eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Proyecto Fin de Carrera

Ingeniería Técnica en Informática de Sistemas

Autor - Iker Beristain Villar

Director - José Miguel Blanco Arbe

Junio 2013

McPhone
para iPhone

Índice

0. REFERENCIAS A TRADEMARKS Pág. 6

1. INTRODUCCIÓN Pág. 8

2. ANTECEDENTES Pág. 10

2.1 Antecedentes de necesidad Pág. 10

2.2 Antecedentes tecnológicos Pág. 11

2.2.1 Recursos en la nube - Google Drive Pág. 11
2.2.2 Plataforma móvil - iOS Pág. 12
2.2.3 Comparativa iOS VS Android OS Pág. 13
2.2.4 XCode SDK 4.6.2 y Objective-C Pág. 15

2.3 Antecedentes Personales Pág. 16

3. OBJETIVOS DEL PROYECTO Pág. 18

3.1 Objetivo de optimización sistema comandas Pág. 18

3.1.1 Análisis del sistema y funcionamiento Pág. 18
3.1.2 Diagramas de flujo de los sistemas Pág. 19

3.2 Alcance Pág. 20

3.3 Diagrama de descomposición del trabajo Pág. 21

3.4 Responsabilidades Pág. 22

3.5 Lista de tareas Pág. 23

4. ANÁLISIS DE REQUISITOS Pág. 25

4.1 Requisitos Funcionales Pág. 25

4.1.1 Casos de Uso Pág. 26
4.1.1.1 Gestión de pedidos Pág. 26
4.1.1.2 Consulta Restaurantes Pág. 29
4.1.1.3 Acceso a la web de la compañía Pág. 31

4.2 Requisitos Tecnológicos Pág. 32

5. DISEÑO DE LA ARQUITECTURA DE LA SOLUCIÓN Pág. 36

6. DISEÑO DE LA APLICACIÓN Pág. 38
6.1 Capa interacción del usuario Pág. 38
6.1.1 Navegación y selección productos Pág. 40
6.1.2 Navegación y selección restaurantes Pág. 44
6.1.3 Acceso a la web de la compañía Pág. 46
6.2 Estructura de datos Pág. 46
6.2.1 Objetos Pág. 47
6.2.2 Carga y Almacenamiento Pág 52
6.2.3 El pedido y JSON Pág. 53
6.3 Lógica de negocio Pág. 55
7. IMPLEMENTACIÓN Pág. 59
7.1 Entorno de programación Pág. 59
7.1.1 IDE, SDK, Xcode Pág. 59
7.1.2 División del entorno de Xcode Pág. 62
7.1.3 Simulador iPhone Pág. 64
7.2 Código Pág. 64
7.3 Valoración del código Pág. 68
8. PUBLICACIÓN EN AppleStore Pág. 70
8.1 Información general Pág. 70
8.2 Experiencia publicación McPhone en AppleStore Pág. 71
9. PRUEBAS Pág. 75
9.1 Pruebas registradas en dispositivos Pág. 75
9.2 Casos de prueba Pág 76

10. GESTIÓN DEL PROYECTO Pág. 78
10.1 Planificación Pág. 78
10.1.1 Planificación inicial Pág. 78
10.1.2 Desarrollo y Seguimiento Pág. 79
10.1.3 Diagramas de GANTT Pág. 81
10.1.4 Valoración - Planificación estimada VS real Pág. 82

11. CONCLUSIONES Pág. 84
12. AGRADECIMIENTOS Pág. 88
13. REFERENCIAS Pág. 90
13.1 Enlaces web Pág. 90
13.2 Bibliografía Pág 90
13.3 Documentación Pág. 90
ANEXO, POSIBLES UPDATES DE MCPHONE Pág. 94
ANEXO, MANUAL DEL USUARIO Pág 97

0.REFERENCIAS A TRADEMARKS

El presente documento incluye nombres comerciales, logos y referencias a marcas registradas por terceros.

LOGOS:

MARCAS:

APPLE

Apple®

Apple® Store

iPhone®

Xcode®

iOS™

GOOGLE

Android™

Google™

GoogleDrive™

MCDONALD'S

McDonald's Sistemas de España, INC

McDonald's®

Big Mac®

Mc Royal Deluxe®

McNuggerts®

McPollo®

OTROS

Coca-Cola®

Fanta®

Lipton Ice Tea®

Windows Phone®

Microsoft® SkyDrive

Dropbox®

SugarSync®

Amazon,INC Cloud Drive

Ubuntu® One

MediaFire®

1. INTRODUCCIÓN

La presente memoria agrupa y presenta la documentación del proyecto fin de carrera llamado "McPhone".

Con la motivación de poder abordar un proyecto y elaborar un producto de cosecha propia, el autor se ha decantado por el mundo de desarrollo de software para plataformas móviles. Mostrándose un entorno atractivo por el perfil actual y vanguardista que denota en estos últimos años, sin dejar de lado el componente de mercado laboral emergente que representa.

La idea original está basada en mejorar de manera sustancial la gestión del servicio de un negocio, apoyándose en las posibilidades que nos ofrecen las plataformas móviles. El proyecto consiste en el diseño e implementación de una aplicación "McPhone" para dispositivos móviles, que mejore y evolucione el sistema de realizar pedidos en un restaurante de comida rápida. Se ha centrado en la empresa multinacional McDonald's, adaptándose en todo momento a sus necesidades.

La motivación por realizar este proyecto y muestra de que operacionalmente es beneficioso, reside en el actual sistema de pedidos bajo un dispositivo tipo PDA[1] que posee la compañía, que va a quedar prácticamente obsoleto, puesto que los Smartphone son muy populares a día de hoy, y están en manos de casi cualquier persona. Es importante mencionar que los PDAs[1] son propiedad del restaurante, con el coste que supone su compra y mantenimiento, además de requerir de un empleado para poder tomar nota a los clientes.

La aplicación está orientada a optimizar la gestión de comandas, permitiendo al cliente conformar un pedido completo desde su dispositivo móvil, de manera desatendida por parte del restaurante. Después será capaz de enviarlo a un restaurante de los disponibles en la aplicación, y el restaurante podrá entonces proceder a la descarga, consulta y tramitación de dicho pedido.

Para poder desarrollar la mencionada aplicación se ha afrontado, por ejemplo, la dirección del proyecto, formación, diseño, análisis de requisitos, desarrollo de la solución, implementación y demás tareas que se expondrán a lo largo de este documento.

[1] *PDA*: *Un ordenador de bolsillo, organizador personal o una agenda electrónica de bolsillo. Suelen llevar pantalla táctil para la navegación. [wikipedia]*

La presente memoria está estructurado por capítulos. Se comienza analizando los antecedentes que motivan este proyecto y los objetivos que se han marcado en la etapa inicial del mismo. Después se explican los requisitos que deberá cumplir la solución ofrecida, así como el diseño de la solución elaborada que satisface los requisitos. A continuación, se habla sobre el diseño dado a la aplicación y de su fase de implementación. Luego se dedica espacio a explicar la publicación en AppleStore y las pruebas a las que ha sido sometida la aplicación para poder publicarla. Antes de llegar a las conclusiones finales, se ha documentado cómo se ha gestionado el proyecto desde el inicio hasta el fin. Al final, tras las conclusiones, contaremos con las referencias y agradecimientos oportunos. Han sido anexados dos documentos que son de gran interés; por un lado, un anexo que trata sobre una entrevista que tuvo lugar en las oficinas de Madrid en la que se contrastó la viabilidad del proyecto; y por otro lado, el manual del usuario para el manejo de McPhone.

2. ANTECEDENTES

En este capítulo se exponen los antecedentes de necesidad, tecnológicos y personales en los cuales se ha basado el autor para el desarrollo del proyecto.

2.1 Antecedente de necesidad

Comencemos analizando detenidamente cuál es la necesidad real por la que es interesante optimizar el sistema actual mediante McPhone.

Tiempos de servicio:

Es necesario hacer una breve explicación sobre los tiempos de servicio que se estiman para poder entender porqué este sistema optimiza operacionalmente el servicio del restaurante. Hay dos tiempos que se miden en la atención al cliente, que no deben ser rebasados. Los denominados "tiempo total de la experiencia" y "tiempo de servicio".

Tiempo total de la experiencia: (210 segundos)

> Tiempo desde que el cliente se pone en la fila hasta que le entregan el pedido completo.

Tiempo de servicio: (60 segundos)

> Tiempo desde que el empleado dice al cliente el importe del pedido, hasta que le entrega el pedido completo. (incluido en el "tiempo total de la experiencia")

Con la aplicación desarrollada se optimiza parte del "tiempo total de la experiencia", es decir, el tiempo que se consume mientras el cliente indica al cajero todos los productos que desea. Por lo que de ese tiempo, 150 segundos que se admiten por la compañía en el peor de los casos, la aplicación puede ahorrar una gran cuantía de tiempo.

2.2 Antecedentes tecnológicos

Analizaremos los antecedentes de las tecnologías que se han escogido, a priori, para la realización del proyecto, aclarando que queda fuera del alcance del mismo el estudio de las alternativas (aunque se mencionarán en algunos casos). Las alternativas existen, son numerosas e interesantes, pero dados los recursos disponibles se ha optado por una selección de partida centrada en herramientas sólidas y de interés para el futuro profesional del autor.

Nos centraremos entonces en los recursos en la nube ofrecidos por GoogleDrive, las plataformas móviles iOS, una comparativa entre iOS y Android, y por último en Xcode y el lenguaje de programación Objective-C.

2.2.1 Recursos en la nube - Google Drive

Ante la necesidad de transmitir información entre dos puntos, es necesario contar con un recurso que nos permita almacenar y acceder a información en la nube. Para ello existen numerosos servicios que cumplen estas características. Algunos de ellos son: SkyDrive (Microsoft), dropBox, SugarSync, Google drive, Amazon Cloud Drive, Ubuntu One, MediaFire, Mega, etc.

Por su carácter experimental, por la integración con otros servicios para una futura posible expansión y por la prestación de manera gratuita de sus servicios se ha optado por utilizar la API de Google Drive. Además, ofrece un soporte multiplataforma y tiene aplicación para todos los sistemas operativos principales con lo que cubriría las necesidades de este proyecto.

Sencillamente se utilizan los servicios Google Drive para almacenar un archivo, el ticket del cliente en formato JSON, y su posterior descarga en el restaurante.

2.2.2 Plataforma móvil - iOS

Se ha decidido desarrollar la aplicación para un dispositivo móvil por las necesidades obvias en el entorno que va ser utilizada. El cliente lo suponemos en cualquier lugar, y por ello una aplicación móvil es lo más adecuado. Se ha basado en iOS 6.x, y como requisito mínimo iOS 5.1, que es el sistema operativo del dispositivo iPhone de Apple. Se ha considerado la posibilidad de desarrollarlo en Android.

En la próxima página, para una evaluación justa y estudiada de a qué plataforma debía dirigirse el desarrollo, se ha hecho la siguiente comparativa entre los dos sistemas operativos móviles actualmente más populares del mercado. La información está basada en testimonios de terceros, y el autor no ha tenido la oportunidad de profundizar en ambos entornos de manera práctica; aun así, le ha sido útil para poder concluir a qué plataforma debía dirigirse este proyecto.

2.2.3 Comparativa iOS VS Android OS

iOS	Android OS
Accesibilidad: -Xcode sólo corre en un Mac para desarrollas las Apps, con último MAC OS. -iOS es exclusivo de terminales de Apple.	Accesibilidad: -Eclipse está disponible en Mac, Linux y Windows y se puede instalar la SDK en cualquiera de esas plataformas. -Android es adaptable por muchos fabricantes. Está orientado a un público masivo
Curva de aprendizaje: Objective-C es un lenguaje exclusivo para sistemas MAC e iOS. Inspirado en C, con una sintaxis muy similar. Objective-C tiene características de bajo nivel de C, y ventajas respecto Java.	Curva de aprendizaje: Java es uno de los lenguajes más extendidos. Java al ser "universal" existe infinidad de documentación y soporte.
Gestión de la memoria: A partir de iOS5 se implementó el ARC(automatic reference counting), que optimiza el rendimiento de la memoria.	Gestión de la memoria: Posee recolector de memoria, no tan eficaz como el de iOS.
Documentación: Existe gran cantidad de documentación, proporcionada por Apple.	Documentación: Existe gran cantidad de documentación, y además Java tiene gran soporte de la comunidad.
Diseño de la interface: Al tratarse de terminales más cerrados, es más fácil diseñar para ellos. El editor de diseño de Xcode es más suave que en Android.	Diseño de la interface: Android corre en terminales de resoluciones completamente dispares. La App tendría que estar adaptada a todas ellas. Diseños en XML de fácil reutilización de código.
SDK: Xcode Menos consumo de la memoria. A pesar de su carácter cerrado, es muy eficaz y suave. Simulación de dispositivo.	SDK: Eclipse Consumo excesivo de memoria. Compatible con Linux, Windows y Mac. Emulación de dispositivo.

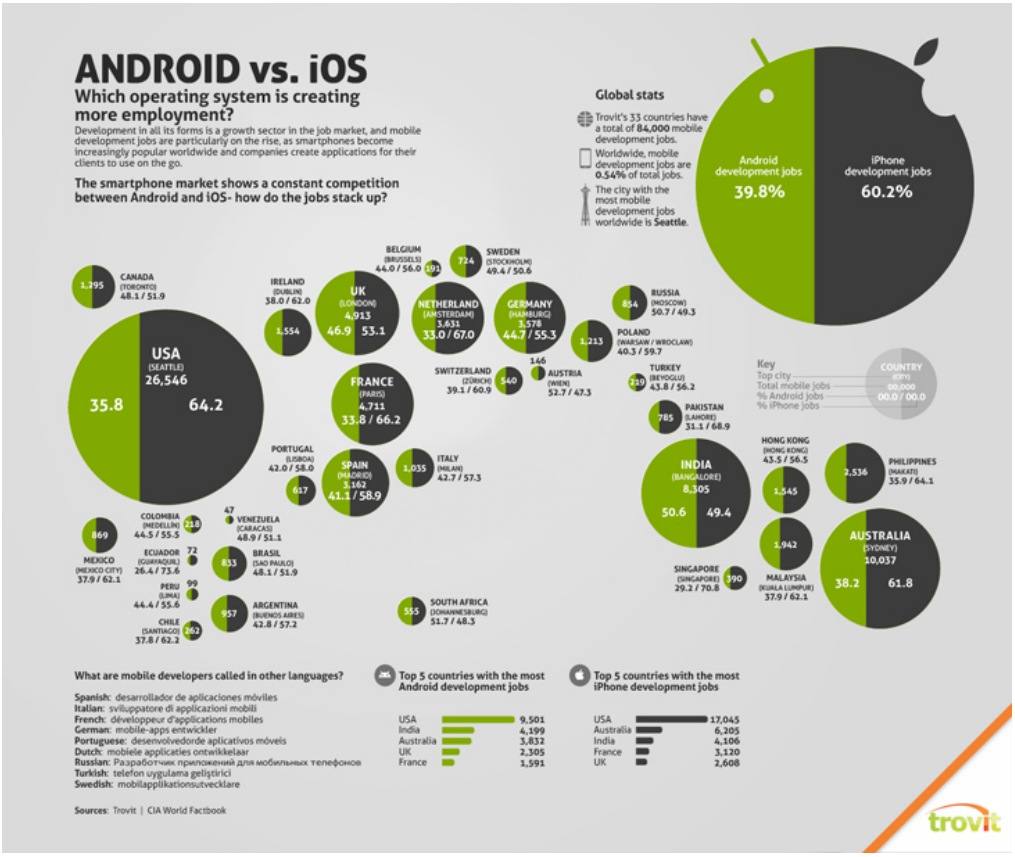
Connotación de los colores: a favor | igualado | en contra

CONCLUSIÓN

Ambas plataformas son perfectamente válidas. Android proporciona más facilidades por la expansión que tiene y su carácter universal, tanto de la plataforma como del entorno de programación. En cambio Apple ofrece un entorno mucho más cerrado, pero que funciona incluso mejor que el de la competencia. Es más complejo adentrarse en el desarrollo para iOS, pero después la probabilidad de éxito laboral es mayor.

La apuesta arriesgada es la que el autor considera más acertada. Viendo el éxito del AppStore respecto a Android Market, los desarrollos se centran principalmente para iOS. Esto supone una más amplia oferta de trabajo, y más exclusiva. Ello se debe a que los conocimientos requeridos para desarrollar en iOS son específicos de esa plataforma; en Android en cambio, la formación general en Java permite que "cualquiera" pueda adentrarse en un desarrollo para esa plataforma.

Por ello el desarrollador se ha decantado por iOS de Apple, a pesar de que Android hubiera sido igual de válida.



Fuente: "Trovit y CIA World FactBook, año 2012."

2.2.4 XCode SDK 4.6.2 y Objective-C

Se ha utilizado la última versión disponible del entorno de desarrollo de Apple. Con las novedades más recientes que aportaron, es decir, los *StoryBoard*s[1] y el *ARC*[2] . Se ha hecho uso de ambas tecnologías, por un lado potenciando el diseño basado en *StoryBoard*s, y por otro lado consiguiendo un flujo de trabajo mucho más eficaz durante el desarrollo. Y la valía del *ARC* para la autogestión de los *Memory Leaks*[3], consigue un incremento en la productividad en fase de implementación, pudiendo desentenderse de las liberaciones de las reservas de memoria en Objective-C. Debido al uso de los *StoryBoard*s, es requisito que la plataforma destino funcione bajo iOS 5.1 o superior.

El lenguaje de programación, ha sido Objective-C irremediablemente, por ser exclusivo de iOS y obligado en todos los desarrollos orientados a las plataformas de Apple. La versión 2.0 de este lenguaje fue presentada en el 2006, y ha sido el lenguaje principal de este desarrollo junto con C, para ciertos matices, ya que también es soportado por el entorno de programación XCode.

[1] *StoryBoard*: *Herramienta de xCode para iOS5+ que nos facilitará enormemente la gestión de múltiples pantallas en nuestra App. No solo podemos ver gráficamente cada una de las vistas de nuestra aplicación, sino que podemos conocer cuál es la navegación que las relaciona.*

[2] *ARC, Automatic Reference Counting: Gestiona automáticamente la memoria, asegura buenas practicas y realiza desatendidamente todas las operativas de RETAIN y RELEASE sin que nos preocupemos de ello. Es una funcionalidad proporcionada por LLVM 3.0, que es el nuevo compilador incorporado por Apple como predeterminado a partir de Xcode 4.2.*

[3] *Memory Leaks: Fugas de memoria o pérdida de la misma.*

2.3 Antecedentes Personales

En primera instancia, el autor no tenía claro hacia dónde quería orientar el proyecto fin de carrera. Lo que sí tenía claro era que a nivel personal debía ser una tarea que le supusiera un reto, una culminación a la altura de lo que se podía esperar de años de estudio, esfuerzo y dedicación.

