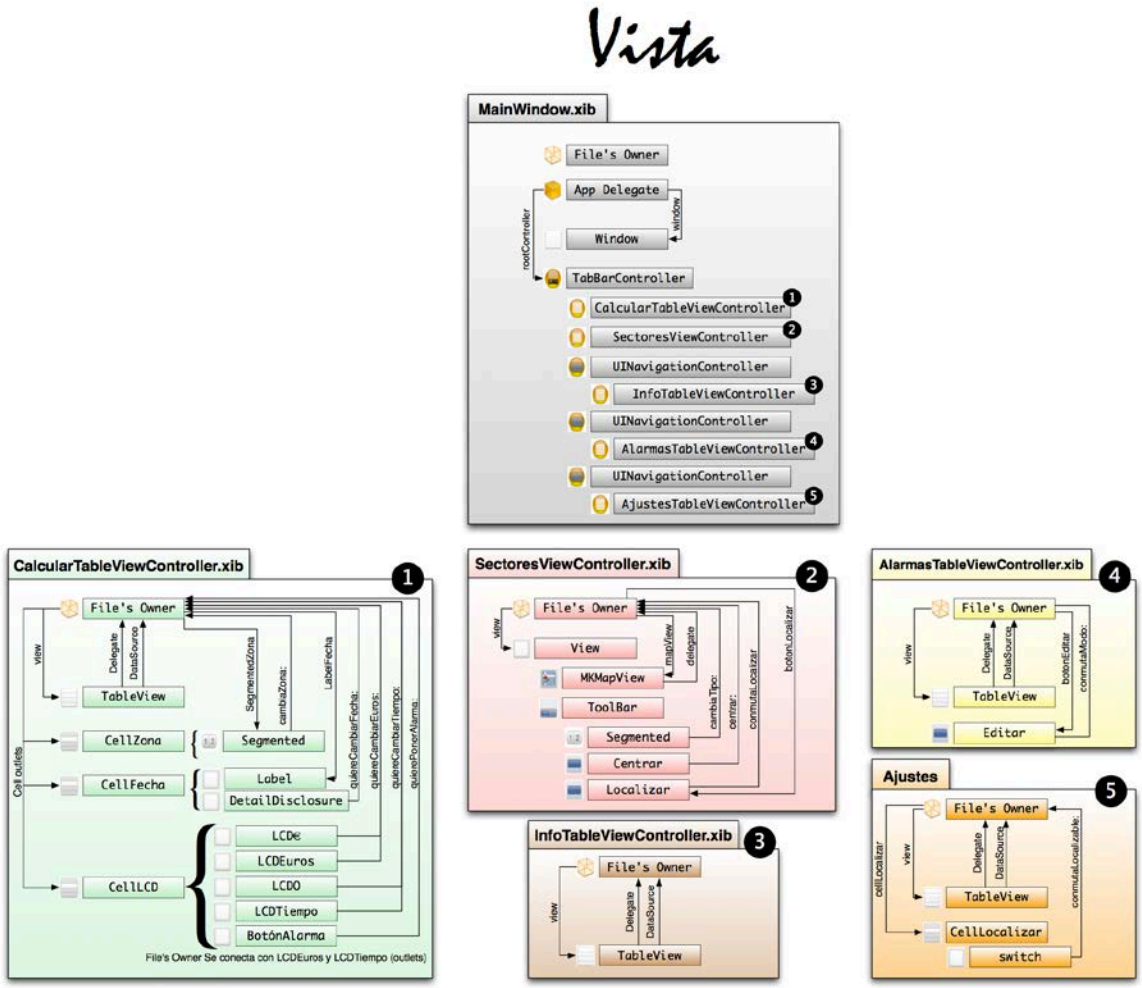


Figura 3-1-3 Diagrama de Vista.

3.1.4. Controlador

La parte del controlador es la que se encarga de coordinar las acciones de usuario (vista) con los datos de la aplicación (modelo). Esta parte es fundamental en el desarrollo de una aplicación para iPhone y suele costar mucho tiempo.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

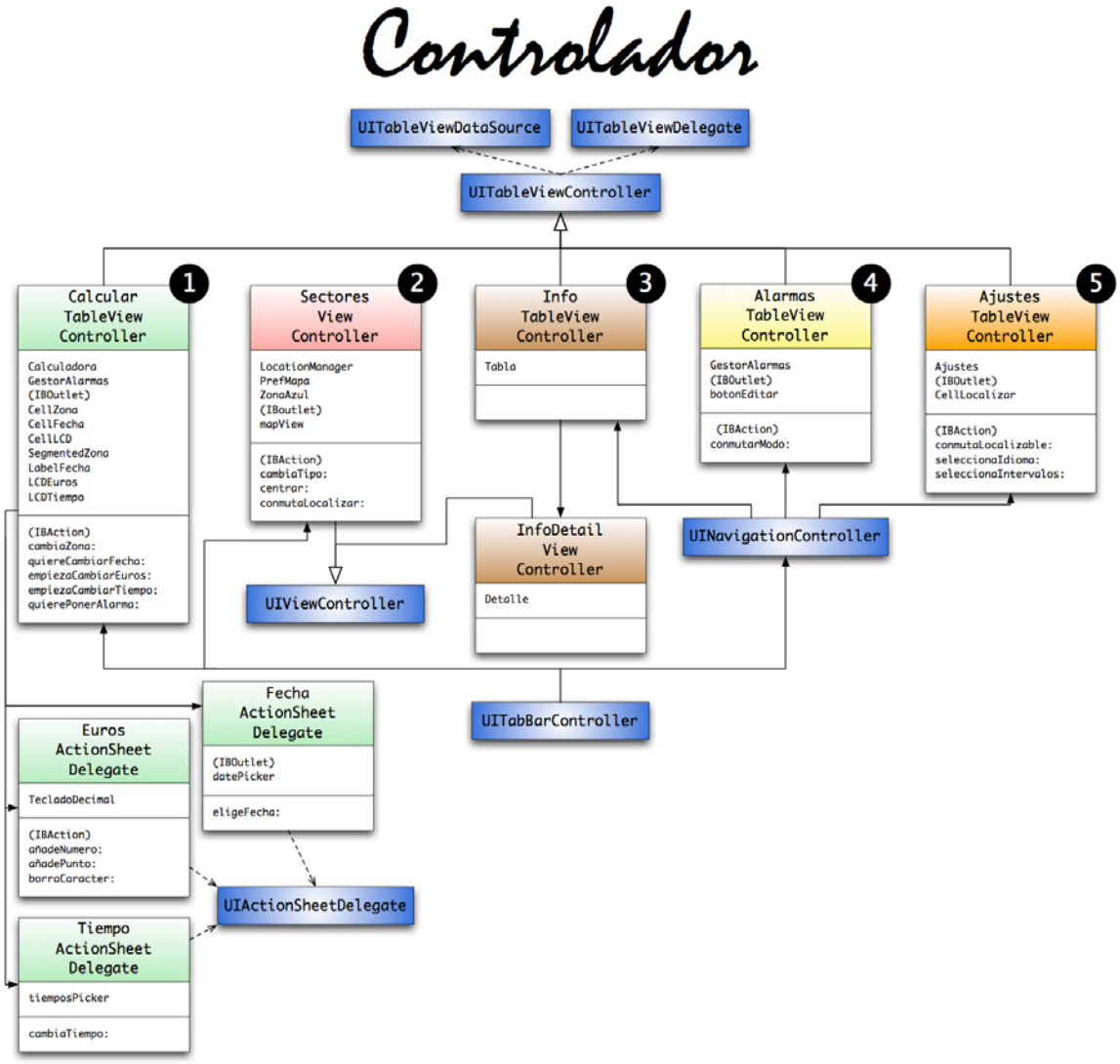


Figura 3-1-4 Diagrama de Controlador.

3.1.5. Desglose de diagramas por funciones

Ya que la aplicación se ha diseñado de manera que puedan implementarse cada una de sus principales características por separado, se ha realizado un diagrama para cada una de ellas. De este modo, podría entregarse cada uno de los diagramas a un equipo para que realicen el desarrollo en paralelo.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

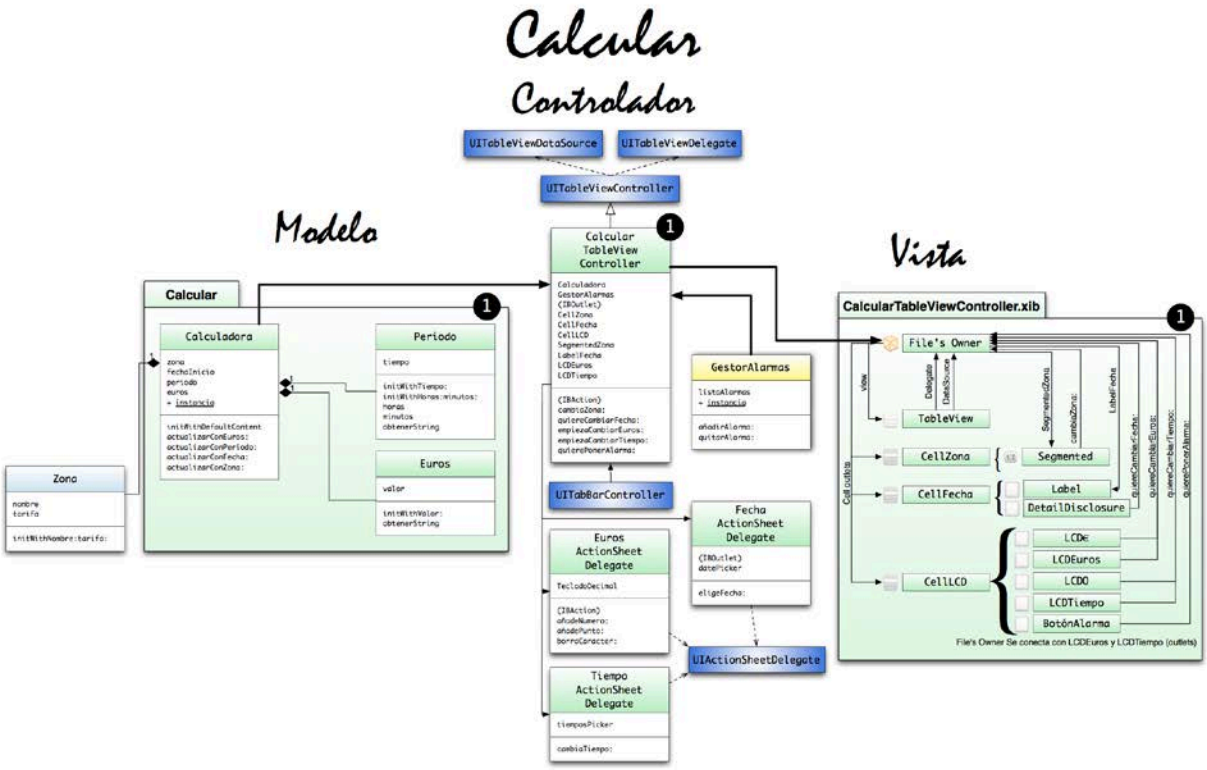


Figura 3-1-5 Diagrama de la pestaña Calcular.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

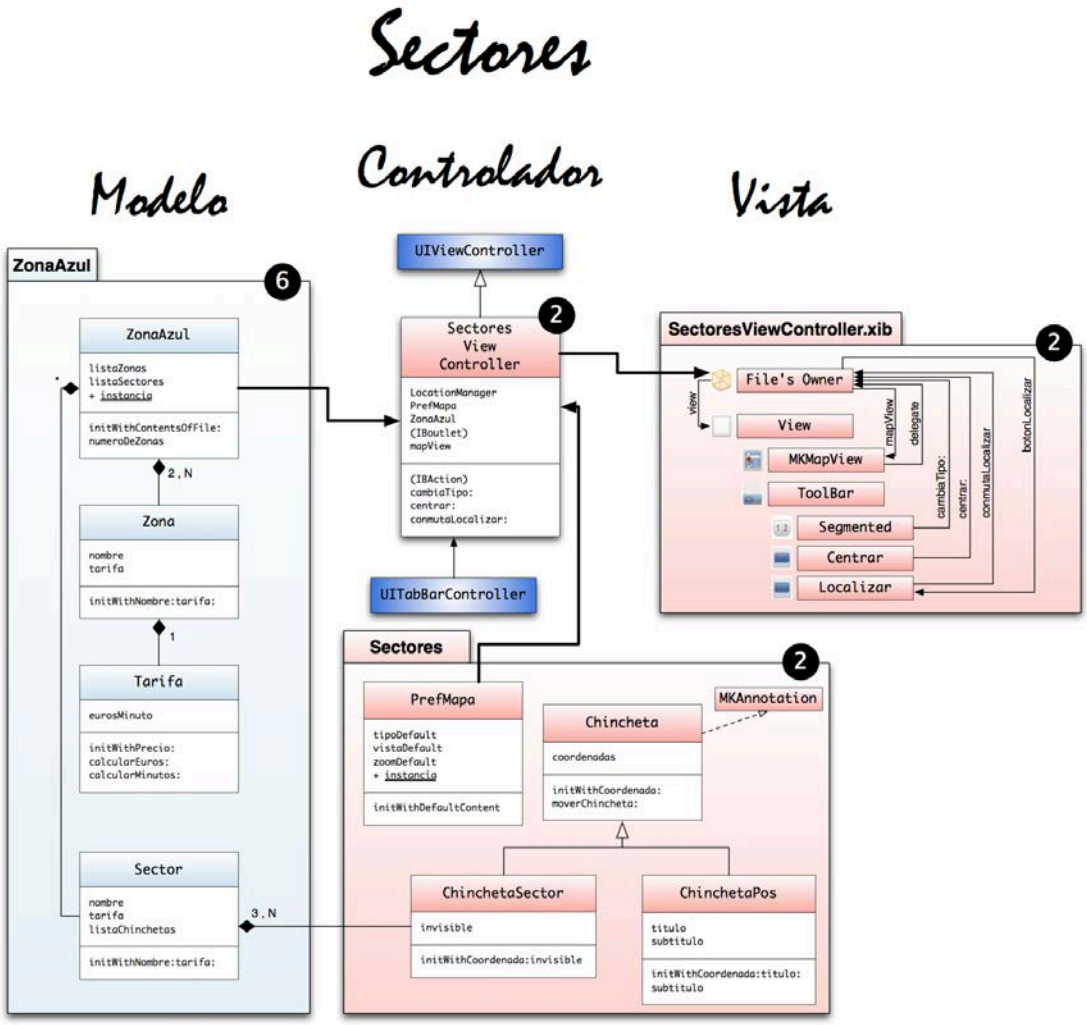


Figura 3-1-6 Diagrama de la pestaña Sectores.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

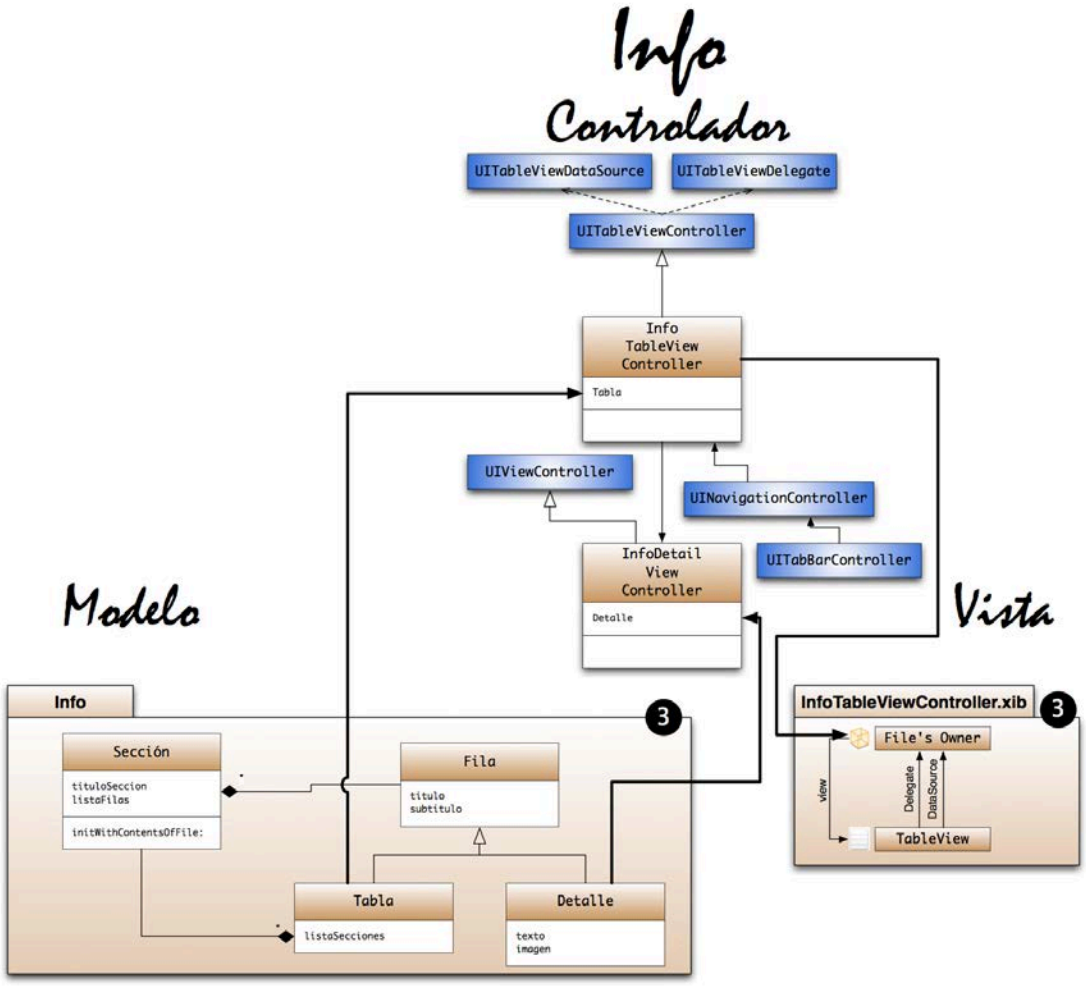


Figura 3-1-7 Diagrama de la pestaña Info.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

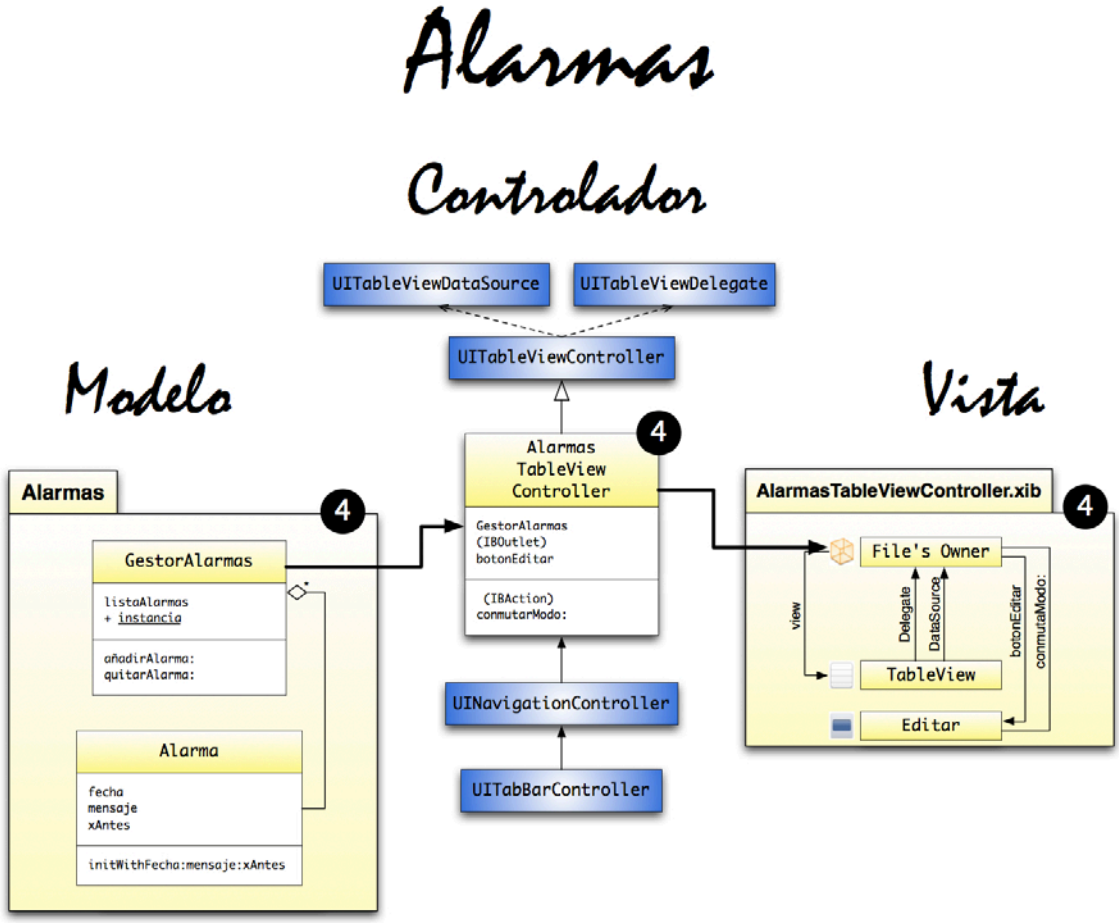


Figura 3-1-8 Diagrama de la pestaña Alarmas.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

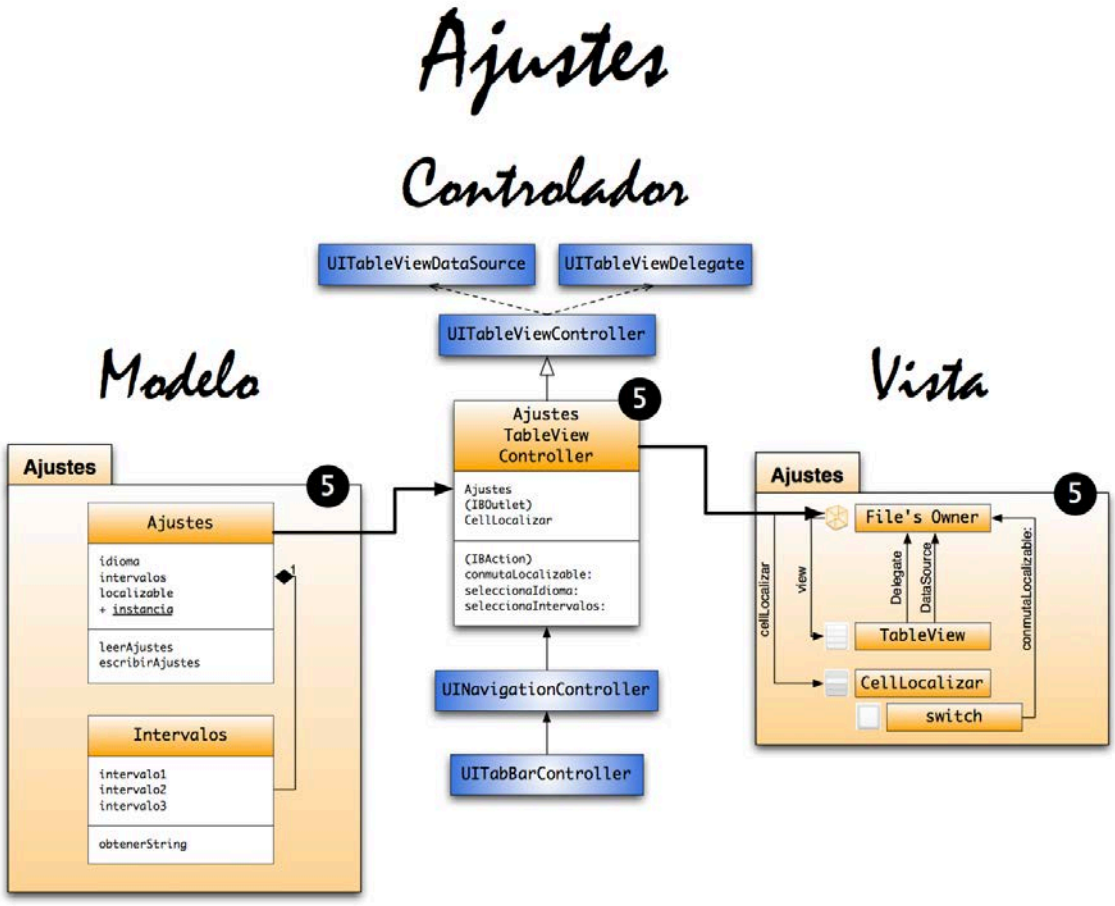


Figura 3-1-9 Diagrama de la pestaña Ajustes.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

4. Codificación: Esqueleto

En esta primera fase de la codificación del programa se ha creado el esqueleto básico de la aplicación en un proyecto de Xcode. Este esqueleto va a servir de base para el desarrollo del resto de la aplicación ya que contiene la funcionalidad más básica del programa.