Ponerse barreras uno a sí mismo, es una ardua tarea, más tras años en los que se ha dedicado gran esfuerzo a sortear las que ajenamente se nos han impuesto. Aun así, había un mundo "desconocido" que le llamaba con fuerza; las aplicaciones orientadas a dispositivos móviles. Con los estudios cursados y teniendo en cuenta que actualmente gran parte de los desarrollos de software se dirigen a ese ámbito, consideró que formaría parte de la evolución natural de su desarrollo académico.

Llegados a este punto... ¿Qué desarrollar? ¿Qué puede tener suficiente densidad y utilidad? ¿Qué podría aportar él? Juntó su deseo de desarrollar una App móvil con una idea que le venía rondando tiempo atrás. El autor cuenta con una experiencia laboral de 8 años en el equipo de gerencia en una multinacional de restauración. Tantos años de trabajo compaginados con los estudios, no podía acabar con mayor motivación que producir algo de cosecha propia, y que le ayudara a cobrar sentido a lo hecho durante todos estos años atrás, tanto laboral como académicamente.

De esta manera, afrontaría un proyecto que no se le podía presentar más atractivo a nivel personal. Así surgió la idea de McPhone.

3. OBJETIVOS DEL PROYECTO

En este apartado se explica el alcance y estructuración del trabajo, así como las responsabilidades y relación de tareas a realizar.

3.1 Objetivo de optimización sistema comandas

Pasemos a analizar el sistema actual y los objetivos que debemos cumplir para que el proyecto resulte un éxito.

3.1.1 Análisis del sistema y funcionamiento

El sistema actual PDA, requiere de un empleado tomando nota, y sólo es viable y rentable a partir de tener un octavo empleado trabajando. Es decir, con menos de ocho empleados no se debe utilizar. Y a partir del empleado diecisieteavo se evalúa si conviene tener un segundo empleado con otro PDA tomando pedidos. Todas estas restricciones quedarían eliminadas con el nuevo sistema "McPhone".

Explicación sistema actual bajo PDA

En una fila de caja, el primer cliente está siendo atendido, y el segundo se estima que se le va a atender en breve. Un empleado es el encargado de ir tomando pedidos a partir de la tercera persona que hay en cada fila de caja. Va conformando el pedido en una PDA según las indicaciones que le va dando el cliente. Una vez terminado el pedido, la PDA devuelve el número de pedido que deberá memorizar el cliente, y manda a las cajas el pedido asociado al número de pedido. Una vez llega el turno del cliente en esa fila de caja, indica al cajero su número de pedido. El cajero lo introduce en la caja y ésta le muestra el pedido a servir. El cajero prepara, entrega el pedido y lo cobra al cliente. De esta manera, se ahorran todos los segundos que inicialmente estaban reservados a la conversación Cliente-Cajero para tomar nota del pedido.

Explicación NUEVO sistema bajo Smartphone

La aplicación en cuestión, ofrece al usuario un sistema para poder hacer un pedido desde su dispositivo móvil, sin necesidad de un empleado, e independientemente de su posición en la fila, de si se encuentra en el local o si está de camino al establecimiento, o en su casa enumerando los productos que desea cada componente

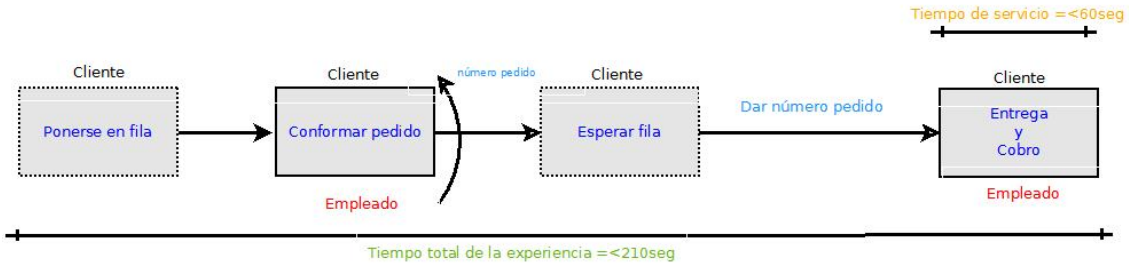
familiar. Con todo el tiempo que necesite, conforma su pedido y queda registrado bajo un número de pedido. Una vez en el restaurante seleccionado, se acerca al mostrador e indicando su número de pedido, el cajero lo introduce en la caja y ésta le muestra lo que el cliente desea. Lo prepara y procede al cobro.

Ventajas

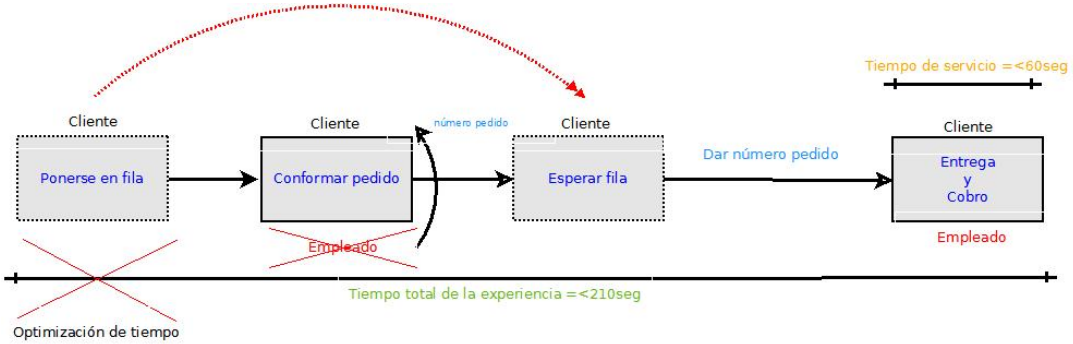
i) No hacen falta empleados dando soporte al cliente para hacer el pedido, lo que supone un ahorro económico.
ii) Potencial ahorro de tiempo en cola para el cliente.
iii) Se puede aplicar tanto en bajo volumen de ventas como en volumen alto de ventas.
iv) Comodidad y entorno de confianza para el cliente.
v) Cualquier cliente con Smartphone tiene acceso al mismo, no se depende de un dispositivo PDA especial.
vi) No requiere la compra o mantenimiento de dispositivos PDA, con la disminución de costes resultante.

3.1.2 Diagramas de flujo de los sistemas

Sistema PDA



Transición a sistema McPhone



Sistema optimizado McPhone

3.2 Alcance

Se implementará una App móvil para iOS que permita agilizar ciertos procesos de la atención a los clientes del restaurante. La aplicación permitirá hacer una gestión de las comandas por parte del cliente en su dispositivo móvil.

Permitirá seleccionar productos del catálogo de McDonald's y reunirlos en un ticket. Soportará la modalidad de McMenú (sándwich, bebida y patatas). Al igual que se podrán añadir el número de productos que se quiera; después habrá posibilidad de eliminar cualquiera de ellos.

La interfaz deberá basarse en imágenes y miniaturas de los productos para ser más atractiva al usuario.

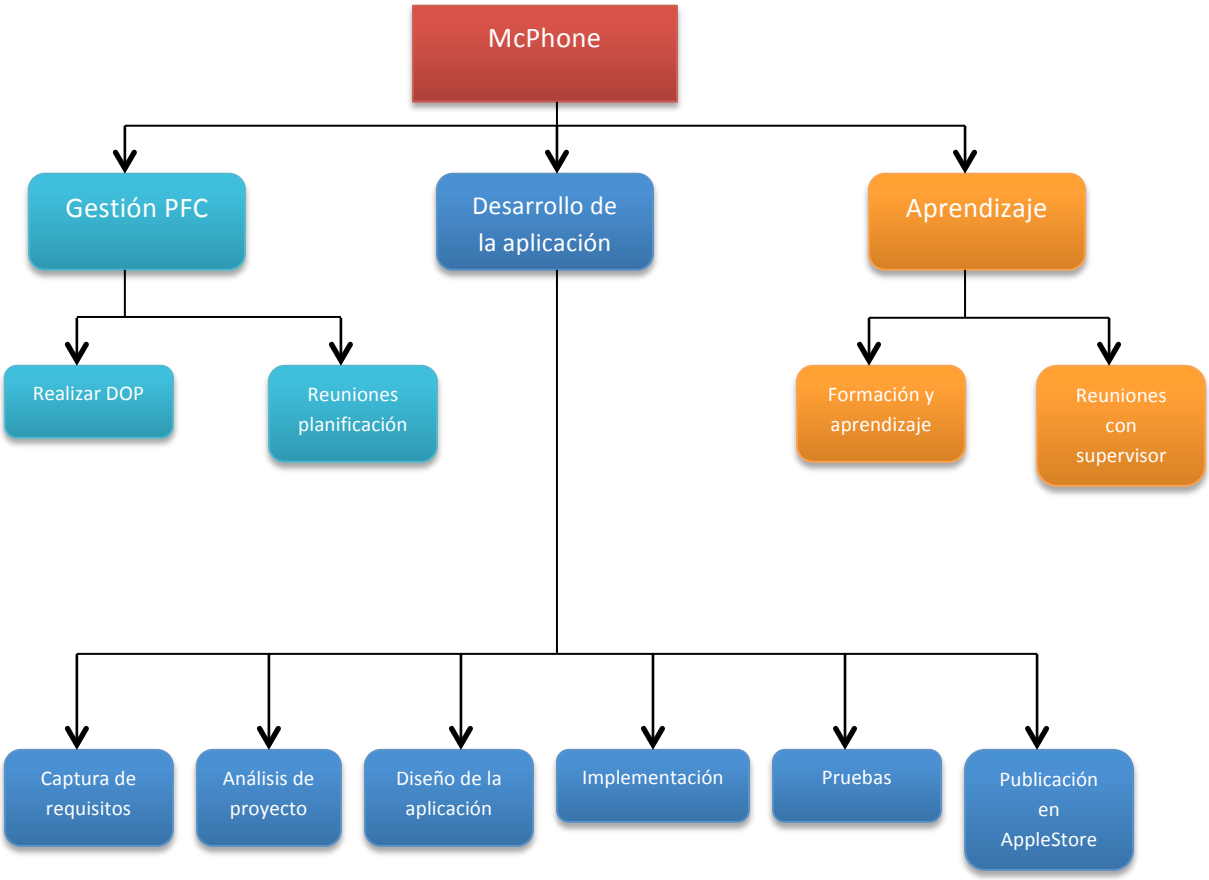
Se podrá realizar un pedido y enviarlo de manera que pueda ser recibido en un restaurante. El restaurante será a elección del cliente. Para el envío se valdrá de algún recurso de almacenamiento en la nube, de tal manera que los pedidos puedan ser descargados a través de internet. La App tendrá un listado de restaurantes disponibles y la información útil relativa a estos, geolocalización inclusive. Permitirá así mismo establecer un restaurante como favorito.

La App tendrá un apartado de consulta de información corporativa, como pueda ser la web de la compañía.

No pertenece a las tareas de este PFC cualquier elemento no mencionado anteriormente, ni la fase de gestión del pedido por parte del restaurante. Se delega en herramientas de terceros partes de la aplicación, por lo que en ellos recae la responsabilidad del buen funcionamiento de las mismas.

3.3 Diagrama de descomposición del trabajo

Estructura jerárquica de la descomposición de trabajo.



3.4 Responsabilidades

A continuación vamos a describir las responsabilidades que deben cumplir cada uno de los implicados en el Proyecto Fin de Carrera, como se describe en el convenio para el desarrollo de cooperación educativa de la universidad.

- **Máximo responsable**

  Iker Beristain Villar.

  Será máximo responsable de la realización y éxito del PFC. Se le atribuirán todas las responsabilidades de este: dirección del proyecto, análisis de requisitos, diseño y desarrollo de la solución, etc.

  Será máximo responsable de que las tareas se lleven a cabo en los plazos y alcance programado.

  Deberá informar de la marcha del Proyecto Fin de Carrera al supervisor del mismo, con la periodicidad entre ellos acordada.

- **Supervisor PFC**

  Jose Miguel Blanco.

  Supervisará el Proyecto Fin de Carrera con la periodicidad acordada entre los implicados.

3.5 Lista de tareas

A continuación la lista de tareas y subtareas realizadas durante el transcurso del proyecto.

1. Gestión del proyecto:

- Realizar el DOP:
  - Primera versión.
  - Modificaciones.
- Reuniones planificación.

2. Desarrollo de la aplicación:

- Captura de requisitos.
- Análisis de proyecto.
- Diseño de la Aplicación
- Implementación.
- Pruebas.
- Publicación en la AppStore.

3. Aprendizaje:

- Reuniones con el supervisor.
- Formación y aprendizaje.

4. ANÁLISIS DE REQUISITOS

En este capítulo se dará una visión sobre los diferentes requisitos que se han tenido en cuenta en el diseño de la aplicación.

4.1 Requisitos Funcionales

La App...

- Muestra el catálogo de productos disponibles y su descripción.

- Permite seleccionar productos, así como la cantidad a añadir.

- Muestra el ticket y puede ser consultado en cualquier momento por el usuario, así como la suma del precio total.

- Se pueden eliminar productos tras haber sido añadidos.

- Permite recuperar el último pedido realizado, aun habiendo cerrado la App.

- Permite hacer pedidos a diferentes restaurantes de la cadena de restauración.

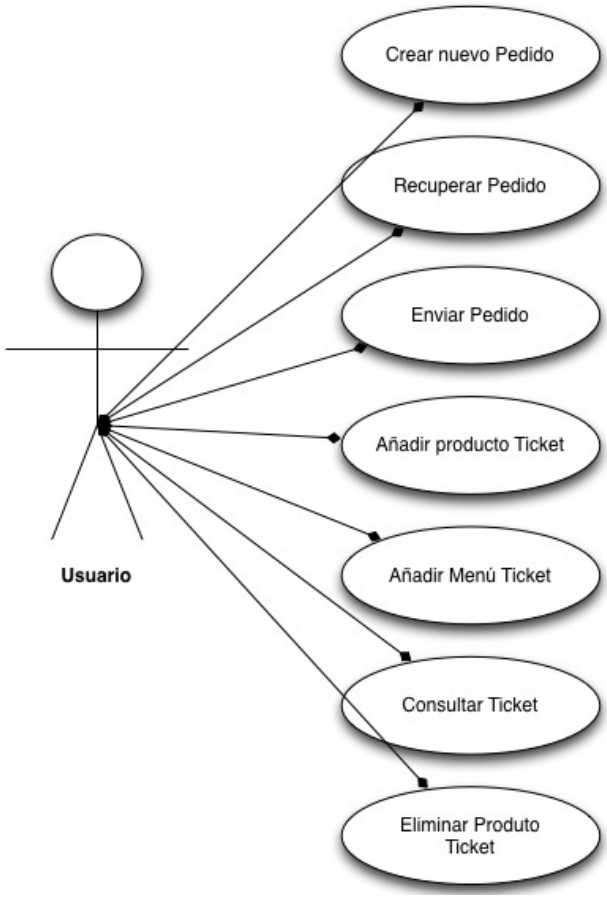
- Permite consultar información sobre los restaurantes, así como sus ubicaciones y la ubicación del usuario.

-Permite seleccionar un restaurante como favorito.

4.1.1 Casos de Uso

Se han ordenado agrupándolos en tres grupos. Gestión de pedidos, consulta restaurantes y acceso a la web de la compañía.

4.1.1.1 Casos de Uso - Gestión de pedidos



Nombre: Crear nuevo pedido

Actores: Usuario

Descripción:
- El usuario pulsa en el Botón de la pantalla principal "Nuevo Pedido".
- El sistema crea un nuevo ticket vacío para poder iniciar un pedido desde el comienzo.

Nombre: Recuperar Pedido

Actores: Usuario

Descripción:
- El usuario pulsa en el Botón de la pantalla principal "Recuperar Pedido".
- El sistema carga el último pedido realizado por el usuario.

Nombre: Enviar Pedido

Actores: Usuario

Descripción:
- Al usuario tras conformar el ticket, se le muestran en pantalla los restaurantes disponibles para el envío y un TextField para un comentario.
- Los rellena y presiona en "Enviar".
- El sistema crea un pedido en formato JSON con: NombreRestaurante, FechaYHora, NombreDispositivo, CodigoUnicoDispositivo, numeroPedido, comentario, productosTicket.
- Se envía el JSON a GoogleDrive y se queda almacenado.

Nombre: Añadir producto ticket

Actores: Usuario

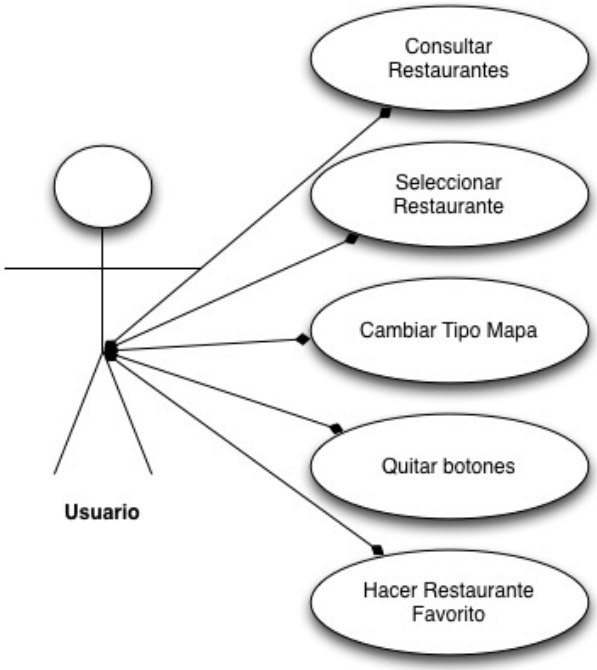
Descripción:
- El sistema muestra por pantalla el catálogo de productos disponibles por categorías: Sándwiches, Complementos, 1X1 PLUS, Bebidas, Postres y Helados.
- El usuario selecciona una categoría y se le muestran los productos dentro de la misma.
- El usuario selecciona el producto deseado.
- El sistema le devuelve por pantalla la descripción y un cuantificador indicando el número de ejemplares que desea añadir.
- El Usuario pulsa en "Añadir al Pedido".
- El sistema añade el producto al ticket. Muestra por pantalla " ¡Producto añadido!".

Nombre: Añadir menú ticket
Actores: Usuario
Descripción:
- El sistema muestra por pantalla los Sándwiches disponibles en modalidad de menú.
- El usuario pulsa sobre el sándwich deseado.
- El sistema registra en el menú actual el sándwich escogido y muestra por pantalla las bebidas disponibles en los menús.
- El usuario selecciona la bebida deseada.
- El sistema registra en el menú actual la bebida escogida, y muestra por pantalla un selector de patatas, fritas o deluxe, y un activador para hacer el menú grande.
- El usuario hace las selecciones oportunas y presiona sobre "Añadir al Pedido".
- El sistema registra el menú completo y lo añade al ticket. Muestra por pantalla "¡Producto añadido!".

Nombre: Consultar ticket
Actores: Usuario
Descripción:
- El usuario presiona sobre la pestaña "Tu Ticket".
- El sistema muestra una lista de todos los productos añadidos al ticket hasta el momento, con el precio desglosado por productos. También muestra el precio total del ticket.

Nombre: Eliminar producto ticket
Actores: Usuario
Descripción:
- El sistema muestra el ticket.
- El usuario desliza el dedo lateralmente sobre un producto o menú.
- El sistema muestra el botón "Eliminar" sobre el producto seleccionado.
- El usuario presiona sobre el botón "Eliminar".
- El sistema elimina del ticket el producto o menú seleccionado y actualiza en tiempo real el ticket mostrado en pantalla.

4.1.1.2 Casos de Uso - Consulta Restaurantes



Nombre: Consultar restaurantes
Actores: Usuario
Descripción:
- El usuario presiona sobre el botón "Restaurantes" en el menú principal de la App.
- El sistema muestra una lista de los restaurantes disponibles.

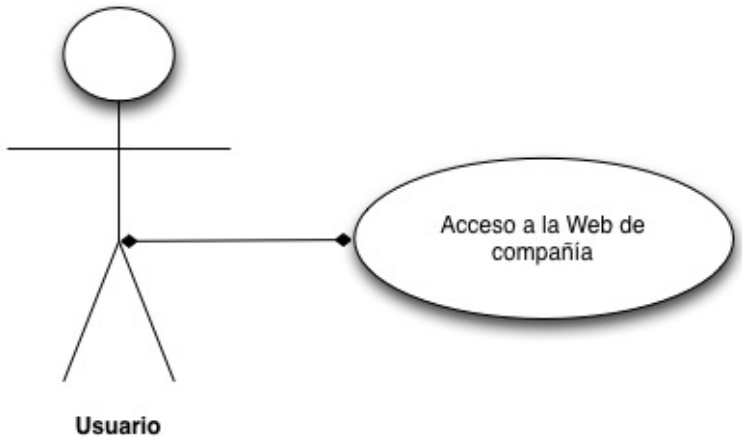
Nombre: Seleccionar restaurantes
Actores: Usuario
Descripción:
- El sistema muestra una lista de los restaurantes disponibles.
- El usuario selecciona uno para ampliar la información.
- El sistema muestra la información relativa al restaurante:
    - Nombre restaurante.
    - Dirección restaurante.
    - Teléfono restaurante.
    - Mapa con la ubicación del restaurante y ubicación relativa del usuario respecto a éste.

Nombre: Cambiar tipo mapa
Actores: Usuario
Descripción:
- El sistema muestra en mapa la ubicación del restaurante.
- El usuario presiona sobre "Cambiar Mapa".
- El sistema alterna entre los diferentes tipos de vistas del mapa en la siguiente secuencia: Mapa, satélite, híbrido.

Nombre: Quitar botones
Actores: Usuario
Descripción:
- El sistema muestra en mapa la ubicación del restaurante.
- El usuario presiona sobre "Quitar botones".
- El sistema limpia la interfaz y despeja la pantalla de los botones superpuestos sobre la vista del mapa. Si ya se había presionado antes, revierte el proceso.

Nombre: Hacer restaurante favorito
Actores: Usuario
Descripción:
- El sistema muestra en mapa la ubicación del restaurante.
- El usuario presiona sobre el botón "Hacer Favorito".
- El sistema registra como restaurante favorito del usuario el que actualmente esté en pantalla.

4.1.1.3 Casos de Uso - Acceso a la web de la compañía



Nombre: Acceder web compañía
Actores: Usuario
Descripción:
- El usuario presiona sobre el botón de información "i" en el menú principal de la App.
- El sistema pliega la vista y muestra un explorador de internet accediendo a la dirección de la compañía: http://www.mcdonalds.es/

4.2 Requisitos Tecnológicos

iPhone de Apple (véase figura 4.1)

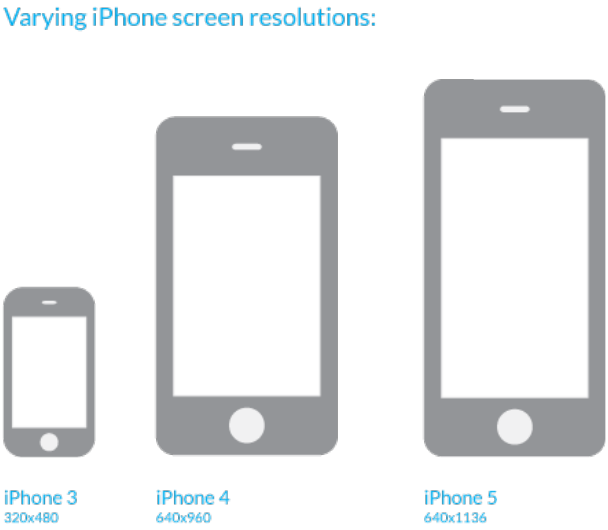
figura 4.1. Imágen del iPhone5 de Apple

Un dispositivo móvil modelo iPhone de la marca Apple, en cualquiera de sus versiones actualmente disponibles (Junio 2013: Iphone3GS, Iphone4, Iphone4S y Iphone5).

La aplicación se adapta a cualquiera de las tres resoluciones existentes (estándar de 320x480, retina de 3,5" con 640x960, retina de 4" con 640x1136) (véase figura 4.2)

.

figura 4.2. Resoluciones soportadas



Funciona de manera fluida con cualquiera de las CPUs ARM que montan los diferentes modelos, ya que no consume recursos de cálculo excesivos.

En cuanto a las diferentes capacidades de RAM, incluso en la configuración más baja de 256MB rinde bien, incluso utilizando la multitarea. (véase figura 4.3)

Son cuatro las configuraciones de hardware que soporta la aplicación:

figura 4.3. Configuraciones de hardware

iPhone 3GS	iPhone 4	iPhone 4S	iPhone 5
Actual[13]	Actual	Actual	Actual
iPhone OS 3.0[16]	iOS 4.0[17]	iOS 5.0[18]	iOS 6.0[19]
iOS 6.1.3[22]			iOS 6.1.4[23]
89 mm (3,5") cristal LCD[24]			102 mm (4,0")
480 × 320 Píxeles	960 × 640 px a 326 ppp		1136 x 640 px a 326 ppp
8, 16 y 32 GB		16, 32 y 64 GB	16, 32 y 64 GB
PowerVR SGX 535		PowerVR SGX 543MP2	PowerVR SGX 543MP3
600 MHz ARM Cortex-A8[26] [27]	Apple A4 800 MHz[28] ARM Cortex-A8[29]	Apple A5 800 MHz ARM Cortex-A9[30] doble núcleo	Apple A6 1,3 GHz ARM Cortex-A15 doble núcleo
256 MB DRAM[26] [27]	512 MB DRAM[32]		1024 MB DRAM

iOS

Se requiere iOS 5.1 o superior. Esta restricción viene dada por la utilización del diseño basado en *StoryBoard*s. iOS es un sistema operativo desarrollado por Apple Inc. para los dispositivos móviles iPod touch, iPhone e iPad. Está basado en una variante del Mach kernel de Mac OS X.

figura 4.4. Actualización de iOS 5.1

Conexión a internet

La App usa la conectividad a través de internet exclusivamente para la transmisión de datos. Así es posible el almacenamiento y la trasmisión del pedido. El resto de carga de datos, menús, precios y etc se hace en tiempo de compilación y tiempo ejecución, consumiendo siempre recursos y datos internos, es decir, instalados con el programa.

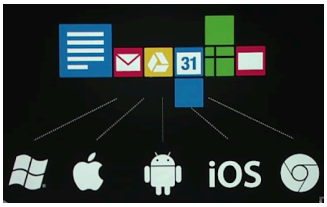
GoogleDrive

Cuenta en GoogleDrive, para poder almacenar los pedidos realizados en la nube a través del sistema ofrecido gratuitamente por Google Inc.

GoogleDrive es un servicio de almacenamiento de archivos en línea que está operativo desde el 24 de abril de 2012.

Plataforma con acceso a internet

figura 4.5. Plataformas disponibles

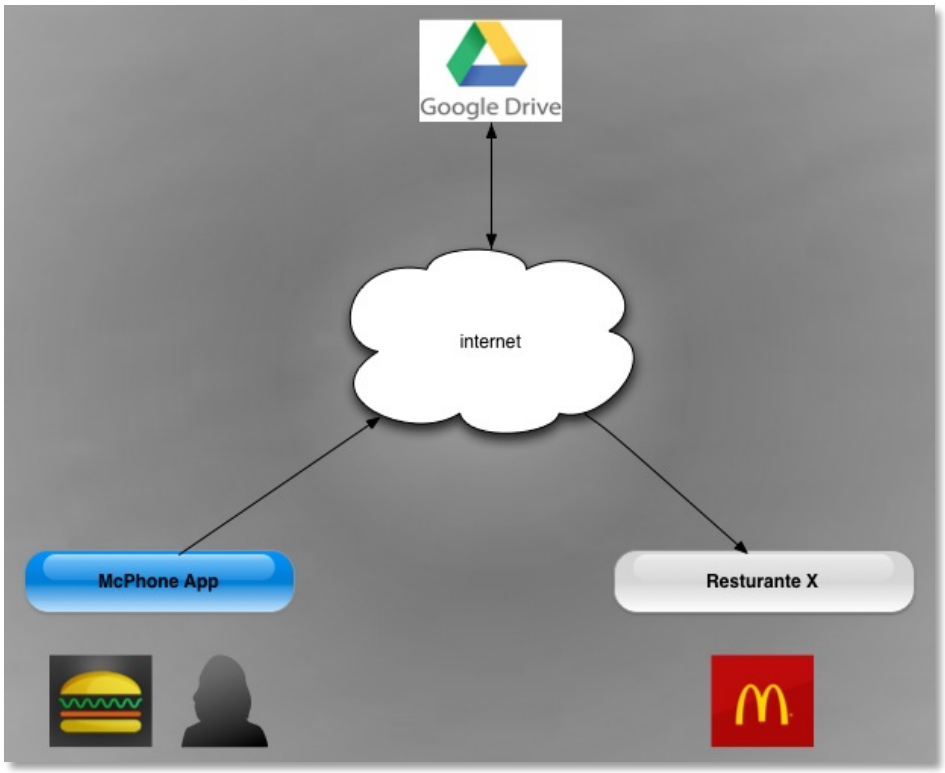


Para la recepción de los pedidos, se requiere de un dispositivo que tenga acceso a internet para conectarse a GoogleDrive vía web. Además, GoogleDrive está disponible para la mayoría de sistemas operativos con aplicación exclusiva. Por ejemplo: Windows, iOS, Max OSX, Android y Chrome. (véase figura 4.5)

Para la consulta de los pedidos desde el restaurante, se podría considerar válida cualquier plataforma con acceso a internet.

5. DISEÑO DE LA ARQUITECTURA DE LA SOLUCIÓN

figura 5.1 Diseño de la arquitectura



Se ha desarrollado una solución directa que nos permite establecer la relación Cliente-restaurante con internet como intermediario. (véase figura 5.1)

Expliquémoslo de la siguiente manera; a grandes rasgos, el cliente quiere transmitir un mensaje al restaurante. Consideremos una comunicación básica de 3 pilares:

1) **Emisor:** McPhone.

2) **Medio divulgador:** Internet y GoogleDrive.

3) **Receptor:** PlataformaX con acceso a GoogleDrive.

El cliente con la app McPhone carga el ticket mediante conectividad a internet móvil en GoogleDrive. Después el RestauranteX consulta los pedidos que se le han realizado y los descarga desde el mismo momento en el que el cliente los pone a su disposición en cualquier tipo de plataforma estandarizada a día de hoy. Es decir, se utiliza la herramienta GoogleDrive como intermediario donde poder cargar y descargar los pedidos.

6. DISEÑO DE LA APLICACIÓN

Se ha hecho un diseño de la aplicación estructurado según diferentes aspectos. Trataremos entonces, sobre las decisiones tomadas en la capa de interacción del usuario, en la estructura de datos y en la lógica del negocio.

6.1 Capa interacción del usuario

figura 6.1. Pantalla principal McPhone

Se ha optado por una implementación de representación gráfica de alto nivel lo más directa e intuitiva posible. Un capa de presentación de interacción táctil basada en imágenes y textos de apoyo. La navegación está diseñada como un flujo de ventanas deslizantes, y en la agrupación de elementos se ha optado por tablas de única columna.

La interacción del usuario se ha diseñado en tres vertientes.

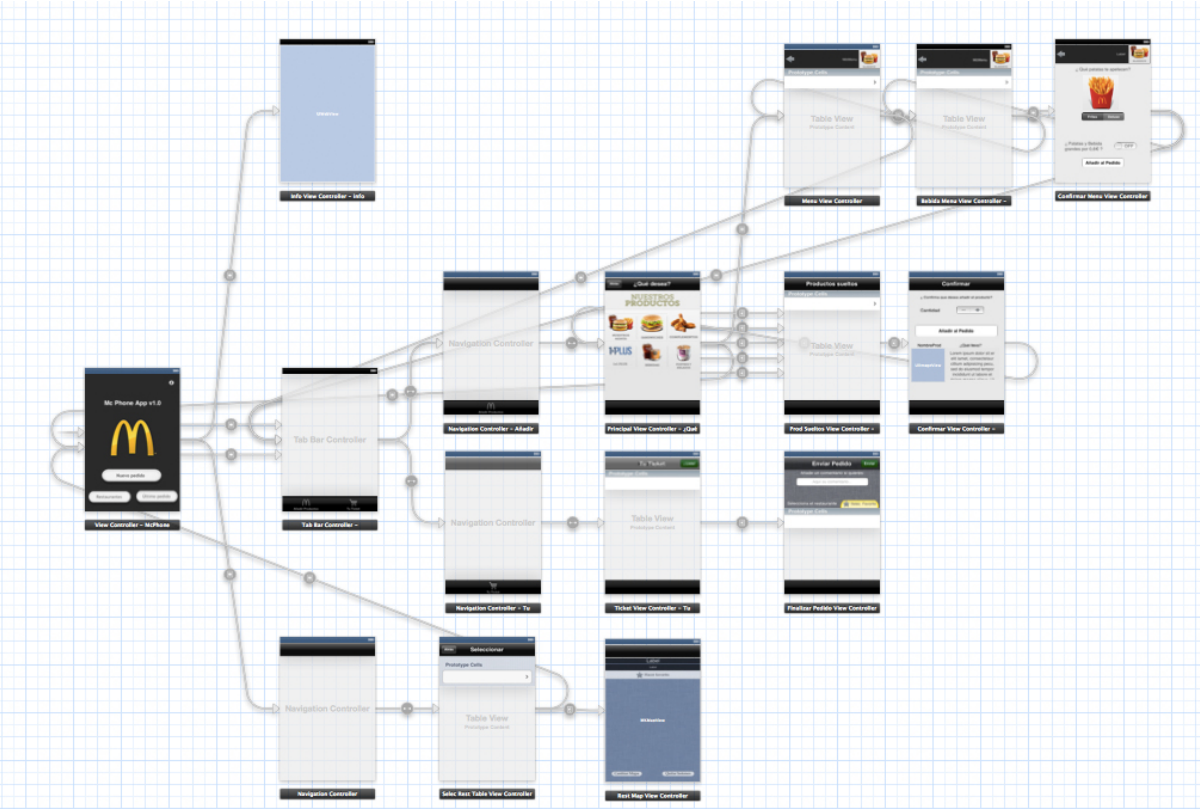
- Navegación y selección de productos.

- Navegación y selección restaurantes.

- Acceso a la web de la compañía.

Se ha diseñado un StoryBoard *(véase figura 6.2)* para la visión general de la aplicación McPhone.

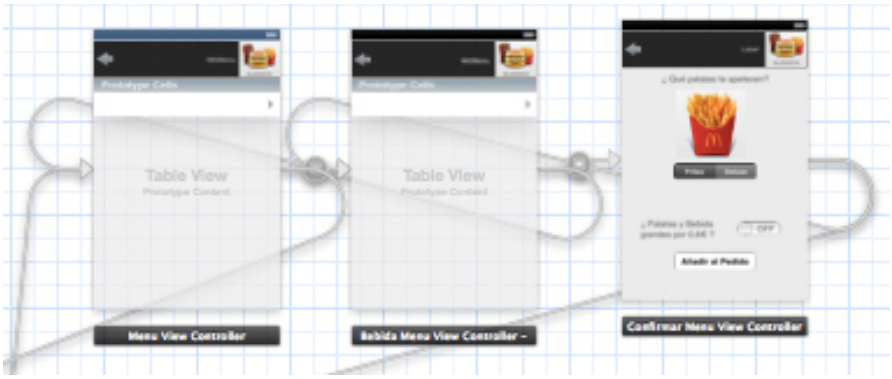
figura 6.2 StoryBoard de McPhone



A continuación, analizamos con más detenimientos las decisiones tomadas en el diseño de cada apartado de la aplicación y de la capa de interacción del usuario.

6.1.1 Capa interacción del usuario - Navegación y selección productos

Se ha elaborado una selección de menús en tres pasos, optimizando al máximo el número de ventanas a recorrer por el usuario. 1) Sándwich 2) Bebida 3) Patatas y tamaño.



Teniendo en cuenta el target destino, es decir, un cliente, se han tenido en cuenta ciertos puntos que benefician su experiencia:

- Ticket siempre disponible y a un click. **(véase figura 6.3)**

- Poder añadir productos y su cuantía. **(véase figura 6.4)**

figura 6.3 Imagen del ticket en McPhone

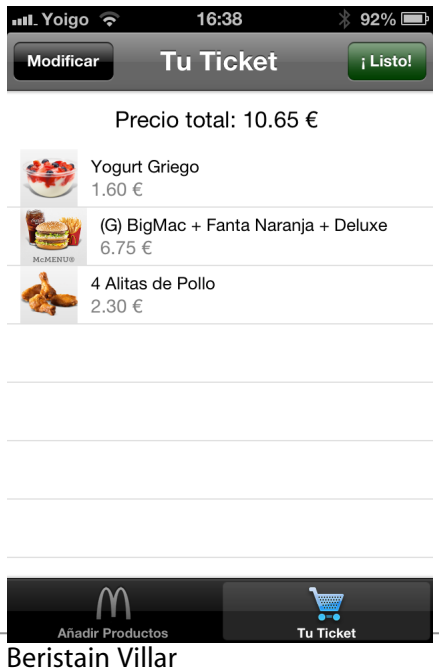


figura 6.4 Cuantificador del producto

Se han implementado y respetado en las tablas representadas, los gestos considerados estándares en plataformas de navegación táctil:

-Botón "edit" para la selección de la celda a eliminar . **(véase figura 6.5)**

-Deslizar lateralmente el dedo sobre una celda para eliminarla . **(véase figura 6.6)**

-Flujo coherente. Botones de retroceso a la izquierda y botones avance a la derecha.

figura 6.5 Método de eliminar 1

figura 6.6 Método de eliminar 2

Para facilitar la selección de productos, se han agrupado por grupos de manera lógica y en cohesión a cómo lo hace la cadena de restauración. **(véase figura 6.7)**

-**Nuestros Menús**.

En este apartado la App es capaz de conformar un menú compuesto por los 3 productos que ofrece la compañía. Sándwich, bebida y patatas.