Se ha implementado la navegación por pestañas con sus distintos controladores y vistas (xib) de acuerdo con la fase anterior del diseño del programa. Siguiendo este diseño también se han añadido las diferentes clases correspondientes al Modelo de la aplicación aunque de momento no se implementen.

Siguiendo el diseño de la interfaz, se han creado las diferentes imágenes e iconos necesarios y se han introducido en el proyecto. Se ha tratado de desarrollar estos iconos con la intención de que sirvan hasta el final de la aplicación y que aparezcan en la versión final de la aplicación, no como simples bocetos.

Al finalizar esta fase se ha conseguido una primera compilación del programa funcional que podría usarse para demostrar parte del diseño de la interfaz y parte de la funcionalidad del producto. Esto permitiría, por una parte, empezar las pruebas del programa en paralelo mientras se desarrolla la siguiente fase y, por otra parte, enseñar el producto al cliente/jefe que nos hubiera encargado la aplicación.

4.1. Creación del proyecto

El proyecto se ha creado utilizando el recién sacado[10] Xcode 4 basándose en iOS 4.3.

[10] 09 de Marzo de 2011
http://itunes.apple.com/us/app/xcode/id422352214

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 4-1-1 Creación del proyecto en la pantalla de bienvenida de Xcode 4

Se ha decidido crear el proyecto basado en la plantilla más simple disponible que nos permita configurar el proyecto desde cero. Por ello se ha elegido la opción *Window-based Application (Figura 4-1-2).*

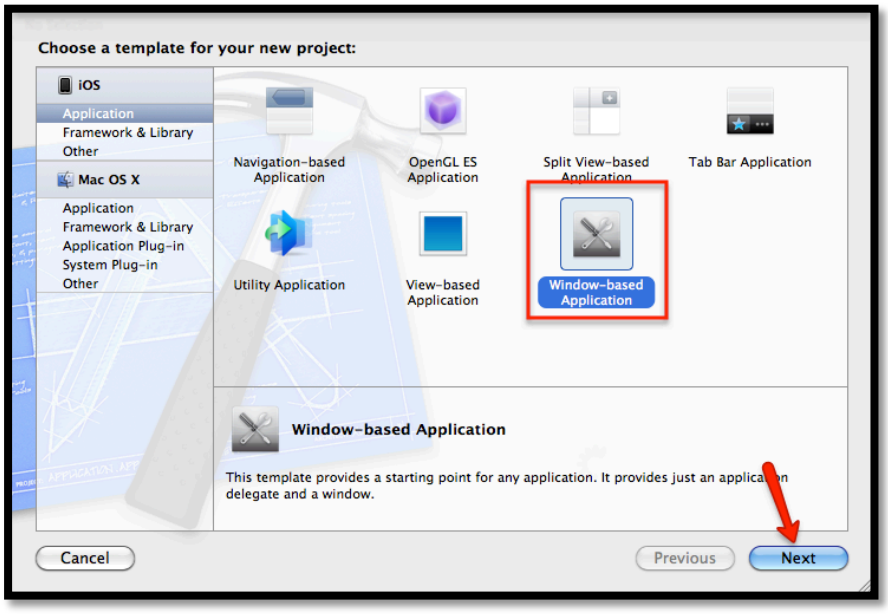


Figura 4-1-2 Seleccionando *Window-based Application*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Se ha escogido *iPark* como nombre del proyecto y se ha seleccionado iPhone como dispositivo para el proyecto. También se ha introducido *upna* como identificador de compañía (Figura 4-1-3).

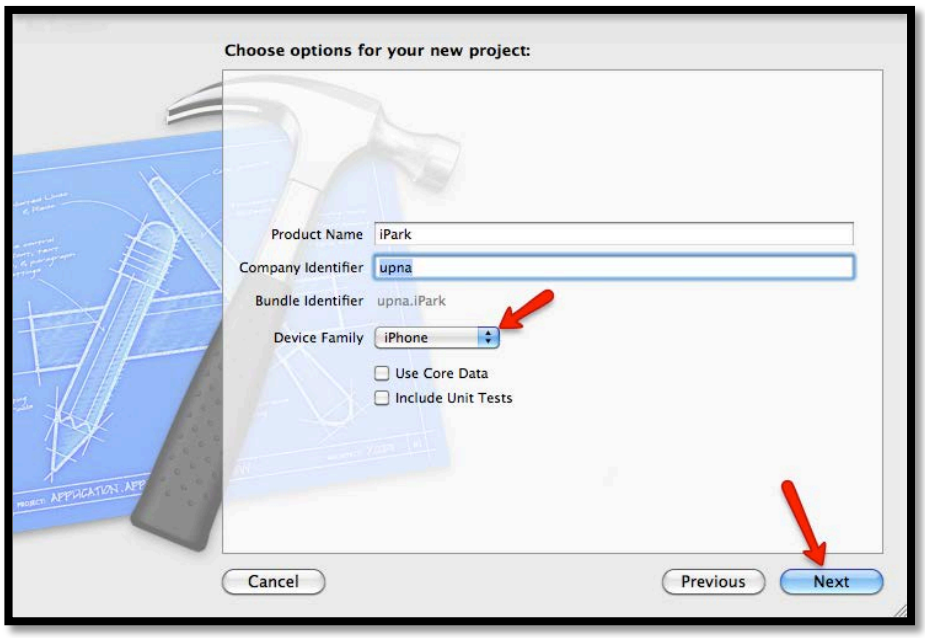


Figura 4-1-3 Pantalla de selección de nombre, compañía y dispositivo.

En la Figura 4-1-4 se muestra el aspecto de la pantalla principal del rediseñado Xcode 4 con el proyecto recién creado.

Figura 4-1-4 Nuevo Xcode 4.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Lo primero que se ha hecho para tener el proyecto y las clases ordenadas ha sido crear una estructura de grupos en el *Project Navigator* (ver Figura 4-1-5). Se ha creado un grupo que englobará todas las clases de la aplicación y dentro de este una grupo para cada una de las funcionalidades básicas de la aplicación (Calcular, Sectores, Info, ...) con sus respectivos grupos para las clases pertenecientes al Modelo, la Vista o el Controlador. También se ha creado otro grupo para los diferentes iconos de la aplicación.

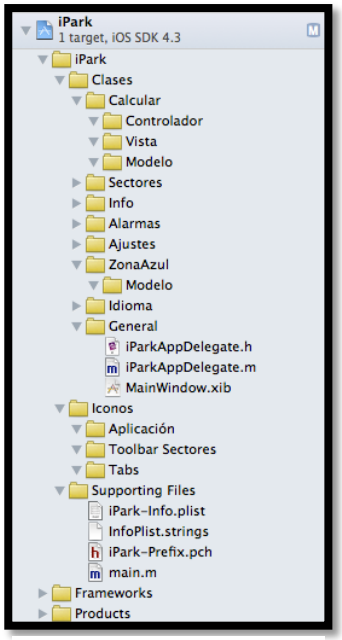


Figura 4-1-5
Estructura del proyecto.

A continuación se han ido añadiendo todas las clases en sus correspondientes carpetas. Las clases de tipo *Controlador* se han creado con las plantillas existentes (Figura 4-1-6) ya que serán subclases de *UIViewController.*

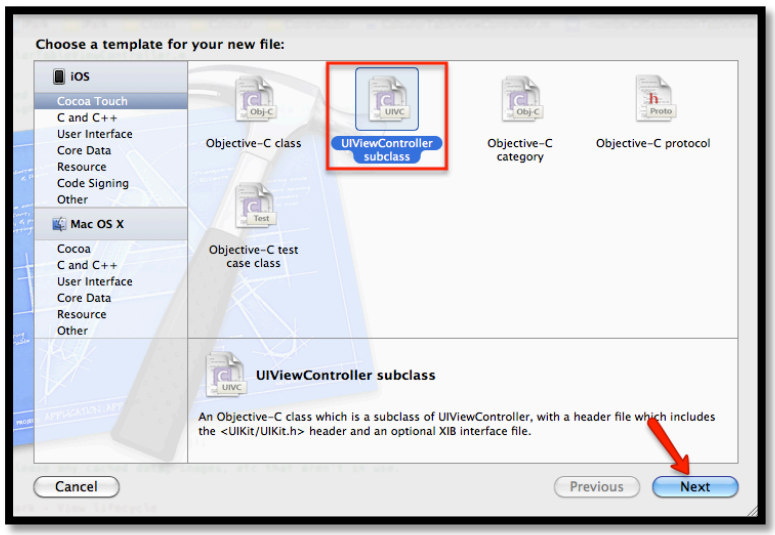


Figura 4-1-6 Creando una subclase de *UIViewController*.

En el siguiente diálogo se elige el tipo de controlador. En los casos de las pestañas de *Calcular*, *Info*, *Alarmas* y *Ajustes* el controlador debe ser subclase de *UI**Table**ViewController* (ver Figura 4-1-7). Mientras que para la pestaña *Sectores* el controlador es subclase de *UIViewController*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

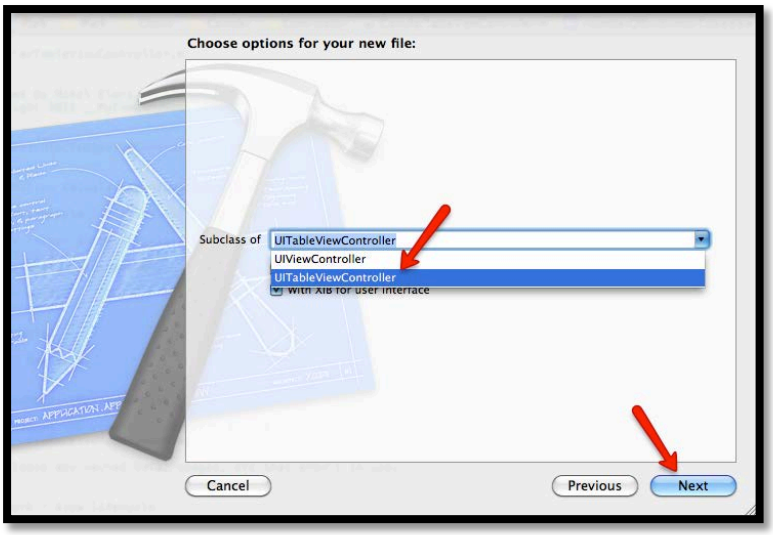


Figura 4-1-7 Eligiendo la superclase de la clase.

Junto con el archivo de cabecera (.h) y el de implementación (.m) también se crea el archivo de interfaz (.xib) que pertenece a la categoría *Vista* y que contendrá un objeto subclase de *UIView* o *UITableView* según lo que hayamos escogido.

El resto de clases se han creado como subclases de *NSObject* en un principio para, más adelante, definir la relación pertinente con las demás clases.

4.2. Creación del mecanismo de pestañas.

Las pestañas son gestionadas por un objeto de tipo *UITabBar* que actúa como controlador principal de la aplicación y se encarga de llamar a los controladores correspondientes para cada una de las pestañas. Se ha añadido este *TabBar* dentro del archivo *MainWindow.xib (*Figura 4-*2-1).*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

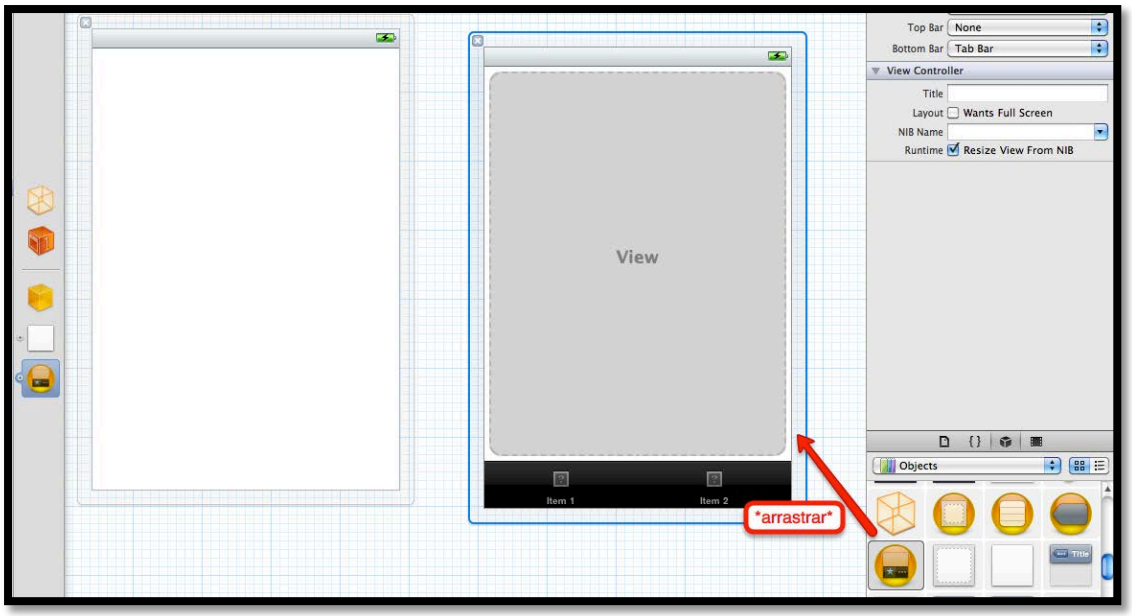


Figura 4-2-1 Añadiendo un *UITabBarController* a *MainWindow.xib.*

Después se han añadido los controladores (ver Figura 4-2-2) que se van a gestionar:

- Para la pestaña *Calcular* se ha añadido un *UITableViewController* y se le ha cambiado la clase en la pestaña *Identity inspector* para que coincida con CalcularTableViewController. También se ha especificado el nombre del *nib* a cargar por dicho controlador en el *Attributes Inspector*.
- Para la pestaña *Sectores* se ha añadido un *UIViewController*, y se ha introducido el nombre de la clase y del *nib*.
- Para cada una de las pestañas *Info*, *Alarmas* y *Ajustes* se ha añadido un *UINavigationController* que contiene, a su vez, una *UINavigationBar* y un *UIViewController.* Se ha cambiado la identidad de ese *UIViewController* al valor correspondiente para cada pestaña y se han especificado todos los nombres de los *nib*s.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

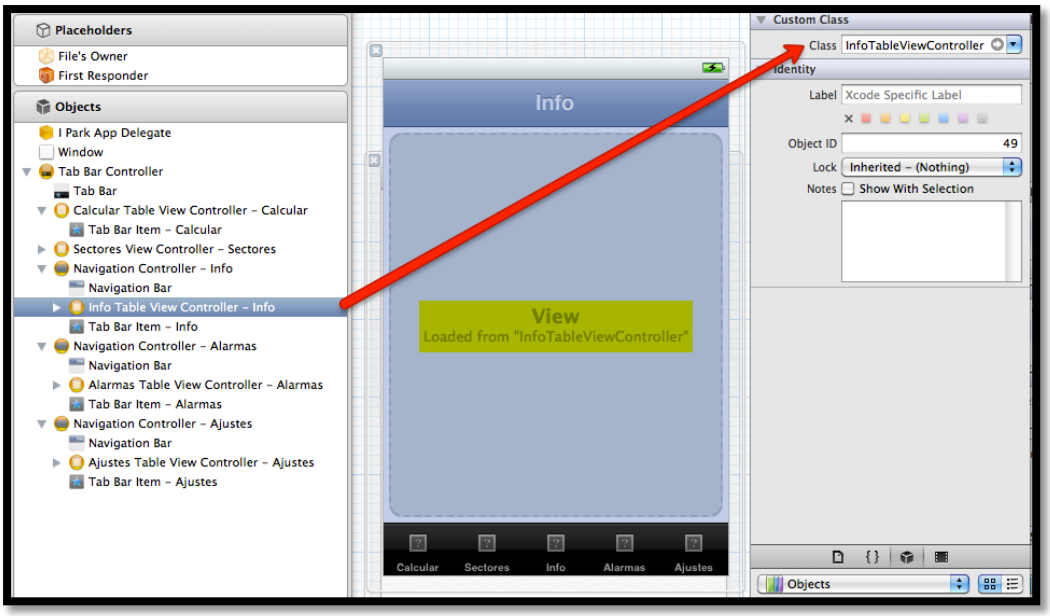


Figura 4-2-2 Controladores añadidos, identidades de las clases cambiadas y vistas cargadas desde los *nib.*

Una vez realizado esto, se ha añadido una propiedad en *iParkAppDelegate* que apunta al controlador de pestañas para, que al iniciarse la aplicación, se muestre en pantalla. En el nuevo Xcode 4 puede hacerse arrastrando desde el *Connections inspector* un nuevo *outlet* en el código del programa como se muestra en la Figura 4-2-3. Esta acción genera el código necesario tanto en la cabecera:

```
@interface iParkAppDelegate : NSObject <UIApplicationDelegate> {
    UITabBarController *_tabBarController;
}
@property (nonatomic, retain) IBOutlet UITabBarController *tabBarController;
(...)
```

como en el archivo .m:

```
(...)
@synthesize tabBarController = _tabBarController;
(...)
- (void)dealloc
{
    [_window release];
    [_tabBarController release];
    [super dealloc];
}
```

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

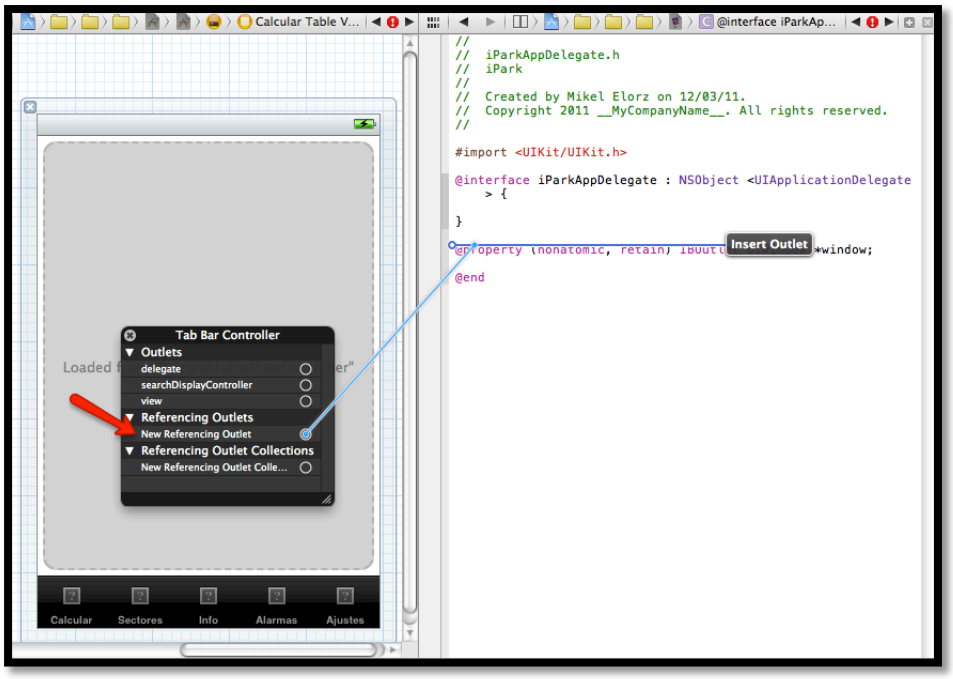


Figura 4-2-3 Añadiendo un nuevo *outlet* en *iParkAppDelegate.h.*

A continuación, se ha añadido el código para mostrar el *tabBarController:*

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

Por último, se comprobado que el *TabBarController* está cargando correctamente las distintas vistas (ver Figura 4-2-4). Para ello se han implementado los métodos siguientes en los controladores que tienen *UITableView:* (se muestra el código de la clase *CalcularTableViewController)*

```
(...)
#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
#warning Metodo con implementación provisional
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{
#warning Metodo con implementación provisional
    // Return the number of rows in the section.
    return 1;
}

- (UITableViewCell *)tableView:(UITableView *)tableView
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
#warning Metodo con implementación provisional
    static NSString *CellIdentifier =
        @"CalcularTableViewControllerCell";
    UITableViewCell *cell = [tableView
        dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc]
                 initWithStyle: UITableViewCellStyleDefault
                 reuseIdentifier: CellIdentifier]
                autorelease];
    }

    // Configure the cell...
    cell.textLabel.text = @"Se ha cargado Calcular";

    return cell;
}
(...)
```

También se ha añadido un *UILabel* en *SectoresViewController*.

Figura 4-2-4 Pestaña *Calcular* mostrando una única celda para comprobar el funcionamiento del *tabBarController*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

4.3. Animación e Iconos

En esta sección se explica cómo se han añadido distintos elementos visuales a la aplicación. Se han creado iconos para cada una de las pestañas, se ha creado un indicador animado para el cambio de pestañas, se ha configurado el icono principal de la aplicación, se ha añadido una imagen de inicio provisional y se ha cambiado la apariencia de la barra de estado superior.

4.3.1. Iconos de las pestañas

Para las pestañas *Sectores*, *Alarmas* y *Ajustes* (ver Figura 3-1) se ha utilizado un pack de iconos disponible en el blog de TWG[11]. Se han modificado para obtener una versión de 60x60 píxeles para pantallas de alta definición y otra de 30x30 para pantallas estándar.

Figura 4-3-1 De derecha a izquierda iconos para las pestañas *Sectores*, *Alarmas* y *Ajustes.*

Para *Sectores* se ha escogido un globo terráqueo ya que esta pestaña contendrá un mapa. El icono de *Alarmas* es obvio e intuitivo, una campana se asocia a una notificación. Los engranajes también son una metáfora ampliamente utilizada para representar ajustes que el usuario puede personalizar, la propia aplicación de ajustes del iPhone muestra una rueda dentada como icono principal.

Para las pestañas *Calcular* e *Info* se han creado sendos iconos desde cero utilizando *Photoshop CS3* y *Pixelmator,* dos aplicaciones para edición de imágenes. Se ha preferido el uso de *Pixelmator*[12] por su sencillez de manejo. Los iconos deben tener un fondo transparente, un dibujo en escala de grises y formato PNG.

El icono de *Calcular* muestra un símbolo de euro que representa el dinero a introducir en el parquímetro y un cronómetro que representa el tiempo

[11] http://blog.twg.ca/2010/11/retina-display-icon-set/twg_retina_icons/
[12] http://www.pixelmator.com/

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

resultante de dicha acción. Las flechas indican que la conversión puede hacerse en los dos sentidos. La *i* de información es un símbolo universal.

Figura 4-3-2 Iconos para Calcular e Información

Se han añadido los iconos al proyecto arrastrándolos a su correspondiente grupo y asegurándose de marcar la opción que hace que los archivos se copien físicamente dentro de la carpeta del proyecto. Las versiones de alta resolución deben llevar el modificador *@2x* en el nombre del archivo. Por ejemplo, el icono *calcular.png* tendrá una resolución de 30x30 píxeles y su versión de 60x60 se deberá llamar *calcular@2x.png.* De esta manera la aplicación detectará automáticamente qué icono cargar según el dispositivo en el que se ejecute.

El siguiente paso ha sido asignar a cada pestaña su icono en el archivo *MainWindow.xib.* Puede hacerse especificando el nombre de la imagen en el *Atributes Inspector* o simplemente arrastrando el icono desde la *Media Library* como se muestra en la Figura 4-3-3.