-**Sándwiches**.

Se muestran los sándwiches disponibles.

-**Complementos**.

Se muestran los complementos disponibles.

-**1x1Plus**.

Se muestran los productos que a pesar de catalogarse en otras áreas, están en promoción y directamente salen en este apartado.

-**Bebidas**.

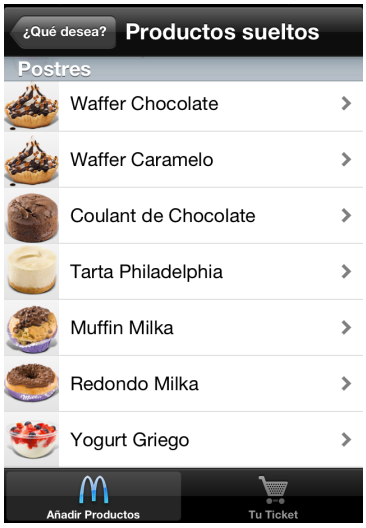
Se muestran las bebidas disponibles.

-**Postres y Helados**. **(véase figura 6.8)**

Se muestran los postres y helados disponibles.

figura 6.7 Selección de productos

figura 6.8 Postres y helados



En la pantalla en la que se formaliza el pedido (véase figura 6.9) y se procede a enviar (véase figuras 6.10 y 6.11), se ha añadido la selección del restaurante a la que se desea realizar el pedido. También se ha añadido un botón para "seleccionar favorito" que auto-selecciona directamente el que hubiera sido previamente seleccionado como tal por el cliente.

Se ha añadido un campo de texto "comentario" por si el cliente quisiera transmitir algún mensaje al restaurante, que puede ser rellenado, o no.

figura 6.9 Pantalla de envío de pedidos

figura 6.10 Enviando pedido

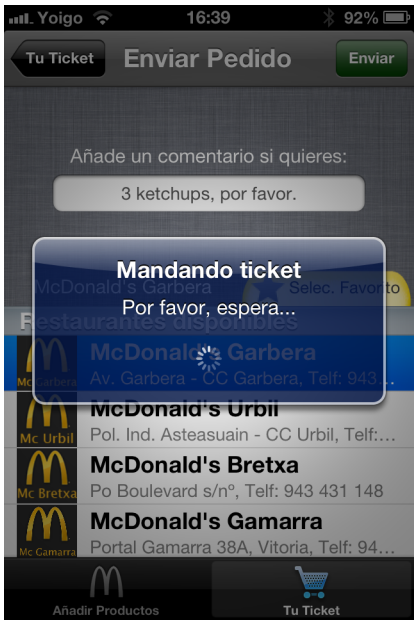


figura 6.11 Pedido correctamente enviado

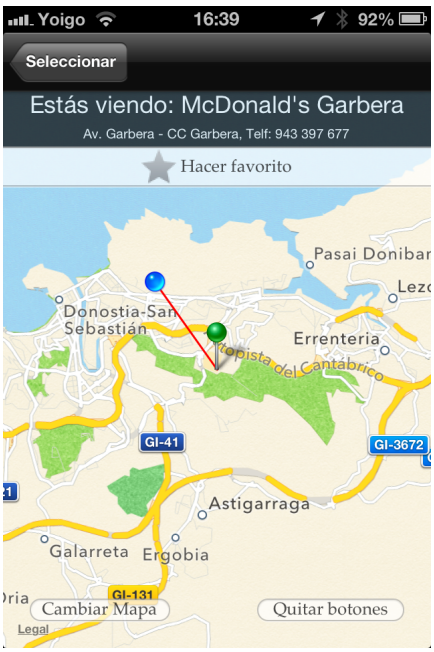
6.1.2 Capa interacción del usuario - Navegación y selección restaurantes

Se muestran listados los restaurantes disponibles.

figura 6.12 Selección de restaurantes

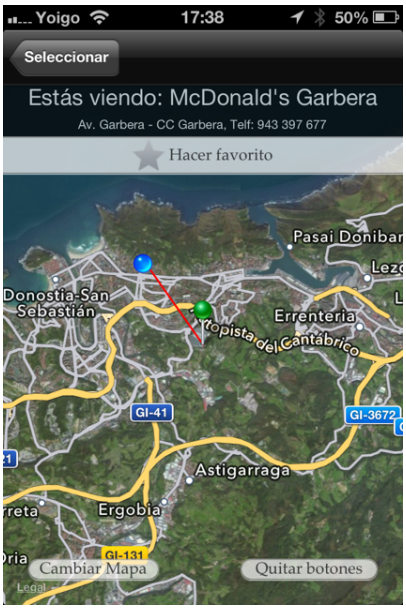
Seleccionando uno, accedemos a toda la información de utilidad relativa al establecimiento, así como nuestra posición respecto a él (véase figura 6.13). Se muestran: Nombre, dirección, teléfono y ubicación en el mapa.

figura 6.13 Vista del restaurante de CC..Garbera



También se ha añadido la posibilidad de cambiar las vistas a gusto del usuario: Mapa, satélite e híbrido. **(véase figura 6.14)**

figura 6.14 Vista híbrida del mapa



Hay disponible un botón para "hacer favorito", con un simple click, el restaurante que actualmente tenemos en pantalla. Esto nos servirá para elegir nuestro restaurante habitual directamente cuando estemos conformando un pedido.

figura 6.15 Garbera seleccionado como favorito

6.1.3 Capa interacción del usuario - Acceso a la web de la compañía

Se ha puesto un acceso mediante la App a la página web oficial de la cadena de restauración. Puede ser consultada sin necesidad de salir de la aplicación. Saldrá un pequeño pliegue de hoja en la parte superior que podemos presionar para volver a McPhone. **(véase figura 6.16)**

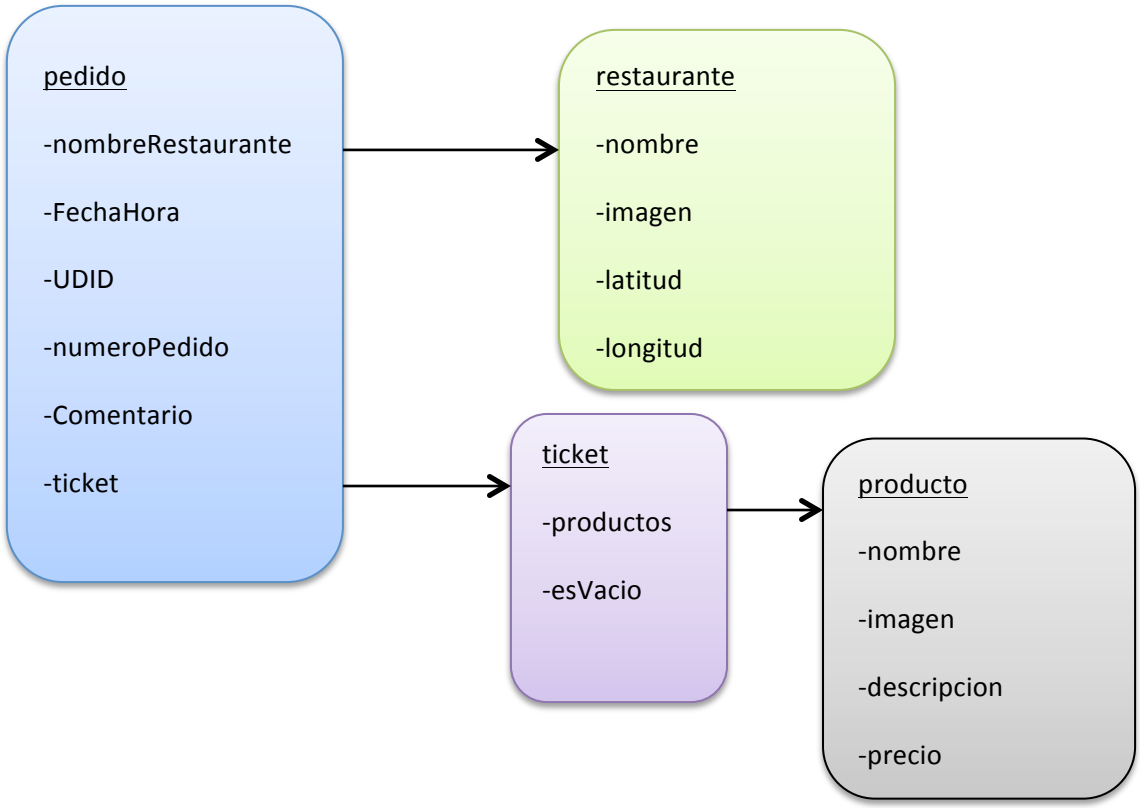
figura 6.16 Acceso a la página web a través de McPhone

6.2 Estructura de datos

Se ha optado por una estructura de datos lo más funcional posible dentro de las necesidades de la aplicación. Utilizando objetos, MutableArrays, estructuras JSON, etc. Cabe destacar que existen varias alternativas a la solución planteada, pero se ha procurado utilizar aquella que se adapta mejor a la naturaleza de la aplicación y sus necesidades. A continuación profundizaremos en la aplicación que se le ha dado a cada estructura principal utilizada.

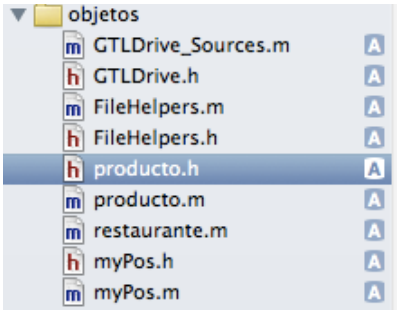
6.2.1 Estructura de datos - Objetos

Representación gráfica de los objetos principales y sus atributos:



Las estructuras de datos principalmente utilizadas para los productos del catálogo son MutableArrays de objetos, es decir, arrays dinámicos en Objective-C. Las clases de objetos `NSObject` (véase figura 6.17) han sido definidos cumpliendo las condiciones de codificación y decodificación propias de `NSCoding` en Objective-C, para poder almacenarlos y recuperarlos. Así son almacenados y recuperados de manera estándar con `NSKeyedArchiver` y `NSKeyedUnarchiver` en la memoria del dispositivo.

figura 6.17 Captura de algunas clases de objetos creados durante la implementación



Se ha creado una clase producto.h/producto.m que sirve como objeto base para almacenar cualquier tipo de producto del restaurante.

Los atributos de la clase *producto* son: Nombre, imagen, descripción y precio.

```
//  producto.h
//  Mcphone2
//
//  Created by Iker Beristain Villar on 25/03/13.
//  Copyright (c) 2013 Iker Beristain Villar. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface producto : NSObject{
    NSString *nombre;
    NSString *imagen;
    NSString *descripcion;
    double precio;

}
@property (nonatomic, copy) NSString *nombre;
@property(nonatomic, copy) NSString *imagen;
@property (nonatomic, copy) NSString *descripcion;
@property (nonatomic, assign) double precio;


-(id) initWithName:(NSString*)n imagen:(NSString*)img descripcion:(NSString*)dsc precio:(double)prc ;
@end
```

Y su consiguiente implementación producto.m, con los patrones de codificación y decodificación para poder ser archivados como estructuras estándar de iOS.

```
//  producto.m
//  Mcphone2
//
//  Created by Iker Beristain Villar on 25/03/13.
//  Copyright (c) 2013 Iker Beristain Villar. All rights reserved.
//

#import "producto.h"

@implementation producto
@synthesize nombre, imagen, descripcion, precio;
-(id) initWithName:(NSString*)n imagen:(NSString*)img descripcion:(NSString*)dsc precio:(double)prc{
    self.nombre=n;
    self.imagen=img;
    self.descripcion=dsc;
    self.precio=prc;
    return self;
}

#define nombreKey @"nombreKey"
#define imagenKey @"imagenKey"
#define descKey @"descKey"
#define precioKey @"precioKey"


- (id)initWithCoder:(NSCoder *)decoder {
    self = [super init];
    if(self) {
        self.nombre = [decoder decodeObjectForKey:nombreKey];
        self.imagen = [decoder decodeObjectForKey:imagenKey];
        self.descripcion = [decoder decodeObjectForKey:descKey];

        //latitud & longitud are a double to we need to do this
        self.precio = [decoder decodeDoubleForKey:precioKey];
    }
    return self;
}

- (void)encodeWithCoder:(NSCoder *)encoder {
    [encoder encodeObject:self.nombre forKey:nombreKey];
    [encoder encodeObject:self.imagen forKey:imagenKey];
    [encoder encodeObject:self.descripcion forKey:descKey];

    //dereferance the pointer to persist the value
    [encoder encodeDouble:self.precio forKey:precioKey];
}

@end
```

También ha sido necesario crear una clase de dato *restaurante* para poder gestionar la información relacionada con estos.

Los atributos de la clase *restaurante* son: Nombre, imagen, dirección, latitud y longitud.

```
//  restaurante.h
//  Mcphone2
//
//  Created by Iker Beristain Villar on 19/03/13.
//  Copyright (c) 2013 Iker Beristain Villar. All rights reserved.
//


@interface restaurante : NSObject<NSCoding>{

    NSString *nombre;
    NSString *imagen;
    NSString *direc;
    double latitud;
    double longitud;

   }

@property (nonatomic, copy) NSString *nombre;
@property(nonatomic, copy) NSString *imagen;
@property (nonatomic, copy) NSString *direc;
@property (nonatomic, assign) double latitud;
@property (nonatomic, assign) double longitud;

-(id) initWithName:(NSString*)n imagen:(NSString*)img direc:(NSString*)drc latitud:(double)lat longitud:(double)lng;


@end
```

Y su consiguiente implementación restaurante.m, con los patrones de codificación y decodificación para poder ser archivados como estructuras estándar de iOS.

```
//  restaurante.m
//  Mcphone2
//
//  Created by Iker Beristain Villar on 19/03/13.
//  Copyright (c) 2013 Iker Beristain Villar. All rights reserved.
//

#import "restaurante.h"

@implementation restaurante
@synthesize  nombre, imagen, direc, latitud,longitud;

-(id) initWithName:(NSString *)n imagen:(NSString*)img direc:(NSString*)drc latitud:(double)lat longitud:(double)lng{
    self.nombre=n;
    self.imagen=img;
    self.direc=drc;
    self.latitud=lat;
    self.longitud=lng;
    return self;
}

#define nombreKey @"nombreKey"
#define imagenKey @"imagenKey"
#define direcKey @"direcKey"
#define latitudKey @"latitudKey"
#define longitudKey @"longitudKey"


- (id)initWithCoder:(NSCoder *)decoder {
    self = [super init];
    if(self) {
        self.nombre = [decoder decodeObjectForKey:nombreKey];
        self.imagen = [decoder decodeObjectForKey:imagenKey];
        self.direc = [decoder decodeObjectForKey:direcKey];

        //latitud & longitud are a double to we need to do this
        self.latitud = [decoder decodeDoubleForKey:latitudKey];
        self.longitud= [decoder decodeDoubleForKey:longitudKey];
    }
    return self;
}

- (void)encodeWithCoder:(NSCoder *)encoder {
    [encoder encodeObject:self.nombre forKey:nombreKey];
    [encoder encodeObject:self.imagen forKey:imagenKey];
    [encoder encodeObject:self.direc forKey:direcKey];

    //dereferance the pointer to persist the value
    [encoder encodeDouble:self.latitud forKey:latitudKey];
    [encoder encodeDouble:self.longitud forKey:longitudKey];
}


@end
```

6.2.2 Estructura de datos - Carga y Almacenamiento

Para el almacenamiento, considerando un catálogo de productos fijo o poco cambiante en el restaurante, se ha optado por cargar los datos en tiempo de ejecución, pudiéndose cambiar las estructuras con actualizaciones de la aplicación. Las actualizaciones tienen que ser aceptadas y aprobadas por Apple. Se requeriría un envío con una semana de antelación para que la actualización estuviese disponible. Se ha considerado asumible, en pro de que la aplicación dependa de datos internos y no consuma más datos de recursos en la nube. Por ello la carga de datos inicial se hace al arrancar la App, de manera desatendida.

Ejemplo de carga de las bebidas:

a) Se crean las bebidas , de la clase *producto*, independientemente.

b) Se añaden a un array de `BebidasMenu`.

c) Se almacenan mediante `NSKeyedArchiver` en la memoria interna del dispositivo en un archivo llamado "arrayBebidasMenuGuardado".

```
[NSKeyedArchiver archiveRootObject: arrayBebidasMenu  toFile:
   pathInDocumentDirectory (@"arrayBebidasMenuGuardado")];
```

Aquí se expone un ejemplo de la carga del catálogo inicial.

```
//BEBIDASMENU ------  Creamos la lista de Bebidas
producto *nuevoBebidaMenu0= [[producto alloc]initWithName:@"Coca Cola"
                                        imagen:@"CocaCola.png"
                                    descripcion:@"Refresco de Coca Cola"
                                        precio:1.80];

producto *nuevoBebidaMenu1= [[producto alloc]initWithName:@"Fanta Naranja"
                                        imagen:@"FantaNaranja.png"
                                    descripcion:@"Refresco de Fanta Naranja"
                                        precio:1.80];
producto *nuevoBebidaMenu2= [[producto alloc]initWithName:@"Lipton Ice"
                                        imagen:@"IceTea.png"
                                    descripcion:@"Refresco de té helado"
                                        precio:1.80];
producto *nuevoBebidaMenu3= [[producto alloc]initWithName:@"Agua"
                                        imagen:@"Agua.png"
                                    descripcion:@"Agua natural"
                                        precio:1.2];
producto *nuevoBebidaMenu4= [[producto alloc]initWithName:@"Cerveza"
                                        imagen:@"Cerveza.png"
                                    descripcion:@"Cerveza Mahou"
                                        precio:2.2];

NSMutableArray *arrayBebidasMenu=[[NSMutableArray alloc] initWithCapacity:1];
[arrayBebidasMenu addObject:nuevoBebidaMenu0];
[arrayBebidasMenu addObject:nuevoBebidaMenu1];
[arrayBebidasMenu addObject:nuevoBebidaMenu2];
[arrayBebidasMenu addObject:nuevoBebidaMenu3];
[arrayBebidasMenu addObject:nuevoBebidaMenu4];


// EL NSMutableArray arrayBebidas está creado y contiene las Bebidas.
[NSKeyedArchiver archiveRootObject:arrayBebidasMenu toFile:pathInDocumentDirectory(@"arrayBebidasMenuGuardado")];
```

En el caso de necesitar cargar una estructura previamente almacenada, recurrimos al mismo sistema en sentido inverso.

Por ejemplo, a continuación se expone como sería la carga de la estructura de las bebidas desde el archivo "arrayBebidasGuardado", ya que la clase *producto* está adaptada a `NSCoding`.

```
arrayBebidas=[NSKeyedUnarchiver
unarchiveObjectWithFile:pathInDocumentDirectory(@"arrayBebidasGuardado")];
```

6.2.3 Estructura de datos - El pedido y JSON

Los productos seleccionados se almacenan en una estructura *ticket,* con los objetos tipo *producto*, que representan el pedido.

Es recomendable que los pedidos una vez conformados, y previa salida a la nube, se encuentren en un formato estándar. Por lo que son adaptados al formato ligero para el intercambio de datos JavaScript Object Notation (JSON). (véase figura 6.18)

Se adaptar los datos salientes a JSON desde un comienzo, con vista a una posible actualización en el sistema de recepción de los pedidos con una base de datos. Esta decisión, aunque no fuera completamente necesaria, pretende evitar doblar el trabajo en una futura posible actualización del sistema de recepción.

Además, en la estructura JSON, se añaden datos imprescindibles para la funcionalidad de la aplicación que se explican a continuación.

Los datos se listan dentro del JSON del pedido que después es enviado vía internet.

figura 6.18 Ejemplo de un pedido estructurado en formato JSON.

```
[
  a"McDonald's Garbera",
  b"30-04-2013 20:47:21 ",
  c"iPhone4 de Beris",
  d"975319E0-43ED-435F-AA81-0B4F051B3768",
  e"9",
  f"3 ketchups, por favor.",
  g[
    [
      "(G) BigMac + Lipton Ice + Deluxe",
      "6.75"
    ],
    [
      "Espresso",
      "1.20"
    ]
  ]
]
```

a) **Restaurante:** El restaurante al que va dirigido el pedido. Imprescindible para poder filtrar los pedidos por restaurantes.

b) **Fecha y hora:** Fecha y hora en la que se envía el pedido. Útil para poder ordenar cronológicamente los pedidos. Dato prescindible mientras nos apoyemos en GoogleDrive, porque ésta herramienta registra fecha y hora de envío, pero útil para un mejor control y futuras actualizaciones.

c y d) **Nombre dispositivo y código único:** Estos dos datos corresponden al nombre del dispositivo y un código único que genera la aplicación por cada par teléfono-aplicación. Se crea mediante el código UDID. La funcionalidad de este código es poder evitar pedidos fraudulentos, es decir, un cliente realiza un pedido y luego no va a recogerlo, podría ser incluido en blacklist y filtrados sus próximos pedidos.

e) **Número pedido:** Número de pedido en el dispositivo.

f) **Comentario:** El espacio del ticket reservado para que el cliente pueda transmitir un mensaje al restaurante (Mensaje limitado a 25 caracteres).

g) **Lista de productos solicitados:** Lista completa de todos los productos que se están solicitando por parte del cliente.

6.3 Lógica de negocio

A continuación se exponen los diagramas más relevantes de la aplicación:

Nombre: enviarPedido

Responsabilidades: El sistema debe crear un pedido a partir del restaurante seleccionado, comentario y ticket conformado por el usuario. Después, deberá dar la orden de enviarlo al almacenamiento en la nube y mostrar una alerta con la confirmación del envío, nombrePedido, numeroPedido.

pre: Ticket no vacío.

post: -

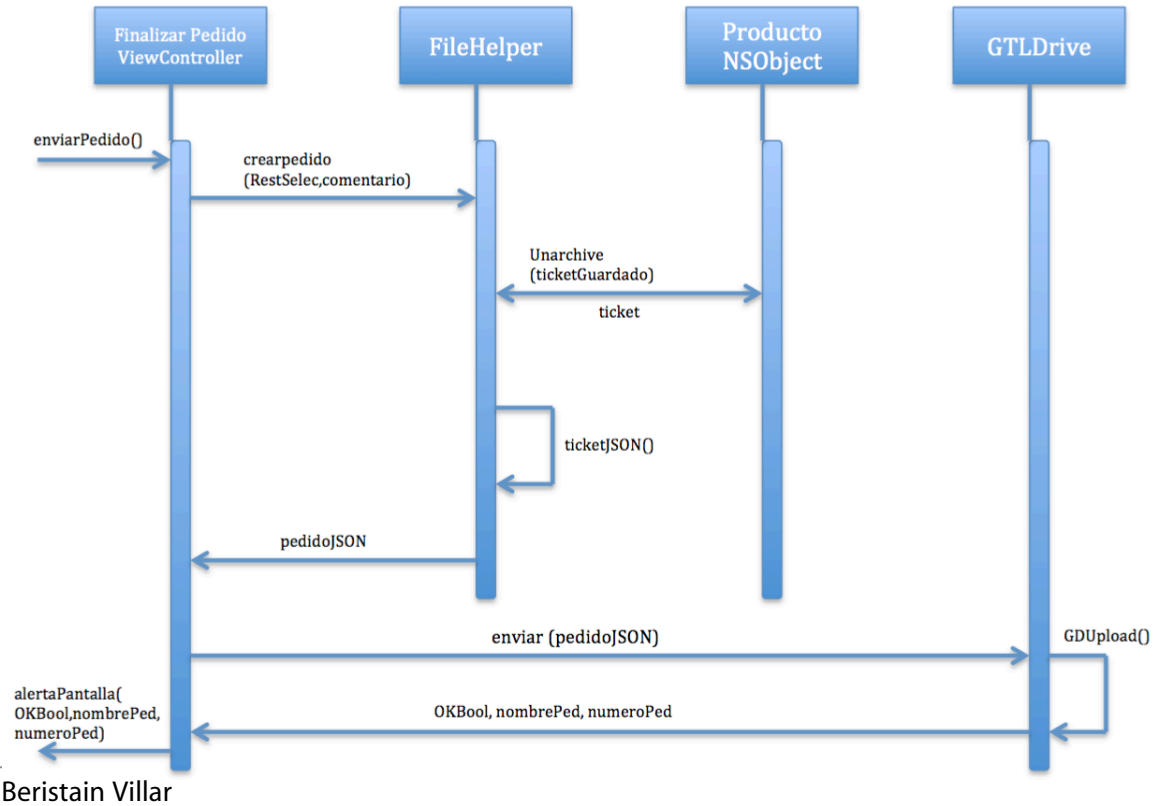
salida: Alerta por pantalla OKBool, nombrePedido y numeroPedido.

OKBool: Booleano para la confirmación de que ha sido enviado, o no.

nombrePedido: Nombre del dispositivo que envía el pedido.

numeroPedido: Número asignado al pedido realizado.

Diagrama de secuencia:



Nombre: eliminarProducto

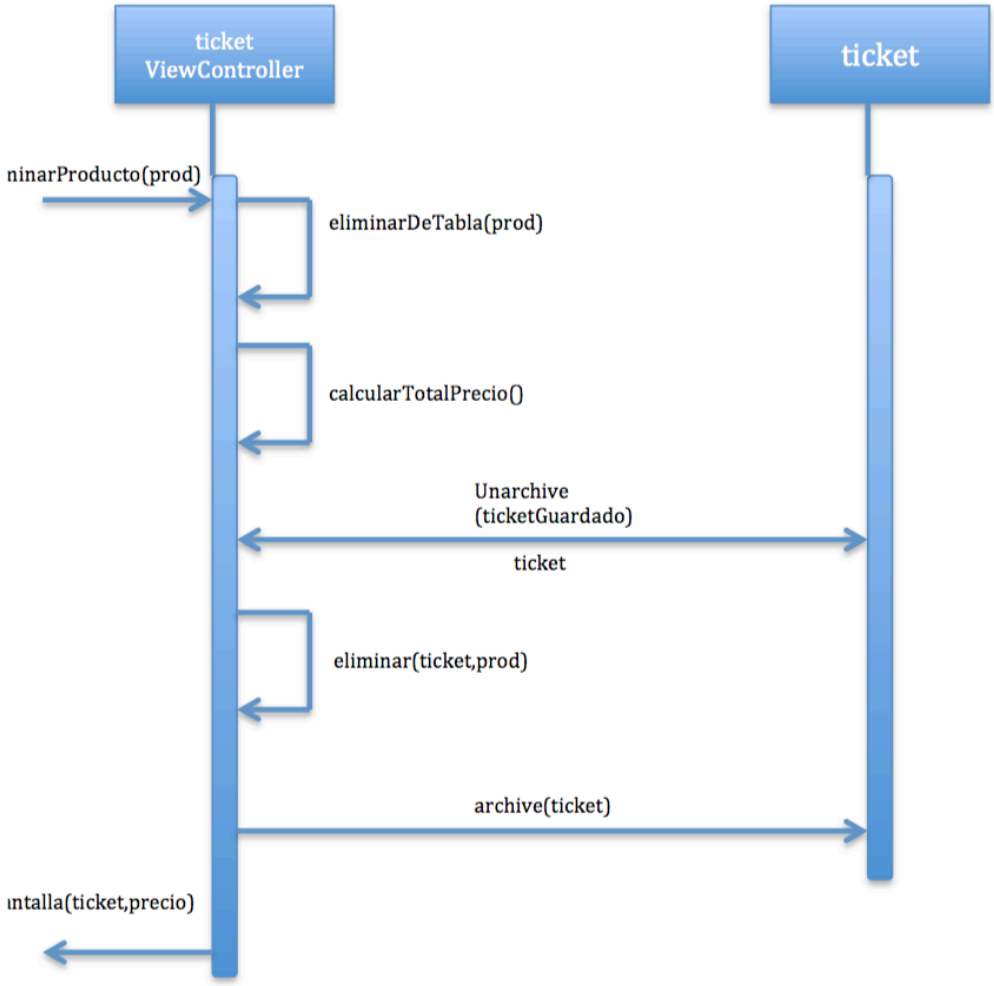
Responsabilidades: El sistema debe eliminar el producto seleccionado de la lista en pantalla, recalcular el precio total, y además debe eliminar ese producto del ticket guardado.

pre: Ticket no vacío.

post: -

salida: Muestra por pantalla la lista del Ticket actualizado y el precio actualizado.

Diagrama de secuencia:



Nombre: añadirMenú

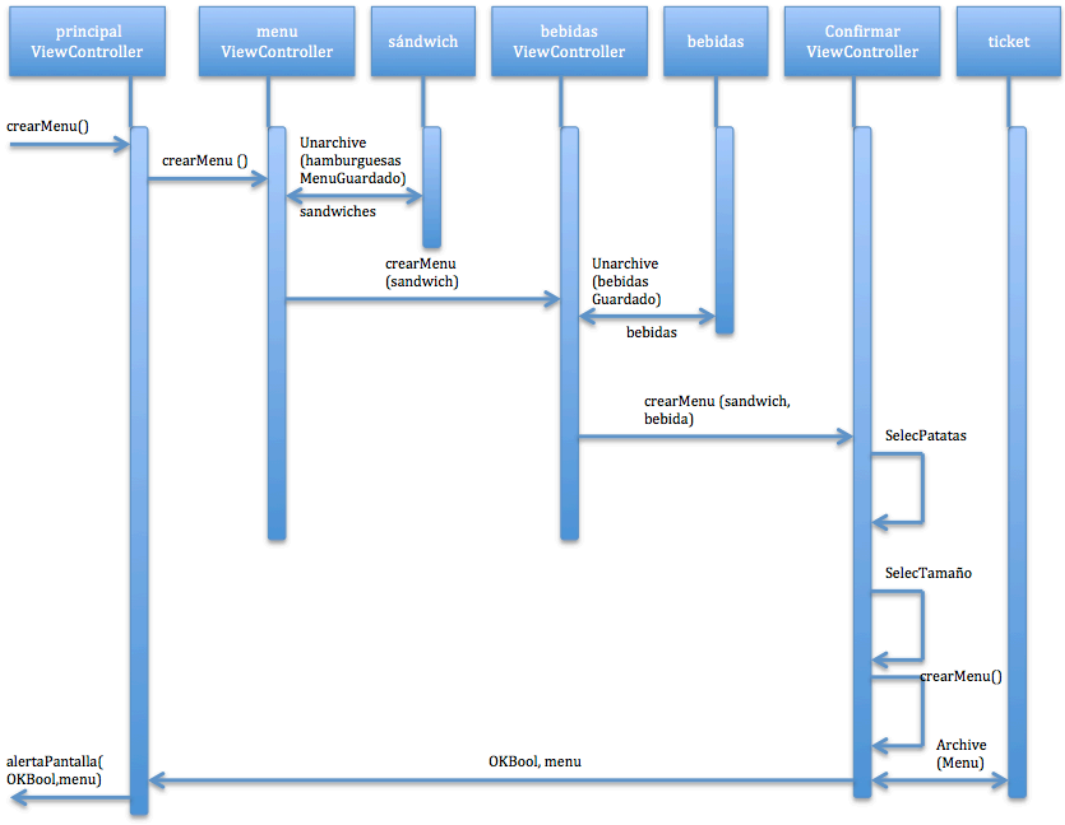
Responsabilidades: El sistema debe almacenar el sándwich, bebida, patatas y tamaño deseado, y añadir un menú al ticket con las elecciones del usuario.

pre: -

post: -

salida: Muestra por pantalla una alerta confirmando que el menú se ha añadido al ticket.

Diagrama de secuencia:



7. IMPLEMENTACIÓN

Durante este capítulo se tratarán los diferentes aspectos relacionados con la implementación de la aplicación, entorno de programación y del código de McPhone.

7.1 Entorno de programación

Apple sólo admite un único entorno de programación para sus dispositivos; analicemos detenidamente qué nos ofrece.

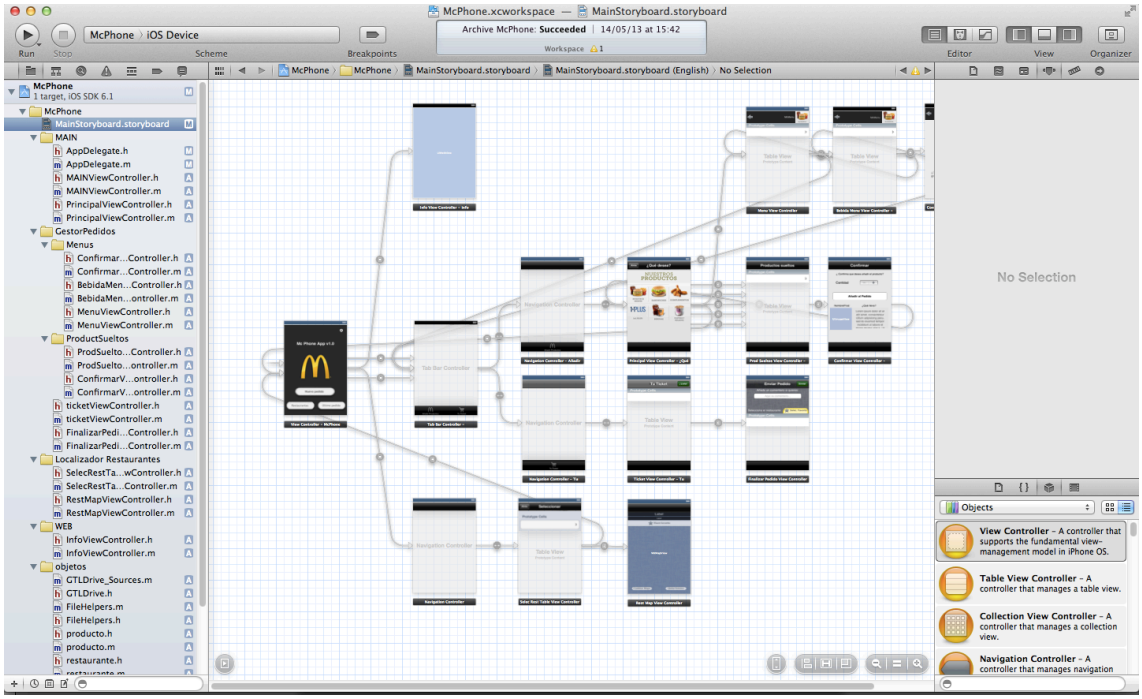
7.1.1 IDE, SDK, Xcode

El entorno de desarrollo viene dado por Apple, sin posibilidad de utilizar ninguna otra variante. Puede parecer limitado, pero la verdad es que han cuidado bien todos y cada uno de los elementos y se muestra realmente sólido. En 5 meses de intensa curva de aprendizaje, fallos, errores y demás contratiempos, el entorno no ha fallado y muestra una consistencia que convence desde el primer instante. El requisito previo es trabajar con un ordenador Mac, y tener OS X 10.7.4 o posterior como indica Apple en AppleStore (véase figura 7.1).

figura 7.1 información de Xcode en AppleStore

El **Xcode** es gratuito y únicamente en inglés. (véase figura 7.2)

figura 7.2 interfaz del Xcode



El responsable del buen funcionamiento es el entorno de desarrollo integrado IDE **Xcode** exclusivo para Mac OSX. Tiene un diseño y distribución de áreas muy cómodo para trabajar. Es práctico, fluido, y funcional desde el momento que se descarga desde el AppleStore. La naturaleza de entorno cerrado le beneficia, sobre todo vista la consistencia que se ha logrado.

Xcode se introdujo el 24 de octubre de 2003, junto con la versión 10.3 de Mac OS X. Es una evolución a partir del anterior entorno de desarrollo de Apple, Project Builder, al que sustituyó. El kit de desarrollo de software para iPhone fue anunciado oficialmente y puesto a disposición de los desarrolladores el 6 de marzo de 2008, en versión beta, mientras que la tienda de aplicaciones entró en funcionamiento el 11 de julio de ese año.

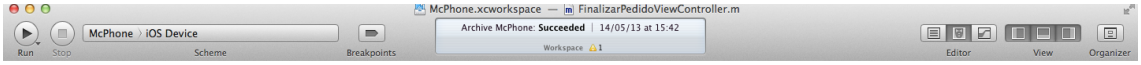
El entorno **Xcode** trabaja indistintamente con el Interface Builder[1] o con los recientemente integrados *StoryBoard*s. Los *StoryBoard*s son la evolución del Interface Builder[1] y nos permiten subir un nivel más y controlar de manera gráfica la transición a rasgos generales entre el flujo de ventanas.

Se ha utilizado la última versión estable, es decir, la versión 4.6.2, que ya incluye las dos grandes características que se incluyeron hará unas cuantas revisiones. Por un lado, los *StoryBoard*s de los que ya hemos hablado. Y por otro, el ARC, que nos facilita mucho la implementación de la aplicación al gestionar, por si sólo, la liberación de la memoria cuando sea necesario. Hasta la integración del ARC, había que tener mucho cuidado con cuándo se liberaba la memoria y cuánta memoria se reservaba. Actualmente, todo esto, si se quiere, se puede hacer de manera desatendida. En McPhone se ha activado esta opción, y realmente funciona de maravilla.

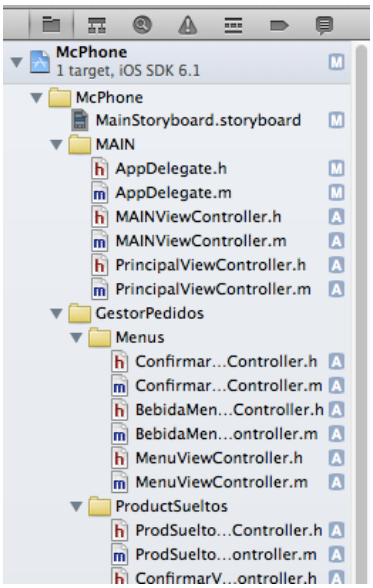
Respecto al contenido del SDK, ya que iPhone comparte base con Mac OS X, la cadena de instrumentos para desarrollar aplicaciones para iPhone está también basada en **Xcode** e incluye compiladores cruzados para el procesador ARM y un simulador de iPhone. El lenguaje de programación principal para iPhone OS, al igual que en Mac OS, es Objective-C.