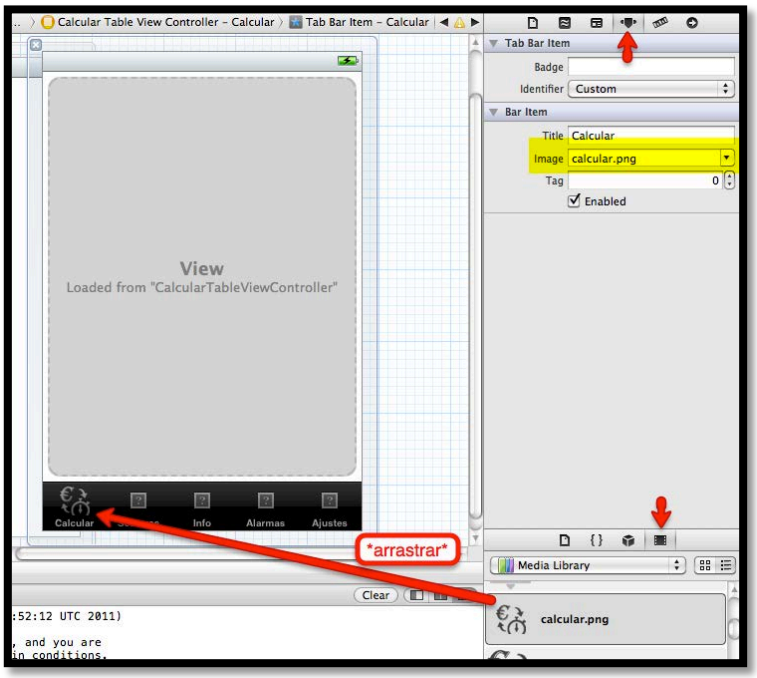


Figura 4-3-3 Asignando un icono a un *UITabBarItem.*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 4-3-4 Resultado después de añadir los iconos de las pestañas.

4.3.2. Indicador de cambio de pestaña

De un tutorial encontrado online[13] se ha sacado la idea de añadir una pequeña animación que sirva para dos propósitos. El primero es ofrecer más feedback al usuario de en qué pestaña está y hacia qué pestaña se mueve. El segundo propósito es meramente estético, pero además sirve para diferenciar la aplicación de otras similares que cuenten con navegación por pestañas. Todo esto se hace de manera no-intrusiva ya que la imagen no ocupará más que unos pocos píxeles de alto.

13 http://howtomakeiphoneapps.com/2011/02/how-to-skin-your-iphone-app-with-core-animation/

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

En la Figura 4-3-5 se aprecia la animación creada por el que realiza el tutorial pero no comparte la imagen ni la explicación de como funciona por lo que ha sido creada desde 0.

Figura 4-3-5 Animación mostrada en la web de origen.

El funcionamiento de la animación es sencillo pero no evidente. Se trata de una imagen larga con un indicador en el centro. Si se mueve esa imagen para que coincida el centro de la imagen con el centro de nuestra pestaña, obtenemos la animación deseada ya que una parte de la imagen queda oculta (fuera de la pantalla). Por lo tanto, la imagen debe ser de ancho:

$$w_{imagen} = (2n - 1) \cdot \ w_{pestaña}$$

Donde n es el número de pestañas y $w_{pestaña}$ es el ancho de una pestaña que a su vez puede calcularse con:

$$w_{pestaña} = \frac{w_{pantalla}}{n}$$

Con $w_{pantalla} = 640\ píxeles$ (para pantallas retina) la ecuación inicial queda:

$$w_{imagen} = (2n - 1) \cdot \frac{640}{n}$$

O lo que es lo mismo:

$$w_{imagen} = 2w_{pantalla} - \ w_{pestaña}$$

En definitiva, con 5 pestañas, el ancho de la imagen debe ser 1152 píxeles.

Se ha creado con Photoshop y Pixelmator la imagen de la Figura 4-3-6:

Figura 4-3-6 Indicador visual de selección de pestaña. 1152x14 píxeles (fondo transparente)

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Una vez se ha añadido la imagen al proyecto, se ha procedido a la programación de la animación. La flecha se mostrará en cuanto la aplicación termine de cargar, por lo cual el propio *iParkAppDelegate* deberá encargarse tanto de mostrarla por primera vez como de moverla cada vez que se cambie a una nueva pestaña.

Para mostrarla al inicio, se ha añadido una llamada a *anadeFlecha* dentro de la función *application:didFinishLaunchingWithOptions*, después de la sentencia que añade el *tabBarController*:

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    // Añadir el tabBarController
    self.window.rootViewController = self.tabBarController;
    // Añadir la flecha indicadora de la posición
    [self anadeFlecha];

    [self.window makeKeyAndVisible];
    return YES;
}
```

Después se han definido tres constantes en *iParkAppDelegate.h*:

- *kPosFlecha*: la posición en píxeles a partir del borde del *tabBar* en la que se mostrará la imagen. Un número más alto implica una posición más baja.
- *kDuración:* duración de la animación al cambiar de pestaña (en segundos)
- *kNombreFlecha*: indica el nombre de la imagen que debe cargarse.

También se ha añadido una propiedad a la clase de tipo *UIImageView* llamada *flechaTabBar* que se utiliza para mantener la referencia a la imagen. Por último se ha declarado el método *anadeFlecha* y el método auxiliar *posHorizontalParaIndice:* (encargado de calcular la posición a la que debe moverse la imagen cuando se cambie de pestaña).

```
// Posición de la flecha relativa al borde superior del tab
// (Un número más alto la hace bajar). Por defecto 5
#define kPosFlecha      5
// Duración de la animación en segundos. Por defecto 0.2
#define kDuracion       0.2
// Nombre de la imagen a carcar para la flecha
#define kNombreFlecha   @"flecha.png"

@interface iParkAppDelegate : NSObject <UIApplicationDelegate> {
    UITabBarController *_tabBarController;
    //Indicador visual del tab seleccionado
```

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
    UIImageView         *flechaTabBar;
}

@property (nonatomic, retain) IBOutlet UITabBarController *tabBarController;
@property (nonatomic, retain) IBOutlet UIWindow *window;

@property (nonatomic, retain) IBOutlet UIImageView *flechaTabBar;

// Función que devuelve la de la flecha para el índice dado
- (CGFloat) posHorizontalParaIndice:(NSUInteger)tabIndex;
// Crea y muestra la imagen de la flecha que señala al tab seleccionado
- (void) anadeFlecha;
```

Después se ha añadido el *synthesize* correspondiente en *iParkAppDelegate.m* y se han implementado las dos funciones declaradas en la cabecera:

```
(...)
@synthesize flechaTabBar;
(...)
#pragma mark -
#pragma mark flecha TabBar

// Crea y muestra la imagen de la flecha que señala al tab seleccionado
- (void) anadeFlecha {
    // Conseguir la imagen con metodo de conveniencia
    UIImage *flechaTabBarImage = [UIImage imageNamed:kNombreFlecha];
    //Crear un UIImageView a partir de la imagen
    self.flechaTabBar = [ [ [UIImageView alloc]
                           initWithImage:flechaTabBarImage]
                         autorelease];
    // Calcular la posición vertical a partir de parámetros disponibles
    CGFloat posVertical =   self.window.frame.size.height
                          - self.tabBarController.tabBar.frame.size.height
                          - flechaTabBarImage.size.height
                          + kPosFlecha;
    // Asignarle su posición
    flechaTabBar.frame = CGRectMake(
        //              x                   ,      y
        [self posHorizontalParaIndice:0] ,  posVertical,
        //              w                   ,      h
        flechaTabBarImage.size.width     ,  flechaTabBarImage.size.height
    );
    // Añadir la subvista
    [self.window addSubview:flechaTabBar];
}

// Función que devuelve la posición del punto medio del tab número indiceTab
- (CGFloat) posHorizontalParaIndice: (NSUInteger)indiceTab {
    // Calcular el ancho de un tab
    CGFloat anchoTab =   self.tabBarController.tabBar.frame.size.width
                       / self.tabBarController.tabBar.items.count;
    // Calcular la mitad del ancho de un tab
    CGFloat anchoMedioTab =   (anchoTab / 2.0)
                            - (flechaTabBar.frame.size.width / 2.0);

    return (indiceTab * anchoTab) + anchoMedioTab;
}
```

Con estas dos funciones se ha conseguido que la flecha se muestre encima de la primera pestaña. Para que la flecha cambie de posición cuando se seleccionan distintas pestañas se ha especificado que *iParkAppDelegate* sea también *delegate* del *tabBar* (ver Figura 4-3-7). De esta manera disponemos de la función

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

tabBarController:didSelectViewController que se llamará cada vez que el usuario elija una nueva pestaña.

Figura 4-3-7 Haciendo que el *tabBar* delegue en *iParkAppDelegate* (en el fichero MainWindow.xib)

```
#pragma mark -
#pragma mark Tab Bar Delegate

// El tabbar nos notifica cuando cambia de tab (metodo del delegate)
- (void) tabBarController:(UITabBarController *)theTabBarController
  didSelectViewController:(UIViewController *)viewController {
    // Empezar la animación con core animation (block approach)
    [UIView animateWithDuration: kDuracion
                          delay: 0
                                 // Que empiece rápido y que
                                 // termine despacio
                        options: (UIViewAnimationOptionCurveEaseOut |
                                 // Que se pueda interactuar con
                                 // la interfaz mientras dure la animación
                                 UIViewAnimationOptionAllowUserInteraction |
                                 // Si se interrumpe, se empieza del
                                 //estado actual, no desde el principio
                                 UIViewAnimationOptionBeginFromCurrentState)
                     animations:^(void) {
                         // flechaTabBar.frame.origin.x
                         // no se puede modificar por lo que
                         // leemos todo el frame en auxFrame,
                         CGRect auxFrame = flechaTabBar.frame;
                         // lo modificamos
                         auxFrame.origin.x = [self posHorizontalParaIndice:
                                              self.tabBarController.selectedIndex ];
                         // y lo volvemos a asignar
                         flechaTabBar.frame = auxFrame;
                     } completion:^(BOOL finished) {
                     }
    ];
}
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 3-8 Flecha en la primera pestaña y después de seleccionar la tercera pestaña.

4.3.3. Icono principal

El icono principal, creado durante la fase de diseño de la interfaz, se puede agregar de manera gráfica o especificando su nombre en el *plist* principal de la aplicación. Sin embargo, para deshabilitar el brillo por defecto del icono, debemos configurar dicho *plist*. Se puede editar el *plist* directamente o ir a la pestaña *Info* del proyecto y cambiar los valores allí.

Como se muestra en la figura 4-3-9, se ha editado el campo *Icon File* con el nombre de la imagen (en este caso icon.png). y se ha añadido un campo *Icon already includes gloss effect* con el valor booleano *YES*.

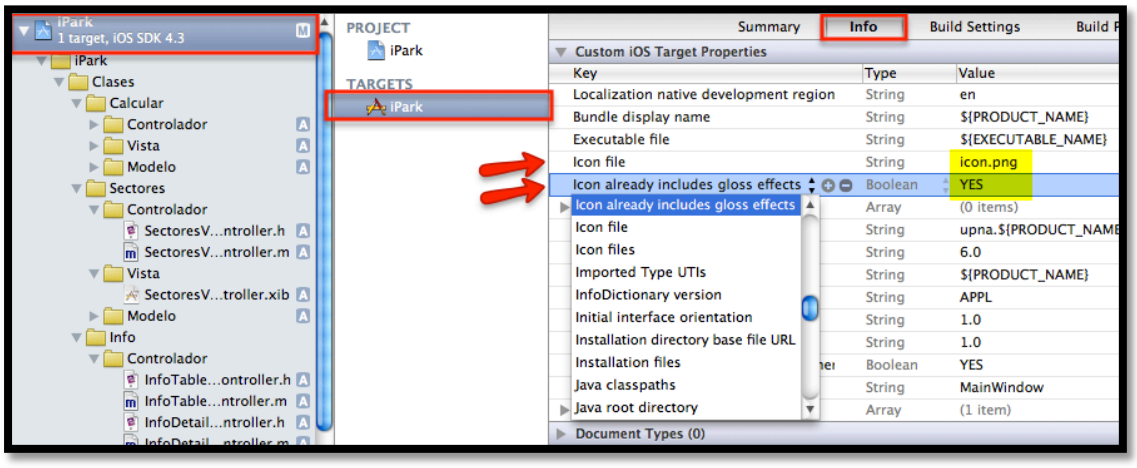


Figura 4-3-9 Eligiendo el tipo de campo al añadir una propiedad al proyecto.

Ahora al ejecutar la aplicación esta buscará la imagen llamada icon.png y la utilizará como icono principal. Además, si el dispositivo cuenta con pantalla retina, buscará el archivo *icon@2x.png* que en nuestro caso tiene el doble de

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

resolución (114x114 píxeles en pantallas retina frente a 57x57 en pantallas estándar).

Figura 4-3-10 Resultado después de añadir el icono principal.

4.3.4. Imagen de Inicio.

La imagen de inicio es una imagen que la aplicación cargará nada más iniciarse antes de que se carguen las vistas. Esta imagen es lo primero que visualiza el usuario y debe asemejarse mucho a la pantalla que el usuario verá cuando todos los elementos de la interfaz terminen de cargarse. Esto hace que la aplicación parezca que se carga más rápidamente cuando realmente lo que se muestra al principio no es más que una imagen estática.

Por lo tanto, se ha creado una imagen de 640x960 píxeles (@2x) a partir de una captura de pantalla de lo primero que se carga. Los 40 píxeles superiores (para la imagen @2x, 20 píxeles para la imagen normal) pertenecen a la barra de estado del dispositivo por lo que se han eliminado y se ha dejado esa parte de la

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

imagen con fondo transparente. De esta manera el usuario seguirá viendo su barra de estado actual. Por último se ha añadido la propiedad *Launch image* justo debajo de la entrada *Icon file* y se le ha asignado el valor *inicio.png.*

Figura 4-3-11 En los primeros instantes en los que se lanza la aplicación se muestra la pantalla de la izquierda. Aunque no lo parezca es una imagen estática que simula la interfaz que se verá a continuación (imagen derecha).

4.3.5. Barras de estado y de navegación negras.

Como se puede observar en la Figura 4-3-11, la barra de estado que informa sobre el operador, la cobertura, la hora, la batería, etc. aparece opaca y de color negro mientras que lo común es que sea gris. Esto se ha conseguido añadiendo una entrada al *plist* principal con la clave *Status bar style.* Esta propiedad ofrece tres valores diferentes (ver Figura 4-3-12): Gray style (default), Transparent black style (alpha of 0.5) y Opaque black style. Se ha escogido esta última.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

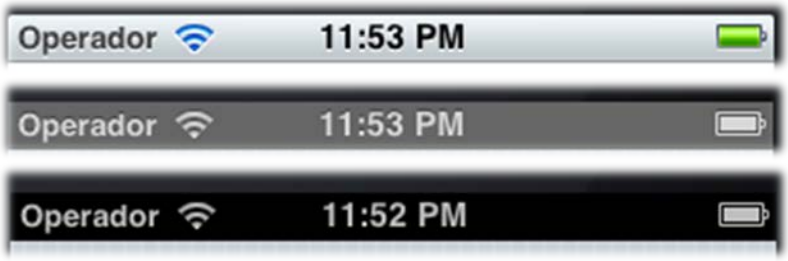


Figura 4-3-12 De arriba a abajo: estilo gris por defecto, negro transparente y negro opaco.

Por último, se ha cambiado el color de las *UINavigationBar* a negro, acorde con la barra de estado. Para ello se han seleccionado todas las instancias de *UINavigationBar* que existen dentro de *MainWindow.xib* y mediante el inspector de atributos se les ha cambiado el estilo a negro opaco tal como se muestra en la Figura 4-3-13.

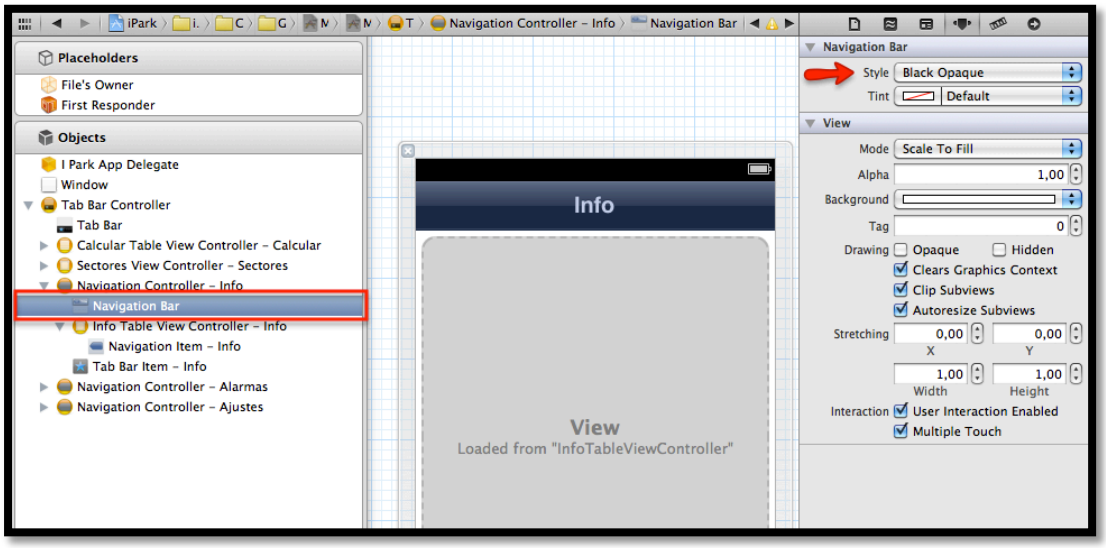


Figura 4-3-13 Cambiando el estilo de una *UINavigationBar.*

Puede verse el efecto conjunto de la barra de navegación negra con la barra de estado negra en la Figura 4-3-14.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 4-3-14 Resultado, barra de estado negra, barra de navegación negra y barra de pestañas negra.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

4.4. Traducción

Se ha implementado un método de traducción de cadenas basado en una estructura *hashmap* (en el mundo de objective-c un *NSDictionary*). La idea es cargar un solo *hashmap* en la aplicación que contenga como *key* el *string* en castellano y como objeto un *array* con sus traducciones.

Así definido, al programar se utilizarán todo el rato las frases en castellano y se traducirán automáticamente en tiempo de ejecución según el idioma seleccionado en preferencias.

La clase *TraductorHash* tendrá un método estático que devuelva la traducción del *string* introducido según el idioma actual. (En caso de estar seleccionado el castellano, devolverá la misma cadena pasada como parámetro).

```
self.navigationItem.title = [TraductorHash traducir:@"Ajustes"];
```

El diccionario se cargará, como se ha dicho antes, una sola vez de manera estática al inicializar la clase *TraductorHash*. Los datos los obtendrá de una *property list* que recreará la estructura de datos necesaria (Figura 4-4-1)

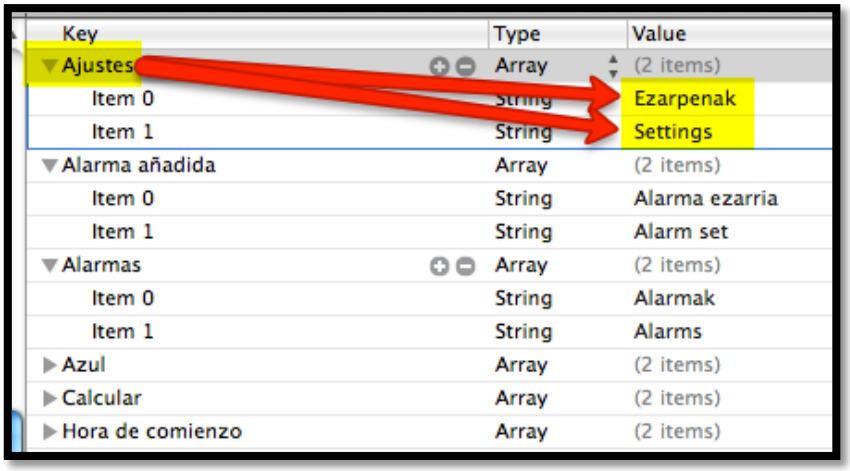


Figura 4-4-1 plist con diferentes traducciones.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

5. Codificación: Calcular

En esta segunda fase de la implementación se han añadido todas las funcionalidades de la primera de las pestañas: *Calcular*. Esta es la pestaña principal, la que implementa las funcionalidades del requisito principal de la aplicación. Aquí es donde los usuarios van a poder realizar los cálculos de tiempo, dinero y fechas de las diferentes zonas de estacionamiento limitadas.

5.1. Objetivo

La apariencia y funcionalidad de esta pestaña se basan en el diseño realizado anteriormente que se puede ver en la Figura 5-1-1.

Figura 5-1-1 Diseño de la interfaz, objetivo a alcanzar.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Como puede apreciarse, la interfaz va a tener cuatro componentes:

- Sección de la zona: permite elegir en qué zona (azul, naranja o roja) quiere aparcar el usuario.
- Sección de la fecha: permite elegir a en qué fecha y hora va a aparcar el usuario, para poder calcular la hora de finalización del periodo.
- Sección de los LCD: permite al usuario realizar el cálculo euros-> tiempo y tiempo->euros.
- Sección de la Alarma: mostrará, a partir de los datos anteriores, la fecha y hora límite hasta la que el usuario puede estacionar. También permitirá poner una alarma.

Cada una de las secciones detallada arriba se corresponde con una sección o grupo de filas de la *UITableView.* Cada sección va a tener una sola *UITableViewCell* que además va a ser estática. Según las guidelines de *Apple*[14] la mejor manera de hacer esto es cargar las cell desde un xib. Por lo tanto, añadimos 4 instancias de *UITableViewCell* al archivo *CalcularTableViewController.xib* (Figura 5-1-2).

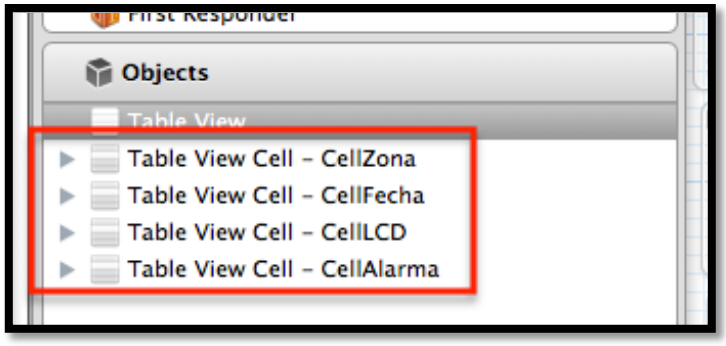


Figura 5-1-2 4 cells añadidas con el Interface Builder.

Una vez hecho esto, se han creado los 4 outlets en *CalcularTableViewController* para tener acceso a ellos. Pueden crearse manualmente o, como se ve en la Figura 1-3 utilizando el *assistant editor* y el *Interface Builder.*

[14] Table View Programming guide http://bit.ly/hymzcw.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

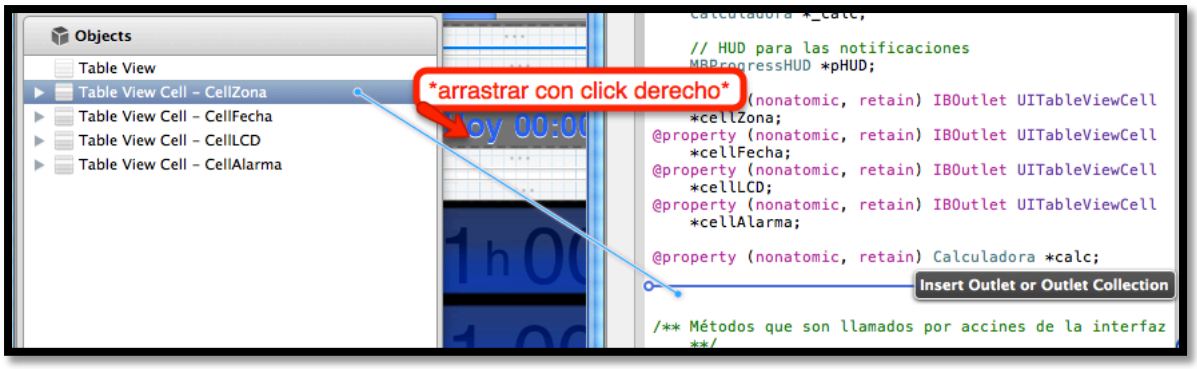


Figura 5-1-3 Añadiendo *outlets* automáticamente desde el *Interface Builder*.

5.2. Mecanismos utilizados

A continuación se detallan una serie de mecanismos y "herramientas" utilizadas a la hora de desarrollar esta pestaña.

5.2.1. Acceso a subvistas

Se han definido una serie de constantes que van a ayudar a encontrar las subvistas dentro de cada cell. Asignando al tags de cada subvista la constante correspondiente en el interface builder tendremos acceso a ella.

Por ejemplo, definimos:

```
#define kZonaLabelTag                11
```

y ponemos el tag a 11 en un *UILabel* dentro de cellZona. Después, cada vez que necesitemos la referencia a ese label, bastará con obtenerla mediante:

```
UILabel *labelZona = (UILabel *) [self.cellZona viewWithTag: kZonaLabelTag];
```

5.2.2. Cambios de datos

Para todo el funcionamiento de esta pestaña necesitaremos de las clases del modelo de datos Calculadora, Fecha, Periodo, Euros y Zona (perteneciente al paquete ZonaAzul) que se detallarán más adelante. La clase Calculadora es la que englobará una Fecha un Periodo de tiempo y unos Euros. Además, tendrá la habilidad de realizar los cálculos pertinentes con las tarifas de las zonas, teniendo en cuenta el periodo, los euros, ...

Uno de los problemas con esta pestaña es que cualquier cambio en los datos de entrada afecta a todos los demás. Por ejemplo, cambiar los euros cambia la duración del aparcamiento pero también la hora de finalización. Pero cambiar la

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

zona también puede cambiar el periodo en el caso de que el periodo seleccionado rebase el periodo máximo permitido en la zona nueva seleccionada.

Así pues, se han implementado diferentes métodos (*actualizaZona:animado:* , *actualizaFecha:animado:* , ...) en el controlador para que, al modificar cualquier parámetro (periodo, zona, ...) se realicen los cambios pertinentes en el resto.

5.2.3. Animación de los cambios en los datos

Se han creado dos pequeñas animaciones para dar feedback al usuario cuando una de sus acciones desencadene algún cambio en otro sitio que no sea justamente lo que ha tocado. Esto ayudará a aumentar la comprensión por parte del cliente de la dinámica de la Zona Azul.

La primera es una animación que mueve una vista dada de izquierda a derecha, como si ésta se agitase. Sirve para indicar que lo que se ha movido ha sufrido un cambio como consecuencia de la última acción realizada.

La animación se ha realizado utilizando la estrategia de bloques:

```
[UIView animateWithDuration: 0.2 animations:^(void) {
    // Mover n píxeles a la derecha
    CGPoint centro = vista.center;
    centro.x += n;
    vista.center = centro;
} completion:^(BOOL finished) {
    // Al terminar otro bloque de animación para mover
    // 2n píxeles a la izquierda
        // Al terminar el segundo bloque, un tercero
        // para mover n píxeles a la derecha de nuevo
}];
```

Es importante que la animación utilice los bloques `^(void){}` ya que estos los gestiona iOS de manera eficiente y los puede paralelizar. Hasta ahora los procesadores de dispositivos con iOS eran de un sólo núcleo y por lo tanto, no se le sacaba un gran partido a esta manera de programar. Pero con la salida del iPad 2 se ha sentado el precedente del primer dispositivo iOS con procesador de doble núcleo, el *Apple A5.* Con este nuevo procesador, que presumiblemente se utilizará en el iPhone 5, se exprimirá el potencial de la implementación de las animaciones como bloques paralelizables.

La segunda hace fluctuar una vista, haciendo la vista casi transparente y opaca de nuevo de manera gradual. Se utiliza para indicar al usuario que la vista

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

que parpadea acaba de actualizarse con el resultado del cálculo asociado a la última acción.

El mecanismo es parecido, en el bloque *animations* se establece el alpha (transparencia) de la vista un valor cercano a 0. En el bloque *completion* se devuelve el alpha al estado inicial.

5.3. Secciones en el controlador y clases asociadas

A continuación se muestran las cuatro secciones de la pestaña calcular y sus características más destacables junto con las clases pertenecientes al modelo que las acompañan. Para conocer el funcionamiento interno en detalle y cómo se implementan los distintos métodos de delegados de la *tableView*, etc. consulte el código adjunto con esta memoria.

Cabe destacar la función *viewWillAppear:* donde se realizan las labores de inicialización de la vista. En este método se devuelven todos los parámetros a sus valores por defecto. Interesa que siempre que se acceda de nuevo a la pestaña de calcular estén configurados los valores por defecto, que son los cálculos que se realizarán con más asiduidad.

También se ha añadido la función *appDidBecomeActive* que se ha configurado para que se le llame cuando la aplicación vuelva del estado inactivo. Esta función llamará a *viewWillAppear:* con el propósito de restablecer todos los valores de la interfaz.

5.3.1. Sección zona

El primer componente va a constar de un *UILabel* que va a mostrar la palabra *Zona* (o sus traducciones) y de un *UISegmentedControl.* El control tendrá tres segmentos para las zonas azul, naranja y roja.

Figura 5-3-1 Apariencia final de la sección.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Para mejorar el feedback que le ofrecemos al usuario, queremos los elementos de la interfaz cambien de color con la zona seleccionada.

Cambiar el color de un label es sencillo, por ejemplo para la Zona Azul:

```
UILabel *labelZona = (UILabel *) [self.cellZona viewWithTag: kZonaLabelTag];
labelZona.textColor = [UIColor blueColor];
```

Sin embargo, no es posible cambiar el color de un *UISegmentedControl.* La solución pasa por hacer una subclase del control segmentado e implementar los métodos de dibujado. Por suerte, se dispone del código gracias a Matteo Caldari[15], que ha creado y compartido el código de un control segmentado que admite diferentes colores. Desde aquí se agradece su labor y se le atribuye todo el mérito.

Por lo tanto, una vez añadida la clase, se puede utilizar este código para cambiar de color al *CustomSegmentedControl.*

```
CustomSegmentedControl *sc = (CustomSegmentedControl *) sender;
sc.tintColor = [UIColor blueColor];
```

La sección zona está relacionada con la clase de modelo del paquete *ZonaAzul* llamada Zona. Esta clase representa una zona (azul, naranja, ...) e implementa unos métodos para realizar los cálculos de tiempo o dinero con las tarifas adecuadas. Estas tarifas han sido consultadas en la sede de Dornier S.A.[16], empresa que gestiona la Zona de Estacionamiento Limitado y Restringido (ZEL) entre otras cosas.

Los métodos

```
// Método que calcula los euros a partir de un periodo dado
- (Euros *) calcularConPeriodo: (Periodo *) p;
// Método que calcula el periodo a partir de unos euros dados
- (Periodo *) calcularConEuros: (Euros *) e;
```

son los encargados de realizar este cálculo. El algoritmo es muy similar en los dos casos (uno lo realiza a la inversa del otro):

---

[15] http://matteocaldari.it/2010/05/a-uisegmentedcontrol-with-custom-color

[16] http://www.pamplona.es/verPagina.asp?idPag=20-40495

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- Tener en cuenta el periodo y coste mínimo de aparcamiento (en estos momentos 10 minutos -> 25 céntimos).

El periodo restante se divide en 2 tramos: las primeras 2 horas (incluyendo los 10 minutos del punto anterior) a una tarifa de 5 céntimos cada 3 minutos (para todas las zonas) y las siguientes horas a 5 céntimos cada 10 minutos (sólo para la zona naranja). Por lo tanto:

- Si el periodo es mayor que 2 horas, se aplica la tarifa de la zona naranja: 5 cent/10 min.
- Llegados a este punto, el tiempo restante es menor o igual a 2 horas y se le debe aplicar la tarifa más cara: 5 cent / 3 min.

En todo este proceso hay que tener cuidado con los redondeos. En el cálculo de euros a periodo, se asume que los céntimos son múltiplos de 5 (porque al usuario no se le permite introducir otros valores) y como los bloques son de 5 céntimos en 5 céntimos, no hay problema. Pero en el cálculo de minutos a euros, los minutos van de uno en uno, por lo tanto debe cogerse la mínima cantidad de euros que permita alcanzar dicho periodo.

Por ejemplo a 1 € le corresponden 55 minutos. Pero al hacer el cálculo inverso, tanto a 53, 54 como a 55 minutos les corresponde la misma cantidad de 1 €.

De todas maneras, se ha observado que en más de un ticket obtenido en las zonas azul y naranja en el año 2011, la cantidad de minutos que las máquinas de la calle permiten aparcar varía en ±1 minuto.

En la Figura 5-3-2 pueden observarse 4 de estos tickets. En todos ellos se ha introducido exactamente 1 € pero en los dos de arriba la máquina ha dado 55 minutos y en los dos de abajo solamente 54. Se cree que esto se debe al momento en el que se introducen las monedas, si el minuto actual está muy avanzado o acaba de empezar. Para la aplicación, se ha decidido ser consistente con los datos oficiales de tarifas[17].

[17] http://www.pamplona.es/verPagina.asp?idPag=20-51881

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 5-3-2 Tickets de aparcamiento con datos inconsistentes.

5.3.2. Sección hora de comienzo.

Esta sección consta de dos *UILabel,* el primero es un título estático que se muestra en color gris y el segundo muestra el día y la hora en la que el usuario

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

desea empezar el periodo de estacionamiento (Figura 5-3-3). Por defecto tendrá el valor de la hora actual y se prevé que este será el valor que la mayoría de gente utilizará.

De todas formas, se le permite al usuario modificar esta fecha en unas cuantas horas (como mucho hasta el día de mañana) con el fin de que planifique el aparcamiento. Podría haberse dejado más rango de fechas, pero se considera que eso confundiría al usuario y sería muy raramente utilizado y por ello se ha limitado esa posibilidad.

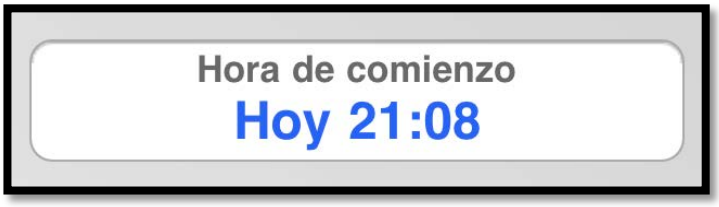


Figura 5-3-3 Sección hora de comienzo con Zona Azul seleccionada.

Cuando el usuario toca la cell en cualquier parte, se muestra un *UIActionSheet* al que se le ha añadido un *UIPicker* personalizado en el que puede elegir el día y la hora (Figura 5-3-4). Además, al seleccionar un nuevo valor, el *UILabel* de la sección cambia dinámica y animadamente para reflejar el cambio.

El cambio en la hora de comienzo no afecta al cálculo de euros ni de periodo pero sí al de la fecha final. Esto lo calcula la clase *Calculadora.*

Figura 5-3-4 Seleccionando una nueva hora con el *UIActionSheet*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

A esta sección se le asocia la clase Fecha; que representa una hora, unos minutos y un string para la fecha ("Hoy"/"Mañana"). Dispone de un método para representar el objeto como string, un método para comparar si dos fechas son iguales y un método para devolver la fecha en forma de *NSDate.*

5.3.3. Sección LCD

Esta es la sección más complicada y que más esfuerzo ha requerido. Aquí es donde el usuario especifica el tiempo que desea estar o los euros que desea gastar. Los cálculos que permiten saber cuántos euros corresponden al periodo y qué periodo corresponde a los euros los realiza la zona seleccionada actualmente. La zona también limita cuál es el número máximo de euros o de tiempo que la zona permite estacionar. Todo esto hace que cuando se cambie una zona, se deban recalcular ambos datos y ver si se pasan de los límites.

Como puede apreciarse en la Figura 5-3-5, se ha intentado dar una apariencia de pantalla LCD con números estilo 7-segmentos. Se han creado los LCD en tres colores para cada una de las zonas y para cada color dos versiones una ligeramente más oscura que la otra para diferenciarlas mejor.

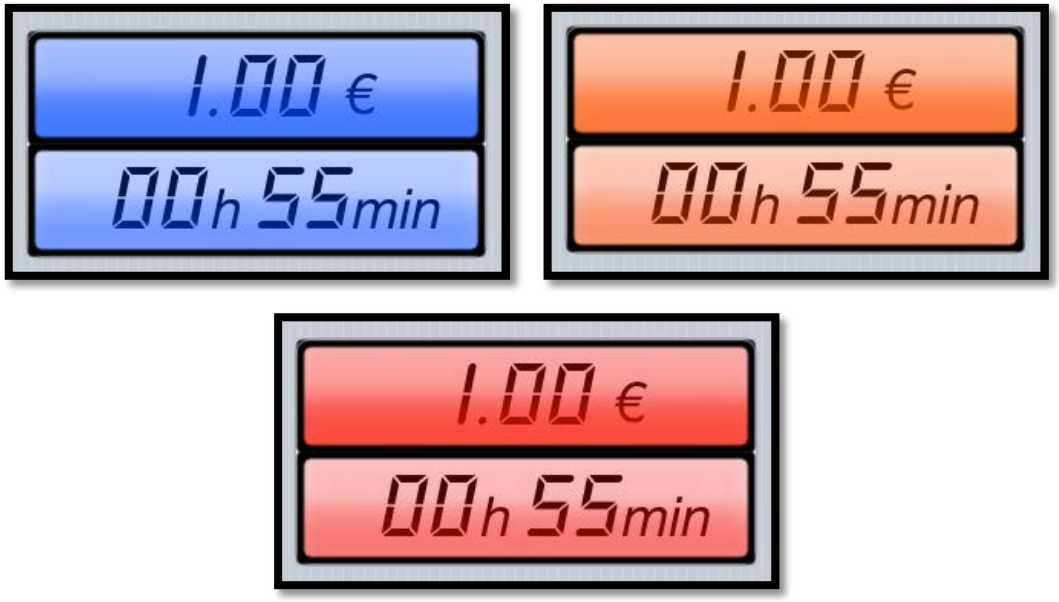


Figura 5-3-5 Los tres colores de los LCD

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Esta apariencia se consigue con:

- Dos vistas principales que actúan de contenedor de cada LCD.
- Dentro de cada contenedor:
  - Los labels necesarios para mostrar la información.
  - Una UIImageView para el color.
  - Un botón transparente arriba del todo para captar los toques del usuario.

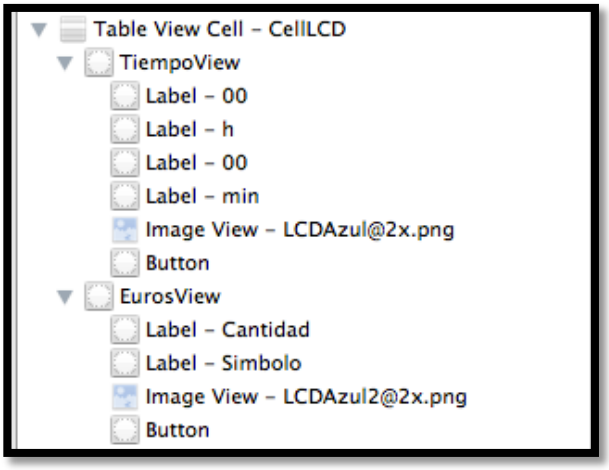


Figura 5-3-6 Jerarquía de vistas para los dos LCD.

También se ha prestado especial interés en la manera de editar los dos valores. Para ofrecer una interfaz consistente, se ha optado por que el método de input sea un *UIActionSheet* con pickers personalizado para los euros y el periodo que tengan en cuenta los valores mínimos y máximos seleccionables.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 5-3-7 Editando el periodo (izquierda) y los euros (derecha).

También se ha conseguido que cuando se cambie el valor de un dato, se refleje instantáneamente en la parte de arriba y se calcule el resultado correspondiente.

Nótese también que el LCD cuyo dato está siendo editado aparece siempre posicionado arriba. Mientras, el LCD cuyo dato se calcula como consecuencia aparece siempre abajo. Para conseguir esto se ha creado una animación que intercambie la posición de los dos LCD cuando sea necesario. Basta con intercambiar los centros de las vistas contenedor entre ellas dentro de un bloque de animación. De esta manera, el usuario sabe en todo momento qué está editando y qué está siendo calculado.

A esta sección se le asocian dos clases: Periodo y Euros. Las dos no son más que una manera de representar la realidad, un periodo de tiempo en horas y minutos y una cantidad monetaria en euros y céntimos. Disponen de métodos para comparar, para obtener su representación como texto, para añadir o restar céntimos o minutos...

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

5.3.4. Sección hora de fin

La última sección es en la que se ve la hora a la que finalizará el tique de aparcamiento. También es desde la que se pueden añadir las alarmas para que los usuarios no olviden volver y retirar el coche.

Para que no se confunda con la sección de la hora de comienzo (que es seleccionable) se le ha dado una apariencia que no invita a ser tocado. Así se diferencia claramente y los usuarios no tendrán tanta tentación de intentar cambiar la hora de fin. Este problema se ha detectado gracias a unas pruebas informales que se hicieron con un grupo reducido de usuarios que expusieron que las secciones se parecían demasiado y causaban confusión.

En cuanto al botón de la alarma, se ha diseñado uno de cada color. Tocándolo aparece un diálogo (ver Figura 5-3-8) que permite elegir con cuánta antelación queremos que nos avise la alarma.

Esta elección sirve tanto como para dar la posibilidad de elegir cuándo se pone la alarma como para realmente confirmar que se desea establecerla. Así se evita que se pongan alarmas sin querer.

Figura 5-3-8 Diálogo de confirmación de alarma.

Una vez elegida la alarma, se presenta una ventana de información (Figura 5-3-9) durante unos segundos para que el usuario sepa exactamente cuándo ha puesto la alarma y dónde ir para cambiarla (se minimiza con una animación hacia la pestaña *Alarmas*).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 5-3-8 HUD que se muestra al añadir una alarma.

Como en los ajustes se pueden cambiar las opciones que se le dan al usuario al poner la alarma, surge un problema. Un error ocurría en la aplicación si el usuario pulsaba el botón de poner la alarma y sin elegir ni cancelar sale de la aplicación y cambia dicho ajuste. Por ello, se ha utilizado una subclase de *UIActionSheet* que detecta automáticamente cuándo el usuario sale de la aplicación para ocultarse automáticamente.

Este otro problema se detectó gracias a que se realizó un exhaustivo análisis de todas las posibles acciones que el usuario puede emprender en cada momento.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

5.4. Resultado

Figura 5-4-1 Pestaña calcular en castellano. Zona azul y zona naranja.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 5-4-2 Pestaña calcular en euskera.
Zona azul (izquierda) y zona naranja editando la fecha de inicio (derecha).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 5-4-3 Pestaña calcular en ingles. Zona roja (izquierda) y zona roja editando el periodo (derecha).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

6. Codificación: Sectores

Esta tercera pestaña es la encargada de mostrar un mapa de Pamplona en la que se pueden consultar los diferentes sectores de residentes. Está pestaña está basada en el uso de *Google maps*, ampliamente conocido por los usuarios de todas las plataformas móviles.

6.1. Objetivo

El principal objetivo es el de ofrecer a los usuarios de la Zona Azul que tengan tarjetas de residentes una herramienta para comprobar si un determinado lugar pertenece a su sector o no.

También puede ser utilizado por el resto de usuarios para comprobar si determinado lugar (como, por ejemplo, su posición actual) es una zona de pago o no.

Se cree necesario que las características del mapa y el modo de interaccionar sean similares a los que podemos encontrar en la aplicación *Mapas.* Así la curva de aprendizaje se suaviza mucho ya que se utilizan mecanismos ya conocidos.

6.2. Mecanismos utilizados

6.2.1. Visualización de mapa, gestos de zoom, ...

Todos los mecanismos de visualizar el mapa, gestionar la localización del usuario, los gestos para hacer zoom, desplazarse por diferentes regiones... están ya implementados en el *framework* MapKit. La clase *MKMapView* es una vista que podemos simplemente añadir en el *nib* correspondiente y funcionará automáticamente mostrando imágenes de *Google maps*.

6.2.2. Posición de los controles del mapa

Es ampliamente conocido que los dispositivos móviles no destacan por lo general por la amplitud de sus pantallas. Uno de los mayores problemas es que añadir ciertas funcionalidades implica añadir controles, botones, ... Esto reduce considerablemente la cantidad de mapa que es visible en cada momento.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Para solucionarlo se ha optado por una solución similar a la que implementa la aplicación *Mapas*: un botón que 'doble' el mapa como una página de papel hacia arriba para revelar los controles del mapa (ver Figura 2-1).

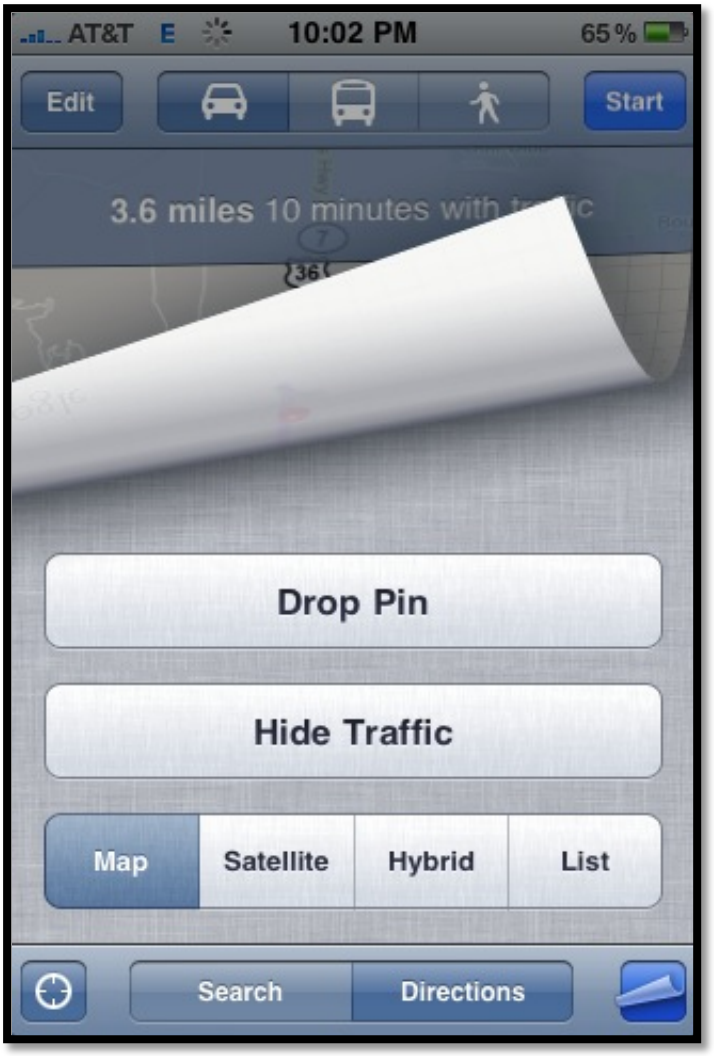


Figura 6-2-1 Aplicación oficial de mapas del iPhone.

Para contar con una funcionalidad parecida se ha utilizado *FDCurlViewControl*[18]. Esta es una clase que especificándole una vista cualquiera (en este caso la instancia de *UIMapView*) se encarga de animar un efecto de doblado parándolo a mitad de camino. El resultado en iPark se puede apreciar en la figura 6-2-2.

[18] https://github.com/FairfaxMobile/FDCurlViewControl

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

De esta manera todos los controles están escondidos debajo del mapa excepto el botón que permite mostrarlos y ocultarlos. La animación es, además, muy fluida y visualmente llamativa.

Figura 6-2-2 iPark mostrando el mapa normal a la izquierda y después de tocar el botón de doblar el mapa.

6.2.3. Control de cambio de tipo de mapa

Los mapas pueden visualizarse de tres maneras diferentes: con la apariencia normal de mapa de papel (como aparece en la Figura 6-2-2), como fotos reales tomadas desde un satélite y mezclando ambas (Figura 6-2-3). Para permitir a los usuarios elegir entre una de estas tres, la mejor manera es mediante un *segmented view*.

Este control tiene un color azul que en principio no debe cambiarse. Puede cambiarse pero el compilador da un aviso y es posible que la aplicación no fuera aceptada en la *App Store* por culpa de este detalle.

Como se quería un color gris más acorde con la estética del programa, se ha utilizado la clase *CustomSegmentedControl* utilizada con anterioridad en la pestaña *Calcular.* El aspecto de este control puede apreciarse en la Figura 6-2-2.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 6-2-3 Tipos de vista satélite (izquierda) e híbrido (derecha) en iPark.

El cambio de un tipo de mapa a otro es sencillo con el framework MapKit. Basta con ajustar la propiedad *mapType* del *mapView* al valor de una constante para cada tipo de mapa:

```
UISegmentedControl *sc = sender;
switch ([sc selectedSegmentIndex]) {
    case kSegmentoMapa:
        self.mapView.mapType = MKMapTypeStandard;
        break;
    case kSegmentoSatelite:
        self.mapView.mapType = MKMapTypeSatellite;
        break;
    case kSegmentoHibrido:
        self.mapView.mapType = MKMapTypeHybrid;
        break;
    default:
        break;
}
```

6.2.4. Centrar el mapa

A menudo, al navegar por un mapa y hacer zoom el usuario se 'pierde' y le cuesta trabajo volver a la posición inicial para obtener una vista general de los sectores.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Para facilitar la acción que se puede plantear el usuario de 'volver al principio, volver a la vista general' se ha creado un botón que al tocarlo centra el mapa en una zona y a un nivel de zoom predefinidos.

Esto se consigue haciendo que el mapa navegue hasta una *región*. La región se define como un registro que incluye una coordenada (CLLocationCoordinate2D) y un espacio (*MKCoordinateSpan*). La coordenada representa el centro y el espacio representa la anchura de la región a mostrar (es una manera de representar el zoom).

Por suerte, existen las funciones *MKCoordinateRegionMake, CLLocationCoordinate2DMake y MKCoordinateSpanMake* que facilitan la creación de estas estructuras de datos.