[1] *Interface Builder*: *Una herramienta gráfica para la creación de interfaces de usuario, herencia de Next.*

7.1.2 División del entorno de Xcode

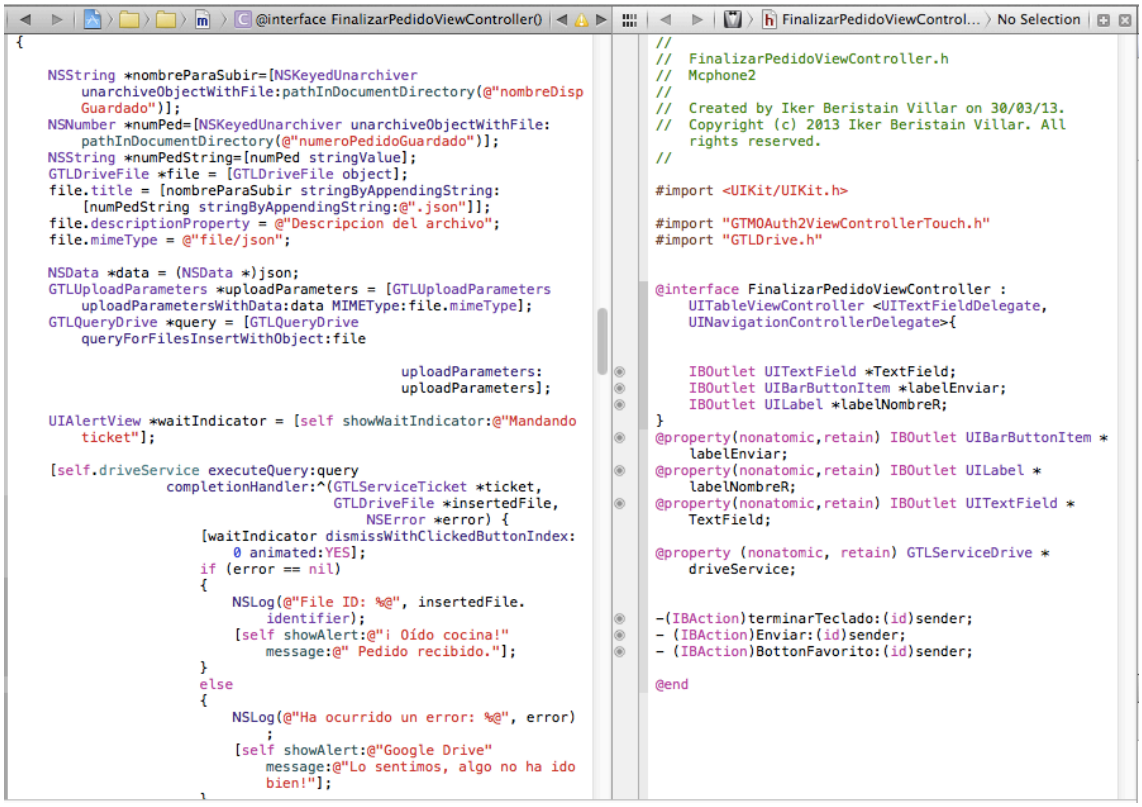


Barra superior: Nos muestra el estado del **Xcode**. Nos permite compilar y ejecutar nuestro proyecto en el simulador o en un dispositivo iOS. Nos notifica los errores o warnings pendientes de revisión. Y en la parte derecha nos permite gestionar las vistas que tenemos en uso.



Navegador del proyecto: Aquí se puede encontrar todos los objetos, clases, subclases, cabeceras, hilos, ViewControllers[1], etc. Desde aquí gestionaremos los archivos y ficheros de nuestro proyecto de manera estructurada.

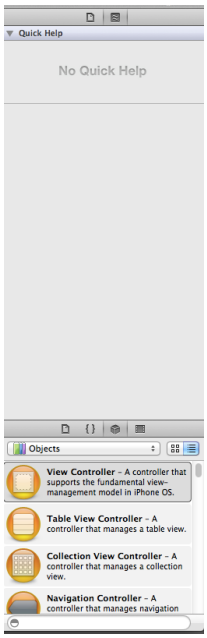
[1] *ViewController*: *Son los controladores que se encargan de gestionar cada vista de la aplicación.*



```
{

    NSString *nombreParaSubir=[NSKeyedUnarchiver
        unarchiveObjectWithFile:pathInDocumentDirectory(@"nombreDisp
        Guardado")];
    NSNumber *numPed=[NSKeyedUnarchiver unarchiveObjectWithFile:
        pathInDocumentDirectory(@"numeroPedidoGuardado")];
    NSString *numPedString=[numPed stringValue];
    GTLDriveFile *file = [GTLDriveFile object];
    file.title = [nombreParaSubir stringByAppendingString:
        [numPedString stringByAppendingString:@".json"]];
    file.descriptionProperty = @"Descripcion del archivo";
    file.mimeType = @"file/json";

    NSData *data = (NSData *)json;
    GTLUploadParameters *uploadParameters = [GTLUploadParameters
        uploadParametersWithData:data MIMEType:file.mimeType];
    GTLQueryDrive *query = [GTLQueryDrive
        queryForFilesInsertWithObject:file

                                        uploadParameters:
                                        uploadParameters];

    UIAlertView *waitIndicator = [self showWaitIndicator:@"Mandando
        ticket"];

    [self.driveService executeQuery:query
                    completionHandler:^(GTLServiceTicket *ticket,
                                        GTLDriveFile *insertedFile,
                                            NSError *error) {
                [waitIndicator dismissWithClickedButtonIndex:
                    0 animated:YES];
                if (error == nil)
                {
                    NSLog(@"File ID: %@", insertedFile.
                        identifier);
                    [self showAlert:@"¡ Oído cocina!"
                        message:@" Pedido recibido."];
                }
                else
                {
                    NSLog(@"Ha ocurrido un error: %@", error)
                        ;
                    [self showAlert:@"Google Drive"
                        message:@"Lo sentimos, algo no ha ido
                        bien!"];
                }
```

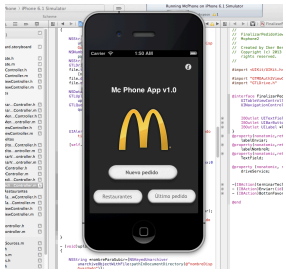
```
//
//  FinalizarPedidoViewController.h
//  Mcphone2
//
//  Created by Iker Beristain Villar on 30/03/13.
//  Copyright (c) 2013 Iker Beristain Villar. All
    rights reserved.
//

#import <UIKit/UIKit.h>

#import "GTMOAuth2ViewControllerTouch.h"
#import "GTLDrive.h"


@interface FinalizarPedidoViewController :
    UITableViewController <UITextFieldDelegate,
    UINavigationControllerDelegate>{


    IBOutlet UITextField *TextField;
    IBOutlet UIBarButtonItem *labelEnviar;
    IBOutlet UILabel *labelNombreR;
}
@property(nonatomic,retain) IBOutlet UIBarButtonItem *
    labelEnviar;
@property(nonatomic,retain) IBOutlet UILabel *
    labelNombreR;
@property(nonatomic,retain) IBOutlet UITextField *
    TextField;

@property (nonatomic, retain) GTLServiceDrive *
    driveService;


-(IBAction)terminarTeclado:(id)sender;
- (IBAction)Enviar:(id)sender;
- (IBAction)BottonFavorito:(id)sender;

@end
```

Assistant editor: Ventana donde podemos crear, eliminar o editar el código de cada componente.



Utilities: En este panel se editan propiedades de los *StoryBoards*; y además, tenemos disponibles todos lo objetos predefinidos que nos da Apple.

7.1.3 Simulador iPhone



Permite simular tanto iPhone como iPad en cualquiera de sus versiones disponibles en el mercado. Es importante remarcar que hace una simulación directa, y nos permite testear de manera inmediata nuestro proyecto.

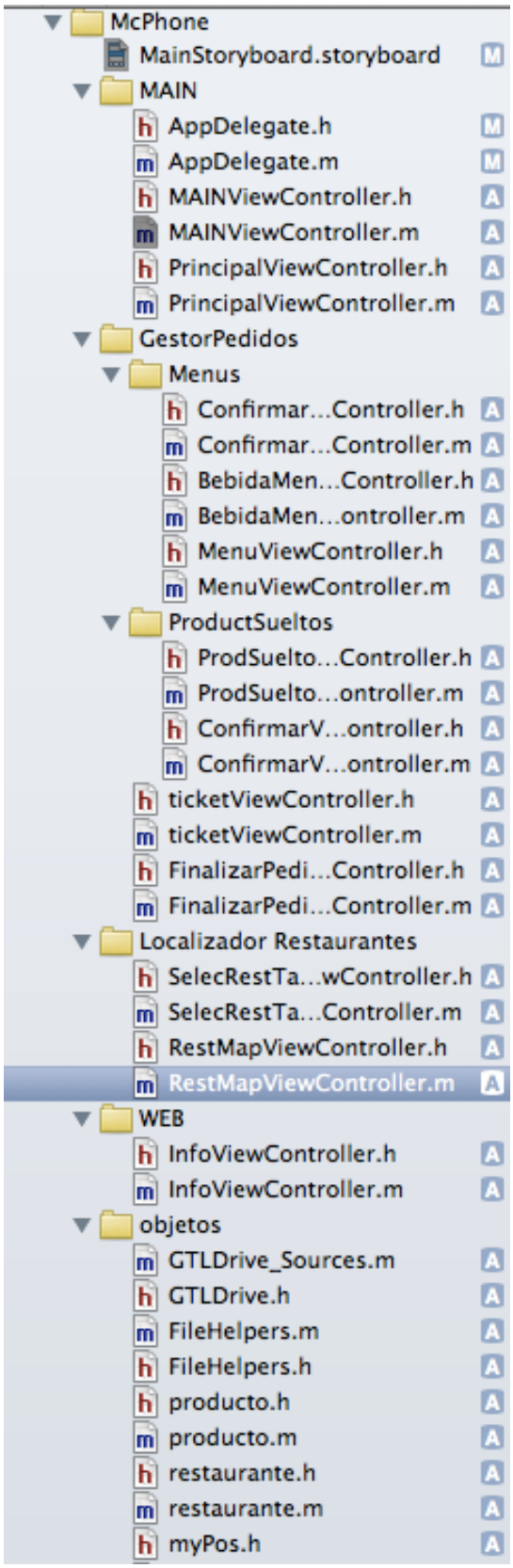
La mayoría del proyecto se ha desarrollado con el simulador iPhone alternando entre las diferentes resoluciones de pantalla. Se ha dejado el hardware real para un testeo periódico de aproximadamente una vez por semana de desarrollo.

Tan solo se ha hallado un caso puntual en el que el iPhone real y el simulado no tenían el mismo comportamiento. La incidencia tenía su origen en la escritura en mayúsculas o minúsculas de las extensiones de los ficheros de las fotografías utilizadas. Dependiendo cómo se escribieran, aparecían o desparecían al ejecutar la aplicación. El iPhone simulado admite minúsculas/mayúsculas indistintamente; pero el hardware real se muestra más caprichoso, atiende a cualquiera de las dos, pero no simultáneamente.

7.2 Código

Para el código se ha utilizado casi en su totalidad el lenguaje de programación Objective-C, a excepción del tipo *float* de C que es compatible con Xcode y un par de instrucciones puntuales más. Se ha procurado estructurar el código lo más intuitivamente posible ya que la App tiene un alto potencial de expansión. Se ha comentado todo lo posible para facilitar las posibles próximas líneas de mejora. A continuación se puede ver, cuando son necesarios en puntos críticos, que la densidad de comentarios explicativos supera a las líneas de código.

```
//Cargar el arrayRest con NSUserDefault
arrayRest=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"arrayRestGuardado")];

//Inicializamos el comentario, sino se escribe nada -sin comentario-


TextField.delegate=self;


// Uncomment the following line to preserve selection between presentations.
// self.clearsSelectionOnViewWillAppear = NO;
```



Se han creado cinco clases de objetos de apoyo diferentes para gestionar el proyecto, un delegado que maneja la App, un *StoryBoard* y se cuenta con doce controladores de vistas para tareas específicas de las ventanas de navegación. Ha sido importante la administración organizada de los controladores de vistas para poder implementar de manera correcta los métodos en donde corresponden.

La base del flujo de ventanas se ha creado inicialmente con el *StoryBoard* y en base a él se han ido implementando los diferentes controladores de vistas. Las funciones específicas se han incluido en cada controlador que las maneja, con el objetivo de facilitar las tareas, y siempre que no fueran funciones complejas. Así pues, se ha tomado la decisión de implementar por separado el GTLDrive, FileHelper y myPos. Son, en orden, el soporte para la utilización de GoogleDrive, el creador del pedido final y conversor a JSON, y por último, la clase del objeto que ayuda a gestionar las posiciones.

Se ha mantenido nomenclatura uniforme en todo el código, con ciertas reglas para normalizarlo. Como ejemplo rápido, la nomenclatura de las variables o nombre de los ficheros : *nombreDeLaVariable* , *NombreFichero.*

Expliquemos, por ejemplo, cómo se crean los pedidos, cómo se adaptan a JSON y cómo son enviados mediante GoogleDrive. Nos centraremos en las fracciones de código principales, a pesar de que éstas hagan llamadas a otras partes de código.

En FileHelpers se ha implementado la creación del pedido donde se recopilan todos los datos necesarios para la gestión del mismo y se integran en un array.

```
NSArray *crearPedido(restaurante *RestSelec,NSString *comentario){
    NSMutableArray *pedido=[[NSMutableArray alloc] initWithCapacity:1];

    NSString *fechaString;
    NSString *UDID;
    NSString *nombreDisp;
    NSNumber *numeroPedido;
    NSMutableArray *ticket;

    //Coger fecha y hora actual y pasarla a un String
    NSDate *fechaAct=[NSDate date];
    NSDateFormatter *formato= [[NSDateFormatter alloc] init];
    [formato setDateFormat:@"dd-MM-YYYY HH:mm:ss "];
    [formato setTimeZone:[NSTimeZone timeZoneWithName:@"Europe/Madrid"]];
    fechaString=[formato stringFromDate:fechaAct];
    //Recuperamos el nombre del dispositivo y UDID(código único de App/dispositivo)
    UDID=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"codUnicoDispGuardado")];
    nombreDisp=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"nombreDispGuardado")];
    //recuperamos el numero pedido
    numeroPedido=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"numeroPedidoGuardado")];
    numeroPedido=[NSNumber numberWithInt:[numeroPedido integerValue]+1];
    [NSKeyedArchiver archiveRootObject:numeroPedido toFile:pathInDocumentDirectory(@"numeroPedidoGuardado")];


    //recuperamos el ticket
    ticket=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"ticketGuardado")];

    [pedido addObject:RestSelec.nombre];
    [pedido addObject:fechaString];
    [pedido addObject:nombreDisp];
    [pedido addObject:UDID];
    [pedido addObject:[numeroPedido stringValue]];
    [pedido addObject:comentario];
```

Se prepara el ticket también, para pasarlo a la estructura JSON .

```
'/Conversion del ticket a Array para poder pasarlo a estructura JSON

    NSMutableArray *ticketJSON=[[NSMutableArray alloc] initWithCapacity:1];
    producto *productoAct;
    NSArray *pedidoAct;
    NSString *precioAct;

    for (int i=0; i<[ticket count]; i++) {
        productoAct=[ticket objectAtIndex:i];
        precioAct=[NSString stringWithFormat:@"%.2f", productoAct.precio];
        pedidoAct=[[NSArray alloc] initWithObjects:productoAct.nombre, precioAct, nil ];
        [ticketJSON addObject:pedidoAct];
    }
    [pedido addObject:ticketJSON];

    return pedido;
```

En el ViewController se envía el pedido y se prepara para transformarlo a una estructura JSON.

```
- (IBAction)Enviar:(id)sender {
        comentario=TextField.text;

    ticket=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"ticketGuardado")];

    if([ticket count]==0){
        UIAlertView *alerta=[[UIAlertView alloc] initWithTitle:@"Pedido vacío" message:@" Por favor, revisa el tiket y recuerda
            añadir al menos un artículo antes de enviar el pedido. ¡ Gracias!" delegate:self cancelButtonTitle:@"ok"
            otherButtonTitles: nil];
        [alerta show];

    }else{

        NSArray *pedidoCreado=crearPedido(RestSelec,comentario);


        //JSON

        //Generar datos JSON
        NSError *error=nil;

        NSData *json =[NSJSONSerialization dataWithJSONObject:pedidoCreado options:NSJSONWritingPrettyPrinted error:&error];

        [NSKeyedArchiver archiveRootObject:json toFile:pathInDocumentDirectory(@"PedidojsonGuardado")];
```

Y finalmente se procede a la subida a GoogleDrive.

```
// Subir un JSON a Google Drive
- (void)uploadJSON:(NSData*)json
{

    NSString *nombreParaSubir=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"nombreDispGuardado")];
    NSNumber *numPed=[NSKeyedUnarchiver unarchiveObjectWithFile:pathInDocumentDirectory(@"numeroPedidoGuardado")];
    NSString *numPedString=[numPed stringValue];
    GTLDriveFile *file = [GTLDriveFile object];
    file.title = [nombreParaSubir stringByAppendingString:[numPedString stringByAppendingString:@".json"]];
    file.descriptionProperty = @"Descripcion del archivo";
    file.mimeType = @"file/json";

    NSData *data = (NSData *)json;
    GTLUploadParameters *uploadParameters = [GTLUploadParameters uploadParametersWithData:data MIMEType:file.mimeType];
    GTLQueryDrive *query = [GTLQueryDrive queryForFilesInsertWithObject:file
                                                       uploadParameters:uploadParameters];

    UIAlertView *waitIndicator = [self showWaitIndicator:@"Mandando ticket"];

    [self.driveService executeQuery:query
                   completionHandler:^(GTLServiceTicket *ticket,
                                       GTLDriveFile *insertedFile, NSError *error) {
                       [waitIndicator dismissWithClickedButtonIndex:0 animated:YES];
                       if (error == nil)
                       {
                           NSLog(@"File ID: %@", insertedFile.identifier);
                           [self showAlert:@"¡ Oído cocina!" message:@" Pedido recibido."];
                       }
                       else
                       {
                           NSLog(@"Ha ocurrido un error: %@", error);
                           [self showAlert:@"Google Drive" message:@"Lo sentimos, algo no ha ido bien!"];
                       }
                   }];
}
```

7.3 Valoración del código

Han sido necesarias entorno a 3200 líneas de código para completar la implementación. Éstas líneas se distribuyen en los siguientes componentes del código:

- **Un delegado de la Aplicación.**

Donde se gestionan las cargas principales de información en la aplicación, y las funcionalidades de integración en el sistema operativo.

Aproximadamente 875 líneas de código.