```
// Mueve el mapa hasta centrarlo en Pamplona
- (void) irAlCentro: (BOOL) animado {
    CLLocationCoordinate2D centro =
                    CLLocationCoordinate2DMake(42.810589, -1.646056);
    MKCoordinateSpan espacio =
                        MKCoordinateSpanMake( 0.031320,  0.033240);
    MKCoordinateRegion region =
                       MKCoordinateRegionMake(  centro ,  espacio );
    [self.mapView setRegion: region animated: animado];
}
```

Los valores de coordenadas y distancias son valores experimentales que se han recogido centrando el mapa a mano y anotando los valores que se registran.

Se ha decidido que al pulsar el botón de centrar, como al elegir el tipo de mapa, este se baje automáticamente para mostrar los cambios de la acción realizada. Esto causaba problemas ya que el mapa se centraba mientras la animación de desdoblarse se efectuaba. Por lo cual, al pulsar en el botón de centrar, este no se debe centrar hasta que el mapa baje. Si no, la animación de centrar el mapa no se realizaba correctamente y el mapa parecía cambiar repentinamente de región.

También se ha añadido una pequeña funcionalidad que consiste en ejecutar la acción de centrar el mapa cuando el teléfono se agite. Si alguna vez el usuario se frustra y decide agitar el dispositivo, la interfaz volverá a su posición inicial.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 6-2-4 Mapa centrado en Pamplona.

6.2.5. Localización del usuario

Esta función permite al usuario conocer su posición actual y determinar en qué sector cae o si no cae en ninguno. Se puede hacer un seguimiento exhaustivo y más completo de la posición del usuario con el *framework CoreLocation. p*ero para hay una manera más sencilla y adecuada para esta aplicación y el tiempo de desarrollo del que se dispone. Simplemente asignando el valor booleano 'YES' a la propiedad *showsUserLocation* del *mapView* el dispositivo localiza al usuario de la manera más adecuada posible. No es necesariamente la más precisa, por cuestiones de consumo de energía.

Lo que si que ha requerido más esfuerzo a la hora de desarrollar ha sido decidir si una coordenada del mapa está dentro o no de un determinado polígono. La solución que se ha encontrada se compone de los siguientes pasos:

- Obtener el punto del mapa para la coordenada:

```
MKMapPoint mapPoint = MKMapPointForCoordinate(mapCoordinate);
```

- Para un polígono dado, encontrar su vista en la pantalla

```
MKPolygonView *polygonView = (MKPolygonView *)
                             [self.mapView viewForOverlay:pol];
```

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- Obtener el punto de la pantalla correspondiente al punto del mapa:

```
CGPoint polygonViewPoint = [polygonView pointForMapPoint: mapPoint];
```

- Utilizando la función CGPathContainsPoint , podemos saber si el *CGPath*[19] de la vista del polígono contiene o no el punto de la coordenada

```
CGPathContainsPoint(polygonView.path, NULL, polygonViewPoint, NO);
```

En resumidas cuentas, obtenemos la representación en la pantalla de la coordenada y el polígono del mapa y nos valemos de las librerías de *Core Graphics* que tienen métodos que realizan la función que estamos buscando.

Puede apreciarse el resultado en la siguiente Figura:

Figura 6-2-5 Botón de localizar seleccionado y el mensaje que aparece al tocar el punto azul.

[19] http://bit.ly/iJ9WTz Una descripción matemática de una serie de formas o líneas.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

6.3. Datos de los sectores.

En este apartado se pretende explicar la descripción y el origen de los datos que componen un sector así como la manera en la que se accede a ellos, se procesan y se muestran en la vista de mapas.

6.3.1. Descripción de los datos.

Un sector tiene únicamente un número, un color y un *array* de coordenadas.

El número es el que identifica a cada sector y actualmente en la Zona Azul existen los sectores 1, 2, 3 y 4[20].

El color es elegido arbitrariamente. La única consideración que se ha tenido en cuenta ha sido la de no utilizar a la vez los colores azul, naranja, rojo y verde ya que de esa manera podrían confundirse con los colores de las diferentes zonas.

Por último se necesita una serie de coordenadas que, en orden, formen un polígono sobre el mapa. Esta es la parte más importante de un sector porque delimita la porción del mapa que es de pago y en la que unos residentes pueden aparcar y otros no. Con estas coordenadas se puede crear un objeto de tipo *MKPolygon* que será necesario para representar el sector en el mapa.

6.3.2. Origen de los datos.

Los datos de la extensión que ocupa cada sector se han obtenido de la *ordenanza reguladora de las zonas de estacionamiento limitado y restringido*[21]. En la última página del documento encontramos el mapa de los sectores (Figura 6-3-1).

---

[20] http://bit.ly/Sector01 , http://bit.ly/Sector02, http://bit.ly/Sector03, http://bit.ly/Sector04

[21] [PDF] http://bit.ly/Ordenanza

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

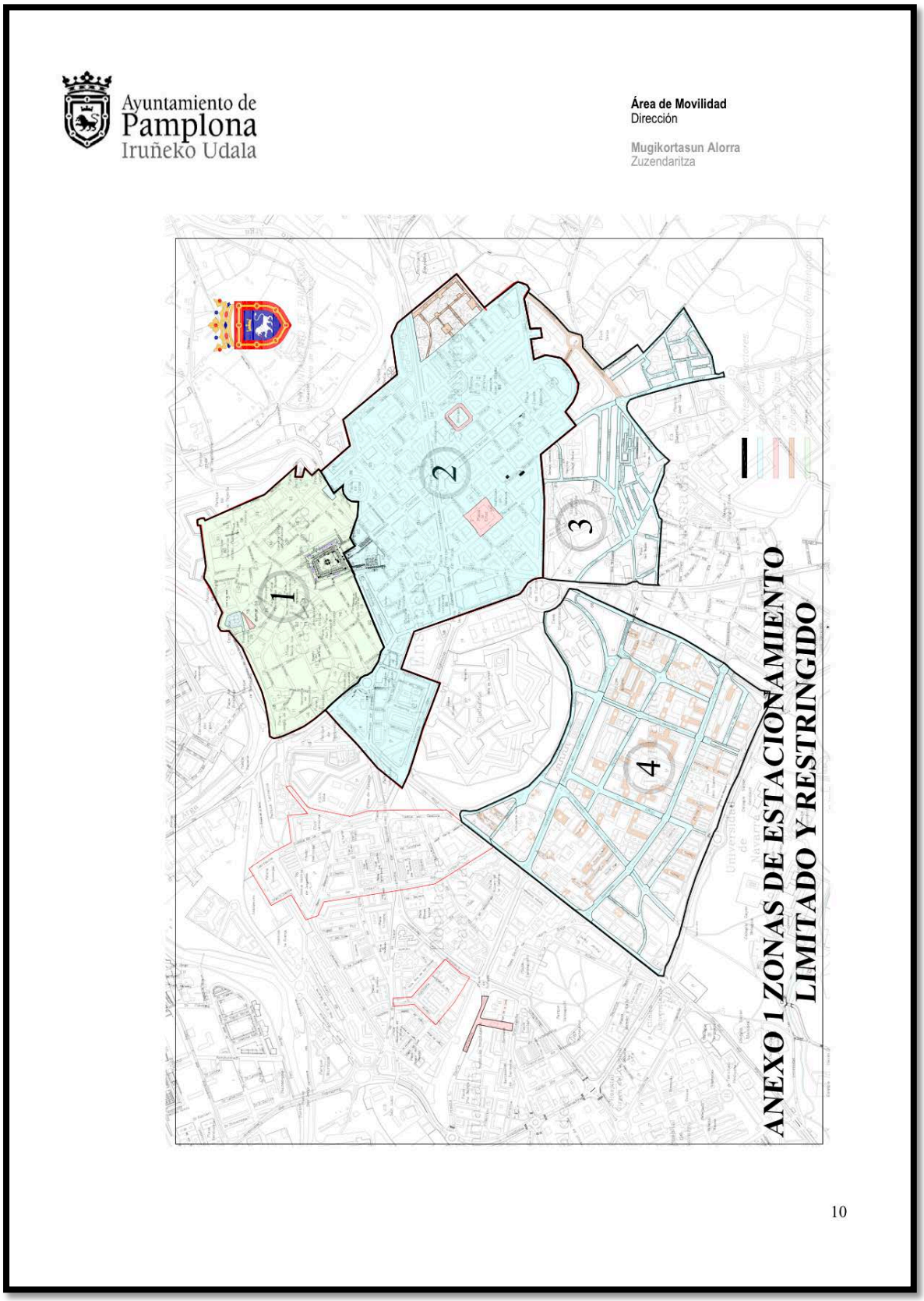


Figura 6-3-1 Anexo de la ordenanza de la que se han sacado los datos de los sectores.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Lo complicado en este punto era obtener las coordenadas que conforman cada uno de esos 4 complejos polígonos. No hay otra manera que introducir los puntos a mano así que hubo que valerse de *Google Maps* y su función de crear mapas personalizados.

De esta manera, se fueron añadiendo todos los puntos con un navegador web en la página web de *Google maps* obteniendo el siguiente resultado[22]:

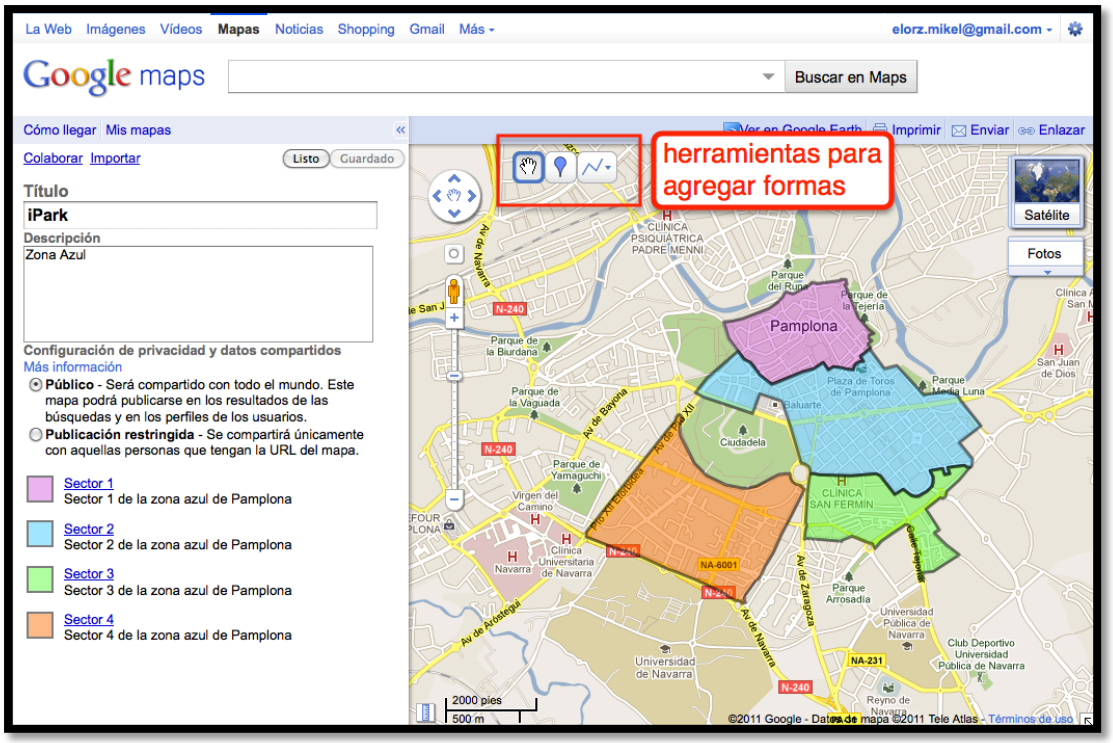


Figura 6-3-2 Editando el mapa con las herramientas de Google maps.

6.3.3. Acceso a los datos.

Google maps ofrece la posibilidad de exportar un mapa en formato KML[23], que no es más que un XML con tags específicos. Para exportar el KML, obtenemos su dirección como se puede apreciar en la Figura 6-3-3.

[22] Consulta del mapa online: http://goo.gl/maps/12Ci
[23] https://code.google.com/intl/es/apis/kml/documentation/

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

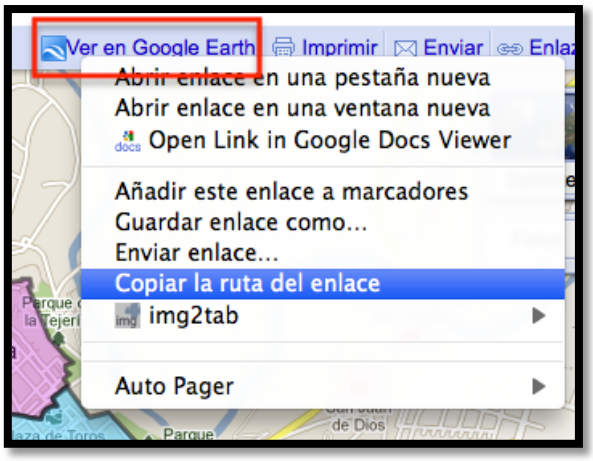


Figura 6-3-3 Copiar el enlace al archivo KML.

Pero el contenido del archivo que obtenemos al seguir el enlace (http://maps.google.com/maps/ms?ie=UTF8&hl=es&vps=2&jsv=346d&authuser=0&msa=0&output=nl&msid=212852501871481854737.0004a3006572d5bf90e2c) no es el esperado:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
    <name>iPark</name>
    <description><![CDATA[Zona Azul]]></description>
    <NetworkLink>
        <name>iPark</name>
        <Link>
<href>http://maps.google.com/maps/ms?ie=UTF8&amp;hl=es&amp;vps=2&amp;jsv=346
d&amp;authuser=0&amp;oe=UTF8&amp;msa=0&amp;msid=212852501871481854737.0004a3
006572d5bf90e2c&amp;output=kml</href>
        </Link>
    </NetworkLink>
</Document>
</kml>
```

El archivo no contiene la información de las coordenada. Sin embargo, si en el enlace anterior cambiamos la opción 'output=nl' por 'output=kml' (http://maps.google.com/maps/ms?ie=UTF8&hl=es&vps=2&jsv=346d&authuser=0&msa=0&output=kml&msid=212852501871481854737.0004a3006572d5bf90e2c) se mostrará el contenido deseado[24]:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
  <name>iPark</name>
  <description><![CDATA[Zona Azul]]></description>
  <Style id="style2">
```

[24] Se han omitido las largas listas de coordenadas.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
    <LineStyle>
      <color>80000000</color>
      <width>3</width>
    </LineStyle>
    <PolyStyle>
      <color>59CC33CC</color>
      <fill>1</fill>
      <outline>1</outline>
    </PolyStyle>
  </Style>
  <Style id="style4">
    <LineStyle>
      <color>80000000</color>
      <width>3</width>
    </LineStyle>
    <PolyStyle>
      <color>73FFCC33</color>
      <fill>1</fill>
      <outline>1</outline>
    </PolyStyle>
  </Style>
  <Style id="style3">
    <LineStyle>
      <color>80000000</color>
      <width>3</width>
    </LineStyle>
    <PolyStyle>
      <color>6633FF33</color>
      <fill>1</fill>
      <outline>1</outline>
    </PolyStyle>
  </Style>
  <Style id="style1">
    <LineStyle>
      <color>80000000</color>
      <width>3</width>
    </LineStyle>
    <PolyStyle>
      <color>730066FF</color>
      <fill>1</fill>
      <outline>1</outline>
    </PolyStyle>
  </Style>
  <Placemark>
    <name>Sector 1</name>
    <Snippet>Sector 1 de la zona azul de Pamplona</Snippet>
    <description><![CDATA[<div dir="ltr">Sector 1 de la zona azul de
Pamplona</div>]]></description>
    <styleUrl>#style2</styleUrl>
    <Polygon>
      <outerBoundaryIs>
        <LinearRing>
          <tessellate>1</tessellate>
          <coordinates>
            -1.641150,42.821331,0.000000
            [...]
            -1.641150,42.821331,0.000000
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
  </Placemark>
  <Placemark>
    <name>Sector 2</name>
    <description><![CDATA[<div dir="ltr">Sector 2 de la zona azul de
Pamplona</div>]]></description>
    <styleUrl>#style4</styleUrl>
```

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
    <Polygon>
      <outerBoundaryIs>
        <LinearRing>
          <tessellate>1</tessellate>
          <coordinates>
            -1.649630,42.817001,0.000000
            [...]
            -1.649630,42.817001,0.000000
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
  </Placemark>
  <Placemark>
    <name>Sector 3</name>
    <description><![CDATA[<div dir="ltr">Sector 3 de la zona azul de
Pamplona</div>]]></description>
    <styleUrl>#style3</styleUrl>
    <Polygon>
      <outerBoundaryIs>
        <LinearRing>
          <tessellate>1</tessellate>
          <coordinates>
            -1.643670,42.809589,0.000000
            [...]
            -1.643670,42.809589,0.000000
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
  </Placemark>
  <Placemark>
    <name>Sector 4</name>
    <description><![CDATA[<div dir="ltr">Sector 4 de la zona azul de
Pamplona</div>]]></description>
    <styleUrl>#style1</styleUrl>
    <Polygon>
      <outerBoundaryIs>
        <LinearRing>
          <tessellate>1</tessellate>
          <coordinates>
            -1.645342,42.809547,0.000000
            [...]
            -1.645342,42.809547,0.000000
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
  </Placemark>
</Document>
</kml>
```

6.3.4. Procesamiento de los datos.

Lo primero ha sido obtener el archivo de internet y guardarlo en un *NSString*. Para ello se ha utilizado la clase *NSUrl* y el método de clase *stringWithContentsOfURL:encoding:error:* . Si obtenerlo de internet falla, tenemos una copia local que se lee de un archivo.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Con el contenido en una cadena de caracteres podemos utilizar la clase *NSScanner* que va recorriendo un *NSString* en busca de diferentes cadenas. Como el *KML* utiliza *tags* estilo *XML*, basta con ir a los *tags* correspondientes para cada cosa que se quiera encontrar. Por ejemplo con *<Placemark>* empieza una nueva forma, *<name>* nos indica su nombre, con *<coordinates>* empiezan la lista de sus coordenadas, etc.

6.3.5. Mostrar los sectores.

El sistema de mostrar formas en un *MKMapView* se basa en la utilización de *MKOverlay.* A estos objetos (que pueden ser caminos, polígonos, ...) se les asocia una *view* que se mostrará por pantalla. Esto se gestiona con los métodos delegados de *MKMapViewDelegate.*

También existen las anotaciones. Así como los *overlay* son formas que aparecen superpuestas en una determinada zona, las anotaciones aparecen en un determinado punto del mapa. Además, con las anotaciones se puede interaccionar si el programador lo desea, con los *overlay* no.

En iPark las formas de los polígonos que conforman un sector se representan con *overlays* mientras que las etiquetas con el título de cada sector son anotaciones. También hay una anotación especial que es la de la localización del usuario.

Al programar estas funcionalidades surgió un problema derivado de la diferencia entre las pantallas normales (iPhone 3GS y anteriores) y la pantalla retina (iPhone 4). Las vistas de los *overlay* de polígonos tienen una propiedad que es el ancho de la línea que las rodea. Se decidió asignar el valor 3 a esta propiedad ya que en un iPhone 4 se visualizaba correctamente. Sin embargo, en los iPhone anteriores la línea es el doble de gruesa y su apariencia no es muy buena.

Para solucionarlo, se ha tenido que añadir una pequeña porción de código que compruebe la escala de la pantalla y con eso determinar si tenemos pantalla retina o no. En el caso de no tener pantalla retina, para que el borde sea igual de fino necesitamos la mitad de anchura.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
if ( [[UIScreen mainScreen] respondsToSelector:@selector(scale)] == YES
  && [[UIScreen mainScreen] scale] == 2.00 )
    // Tenemos pantalla retina
    polView.lineWidth = kAnchoLineaOverlay;
else
    // No Tenemos pantalla retina, el ancho debe ser la mitad
   polView.lineWidth = kAnchoLineaOverlay / 2;
```

6.4. Resultado

Figura 6-4-1 Vista centrada en Pamplona (izquierda) y detalle de la frontera entre 2 sectores (derecha).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 6-4-2 Sectores con vista de satélite (izquierda) y localización del usuario traducido al inglés (derecha)

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 6-4-3 Mapa mostrando el botón seleccionado de localizar (izquierda) y completamente dentro del sector 4 traducido a euskera (derecha).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

7. Codificación: Información

En esta tercera fase de la implementación se ha añadido todo el mecanismo de navegación y se ha redactado toda la información que se incluirá de manera estática con la aplicación

7.1. Objetivos

El objetivo de esta pestaña es ofrecer información útil acerca de distintos aspectos de la Zona Azul. A menudo los usuarios desconocen, por poner algunos ejemplos reales, cómo se anulan las multas, las diferencias entre Zona Azul y naranja, que con un mismo tique se puede cambiar de un sector a otro, etc.

7.2. Mecanismos utilizados

7.2.1. Clases y controladores utilizados.

Una de las características que tiene la información a ser mostrada es que se reparte en múltiples niveles. Una fila puede dar paso una nueva tabla de información con contenido adicional. También puede ser una fila que lleve a mostrar una imagen y un texto en más detalle. O simplemente no ser navegable y sólo mostrar el título y/o subtítulo de la fila.

Teniendo esto en cuenta se diseñó en el diagrama de clases una estructura jerárquica adecuada a ello:

- Una *Tabla* tiene secciones.
- Una *Sección* tiene filas.
- Una *Fila* puede ser Detalle u otra Tabla
- (Un Detalle sin texto es un Detalle no navegable)

Además, por motivos de implementación, se ha definido otro elemento superior denominado *ElementoInfo*. Es una abstracción de cualquiera de los elementos antes mencionados, da acceso a las fuentes de donde se lee información y da un índice único que identifica a cada elemento. Este índice nos permite una obtención rápida de un *NSDictionary*, una tabla *SQL*, ...

Como puede verse, hay dos formas distintas en las que se tiene que visualizar la información: en forma de tabla y en forma de detalle.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

La forma de tabla se basa en un *UITableView* para mostrar los contenidos en diferentes filas organizadas por secciones. Esto se ha implementado en la clase *InfotableViewController* y en su correspondiente xib.

La forma de detalle se basa en un *UISCrollView*, para mostrar la imagen y el texto de un *Detalle* de manera que se pueda visualizar con comodidad e incluso hacer zoom a partes específicas.

7.2.2. Almacenamiento de la información

Una de los primeras cuestiones a las que se ha hecho frente ha sido: ¿Dónde y cómo se guarda la información para que la aplicación acceda a ella?

Las opciones valoradas han sido 3: *CoreData, SQLite y Property Lists*.

CoreData es el API para el almacenamiento persistente de datos de *Cocoa.* Su uso es más rentable con grandes volúmenes de información y sobre todo si la información es dinámica. La información que requiere ser almacenada va a ser de sólo lectura por lo que se necesita una estructura compleja que facilite la persistencia de nuevos objetos.