- **Doce controladores de vistas.**

Controlan el comportamiento de cada ventana/vista. Dentro de ellas también se han definido las funcionalidades que realizan.

Aproximadamente 1885 líneas de código.

- **Cinco clases.**

Unas son clases de objetos definidos para gestionar la información, y otras como apoyo para que las funciones más complejas se realicen fuera de los controladores de vistas.

Aproximadamente 440 líneas de código.

- **Un StoryBoard.**

Forma parte de la implementación a pesar de no ser código. Representa el diseño general de la aplicación, y el modo en que se gestionan los flujos de ventanas.

- **Ochenta imágenes y un audio.**

Se han preparado 80 imágenes para ilustrar la App, el catálogo de productos, fondos, botones, etc. Y se ha adaptado un archivo de audio con el silbido característico de la compañía de restauración como presentación de la aplicación.

8. PUBLICACIÓN EN AppleStore

La aplicación se ha publicado en AppleStore, por lo que en este capítulo se explicará lo relacionado con la publicación en la tienda de Apple.

8.1 Información general

El SDK se puede descargar gratis, pero para publicar el software es necesario registrarse en el *Programa de Desarrollo del iPhone*, un paso que requiere el pago y la aprobación por parte de Apple. Durante el proceso, se entregan al desarrollador unas claves firmadas que permiten subir una aplicación a la tienda de aplicaciones de Apple.

figura 8.1 McPhone en el catálogo de AppleStore

Las aplicaciones pueden ser distribuidas de tres formas: a través de la AppStore de Apple para todo el público, por parte de una empresa a sus empleados en uso privado, o sobre una red *Ad-hoc* de hasta cien iPhones.

Los desarrolladores son libres de poner cualquier tarifa para que sus aplicaciones sean distribuidas por la tienda de Apple. A cambio de ello la compañía se queda con un porcentaje del 30% de los ingresos. Los desarrolladores pueden optar por ofrecer sus aplicaciones gratis y no pagar así nada por la distribución del programa más allá de la cuota de socio.

8.2 Experiencia publicación McPhone en AppleStore

Se ha hecho el registro de rigor (Apple ID) en el centro de desarrollo iPhone y se ha pasado por caja con la suscripción anual de 99$. Ya se disponía del kit (SDK) de desarrollo. Con ello ya tenemos acceso a iTunesConnect[1], imprescindible para gestionar las aplicaciones que queramos publicar.

Una vez finalizada la aplicación, lo primero es asegurarse de no incumplir ninguna de las condiciones de rechazo impuestas por Apple. Algunas podrían considerarse evidentes, pero conviene leer con detenimiento los veintitrés apartados por los que se despliegan las condiciones.

iTunes Connect

Apple Review Guidelines:
https://developer.apple.com/appstore/resources/approval/guidelines.html

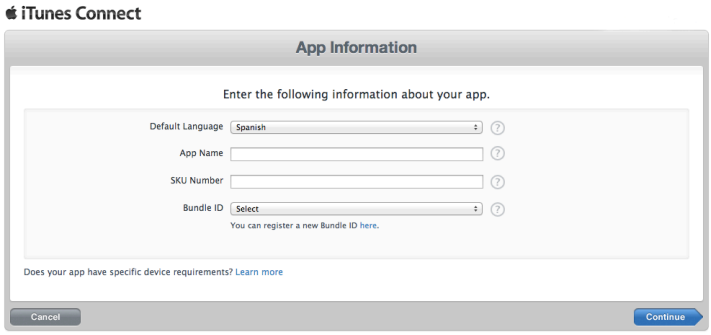
Se ha diseñado un logo para la aplicación, que será su carta de presentación de cara al cliente. Como inspiración, se han seguido las líneas de los últimos logos utilizados por Google en sus Apps, predominando la simplicidad del diseño, colores vivos y leves sombreados.

figura 8.2 Logo diseñado para McPhone

Se ha creado un certificado para McPhone acorde con las condiciones de Apple. Se ha descargado el certificado y registrado en el Organizer del Xcode.

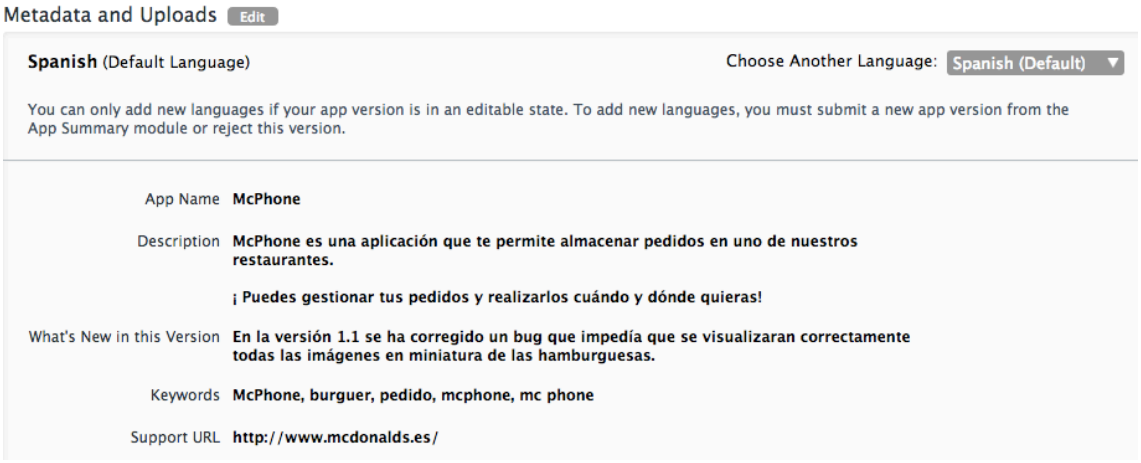
[1] *iTunesConnect*: *Es la herramienta web que nos ofrece Apple para gestionar todo lo relacionado con nuestros desarrollos.*

Una vez correctamente certificada, se da de alta la App en iTunesConnect proporcionando toda la información relativa a la aplicación. Hay que tener especial atención al BundleIdentifier[1] que es un habitual causante de problemas, ya que éste debe coincidir en todos los lugares que se pide.



Hay que introducir: Nombre, descripción, keywords de busqueda, URL soporte, calificación por edades, y capturas de pantalla en formato 3'5" y 4".

En este caso la aplicación es gratuita, por lo que se ajusta a la *tarifa 0* de Apple.



[1] *BundleIdentifier*: *Es el identificador general de la App. Nos dejan seleccionarlo, y a su vez nos lo piden reiteradas veces en distintos momentos del desarrollo, por lo que no conviene dejarlo al azar ya que debe coincidir en todos lados.*

Una vez esté preparada para subir la App, itunesConnect pasa por cinco estados; en el caso de McPhone supuso un total de cinco días.

1) Waiting For Upload: iTunesConnect está a la espera de que se suba la App mediante el Organizer del Xcode. [08-05-2013]

2) Waiting For Review: Probablemente es el proceso más largo. La App se pone en lista de espera para que Apple la revise y le dé el último visto bueno. Unos cuatro o cinco días naturales. [08-05-2013]

3) In review: Este estado se muestra cuando Apple comienza la revisión de la App; el tiempo en este proceso dependerá de la densidad de la App. Una o dos horas. [13-05-2013]

4) Processing For AppleStore: La App ha sido aceptada y está siendo preparada para su publicación en el AppleStore. [13-05-2013]

5) Approved, Ready for Sale: Aplicación aprobada y disponible en el AppleStore. [13-05-2013]

9. PRUEBAS

9.1 Pruebas registradas en dispositivos

Dispositivo	Versión iOS	Resultado [motivo]
iphone4 de Marta	iOS 4.6	NO OK [iOS <5.1]
iPhone4S de Egaña	iOS 6.1.3	OK
iPhone4 de JonG	iOS 6.1.3	OK
iPhone4 de Beris	iOS 6.1.3 (jailbreak[1])	OK
iPhone4 de JonB	iOS 5.12 (jailbreak[1])	NO OK [Jailbreak[1]]
iPod Touch Iñaki	iOS 6.1.3	OK
iPhone3GS de Diego	iOS 6.1.2	OK
iPhone4 de Beris	iOS 7 [Beta Junio'13]	OK
iPhone5 de Andrés	iOS 6.1.3	OK

Los resultados son satisfactorios. Funciona en toda la gama de dispositivos que corren bajo iOS 5.1 o superior, incluso en la Beta del próximo iOS7. El caso del dispositivo "iPhone4 de JonB" no ha sido considerado, por estar modificado y la salida repentina que hacía de la App seguramente era fruto del *jailbreak*[1].

Hubo un fallo en la versión 1.0 de McPhone subida el 13-05-2013, que impedía que se visualizaran las imágenes en miniatura de los sándwiches. Fue corregido en una versión 1.1, aceptada y publicada por Apple el 18-05-2013, unos cuatro días después de enviarles la revisión.

Actualmente sigue en la versión 1.1, se muestra estable y funciona correctamente. Próximas posibles versiones estarían enfocadas a incluir o mejorar funcionalidades.

Nota: Aplicación disponible en la presentación del proyecto ante tribunal. Si hubiera la posibilidad, el tribunal podría formar parte de las pruebas.

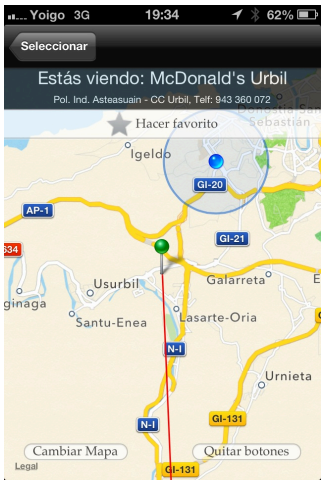
[1] *jailbreak: Se refiere a un dispositivo que corre bajo un sistema operativo iOS pirateado.*

9.2 Casos de prueba

Aparte de las pruebas definitivas y testeo sobre los diferentes hardware existentes de iPhone, durante la fase de implementación se hicieron numerosos casos de prueba para testear cada caso de uso. Se han realizado en torno a cuarenta pruebas. Se han encontrado así dos errores clave en los siguientes casos de uso:

Enviar Pedido: Si en el ticket no se había añadido productos, se enviaba un pedido vacío. Se ha corregido y el sistema muestra por pantalla el siguiente mensaje hasta que se añada al menos un producto:

Consultar restaurante: En la representación en el mapa del restaurante, se encontró un error en la línea que se sobreimprime en el mapa, desde la posición del usuario a la del restaurante. Existía un error de implementación que dibujaba la línea desde el restaurante hasta África, aproximadamente, cuando debería ir dirigida a la posición del usuario. Se corrigió y actualmente se dibuja de manera correcta. A continuación una captura del error:



10. GESTIÓN DEL PROYECTO

Este capítulo tratará sobre la gestión y planificación realizada durante el proyecto.

10.1 Planificación

Se ha llevado un control sobre la planificación y desarrollo para poder hacer un contraste y balance válidos.

10.1.1 Planificación inicial - Lista de entregables estimada

Fecha	Tarea	Coste Horario (estimado)
Septiembre 2012	Presentar documento de propuesta formal de proyecto fin de carrera al tutor.	20h
Octubre 2012	PFC aceptado e inscrito en la Universidad.	2h
15 Octubre 2012	Captura de requisitos.	25h
25 Octubre 2012	Análisis.	25h
05 noviembre 2012	Diseño.	25h
05Nov'12 - 15 enero'13	Formación en el lenguaje de programación propio de MAC OSX , Objective-C. Aprender a utilizar el entorno de desarrollo Xcode. Flujo de desarrollo de una aplicación completa. Conocer el diseño de aplicaciones táctiles y la construcción de su interfaz gráfica. Formarse sobre la publicación de Apps en AppStore, firmas digitales, derechos y responsabilidades. Realizar ejemplos prácticos de aplicaciones similares a la proyectada para poder abordarla con suficiente conocimiento.	200h

04 Marzo 2013	Entrega Fase 1 App. Aplicación 100% funcional.	180h
18 Marzo 2013	Fase2: Implementar la integración de la App con un sistema de almacenamiento en la nube (presumiblemente GoogleDrive).	20h(25h formación)
25 Marzo 2013	Se preparará y solicitará la publicación en AppStore.	10h
29 Abril 2013	Documentación del PFC completada. A su vez, si no entorpeciera en el desarrollo de la misma, podría mejorarse la interfaz y experiencia de usuario de la App.	80h

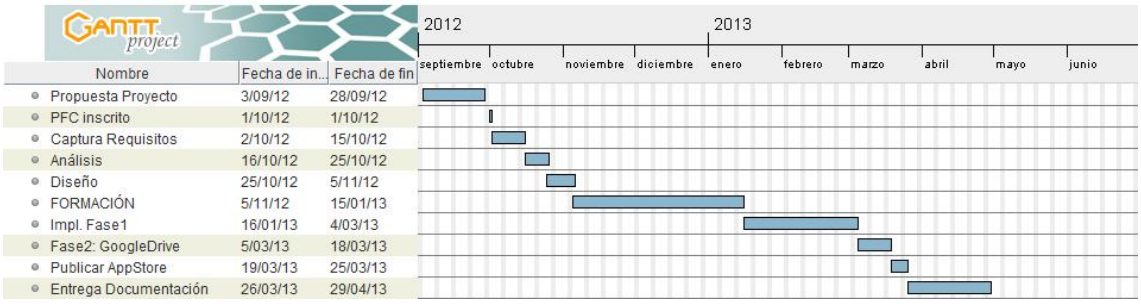
10.1.2 Desarrollo y Seguimiento - Lista de entregables real

Fecha	Tarea	Coste Horario (real)
25 Septiembre 2012	Presentar propuesta formal de proyecto fin de carrera al tutor.	20h
4 Octubre 2012	PFC aceptado e inscrito en la Universidad.	2h
15 Octubre 2012	Captura de requisitos.	15h
25 Octubre 2012	Análisis.	15h
05 noviembre 2012	Diseño.	25h
05Nov'12 - 12Feb'13	Formación en el lenguaje de programación propio de MAC OSX , Objective-C. Aprender a utilizar el entorno de desarrollo Xcode. Flujo de desarrollo de una aplicación completa. Conocer el diseño de aplicaciones táctiles y la construcción de su interfaz gráfica. Formarse sobre la publicación de Apps en AppStore, firmas digitales, derechos y responsabilidades. Realizar ejemplos prácticos de aplicaciones similares a la proyectada para poder abordarla con suficiente conocimiento.	255h
18 Febrero 2013	*StoryBoard* y diseño general de la aplicación terminado. Implementación del flujo de ventanas.	30h

04 Marzo 2013	Implementación del ticket. Listado correcto de la elección de los productos, capacidad de añadirse productos y eliminarse.	50h
11 Marzo 2013	Implementación del sistema de menús.	20h
18 Marzo 2013	Implementación del localizador de restaurantes. posición relativa del usuario inclusive.	30h
25 Marzo 2013	Primera revisión del producto.	-
5 de Abril 2013	Fase1: App totalmente funcional, y capaz de imprimir por pantalla el ticket del pedido realizado por el usuario. A su vez, esa información deberá ser almacenada en un formato normalizado para la próxima transferencia (presumiblemente JSON).	30h
19 Abril 2013	Fase2: Implementar la integración de la App con un sistema de almacenamiento en la nube (presumiblemente GoogleDrive).	15h(25h)
4 Mayo 2013	Se solicitará la publicación en AppStore.	8h
6 Junio 2013	Documentación del PFC completada. A su vez, si no entorpeciera en el desarrollo de la misma, podría mejorarse la interfaz y experiencia de usuario de la App.	60h

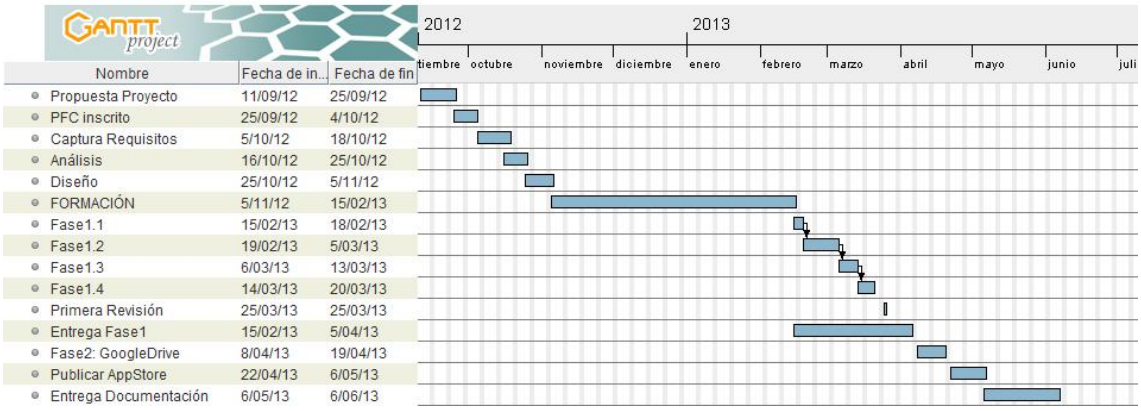
10.1.3 Diagramas de GANTT

- Diagrama de Gantt planificado



Nombre	Fecha de in...	Fecha de fin
Propuesta Proyecto	3/09/12	28/09/12
PFC inscrito	1/10/12	1/10/12
Captura Requisitos	2/10/12	15/10/12
Análisis	16/10/12	25/10/12
Diseño	25/10/12	5/11/12
FORMACIÓN	5/11/12	15/01/13
Impl. Fase1	16/01/13	4/03/13
Fase2: GoogleDrive	5/03/13	18/03/13
Publicar AppStore	19/03/13	25/03/13
Entrega Documentación	26/03/13	29/04/13

- Diagrama de Gantt Real



Nombre	Fecha de in...	Fecha de fin
Propuesta Proyecto	11/09/12	25/09/12
PFC inscrito	25/09/12	4/10/12
Captura Requisitos	5/10/12	18/10/12
Análisis	16/10/12	25/10/12
Diseño	25/10/12	5/11/12
FORMACIÓN	5/11/12	15/02/13
Fase1.1	15/02/13	18/02/13
Fase1.2	19/02/13	5/03/13
Fase1.3	6/03/13	13/03/13
Fase1.4	14/03/13	20/03/13
Primera Revisión	25/03/13	25/03/13
Entrega Fase1	15/02/13	5/04/13
Fase2: GoogleDrive	8/04/13	19/04/13
Publicar AppStore	22/04/13	6/05/13
Entrega Documentación	6/05/13	6/06/13

10.1.4 Valoración - Planificación estimada VS real

- **RESUMEN COSTE HORAS PLANIFICADAS:**
  - Formación : 220h
  - Trabajo:387h
  - Tácticas: 10 reuniones de 1 hora con tutor= 10h

  TOTAL HORAS ESTIMADAS: 617h

- **RESUMEN COSTE HORAS REALES**
  - Formación : 280h
  - Trabajo:320h
  - Tácticas: 6 reuniones de 1,5 hora aprox. con tutor: 9h

    Septiembre 2012

    Octubre 2012

    26 Marzo 2013

    4 Abril 2013

    6 Mayo 2013

    6 Junio 2013

  TOTAL HORAS REALES: 609h

- **VALORACIÓN**

El número total de horas estimadas e invertidas ha sido muy similar. Pero se puede sacar una clara conclusión. Se subestimó la densidad de contenido a asimilar y estudiar, por lo que se ha tardado en torno a sesenta horas más de las previstas en digerir y prepararse para abordar la App. El autor era consciente de la carga que iba a suponer adaptarse a un nuevo lenguaje de programación y a un entorno desconocido de desarrollo. Innumerables horas invertidas entre libros, tutoriales, pruebas, y solucionando errores en búsquedas vía internet. Ha sido un duro y largo proceso, pero necesario para poder rendir bien en la etapa de diseño e implementación de la App. De hecho, en torno a otras sesenta horas han sido reducidas respecto a las planificadas en el trabajo en sí, fruto de una buena inversión de tiempo en formación.