SQLite parece más adecuado para las necesidades de este proyecto. Este motor de base de datos es muy ligero y provee de mecanismos relativamente sencillos para la creación de tablas y la recuperación de la información. Si le sumamos que el proyectista ya tiene conocimientos de lenguajes de consultas relacionales, parece la solución adecuada. Pero se presenta el inconveniente de cómo introducir los datos por primera vez. Además, la cantidad de datos con la que se va a trabajar justificaría a duras penas el uso de *SQLite*.

Por lo tanto se ha optado por la solución de los *Property Lists*. Los también conocidos como *plists* son, en esencia, un archivo XML con una estructura concreta y definida. Su función es la de serializar una serie de objetos básicos del framework *Foundation*. Pueden almacenar:

- *NSString*
- *NSNumber* (real o integer)
- *NSNumber* (booleano)
- *NSDate*
- *NSData*

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- *NSArray*
- *NSDictionary*

Para almacenar la información no necesitamos más que dos archivos de este tipo. El primero que contenga un esquema formado solamente por índices para poder establecer la jerarquía de tablas y secciones (Figura 7-2-1). Cada entrada del diccionario tiene como clave su índice y como contenido un *NSArray* con sus índices subordinados (ya sea *Tabla* o *Sección*).

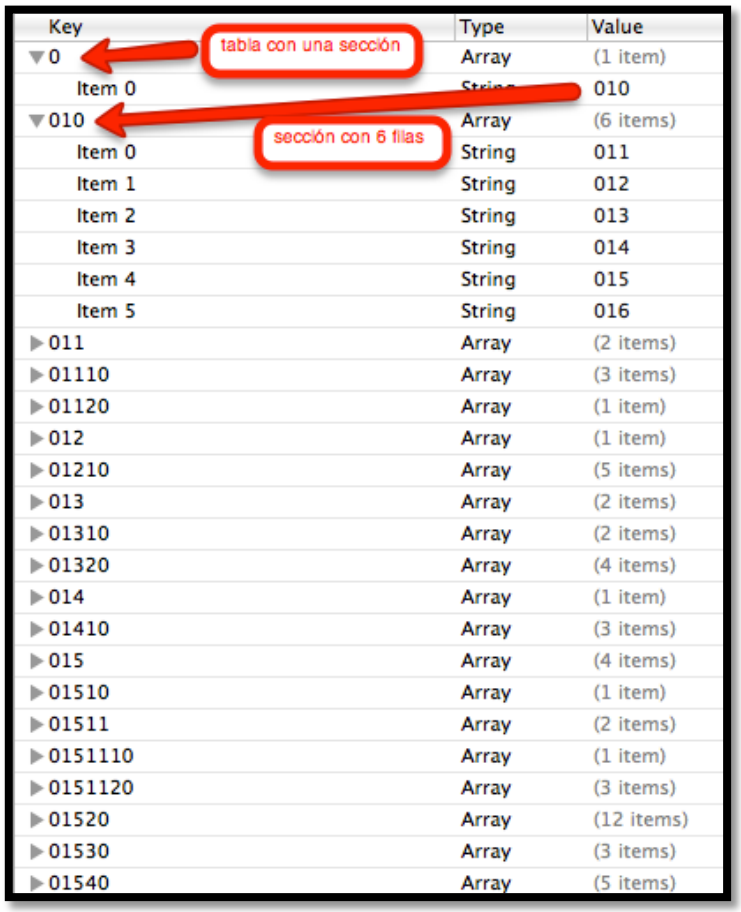


Figura 7-2-1 *Esquema.plist* con una *tabla* y una *sección* desplegadas.

El segundo contiene todos los datos accesibles por el índice de cada elemento de información. Están todos los datos al mismo nivel y dependiendo del tipo de objeto tendrá unos atributos u otros (Figura 7-2-2 y 7-2-3). Por ejemplo, una *Sección* tendrá sólo *tituloSeccion*, pero un Detalle tendrá *titulo, subtitulo, imagen* y *texto*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

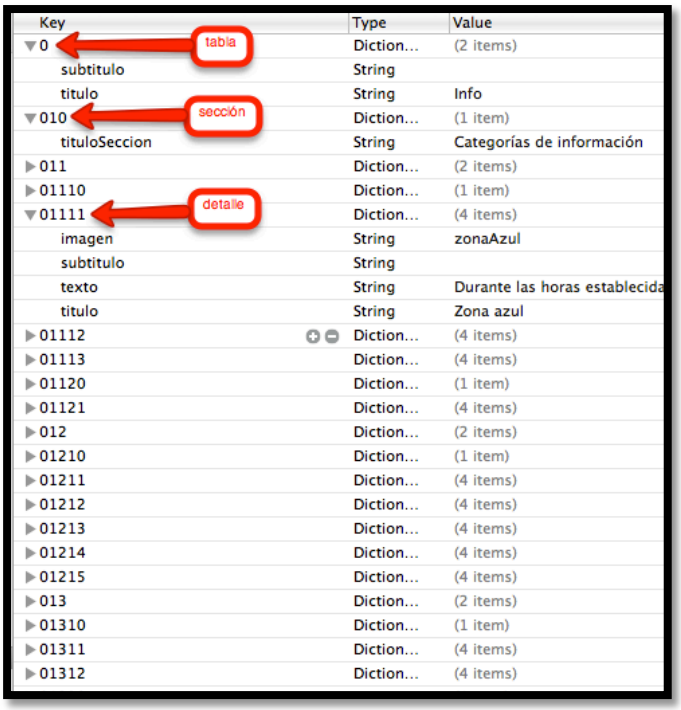


Figura 7-2-2 *Infodatos.plist* con una *tabla,* una *sección* y un *detalle* desplegados.

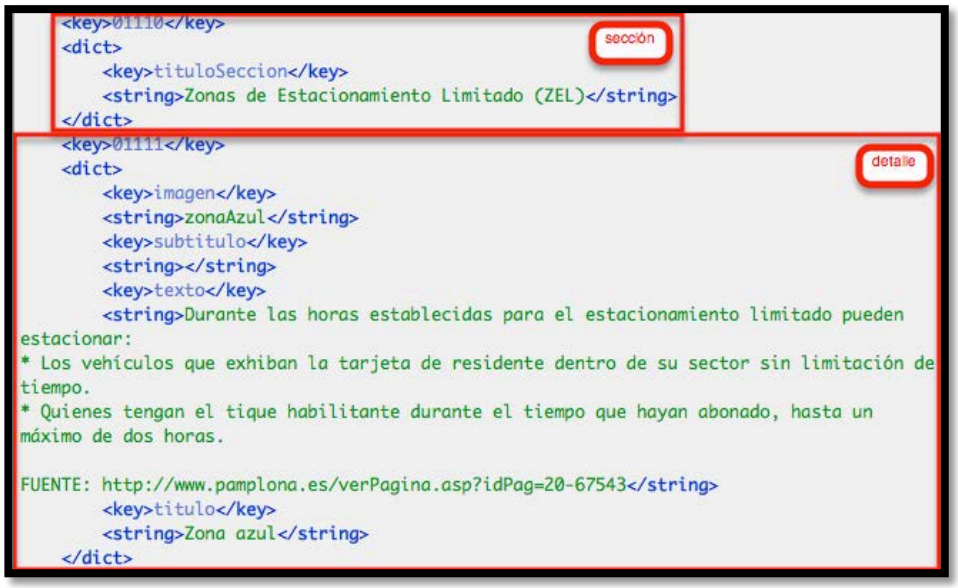


Figura 7-2-3 *Infodatos.plist* visto con un editor de texto. Se puede apreciar las etiquetas XML que forman una *sección* y un *detalle*.

Además, en vez de usar el método de traducción usual (la clase *TraductorHash*), se ha optado por crear tres versiones del *plist* de datos, uno para cada idioma. Esta decisión se ha tomado por cuestiones de eficiencia.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

7.2.3. InfoTableViewController

En esta clase se controla una *UITableView* que se instancia en el xib. El funcionamiento es el típico con dos particularidades.

La primera es que la tabla necesita actualizarse si el idioma ha cambiado. Por lo tanto se ha implementado el refresco de los datos en la función *viewWillAppear*. También se ha hecho que el diccionario de datos tenga versión es, en y eu traducidas y que la clase *ElementoInfo* detecte automáticamente si el idioma ha cambiado frente al que tenía guardado y de ser así cargar el diccionario correspondiente. El resultado es que sin importar donde estemos, si cambiamos el idioma, al volver a la pestaña de *info* se llamará a *viewWillAppear* y se mostrará automáticamente el texto traducido. Esto evita que el usuario deba reiniciar la aplicación para que se aplique la traducción.

La segunda es que se han creado unas imágenes de fondo tanto para las celdas como para el fondo para darle un aspecto distintivo a la tabla que se diferencie de la típica apariencia por defecto. Las imágenes son una serie de gradientes y se puede apreciar el resultado en la Figura 7-2-4.

Además, en vez de utilizar una *UITableView* normal, se ha utilizado una subclase de *UITableView* (*SombrasTableView*) que añade una sombra en la parte de arriba de la tabla y una sombra después de la última fila. Esto se consigue mediante *CAGradientLayers* que crean un degradado de dos colores que se especifican en un array. Para utilizar estas funciones es necesario importar el framework QuartzCore (en Xcode 4, añadiéndolo en la *Build Phase* que se encarga de linkar).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 7-2-4 Resultado *InfoTableViewController* en dos idiomas. Puede apreciarse la sombra en la imagen de la derecha.

7.2.4. InfoDetailViewController

Clase que permite ver la imagen y el texto de un objeto tipo detalle. Como la imagen puede ser grande (en el caso de los mapas de los sectores), se necesita un mecanismo para hacer zoom. Como el texto puede ser largo y no caber en la pantalla, se necesita un mecanismo de *scroll*. Todo esto se puede hacer con un UIScrollView.

En el xib se ha añadido una vista que contiene el *UIScrollView* y el *UIImageView* con la imagen de fondo.

Dentro del *UIScrollView* se ha definido también una vista que aglutine todo el contenido y se le cambiará el tamaño según cambien sus subvistas: otro *UIImageView* y un *UITextView*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Es importante que el *contentSize* del *UIScrollView* se defina correctamente para que el scroll funcione correctamente. También es importante implementar el método de delegado de *UIScrollView* para que se active el zoom.

```
- (UIView *)viewForZoomingInScrollView:(UIScrollView *)scrollView {
    return self.viewContenido;
}
```

Otros de los mecanismos comunes en las vistas que admiten zoom son el doble tap y el tap de dos dedos para acercar y alejar el zoom respectivamente. Para ello se necesitan unos *UITapGestureRecognizer* que detectan el gesto requerido y que hagan programáticamente el zoom a la zona señalada. El código para estos gestos se ha conseguido del proyecto de ejemplo de *Apple TapToZoom* (parte de *ScrollViewSuite*)[25] pero es sorprendentemente sencillo implementar dichos gestos.

Uno de los mayores problemas ha surgido con la traducción. Si se está viendo un detalle y se cambia el idioma, el texto cambia. Por lo cual, hay que recalcular de nuevo el tamaño de la vista del texto, de la vista de contendido... Además, no tiene sentido mantener el zoom en la posición actual por lo que si se detecta que el idioma ha cambiado, se devuelve el zoom a la posición inicial. Esto también sirve para que se note que el texto ha cambiado.

[25] Link acortado: http://bit.ly/mpcdaS

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 7-2-5 Resultado *InfoDetailViewController.* A la izquierda una de las vistas de detalle, a la derecha la misma vista después de haber hecho zoom en la imagen.

7.2.5. Gestos multitáctiles.

Se ha añadido como funcionalidad extra, la posibilidad de navegar hacia atrás en la jerarquía de vistas con el simple gesto de deslizar tres dedos de izquierda a derecha. Al realizar este gesto de barrido, se navega hacia atrás, al nivel anterior tanto si estamos en una vista de detalle como si estamos en una tabla de elementos de información.

Se ha decidido que sea de izquierda a derecha porque así se asemeja a la animación que ocurre cuando volvemos hacia atrás, que la vista del nivel superior aparece por la izquierda desplazando la vista actual hacia la derecha.

Basta con añadir este código que hace que se le mande la acción *popView* a la instancia del controlador de navegación. Este objeto de tipo

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

UISwipeGestureRecognizer debe tener la dirección hacia la derecha. El número de dedos necesarios para realizar el gesto se ha elegido que sea 3 para que no sea fácil de ejecutarlo por accidente. Por último se añade el reconocedor de gestos a la vista que se desee que lo detecte.

```
// Con tres dedos hacia la derecha, volvemos hacia atrás
UISwipeGestureRecognizer *swipeDerecha = [[UISwipeGestureRecognizer alloc]
                      initWithTarget: self.navigationController
                              action: @selector(popViewControllerAnimated:)];

swipeDerecha.direction = UISwipeGestureRecognizerDirectionRight;
swipeDerecha.numberOfTouchesRequired = 3;
[self.view addGestureRecognizer: swipeDerecha];
[swipeDerecha release];
```

7.3. Información

A continuación se muestra toda la información recopilada que se ha introducido. Se muestra con la misma jerarquía con la que aparece en la aplicación.

1. Sección: Categorías de información

1.1. Zonas [Azul, naranja, roja, ...]

1.1.1. Sección: Zonas de Estacionamiento Limitado (ZEL)

1.1.1.1. Zona azul

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Durante las horas establecidas para el estacionamiento limitado pueden estacionar:

* Los vehículos que exhiban la tarjeta de residente dentro de su sector sin limitación de tiempo.

* Quienes tengan el tique habilitante durante el tiempo que hayan abonado, hasta un máximo de dos horas.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-67543

1.1.1.2. Zona naranja

Durante las horas establecidas para el estacionamiento limitado pueden estacionar:

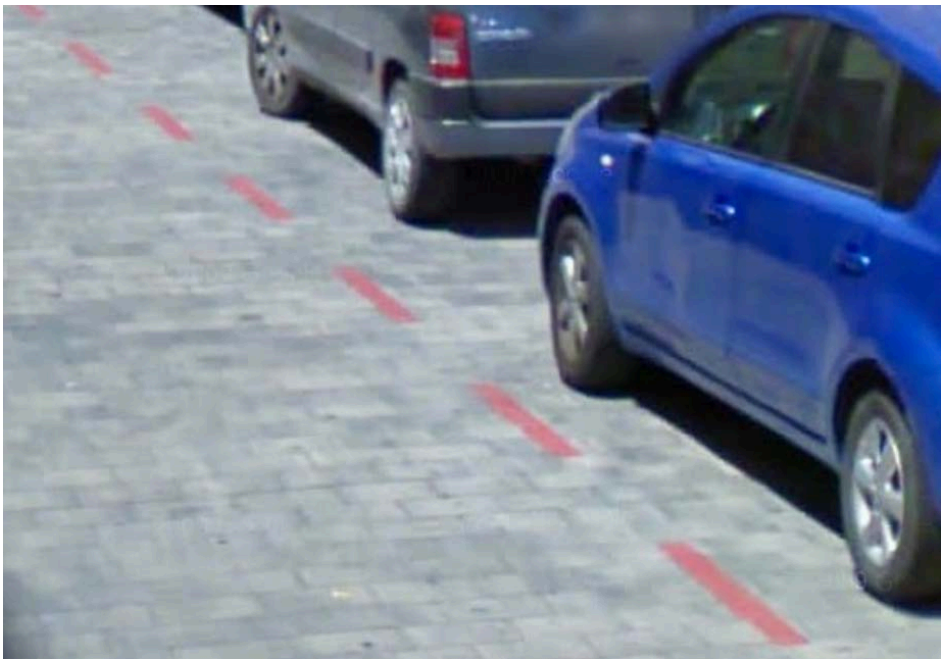
* Los vehículos que exhiban la tarjeta de residente dentro de su sector sin limitación de tiempo.

* Quienes tengan el tique habilitante durante el tiempo que hayan abonado, hasta un máximo de 9 horas y media (lo cual permite estacionar durante un día entero).

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-67543

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.1.1.3. Zona roja [Rotación]



** Zona exclusiva para la rotación **

Durante las horas establecidas para el estacionamiento limitado pueden estacionar:

* Solamente quienes tengan el tique habilitante durante el tiempo que hayan abonado, hasta un máximo de dos horas.

NO PODRÁN estacionar vehículos con tarjeta de residente sin haber sacado el tique correspondiente. Es decir, a los residentes se les aplicará las mismas normas que al resto de los usuarios.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-67543

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.1.2. Sección: Zonas de Estacionamiento Restringido (ZER)

1.1.2.1. Zona verde / mixta [Residentes]

** Exclusivamente para vecinos residentes **

Dentro de las zonas de estacionamiento restringido a residentes también podrán establecerse zonas azules y rojas aplicándose sus respectivas normas. Fuera de los horarios de regulación, en estas zonas únicamente podrán estacionar los residentes.

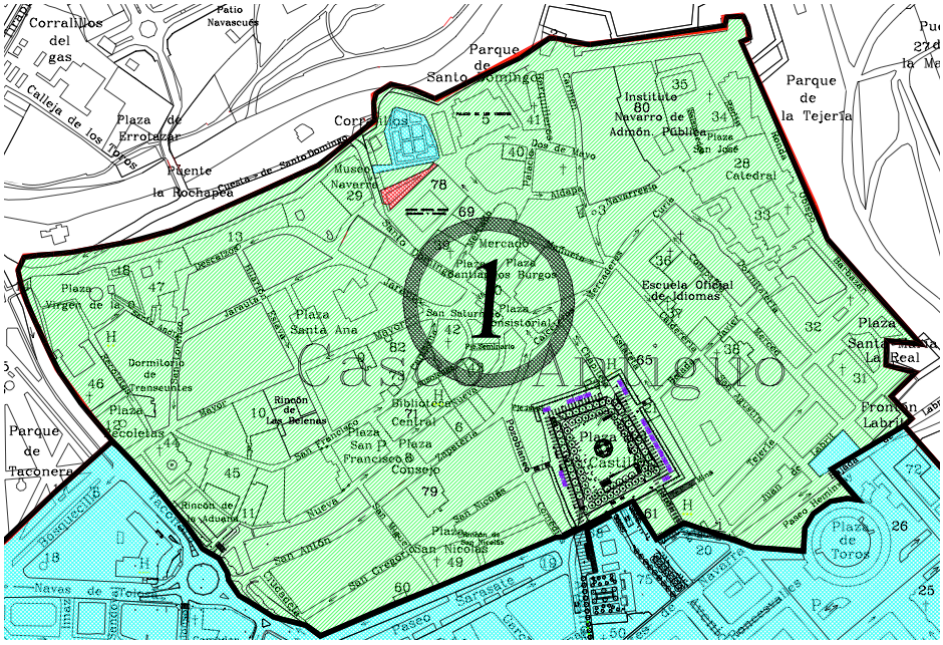
FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-67543

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.2. Sectores [Relación de calles]

1.2.1. Sección: (sin título)

1.2.1.1. Sector 1 [Casco antiguo]



****

En las zonas azules y naranjas del sector 2 pueden aparcar también los del sector 1.

****

ZONA VERDE:

Aldapa, Ansoleaga, Bajada Javier, Barquilleros, Calderería, Carmen, Ciudadela, Compañía, Cuesta del Palacio, Curia, Descalzos, Dormitalería, Dos de Mayo, Eslava, Jarauta, Juan de Labrit (del 1 al 25 y del 31 al 33), Mañueta, Mayor, Mercado, Merced, Navarrería, Nueva, Plaza de Recoletas, Parque Taconera, Paseo Heminway, Plaza San Francisco, Plaza del Castillo, Plaza de Santa M. La Real, Plaza Virgen de la O, Redín, San Agustín, Aparcamiento San Agustín, Santo Domingo, Tejería y Solar Vista Bella..

ZONA ROJA:

Aparcamiento del Departamento de Educación.

Las Huertas de Santo Domingo en horario de lunes a sábado de 08:30 h. a 14:00 h.

ZONA AZUL:

Aparcamiento Juan de Labrit (del 25 al 31).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

ZONA NARANJA:

Parque Larraina.

APARCAMIENTO EXCLUSIVO PARA RESIDENTES DEL SECTOR 1 LAS 24 HORAS:

Zona norte del Parque de la Runa, paralela a la calle Río Arga y solar de Vista Bella.

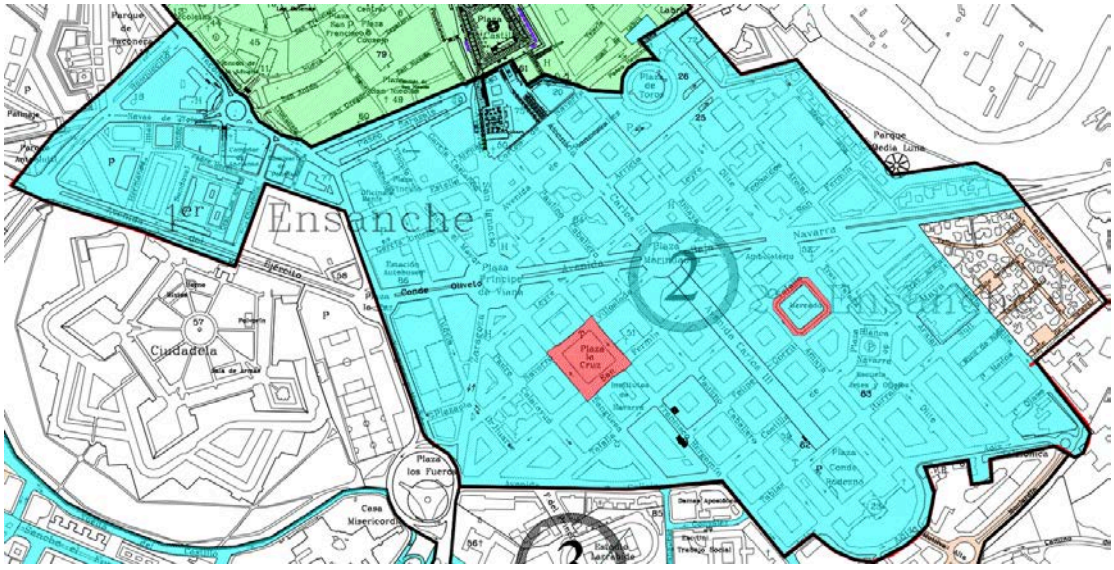
NOTAS:

Las Huertas de la calle Santo Domingo en horario de lunes a viernes de 16:00 h. a 20:00 h. es ZONA VERDE.

El aparcamiento del Departamento de Educación, después de la hora de estacionamiento restringido es ZONA VERDE.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-68071

1.2.1.2. Sector 2 [Ensanches]



ZONA AZUL:

Amaya, Aoiz, Aralar, Arrieta, Avenida de Zaragoza (del 2 al 21), Baja Navarra, Bergamín (1 al 47 y 2 al 28), Bosquecillo, Castillo de Maya, Cipriano Olaso, Conde Oliveto, Conde Rodezno, Cortes de Navarra, Doctor Huarte, Estella, Galicia (impares), García Castañón, García Ximénez, General Los Arcos, González Tablas, Gorriti (excepto la acera del Mercado Nuevo), Hermanos Imaz, Iturralde y Suit, José Alonso, Leire, Marqués de Rozalejo, Media Luna, Navarro Villoslada, Navas de Tolosa, Olite (excepto la acera del Mercado Nuevo), Padre Calatayud, Padre Moret, Pascual Madoz, Paseo Sarasate, Paulino Caballero, Pío XII (número 1),

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Plaza Blanca de Navarra, Plaza de la Paz, Plaza de los Fueros (número 1), Plaza de Merindades, Plaza Príncipe de Viana, Plazaola, Roncesvalles, Ruiz de Alda, San Fermín, San Ignacio, Sancho el Mayor, Sandoval, Sangüesa (del 1 al 19 y del 2 al 30), Tafalla (excepto la acera del Mercado Nuevo), Teobaldos, Trasera Colegio de Médicos, Travesía de Aralar, Travesía de Tafalla, Trinidad Fernández Arenas, Tudela y Yanguas y Miranda (del 17 al 29).

ZONA ROJA:

Aceras circundantes al Mercado Nuevo.

Plaza de la Cruz.

ZONA NARANJA:

Valle de Araquil.

Valle de Baztán.

Valle de Egüés.

Valle de Roncal.

Valle de Salazar.

Valle de Yerri.

Desde rotonda junto a Colegio Ursulinas (cruce c/ Monte Monjardín con c/ Media Luna) hasta rotonda de ctra. Badostáin.

c/ Media Luna (entre calles Gorriti y Monjardín).

Entorno del Club Larraina y Piscinas Militares.

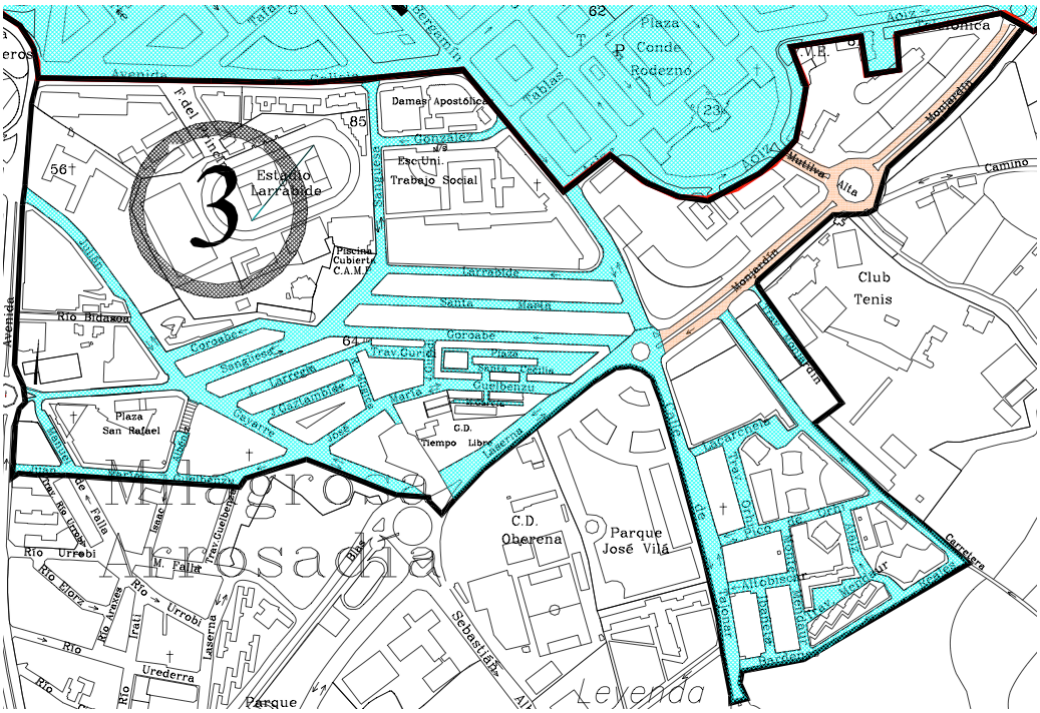
Partes adyacentes a instalaciones deportivas de la Sociedad Larraina y al Centro Deportivo Sociocultural Militar La Ciudadela.

NOTA: en las zonas azules y naranjas del sector 2 pueden aparcar también los del sector 1.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-68072

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.2.1.3. Sector 3 [Milagrosa]



ZONA AZUL: Astobizkar, Bardenas Reales, Bergamín (del 49 al 53 y del 30 al 32, Jesuitas), Blas Laserna (de 2 al 6), Felisa Munárriz, Goroabe, Ibañeta, Isaac Albéniz (del 1 al 5 y del 2 al 10), J Gaztambide, J Guridi, Joaquín Larregla, José María Guelbenzu, Julián Gayarre, Lacarchela, Larrabide, Manuel de Falla (del 1 al 5 y del 2 al 4), Mariano García, Monte Alaiz, Monte Mendaur, Mutilva Alta, Mutilva Baja, Remigio Múgica, San Rafael, Sangüesa (del 32 al 44 y del 21 al 51), Santa Cecilia, Santa Marta, Tajonar, Travesía Guridi, Travesía Mendaur, Travesía Monjardín y Zaragoza (del 25 al 47).

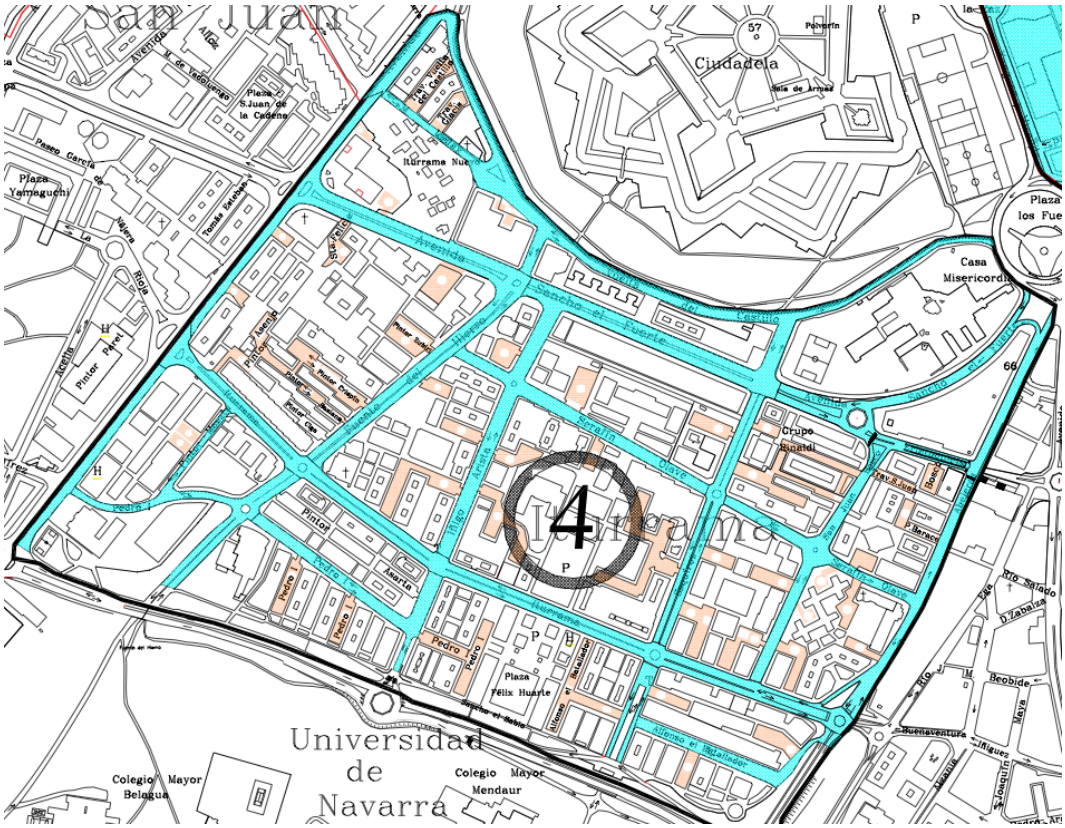
ZONA NARANJA: Monjardín, Pico de Orhi, Travesía Orhi y Avda. Juan Pablo II.

NOTA: en la zona naranja de Avda. Juan Pablo II se permite también el estacionamiento sin tiquet a los residentes del sector 1 y 2 mientras duren las obras del aparcamiento subterráneo de la C/ Olite.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-68421

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.2.1.4. Sector 4 [Iturrama]



Abejeras, Alfonso el Batallador, Erletoquieta, Esquíroz, Felix Huarte, FUENTE del Hierro, Grupo Rinaldi, Iñigo Arista, Iturrama, Monasterio Urdax (del 2 al 8), Padre Barace, Pedro I, Pintor Asarta, Pintor Asenjo, Pintor Basiano, Pintor Ciga, Pintor Crispín, Pintor Maeztu, Pintor Zubiri, Pío XII (del 3 al 45), San Juan Bosco, Sancho el Fuerte (del 1 al 71 y del 2 al 24), Santa Felicia, Serafín Olave, Travesía Glacis, Travesía San Juan Bosco, Travesía Vuelta del Castillo y Vuelta del Castillo.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-68422

1.2.1.5. Otros [San Juan y Ermitagaña]

Estas calles no pertenecen a ningún sector.

SAN JUAN, Zona Roja:

Martín Azpilcueta.

Parte de la Travesía Martín Azpilcueta.

Ermitagaña, Zona Naranja:

upna Universidad Pública de Nafarroa Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Trasera de los Golem Baiona.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=43563VA

y http://www.pamplona.es/verPagina.asp?idPag=20-68955

1.3. Horario [y días exentos]

1.3.1. Sección: Horario

1.3.1.1. Lunes a Viernes [8:30 - 14:00 y 16:00 - 20:00]

1.3.1.2. Sábado [8:30 - 14:00]

1.3.2. Sección: Días exentos.

1.3.2.1. Domingos

1.3.2.2. Festivos

1.3.2.3. Sábado santo

1.3.2.4. Del 6 al 25 de Julio [Ambos inclusive]

FUENTE: Ordenanza de estacionamiento limitado y restringido (ZEL) http://bit.ly/fQ76nm y http://www.pamplona.es/verPagina.asp?idPag=20-40615

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.4. Tarifas

1.4.1. Sección: Por zonas

1.4.1.1. Azul y roja

El tique mínimo es para 10 minutos y cuesta 0.25 euros. A partir de ahí la máquina expendedora irá sumando fracciones de 3 minutos por cada 5 minutos introducidos. Tiene un máximo de 2 horas.

Mínimo 10 min: 0.25 €

+3 min: +0.05 €

Máximo 2 horas: 2.10 €

FUENTE: Tarifas año 2011

http://www.pamplona.es/verPagina.asp?idPag=20-51881

1.4.1.2. Naranja

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Durante las dos primeras horas de estacionamiento se mantiene la tarifa normal de la Zona Azul. A partir de ahí, por cada 0.05 € se obtienen 10 minutos en vez de 3. Puede pagarse un periodo máximo de 9 horas y media (tiempo necesario para estacionar un día entero).

(Primeras 2 horas)

Mínimo 10 min: 0.25 €

+3 min: +0.05 €

2 horas: 2.10 €.

(A partir de 2 horas)

+10 min: +0.05 €

9 horas y media: 4.35 €

FUENTE: Tarifas año 2011

http://www.pamplona.es/verPagina.asp?idPag=20-51881

1.4.1.3. Verde / mixta.

Si se han establecido zonas azules y rojas, se aplican las tarifas de éstas.

Fuera del horario de estas zonas o si no se han establecido dichas zonas, únicamente podrán estacionar los residentes.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-67543

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

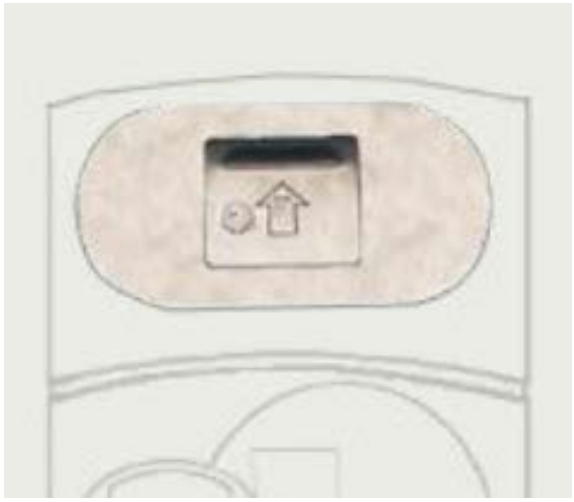
1.5. Multas y sanciones [Anulación de denuncias, importes, ...]

1.5.1. Sección: (Sin título)

1.5.1.1. Anulación de la denuncia

1.5.1.1.1. Sección: (Sin título)

1.5.1.1.1.1. Información general [Cómo devolver las denuncias]



Si se ha superado el tiempo de aparcamiento que marca el tique, se estaciona sin tique o sin tarjeta existe la posibilidad de anular la denuncia.

El vigilante habrá firmado y depositado la denuncia dentro de un sobre en el parabrisas de su vehículo.

El método para evitar esta denuncia consiste en ir a la máquina expendedora y apretar el botón amarillo, luego introducir el importe indicado en la denuncia. Si el pago se hace con tarjeta (no de crédito) hay que apretar el botón azul hasta llegar al importe indicado.

Posteriormente, la máquina imprimirá un tique nuevo como justificante de que el conductor ha pagado el importe correspondiente.

El conductor deberá introducir el tique en el sobre de la denuncia y éste, a su vez, en la ranura superior (foto) para que la multa no sea tramitada.

Si el vehículo es retirado con grúa no se puede anular la denuncia.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-40865

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.5.1.1.2. Sección: Precio anulación

1.5.1.1.2.1. Exceso de tiempo [4 €]

1.5.1.1.2.2. Carecer de tique [11.65 €]

1.5.1.1.2.3. Rebasar el doble del tiempo [11.65 €]

1.5.2. Sección: 200 € por estacionar:

1.5.2.1. En zona de carga y descarga. [Dentro del horario habilitado]

1.5.2.2. En doble fila, sin conductor. [Obstaculizando a otros vehículos]-

1.5.2.3. Impidiendo realizar un giro. [A otros vehículos]

1.5.2.4. En paradas de bus, taxi. [En sus horarios de funcionamiento]

1.5.2.5. En medianas, separadores. [U otros elementos de canalización del tráfico]

1.5.2.6. En intersecciones. [O zonas inmediatas]

1.5.2.7. Plazas de minusválidos.

1.5.2.8. Sobre la acera. [Obstaculizando el tráfico de peatones]

1.5.2.9. Sobre un paso de peatones. [Totalmente sobre él]

1.5.2.10. En un vado.

1.5.2.11. Tiques o tarjetas falsas. [O modificadas]

1.5.3. Sección: 60 € por estacionar:

1.5.3.1. Sobre un paso de peatones. [Parcialmente sobre él]

1.5.3.2. Sin tique o tarjeta residente.

1.5.3.3. Rebasando el tiempo. [En más del doble de lo abonado]

1.5.4. Sección: 30 € por estacionar:

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.5.4.1. Sin el tique bien visible.

1.5.4.2. Rebasando el tiempo. [En menos el doble de lo abonado]

1.5.4.3. En un radio de 200 m. [Una vez rebasado el tiempo máximo]

1.5.4.4. Fuera de las líneas. [Ocupando parte de otra plaza]

1.5.4.5. Bicis o motos. [Fuera de los lugares habilitados para ello]

FUENTE de estas tres secciones:
http://www.pamplona.es/verPagina.asp?idPag=20-40618

1.6. Otros [Tarjetas, pagos, cambio de plaza, ...]

1.6.1. Sección: Tarjetas y colectivos

1.6.1.1. Residentes.

BENEFICIARIOS:

Tendrán derecho todas las personas físicas empadronadas y que tengan su residencia habitual en un domicilio situado dentro de la zona regulada.

REQUISITOS:

Ser titular de un vehículo (salvo en el caso de que el vehículo pertenezca a la empresa).

Estar empadronado en un domicilio situado dentro de la zona regulada, que constituya su residencia habitual, y en el que también tenga domiciliado el coche.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Quedan excluidos los beneficiarios de una plaza de aparcamiento para residentes de concesión municipal. Únicamente se dará una tarjeta por persona y vehículo.

PRECIO:

Año 2011: 48,35 euros, pudiendo dividirse por meses.

Más info, solicitudes, etc. http://www.pamplona.es/verPagina.asp?idPag=20-44547

1.6.1.2. Más de 2 horas [Para colectivos profesionales

Tarjeta para varios colectivos profesionales para pagar estacionamientos por espacio superior a 120 minutos en las zonas reguladas azules y naranjas. Consiste en una tarjeta monedero especial y un adhesivo identificativo para colocar en el coche acreditado.

BENEFICIARIOS:

Vehículos vinculados a actividades empresariales y profesionales que para realizar ese trabajo precisan necesariamente de ese transporte.

REQUISITOS:

Ser persona física o jurídica domiciliada en Navarra.

Estar dado de alta en el Impuesto de Actividades Económicas para la actividad de fontanería, electricidad u otras puedan necesitar estacionar más de ciento veinte minutos.

Estar al corriente de las obligaciones tributarias y de Seguridad Social.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

No tener deudas con el Ayuntamiento de Pamplona en período ejecutivo.

El vehículo tiene que estar a nombre de quien solicita el permiso y al corriente de sus obligaciones de seguro e inspección técnica.

PRECIO:

Obtener la tarjeta es gratuito

Más info, solicitudes, etc.
http://www.pamplona.es/verPagina.asp?idPag=123576TA

1.6.1.3. Actividades comerciales, [industriales, profesionales o de servicios]

Tarjeta de aparcamiento por actividad comercial, industrial, profesional o de servicios para las zonas NARANJAS de su sector de estacionamiento.

BENEFICIARIOS:

Tendrán derecho todas las personas físicas o jurídicas con domicilio fiscal y que tengan su lugar de trabajo habitual en una domicilio situado dentro de la zona regulada.

REQUISITOS:

El vehículo tiene que estar a nombre del titular de la actividad.

Estar debidamente asegurado y al corriente de la inspección técnica.

Estar dado de alta en el Impuesto de Actividades Económicas y en el lugar donde está ubicado el establecimiento.

Tener la correspondiente licencia de apertura del local donde se ejerce la actividad.

Quedan excluidos los beneficiarios de una plaza de aparcamiento para residentes de concesión municipal.

Únicamente se dará una tarjeta por comercio.

PRECIO:

Año 2011: 466,20 euros pudiendo dividirse por meses.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Más info, solicitudes, etc.
http://www.pamplona.es/verPagina.asp?idPag=60085TA

1.6.1.4. Exenciones.

Están eximidos de utilizar los parquímetros:

Los vehículos de residentes con tarjeta en vigor, dentro de su sector.

Las motocicletas, ciclomotores excepto los de 3 y 4 ruedas y bicicletas, que deberán estacionar en lugares señalados.

Vehículos comerciales en zonas y horario de carga y descarga.

Autotaxis, ambulancias, bomberos (que se encuentren en servicio).

Vehículos con tarjeta de minusválido, tanto dentro como fuera de las plazas específicas de minusválidos.

Vehículos autorizados en reservados para edificios oficiales, hoteles, etc.

Vehículos eléctricos con el distintivo correspondiente.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-40617

1.6.2. Sección: Pagos

1.6.2.1. Formas de pago.

Efectivo (no acepta monedas de 1 y 2 céntimos).

Tarjetas monedero de la red EURO6000: Deberán recargarse en cualquier cajero de Caja Navarra (CAN) aunque sea con cargo a cuentas de otras entidades.

Tarjeta monedero de la propia empresa adjudicataria: Será facilitada gratuitamente en las Oficinas de la empresa Dornier. Esta tarjeta deberemos cargarla la primera vez con un mínimo de 3 euros y después podrá recargarse cuantas veces se quiera en los parquímetros y hasta un máximo de 30 euros. Hay que tener en cuenta que al tratarse de tarjetas no nominales, su pérdida implica que puede ser utilizada por cualquier persona, por el importe que reste.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-40614

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.6.2.2. Funcionamiento parquímetro.

MÉTODO PARA PAGAR CON DINERO:

Introducir las monedas por la ranura (admite monedas de 5 céntimos en adelante) desde un mínimo de 0.25 euros. La máquina no devuelve cambios ni acepta tarjetas de crédito.

Una vez pagada la cantidad equivalente al tiempo que se quiera estar aparcado, pulsar el botón verde. Para cancelar, el botón rojo.

Por otra ranura sale el tique que, tiene dos partes, una se debe quedar en el coche, A LA VISTA, y la otra sirve para que el usuario se acuerde de la hora hasta la que ha pagado el estacionamiento.

MÉTODO PARA PAGAR CON TARJETA DE LA EMPRESA:

Introducir la tarjeta por la ranura correspondiente.

Presionar el botón azul para decidir el tiempo que va a durar el estacionamiento. El mínimo es de 10 minutos (0.25 euros) y cada vez que pulse el botón nuevamente, avanzará en fracciones de 0,05 euros. Una vez decidido el tiempo que estará aparcado, presionar el botón verde. Para cancelar cualquier operación, el botón rojo.

Por la otra ranura saldrá el tique justificante del pago por el estacionamiento. Deberá dejar una de las partes en un sitio visible del vehículo.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

SOLICITUD, PRIMERA CARGA Y RECARGAS DE LA TARJETA:

La solicitud y primera carga de la tarjeta se debe realizar en las propias oficinas de la empresa Dornier. Las recargas sucesivas, se realizan en la propia máquina expendedora.

La carga máxima es de 30 euros.

Para consultar el saldo de la tarjeta hay que presionar el botón del interrogante.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-40864

1.6.3. Sección: Aparcamiento

1.6.3.1. Condiciones.

Zonas azules y rojas: tiempo máximo 120 minutos.

Si seguidamente vuelve a aparcar, deberá hacerlo a no menos de 200 metros del lugar que ocupaba.

No se pueden sacar varios tiques sucesivos para una misma plaza por más de 120 minutos.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-40616

1.6.3.2. Cambio de plaza.

Si no se ha agotado el tiempo señalado en el tique y se vuelve a aparcar en otra plaza (aunque se cambie de sector), se puede utilizar el mismo tique hasta agotar el tiempo que indica.

Suelen permitirse tiques de Zona Azul en zona naranja, ya que para las dos primeras horas la tarifa es la misma.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-40883

1.6.4. Sección: -

1.6.4.1. Dornier, S.A. [Empresa adjudicataria]

Dirección:

C/ ESQUIROZ 30, PLANTA BAJA

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

31007 - PAMPLONA

Trasera.

Teléfono

948-153597

Horario:

De lunes a viernes de 08:30 h. a 14:00 h. y de 16:00 h. a 20:00 h.

Sábados de 08:30 h. a 14:00 h.

FUENTE: http://www.pamplona.es/verPagina.asp?idPag=20-40495

Página principal: http://www.empark.es/inicio.asp

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

8. Codificación: Alarmas

La cuarta pestaña de la aplicación, un centro de control de las alarmas añadidas en la pestaña *Calcular.*

8.1. Objetivo

Esta pestaña pretende ofrecer un lugar en el que los usuarios consulten las alarmas que han añadido al calcular la hora en la que se les acaba el aparcamiento.

En principio las únicas acciones que se le permiten al usuario son las de consultar y borrar alarmas, pero no modificarlas. Esto se ha decidido por cuestiones de sencillez. No son alarmas complejas de citas, reuniones o eventos que pueden variar mucho. Son simples recordatorios, tiene sentido que se puedan consultar y es necesario que se puedan anular pero editarlas no es necesario. No es habitual, pero si un usuario decide modificar una alarma, le bastará con borrarla y volver a ponerla a la hora deseada.

8.2. Mecanismos utilizados

8.2.1. Añadir alarmas

Las alarmas se añaden en la pestaña *Calcular* y se forman con un objeto del tipo *UILocalNotification.* Se debe ajustar cuándo aparecerá la alarma (fireDate), la zona horaria y un mensaje. Además no se desea que la notificación tenga una acción, es decir, el usuario solo podrá elegir 'OK' y no tendrá la posibilidad de 'ver' (Figura 8-2-1). Esto es así porque son notificaciones sencillas que no ofrecen más información, no como un mensaje, correo electrónico, ...

Figura 8-2-1 A la izquierda se ve una aplicación que notifica con un mensaje con una acción 'view'. A la derecha las notificaciones de iPark sin acción asociada.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

También existe la posibilidad de añadir más información a la alarma mediante un *NSDictionary* en la propiedad *userInfo.* Esto se utiliza para que la alarma también contenga cuándo es la hora de fin de estacionamiento, que es diferente de la hora a la que suena el recordatorio. Por último se especifica el sonido de la alarma, en este caso existe una constante para especificar el sonido por defecto de las alertas.

Una vez tenemos la notificación correctamente configurada, basta con añadirla como notificación local con la instancia compartida de *UIApplication.*

A continuación el código para añadir una alarma:

```
UILocalNotification *localNotif = [[UILocalNotification alloc] init];
localNotif.fireDate  = fechaAlarma;
localNotif.timeZone  = [NSTimeZone defaultTimeZone];
localNotif.alertBody = mensaje;
localNotif.hasAction = NO;

// Poner la hora de fin en el diccionario de userinfo
localNotif.userInfo = [NSDictionary dictionaryWithObjectsAndKeys:
                       fechaFinal , @"horaFin", nil ];

localNotif.soundName = UILocalNotificationDefaultSoundName;

// Añadir la notificación
[[UIApplication sharedApplication] scheduleLocalNotification:localNotif];
[localNotif release];
```

8.2.2. Mostrar alarmas

Las alarmas se muestran en una *UItableView* usando las mismas imágenes de fondo para las *cell* que se utilizan en la pestaña *Info.* Los datos de las alarmas se cogen directamente de las alarmas que están programadas en la aplicación:

```
[[UIApplication sharedApplication] scheduledLocalNotifications];
```

Este array es el único *datasource* de esta tabla. Así, cuando una notificación se cancele o pase ya no aparecerá más en la aplicación. Además, está la ventaja que este array no contiene nunca alarmas duplicadas y estarán ordenadas por la fecha en la que van a sonar.