Otro elemento que destaca en los diagramas de Gantt, es que las fechas tampoco se han estimado adecuadamente. Hubo una previsión inicial de que el trabajo finalizaría por completo el 29 de Abril de 2013, cuando realmente no se ha completado hasta el 6 de Junio del 2013, un mes más tarde de lo previsto. Esto es consecuencia de la mayor inversión en tiempo debido a lo dilatada que ha resultado la curva de aprendizaje; esperada, pero subestimada.

Si se volviera a abordar un proyecto similar, se tendría especial atención a elementos desconocidos, aquellos que requieren de estudio y formación previa, procurando estimar el tiempo suficiente para su aprendizaje. La formación en las herramientas a utilizar, podría considerarse la base del éxito del proyecto. Una inversión de tiempo escasa en formación podría repercutir en el rendimiento en horas de trabajo, y lo que es peor aun, en la calidad del producto final.

11. CONCLUSIONES

Concluyo el proyecto con un balance positivo y con un producto estable fruto del trabajo realizado. Merece la pena estructurar las conclusiones por apartados pormenorizados, antes de ofrecer mi visión global una vez acabado el proyecto.

iOS

El sistema operativo iOS brinda unas posibilidades de desarrollo muy amplias, y reconoce funciones predefinidas en *frameworks* ofrecidos por Apple, con las que te aseguran un perfecto funcionamiento. A pesar de contar con numerosas restricciones, incluso obligándote algunas de ellas a utilizar exclusivamente sus *frameworks* para determinadas tareas, se muestra amplio en funcionalidades. En determinadas ocasiones, incomoda no poder realizar las cosas de manera alternativa a Apple. Pero realmente, sus *frameworks* funcionan perfectamente y vienen bien documentados, por lo que una vez aceptemos las restricciones, reconoceremos que funcionan a las mil y una maravillas.

Ejemplo en la restricción 9.1 de Apple*: Apps that do not use the MediaPlayer framework to access media in the Music Library will be rejected.*

Recursos en la nube

Respecto a los recursos en la nube, son un elemento muy útil cuando buscamos que dos o más dispositivos establezcan una comunicación independientes entre si y desatendidamente. Más concretamente GoogleDrive, es muy potente y funciona a la perfección. Ahora bien, ojalá la documentación estuviera a la misma altura. Es inexplicable por parte de una compañía de la envergadura de Google, idolatrada compañía por tantos y tantos. No esperaba encontrarme con una documentación (de las más recientemente actualizadas) como es la del API de GoogleDrive, confusa y con errores. Estoy hablando de la versión de Marzo-Abril 2013, que incluso venía acompañada de un vídeo-tutorial en el cual Steve Bazyl de Google demostraba como integrar la API en Xcode, y a él mismo le aparecen cinco errores, exactamente los mismo cinco errores que te saldrán si sigues al pie de la letra los pasos proporcionados en la documentación de Google. Los soluciona pidiendo disculpas y abriendo un proyecto hecho fuera de cámaras; nada profesional y menos didáctico aun.

El vídeo se puede ver aquí:
http://www.youtube.com/watch?feature=player_embedded&v=owlifMbV47k

Salvados los obstáculos de implementación, GoogleDrive se ha mostrado como una herramienta muy funcional y responde perfectamente.

Entorno de programación

Apple, con filosofía alternativa y cerrada, te ofrece adentrarte en un mundo de desarrollo con ciertos hándicaps iniciales (lenguaje exclusivo, entorno de desarrollo exclusivo, restricciones exclusivas, sólo en MacOS...), pero hay que reconocer que una vez se da el paso, sientes como te lleva de la mano y te guía en todo momento. Una vez traspasas esas barreras, las sensaciones no pueden ser mejores.

Xcode es un entorno que funciona muy bien, con una distribución de la interfaz intuitiva y funcional. No he experimentado ningún cuelgue ni irregularidad en meses de desarrollo. Muestra unos errores bien identificados y facilita su resolución. Todo funciona como cabría esperar, y eso, en contraste con experiencias en otros entornos, es una maravilla.

Tal vez el contrapunto que podría resaltar, es el proceso de publicación de la App, ya que las firmas y datos relativos se deben introducir manualmente. Es un proceso en el que las primeras veces que se realiza predomina la confusión. En cualquier caso, parece que la automatización del proceso de firmas es una de las novedades del xCode5.0, que estará próximamente disponible.

En todo momento, como he comentado, contamos con una documentación (únicamente en inglés) perfectamente estructurada y útil. Ha sido mi guía de referencia desde el inicio; y por supuesto, cada vez que se experimenta con un elemento nuevo, comenzaremos por su documentación, ya que merece la pena ser consultada. En caso de errores, o algún caso más particularizado del uso de ciertos métodos, existe una gran comunidad activa en internet. Prácticamente "*http://stackoverflow.com*/", se ha convertido en mi página de inicio en el transcurso del proyecto. He resuelto el 90% de las dudas, incidencias o errores que he tenido gracias a la comunidad que respalda esta web. Encarecidamente recomendada si te embarcas en el desarrollo para iOS.

En definitiva, a pesar de los reparos iniciales por su naturaleza cerrada, Apple ha sabido convencerme y me ha gustado la experiencia que me ha ofrecido en el desarrollo orientado a su sistema operativo.

Gestión

El planteamiento e idea iniciales se han mantenido en todo momento, y se han alcanzado todas las metas marcadas. El fijar unas fechas de entregas parciales, ha ayudado a medir y alcanzar las metas progresivamente. Ha sido crucial marcar ciertas pautas para poder avanzar de una manera segura y siendo consciente del estado en el que se encontraba el proyecto en todo momento. Para ello, la inversión en tiempo en la preparación del proyecto ha sido importante y recompensada.

El desarrollo del proyecto ha transcurrido en general en las líneas esperadas. Era consciente de que la fase de formación iba a ser dura y larga, a pesar de no haber estimado en el planteamiento inicial el tiempo necesario. En cualquier caso, he tenido la suerte de que la inversión de tiempo extra en el aprendizaje, me ha facilitado la implementación haciendo las horas dedicadas más eficientes. Por lo que el total de horas estimado ha resultado ser acertado, y se ha cumplido con los requisitos planteados inicialmente. En cualquier caso, para un futuro planteamiento considero que deberé poner especial atención ante elementos desconocidos, y estimar el tiempo suficiente para el dominio de los mismos.

Presente y futuro de McPhone

Se han marcado unas líneas de trabajo para la ampliación futura de McPhone, en un anexo, ya que quedan todas ellas fuera del alcance de este proyecto. Se tuvo la oportunidad de concertar una entrevista con los dos máximos responsables del área de la informática de McDonald's España. Están trabajando en una App con los mismos objetivos que los expuestos en este PFC, por lo que las necesidades y viabilidad de este proyecto son reales. En McDonald's España, barajan una gran cantidad de datos y estadísticas reales. Que las líneas de trabajo e ideas concebidas en este proyecto sean similares a las barajadas por ellos, son muestra de un planteamiento acertado.

Conclusiones finales

Se hace un balance positivo de los elementos que forman parte del PFC. Por lo que una vez alcanzadas las metas marcadas, el desarrollo ha concluido con una satisfacción personal significativa.

En el apartado de "antecedentes personales" ya se mencionaba cual era mi motivación. Esos mismos argumentos son los que ahora se convierten en una realidad y una gran satisfacción personal. Por ello, me enorgullece poder afirmar, que tras años de estudio y trabajo compaginados, he sido capaz de afrontar un proyecto, desde el comienzo hasta el final, que da sentido al esfuerzo invertido. Todo ello fruto de una idea propia, que además tiene una utilidad real y contrastada.

Con el cierre de este documento, soy consciente de que cierro a su vez una larga e importante etapa de formación académica en mi vida, y el broche final no podía ser otro que McPhone.

12. AGRADECIMIENTOS

Un sentido agradecimiento a todas aquellas personas que han ayudado en el desarrollo de este proyecto. Por pequeña que fuera la aportación; gracias.

Marta Villar
Bernardo Beristain
Alejandro Expósito
Salvador Badía
Jose Miguel Blanco
Jon Gonzalez
Ibai Valencia

Así como a los que ayudaron en las pruebas de la aplicación prestando sus dispositivos para dicho fin; gracias.

Gorka Egaña
Iñaki Martín
Diego Villar
Andrés Cavero
Jon Beristain

Y por último, a todas aquellas personas que han apoyado y creído en este proyecto de manera pasiva; gracias.

13.REFERENCIAS

13.1 Enlaces web

https://developer.apple.com/

http://stackoverflow.com/

http://www.desarrolloweb.com/

https://developers.google.com/

http://www.nscodecenter.com/

http://thxou.com/2012/09/11/parsear-y-crear-ficheros-en-formato-json-en-ios/

http://www.programacionios.es/

http://www.google.es

http://wikipedia.org

13.2 Bibliografía

Desarrollo de aplicaciones para iPhone & iPad [Joe Conway & Aaron Hillegas, ANAYA 2010]

Desarrollo de aplicaciones para iPhone & iPad sobre iOS 5 [Rory Lewis, ANAYA 2012]

Desarrollo de aplicaciones para iOS 5 [Wei-Meng Lee, ANAYA 2012]

13.3 Documentación

Documentación propia elaborada en la etapa de formación del proyecto.

Documentación a disposición de los desarrolladores, creada por Apple.

Documentación libre para las APIS de Google, creada por Google Inc.

Proyecto realizado por Iker Beristain Villar.

ANEXO, POSIBLES UPDATES DE MCPHONE

Se tuvo la oportunidad de concertar una entrevista personal en las oficinas de McDonald's España en Madrid, donde se gestiona la marca a nivel nacional. El 28 de mayo del 2013 se mantuvo una reunión con Alejandro Expósito, director digital y de nuevas tecnologías, y Salvador Badía, manager de tecnología de la información.

Se presentó el planteamiento e ideas concebidas para McPhone. Según expusieron, actualmente existen dos desarrollos piloto a nivel mundial para hacer pruebas con la App móvil, una en Francia y otra en Austria. En España, por estrategias corporativas de McDonald's, se aplicará la versión piloto de Francia. Tras analizarlas junto a ellos, básicamente sólo existe una diferencia remarcable en el diseño de la solución que se les ha dado. Es en la detección de la llegada del cliente al restaurante.

> Francia: El teléfono devuelve un código por pantalla que debe ser escaneado en un dispositivo destinado exclusivamente a ello dentro del restaurante, para confirmar que el cliente ha llegado al establecimiento.
>
> Austria: El proceso de detección de la llegada del cliente se realiza de manera desatendida, por medio de geolocalización.

En resumen, se trató de contrastar las ideas iniciales de este proyecto con las que se obtienen de unas necesidades reales basadas en datos. Se confirmó que McPhone estaba desarrollada de manera realista y atendiendo a unas necesidades reales. Se podría considerar que éstos que continúan deberían ser los próximos updates o actualizaciones de McPhone.

UPDATES

- Geolocalización...

  ... como herramienta para la detección de la llegada del cliente al restaurante.

  ... para sugerir el restaurante más cercano.

- Centralizar la base de datos de los productos...

  ... para la carga en tiempo real mediante conexión internet.

  ... para la localización de ofertas y precio por restaurantes.

- Sustituir el recurso en la nube como almacenamiento de pedidos, y adaptarlo a una base de datos en un servidor.

- Pago mediante banca online, paypal, etc. Ofrecer un sistema de reserva del dinero. Sólo se materializa la transacción cuando el sistema reconoce que el cliente está en el restaurante. Sino, se cancelan las transacciones no completadas cada 24h.

- Obtener datos de hábitos de los clientes, para estadísticas internas y posibles promociones.

- Ofrecer alicientes o alianzas con otros servicios para conseguir ser una App atractiva fuera del ámbito de la marca, y alcanzar la meta de ser una de las más descargadas.

ANEXO, MANUAL DEL USUARIO

A continuación se adjunta el documento del manual del usuario de McPhone.

MANUAL DEL USUARIO

McPhone

para iPhone

INDICE

McPhone 

- INICIO ____ 1
- CONSULTA RESTAURANTES (1) ____ 2
- CONSULTA RESTAURANTES (2) ____ 3
- CREAR UN NUEVO PEDIDO (1) ____ 4
- CREAR UN NUEVO PEDIDO (2) ____ 5
- AÑADIR MENÚ AL TICKET (1) ____ 6
- AÑADIR MENÚ AL TICKET (2) ____ 7
- QUITAR PRODUCTO DEL TICKET ____ 8
- ENVIAR PEDIDO (1) ____ 9
- ENVIAR PEDIDO (2) ____ 10
- RECUPERAR ÚLTIMO PEDIDO ____ 11
- CONSULTA WEB DE LA COMPAÑÍA ____ 12

INICIO

McPhone

McPhone es una aplicación móvil que permite gestionar y realizar pedidos a McDonald's desde dónde quiera que estés.

Podrás recoger el pedido en tu restaurante favorito nada más llegar, sin esperar en la cola.

CONSULTA RESTAURANTES (1)

McPhone

a) Pulsa sobre "Restaurantes" en el menú principal de la aplicación.

b) Selecciona el restaurante del que quieras información.

CONSULTA RESTAURANTES (2)

McPhone

c) Puedes cambiar entre diferentes tipos de vistas con el botón "Cambiar Mapa".
Puedes despejar la pantalla con el botón "Quitar botones".

d) Si quieres que el restaurante que tienes en pantalla sea tu restasurante favorito, pulsa "Hacer Favorito".

CREAR UN NUEVO PEDIDO (1)

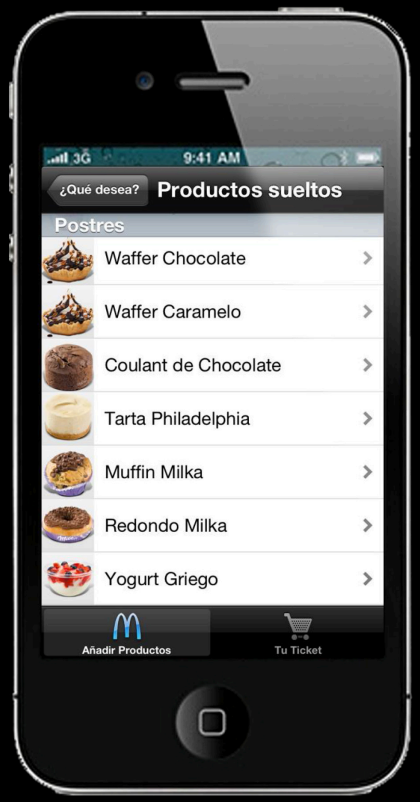
McPhone

a) Presiona sobre "Nuevo Pedido" en el menú principal de la aplicación.

b) Después, selecciona la categoría de productos que quieras del catálogo disponible.

ENVIAR PEDIDO (2)

McPhone



c) Selecciona el producto dentro de la categoría.

d) Pon la cantidad de unidades deseada de ese mismo producto.

e) Pulsa sobre "Añadir al Pedido".

f) Repite los pasos b) a e) hasta haber añadido todos los productos desados.

AÑADIR UN MENÚ AL TICKET (1)

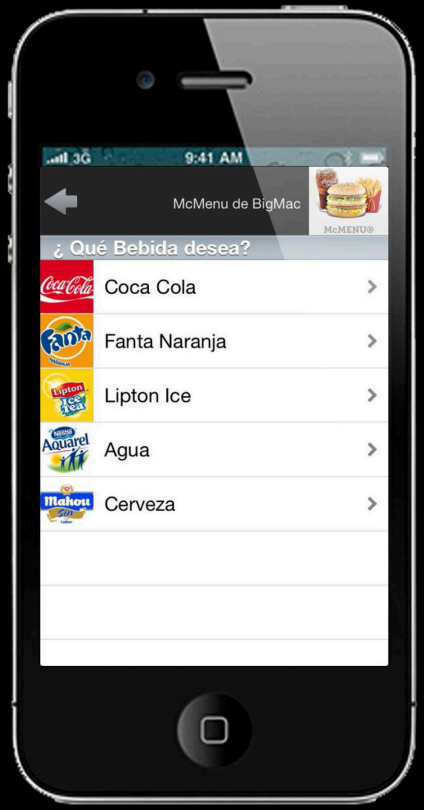
McPhone

A) En el catálogo de productos, pulsa sobre "Nuestros Menús".

B) Ahora selecciona el Sándwich que quieras, o Nuggets si lo prefieres.

AÑADIR UN MENÚ AL TICKET (2)

McPhone



C) Selecciona la bebida que quieras.

D) Elige entre Patatas fritas o Deluxe. Si quieres patatas y bebidas grandes activa el interruptor destinado a ello.

E) Presiona sobre "Añadir al pedido" y el Menú saldrá en Tu Ticket.

QUITAR PRODUCTO DEL TICKET

McPhone

Una vez en "Tu Ticket"...

Opción 1)

Desliza el dedo lateralmente sobre el producto que quieras eliminar del ticket.
Cuando salga el botón rojo, pulsa sobre él.

Opción 2)

Pulsa sobre el botón "edit".
Luego sobre "⛔" en el producto que quieras eliminar.
Cuando salga el botón rojo, pulsa sobre él.

ENVIAR PEDIDO (1)

McPhone

a) Pulsa sobre el botón "¡Listo!" de la pantalla "Tu Ticket".

b) Tienes la opción de añadir el comentario que quieras hacer llegar al restaurante.
Selecciona el restaurante en el que recogerás tu pedido.
Presionando en "Selec. Favorito", la aplicación selecciona automáticamente el restaurante favorito, si lo tuvieras.

ENVIAR PEDIDO (2)

McPhone

c) Pulsa "Enviar".

d) Espera a que se envíe y salga por pantalla tu identificador de pedido

e) Acude a tu restaurante, ¡Buen provecho!

RECUPERAR ÚLTIMO PEDIDO

Pulsa en el menú principal sobre el botón “Último pedido” y verás cómo la aplicación te muestra el Ticket tal y como lo dejaste tras tu último pedido.

CONSULTAR WEB DE LA COMPAÑÍA

McPhone

En el menú principal de la aplicación, presiona el botón "ⓘ" y saldrá un pliegue con la página web de la compañía para que puedas navegar por ella.