Cada fila de una alarma debe mostrar por una parte cuándo va a sonar la alarma y por otra cuándo termina el periodo de estacionamiento exactamente (Figura 8-2-2).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

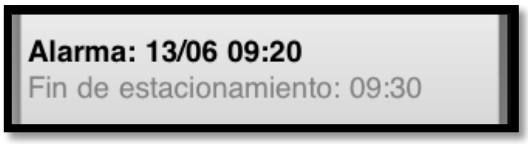


Figura 8-2-2 Una cell mostrando que la alarma sonará el 13 de Junio a las 9:20, 10 minutos antes del fin del estacionamiento.

8.2.3. Borrar alarmas

Para proceder al borrado de las alarmas se debe poner la *tableView* en modo editar. Para ello se ha añadido un botón en la parte superior del *navigationController* que conmuta entre el modo editar y el modo visualizar. Se pueden apreciar los dos modos en la siguiente Figura:

Figura 8-2-3 Tabla en el modo normal (izquierda) y en el modo editar (derecha)

Tocando en el icono de prohibido el paso se nos muestra la opción de confirmar el borrado de esa alarma. Así podemos borrar las alarmas individualmente.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

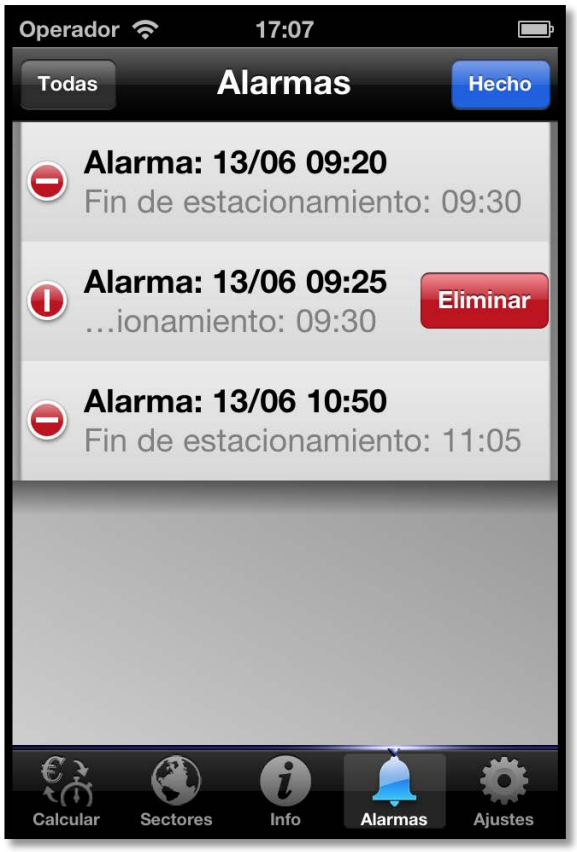


Figura 8-2-4 Botón para confirmar el borrado.

Con esta funcionalidad surgió un pequeño problema que costó solucionar. El botón que reza 'eliminar' era imposible encontrarlo para cambiarle el nombre ya que había que traducirlo a euskera e inglés. Al final la solución fue sencilla pero hubo que investigar, resulta que existe una método delegado de *tableView* que pregunta por el título del botón:

```
// Meted para devolver el título del botón rojo de borrar traducido.
- (NSString *)tableView:(UITableView *)tableView
titleForDeleteConfirmationButtonForRowAtIndexPath:(NSIndexPath *)indexPath {
    return [TraductorHash traducir:@"Eliminar"];
}
```

Como se puede apreciar en la Figura 8-2-4 anterior, se ha añadido también la posibilidad de borrar todas las alarmas de una sola vez con la finalidad de facilitar al usuario la tarea repetitiva en caso de que quiera anular todos los recordatorios.

8.2.4. Problema al añadir alarmas

En uno de los test informales a los que se ha sometido la aplicación con familiares y amigos, se descubrió que el usuario al querer añadir una alarma iba

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

directamente a la pestaña de alarmas. Eso se debe a que desconocía que la manera de añadir recordatorios es mediante la pestaña *Calcular.*

Por ello, cuando no hay alarmas que mostrar al usuario, se aprovecha el espacio libre para mostrar un mensaje de ayuda con un icono que permite identificar rápidamente dónde podemos añadir alarmas.

Figura 8-2-4 Pestaña de alarmas cuando no hay alarmas que mostrar

Esta cell no tiene controles de borrado. Aun así, cuando estamos con esta única *cell* en modo editar y el usuario toca el botón de borrar todas, una animación agita la cell de izquierda a derecha. De esta manera transmitimos al usuario que no se puede borrar y le recordamos cómo se añaden las alarmas.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

9. Codificación: Ajustes

La última funcionalidad de la aplicación son los ajustes, la manera en la que el usuario puede ajustar determinados aspectos de la aplicación a su gusto.

9.1. Objetivo

Se desea ofrecer una pantalla para especificar las preferencias del usuario tanto dentro de iPark (en la quinta pestaña) como en la aplicación de ajustes nativa de los dispositivos iOS. Se desea también que ambas pantallas sean idénticas.

La principal función de esta pantalla será la de poder cambiar el idioma de la aplicación. También se podrán especificar los intervalos de antelación que se presentan al confirmar una nueva alarma. Por último, podrá deshabilitarse la función de localización de usuario por cuestiones de privacidad y ahorro de batería.

9.2. Mecanismos utilizados

9.2.1. Settings.bundle

La manera de añadir una pantalla de ajustes en los ajustes del sistema es añadir un *bundle* de este tipo al proyecto.

Este *bundle* no es más que una carpeta que contiene, entre otras cosas, un archivo denominado *Root.plist.* Esta lista de propiedades debe seguir un formato concreto[26] para que los ajustes se visualicen de manera correcta. Se pueden añadir *switches*, campos de texto, *sliders*, pantallas para elegir entre múltiples valores (*multivalue*), ... También es posible anidar diferentes paneles de ajustes.

Para esta aplicación, necesitaremos únicamente dos *multivalues* para el idioma y los avisos previos y un *switch* para la localización. En la Figura 9-2-1 se puede apreciar la manera en la que se debe especificar cada ajuste y cómo se separan en grupos.

[26] Guía de la interfaz de ajustes del sistema de Apple: http://bit.ly/kgQW8i

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Key	Type	Value
Preference Items	Array	(5 items)
Item 0 (Group -)	Diction...	(2 items)
Item 1 (Multi Value - Idioma)	Diction...	(6 items)
Default Value	Number	0
Identifier	String	idioma
Title	String	Idioma
Titles	Array	(3 items)
Type	String	Multi Value
Values	Array	(3 items)
Item 2 (Group -)	Diction...	(2 items)
Item 3 (Multi Value - Avisos	Diction...	(6 items)
Default Value	Array	(3 items)
Identifier	String	intervalos
Title	String	Avisos previos
Titles	Array	(3 items)
Type	String	Multi Value
Values	Array	(3 items)
Item 4 (Toggle Switch - Permitir	Diction...	(4 items)
Default Value	Boolean	YES
Identifier	String	localizar
Title	String	Permitir localizar
Type	String	Toggle Switch
Strings Filename	String	Root

Figura 9-2-1 Archivo *Root.plist* que contiene el formato concreto para los ajustes del sistema.

Una vez configurado este archivo, basta con ejecutar la aplicación y se obtiene una entrada en los ajustes del sistema (Figura 9-2-2).

Esta es una manera relativamente rápida de añadir los ajustes. Sin embargo, está muy limitado. Este archivo se compila y firma automáticamente por lo que es imposible hacerle cambios en tiempo de ejecución. Esto significa que los ajustes son estáticos y es imposible traducirlos al euskera. Podrían traducirse al inglés utilizando el sistema nativo de traducción de Apple, pero este sistema no soporta el euskera. Por ello, se ha preferido dejar los ajustes de Settings.bundle solamente en castellano.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 9-2-2 A la izquierda la pantalla principal de la aplicación *Ajustes*. A la derecha la pantalla a la que se accede con los ajustes personalizados de iPark.

9.2.2. Ajustes dentro de la aplicación.

Dentro de la aplicación se va a replicar el comportamiento de los ajustes del apartado anterior. Pero va a haber una diferencia, se va a poder controlar cada aspecto de estas pantallas por lo que se actualizarán correctamente con el idioma que seleccionemos.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 9-2-3 Seleccionando idioma inglés en los *settings* del sistema (no se traduce) y en los *settings* de la aplicación.

Además, deben estar sincronizadas ambas pantallas de ajustes, mostrando en todo momento la misma información que muestra la otra. Esto es imprescindible para evitar confusión entre los usuarios.

La programación de esta pantalla de ajuste en la aplicación se basa en un controlador principal (*AjustesTableViewController*) que tiene un *cell* para cada ajuste. Además, para los ajustes *multivalue* se ha creado un nuevo controlador genérico (*AjustesSeleccionarViewController*) que, dado un *array* de valores y su correspondiente *array* de títulos, los visualiza y permite elegir entre ellos. También se especificará un objetivo (*target*) y una acción (*selector*) a realizar. Esto permite que el controlador no deba saber a priori qué método de qué objeto debe ejecutar a la hora de actualizar un valor.

La clase *Ajustes* es una clase *singleton* que tiene una sola instancia. Cuenta con distintos métodos para actualizar cada preferencia, por ejemplo:

```
- (void) ajustarIdiomaConInt: (int) idi;
- (void) ajustarIntervalos: (NSArray *) intervalos;
```

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Así, cuando se seleccione *Idioma* en la pantalla de ajuste, se creará un *AjustesSeleccionarViewController* al que se le pasará `[Ajustes instancia]` como *target* y `@selector(ajustarIdioma:)` como acción. De esta manera, cuando el usuario seleccione un nuevo idioma se ejecutará la acción dada sobre el objetivo dado con el valor correspondiente del *array* de valores:

```
[self.target performSelector: self.action
                  withObject: [self.valores objectAtIndex: indexPath.row]];
```

Figura 9-2-4 Comparación entre las pantallas principales de ajustes.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

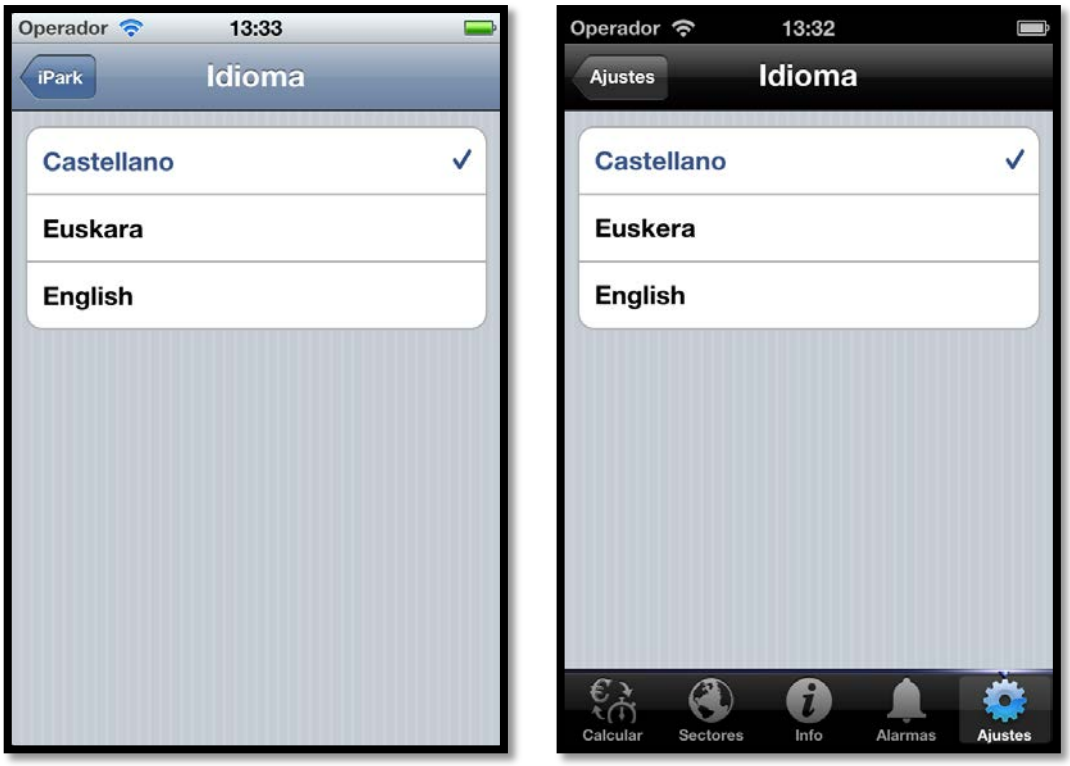


Figura 9-2-5 Comparación entre las pantallas de selección de idioma.

9.2.3. Sincronizar cambios en ajustes.

Cuando los ajustes cambian, la aplicación no ve los cambios hasta que se sincroniza:

```
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
[defaults synchronize];
```

Pero sincronizar los ajustes es una operación cara. Si los ajustes no son accedidos con mucha frecuencia, esto no supondría un problema. Pero en esta aplicación, se accede a los ajustes para consultar el idioma cada vez que se carga una cadena de caracteres. Por lo tanto, los ajustes se cargarán una única vez y se mantendrán como propiedades de instancia del *singleton* Ajustes.

Para mantener los ajustes actualizados a tiempo real, se ha hecho que el delegado de la aplicación incluya este código:

```
// Registrar para cambios en defaults
NSNotificationCenter *center = [NSNotificationCenter defaultCenter];
[center addObserver: [Ajustes instancia]
           selector: @selector(cambiosEnPreferendias)
               name: NSUserDefaultsDidChangeNotification
```

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
                object: nil];
```

De esta manera, cada vez que haya cambios en las preferencias, se llamará automáticamente al método *cambiosEnPreferencias* de la clase *Ajustes*. Esto habría sido más difícil de implementar si *Ajustes* no hubiera sido una clase singleton.

En el método *cambiosEnPreferencias,* no tenemos más que actualizar los valores de las propiedades con los nuevos valores. De esta manera hacemos que el acceso a los ajustes sea eficiente y que además esté actualizado.

A la hora de escribir nuevos ajustes, no hay ningún problema en sincronizar cada vez ya que solo ocurren con acciones del usuario (pasan muchos milisegundos entre una acción del usuario y otra, por muy rápido que las intente hacer).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

10. Conclusiones

10.1. Objetivos conseguidos

Al final, se ha obtenido una aplicación que es capaz de ayudar en los cálculos de los tiques de aparcamiento y mostrar recordatorios si el usuario lo desea. Muestra de manera interactiva el mapa de sectores de residentes que se puede actualizar sin necesidad de sacar una nueva versión del programa. Incluye información relevante y actual acerca de todos los detalles de la Zona Azul. Todo esto disponible en tres idiomas configurables mediante los ajustes de la aplicación y con una interfaz cuidada, atractiva, intuitiva y fácil.

iPark es completamente funcional y ha sido probada en diferentes dispositivos (*iPhone, iPad* y *iPod Touch*) ofreciendo un rendimiento excelente. Por lo tanto, se puede decir que está lista para ser vendida en la *aplication store,* al ayuntamiento de Pamplona o a la empresa adjudicataria Dornier, S.A.

Además, se ha obtenido una sólida base de conocimiento tanto en la plataforma como en el lenguaje y los dispositivos utilizados. Se ha adquirido una amplia experiencia con los mecanismos y particularidades de *Objective-C*, con el IDE *Xcode 4* y el sistema de control de versiones *subversion*. Se ha aprendido a diseñar y programar utilizando el patrón Modelo-Vista-Controlador. También se ha utilizado y dominado gran parte de los frameworks:

- *UIKit (UIView, UIButton, UITableView, ...)*
- *Foundation (NSArray, NSDictionary, NSString, ...)*
- *MapKit (MKMapView, MKPolygon, MKCoordinate, ...)*
- *CoreLocation (CLLocationCoordinate2D, CLLocationManaer, ...)*

Se ha comprendido y utilizado el formato *KML*, parte del *API* de *Google Maps.* También se ha aprendido un manejo básico de aplicaciones de diseño y retoque fotográfico (*Pixelmator, Omnigraffle* y *Photoshop*). Por último, se ha podido obtener un atisbo de lo complicado que puede resultar desarrollar un proyecto software medianamente complejo y las dificultades que entraña la gestión del tiempo, la productividad, etc.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Por tanto, se puede concluir que al finalizar el presente proyecto se han cumplido los objetivos y expectativas declarados al principio del mismo.

10.2. Análisis DAFO

También se ha realizado un análisis[27] sobre el producto software conseguido

10.2.1. Amenazas

La plataforma *Android* es una competidora directa de *Apple.* Está sufriendo un desarrollo y crecimiento espectaculares y cada vez cuentan con más y más usuarios. En principio no se prevé como una amenaza importante ya que aunque comparten un segmento parecido del mercado de *smartphones,* el tipo de usuario de una y otra plataforma suelen diferenciarse.

10.2.2. Oportunidades

Unida a la amenaza del punto anterior, surge una oportunidad de expansión de la aplicación: el desarrollo de esta para *Android*. Esto permitiría llegar a un mayor número de usuarios y extenderse más rápidamente. También pueden considerarse otras plataformas como *Windows Phone, webOS,* etc. pero de momento se considera que están todavía en fases tempranas y no se conoce si se afianzarán satisfactoriamente en el mercado.

Otra oportunidad es la de expandir la aplicación para otras ciudades en las que existen mecanismos similares a la *Zona Azul.*

10.2.3. Fortalezas

La interfaz cuidada hace que el manejo de la aplicación sea realmente sencillo e intuitivo. Por ejemplo, el usuario nunca deberá preocuparse por introducir texto mediante un teclado ya que la aplicación está pensada para ser utilizada de manera ágil y que en segundos haya cumplido su cometido.

La integración de diferentes idiomas con mecanismos propios permite ser una de las primeras aplicaciones traducidas completamente a Euskera. Además,

---

[27] Inspiración en el análisis DAFO:
http://www2.uca.es/serv/dafo/DAFOhelp.html

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

es muy sencillo introducir un nuevo idioma en la aplicación, incluso sin apenas conocimiento de programación.

10.2.4. Debilidades

La aplicación está muy optimizada para dispositivos iOS lo que dificulta la reutilización del código en otras plataformas. Además, el lenguaje utilizado, Objective-C, es muy diferente a la mayoría de otros lenguajes (incluido el propio C, Java, ...)

La aplicación tiene un alcance limitado y un grupo de usuarios reducido: gente que tenga un iPhone y un coche en Pamplona. Sin una expansión a otras ciudades no se puede desplegar su verdadero potencial.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

11. Bibliografía

Sin duda, el libro de referencia que ha servido para obtener la base de conocimiento del desarrollo de aplicaciones para iOS:

- BEGINNING IPHONE 4 DEVELOPMENT. EXPLORING THE IPHONE SDK. Dave Mark, Jeff LaMarche, Jack Nutting. Apress.
  Edición digital: http://www.apress.com/9781430230243

Otros libros acerca de programación para dispositivos iOS:

- MORE IPHONE 4 DEVELOPMENT: FURTHER EXPLORATIONS OF THE IOS SDK. Dave Mark, Jeff LaMarche., Jack Nutting Apress.
- BEGINNING IPHONE® SDK PROGRAMMING WITH OBJECTIVE-C®. Wei-Meng Lee. Wrox.
- IPHONE AND IPAD APPS FOR ABSOLUTE BEGINNERS. Dr. Rory Lewis. Apress.
- IPHONE DESIGN AWARD-WINNING PROJECTS. Chris Dannen. Apress
- MORE IPHONE COOL PROJECTS. (Varios autores). Apress.

Libros acerca del lenguaje Objective-C:

- LEARN OBJECTIVE-C ON THE MAC. Mark Dalrymple, Scott Knaster. Apress
- LEARNING OBJECTIVE-C 2.0. Robert Clair. Addison-Wesley.

Otra bibliografía:

- COCOA PROGRAMMING FOR MAC OS X 3RD EDITION. Aaron Hillegass. Addison Wesley Professional.
- SHNEIDERMAN, B. EIGHT GOLDEN RULES FOR INTERFACE DESIGN. EN: DESIGNING THE USER INTERFACE, 3ª EDN. Addison Wesley, U.S.A.

11.1. Bibliografía digital

Toda la información sobre la Zona Azul se ha obtenido visitando las oficinas de Dornier, S.A. y consultando las páginas que aparecen en el siguiente enlace. Además, estas páginas son las que aparecen como fuentes de la información en la pestaña *Info* (apartado 7).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- http://www.pamplona.es/verPagina.asp?idPag=20-52870

También se merece una mención especial la página:

- http://stackoverflow.com/

Gracias a esta última página se han solucionado la mayor parte de dudas técnicas que surgen a la hora de programar. También ayuda mucho ver técnicas y algoritmos de otros desarrolladores ya que la página se basa en un entorno colaborativo y abierto de preguntas y respuestas.

Asimismo, han sido literalmente innumerables las consultas que se han hecho a la propia documentación de *Apple*. Gracias al entorno de desarrollo integrado *Xcode 4*, la mayor parte de la documentación se consulta sin moverse del código que se está escribiendo.

A continuación se muestran los links sin acortar de los enlaces recogidos en las notas a pie de página de todo el documento (en orden de aparición):

- http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html
- http://fullofdesign.ca/uploads/iphone_notepad_horizontal.pdf
- http://www.omnigroup.com/products/omnigraffle/.
- http://graffletopia.com/
- http://developer.apple.com/library/ios/#documentation/MacOSX/Conceptual/BPInternational/Articles/InternatSupport.html#//apple_ref/doc/uid/20000278-SW1
- http://developer.apple.com/library/ios/#documentation/MacOSX/Conceptual/BPInternational/Articles/LanguageDesignations.html#//apple_ref/doc/uid/20002144-SW1
- http://developer.apple.com/library/ios/#documentation/MacOSX/Conceptual/BPInternational/Articles/InternationalGuidelines.html#//apple_ref/doc/uid/20002142-BBCFJBFB
- http://pixelresort.com/blog/iphone-app-icon-design-best-practises/
- http://developer.apple.com/library/ios/#documentation/general/conceptual/DevPedia-CocoaCore/MVC.html
- http://itunes.apple.com/us/app/xcode/id422352214

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- http://blog.twg.ca/2010/11/retina-display-icon-set/twg_retina_icons/
- http://www.pixelmator.com/
- http://howtomakeiphoneapps.com/2011/02/how-to-skin-your-iphone-app-with-core-animation/
- http://developer.apple.com/library/ios/#documentation/general/conceptual/DevPedia-CocoaCore/MVC.html
- http://matteocaldari.it/2010/05/a-uisegmentedcontrol-with-custom-color
- http://www.pamplona.es/verPagina.asp?idPag=20-40495
- http://www.pamplona.es/verPagina.asp?idPag=20-51881
- https://github.com/FairfaxMobile/FDCurlViewControl
- http://developer.apple.com/library/mac/#documentation/GraphicsImaging/Reference/CGPath/Reference/reference.html
- http://www.pamplona.es/verPagina.asp?idPag=20-68071
- http://www.pamplona.es/verPagina.asp?idPag=20-68072
- http://www.pamplona.es/verPagina.asp?idPag=20-68421
- http://www.pamplona.es/verPagina.asp?idPag=20-68422
- http://www.pamplona.es/pdf/ordenanza_zona_azul_mayo2009.pdf
- http://goo.gl/maps/12Ci
- https://code.google.com/intl/es/apis/kml/documentation/
- http://developer.apple.com/library/ios/#samplecode/ScrollViewSuite/Introduction/Intro.html#//apple_ref/doc/uid/DTS40008904
- http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/Preferences/Preferences.html
- http://www2.uca.es/serv/dafo/DAFOhelp.html

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Mikel Elorz Berástegui

Pamplona, 1 de julio de 2011

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira