



ESTUDIOS DE INGENIERÍA DE TELECOMUNICACIÓN

COMPRARADOR



REALIZADO POR:
ALBERTO MATEOS CHECA

DIRIGIDO POR:
HÉCTOR POMARES CINTAS Y JAVIER PÉREZ FLORIDO

DEPARTAMENTO:
ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

Granada, septiembre de 2012

COMPRARADOR

ALBERTO MATEOS CHECA

PALABRAS CLAVE:

APLICACIÓN MÓVIL, ANDROID, IOS, BLACKBERRY, SYMBIAN, WINDOWS PHONE,
SUPERMERCADO, PRODUCTO, COMPARADOR

RESUMEN:

El presente proyecto tiene como objetivo el diseño y desarrollo de un sistema que permita al usuario visualizar en un dispositivo móvil de última generación, smartphone o tablet, información relativa a productos que son comercializados en diferentes supermercados del país. Asimismo, será posible realizar comparativas de precios de listas de productos creadas por el usuario con el objetivo de facilitarle la elección del establecimiento más económico para realizar la compra.

KEYWORDS:

MOBILE APPLICATION, ANDROID, IOS, BLACKBERRY, SYMBIAN, WINDOWS
PHONE, SUPERMARKET, PRODUCT, COMPARATOR

ABSTRACT:

This project has as main objective the design and development of a system which allows the user to check on a mobile device, smartphone or tablet, information about several products which are traded around different supermarkets in the country. Furthermore, the user will be able to compare prices of products from different lists created by himself and other users in order to find and choose the establishment with lowest prices for the shopping.

D. _____
Profesor del departamento de _____
de la Universidad de Granada, como director del Proyecto Fin de Carrera de D/Dña.

Informa:

que el presente trabajo, titulado:

Ha sido realizado y redactado por el/la mencionado/a alumno/a bajo nuestra dirección, y con esta fecha autorizo a su presentación.

Granada, a ____ de _____ de 20____

Fdo. _____

Los abajo firmantes autorizan a que la presente copia de Proyecto Fin de Carrera se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a ____ de _____ de 20____

(Firmas y números de DNI del alumno y de los tutores)

AGRADECIMIENTOS

En primer lugar quiero agradecer a mis padres, hermano y hermana todo el apoyo y ánimos recibidos así como las facilidades y valores humanos que me han proporcionado, no sólo durante la realización de este proyecto, sino durante toda mi vida.

En segundo lugar a Mati, por haber estado siempre a mi lado ayudándome cuando lo necesitaba, por su inestimable ayuda para grabar los vídeos demostrativos, por compartir su vida conmigo y por los maravillosos momentos que he vivido junto a ella durante los tres últimos años.

A todos aquellos que considero mis amigos por aguantarme y ser como son.
A todos aquellos que en algún momento durante la carrera me han ayudado.

A la comunidad del software libre sin cuyo trabajo desinteresado no habría sido posible realizar este proyecto.

A la música, sin la que la elaboración de este proyecto habría sido una tarea infinitamente más aburrida y por ser uno de los ejes sobre los que gira mi vida.

Para finalizar quiero agradecer a Héctor especialmente por su trato cercano y amigable, por su labor y por todas las facilidades que me ha dado.

A todos y cada uno de ellos, gracias.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	2
1.3. Objetivos	3
2. Estado del arte	5
2.1. Dispositivos móviles	5
2.1.1. Sistemas operativos para dispositivos móviles	6
2.2. Software y servicios relacionados	9
2.2.1. Comparadores de Internet	9
2.2.2. Comparadores para dispositivos móviles	11
3. Proyecto Comprador	19
3.1. Análisis	19
3.1.1. Propósito y alcance	19
3.1.2. Requisitos funcionales	20
3.1.3. Requisitos no funcionales	23
3.1.4. Casos de uso	28
3.1.5. Otros diagramas	58
3.1.5.1. Diagramas de estado	58
3.1.5.2. Diagramas de secuencia	59
3.2. Diseño del sistema	61
3.2.1. Descripción general y arquitectura del sistema	61
3.2.2. Diseño de la base de datos	63
3.2.2.1. Diagrama entidad - relación	63
3.2.2.2. Modelo relacional	66
3.2.3. Diseño del scraper	68
3.2.3.1. Estructura	68
3.2.3.2. Interfaz gráfica	71
3.2.4. Diseño del cliente	71
3.2.4.1. Estructura	71
3.2.4.2. Base de datos	75
3.2.4.3. Interfaz gráfica	76
3.2.5. Diseño del servidor	84
3.2.5.1. Estructura	84
3.2.6. Diseño de las comunicaciones	90
3.2.6.1. Servidor tipo servidor Web	90
3.2.6.2. Servicio web RESTful	95
3.2.7. Matriz de trazabilidad de requisitos-módulos	98

3.3. Implementación	99
3.3.1. Base de datos	100
3.3.2. Programa scraper	100
3.3.2.1. Lenguaje de programación y tecnologías	100
3.3.2.2. Entorno de desarrollo y librerías utilizadas	102
3.3.2.3. Otros aspectos	103
3.3.3. Programa cliente	105
3.3.3.1. Lenguaje de programación y tecnologías	105
3.3.3.2. Entorno de desarrollo y librerías utilizadas	107
3.3.3.3. Otros aspectos	108
3.3.4. Programa servidor	110
3.3.4.1. Lenguaje de programación y tecnologías	110
3.3.4.2. Entorno de desarrollo	112
3.3.4.3. Otros aspectos	113
4. Pruebas y resultados	119
4.1. Entorno de pruebas	119
4.2. Pruebas de funcionamiento de las aplicaciones	120
4.2.1. Pruebas en la aplicación cliente móvil	120
4.2.2. Pruebas en la aplicación scraper	125
4.3. Mediciones de rendimiento	128
5. Planificación y presupuesto	131
5.1. Planificación	131
5.1.1. Fases del proyecto	131
5.2. Presupuesto	133
6. Conclusiones y trabajos futuros	135
6.1. Conclusiones	135
6.2. Trabajos futuros	137
A. Manual de usuario Cliente móvil	141
A.1. Conceptos básicos	141
A.1.1. Apertura del programa y pantalla inicial	141
A.1.2. Navegación	142
A.2. Búsqueda de productos	142
A.2.1. Búsqueda mediante categorías	143
A.2.2. Búsqueda mediante código de barras	147
A.2.3. Búsqueda mediante texto	148
A.3. Gestión de listas de productos	149
A.3.1. Creación de listas	150
A.3.2. Agregar producto a lista	150
A.3.3. Eliminar producto de lista	152
A.3.4. Eliminar lista	152
A.3.5. Modo compra	153
A.4. Comparación de precios de productos	154
A.5. Gestión de favoritos	155
A.5.1. Agregar producto a favoritos	155
A.5.2. Eliminar producto de favoritos	157
A.6. Filtrado	157

B. Manual de usuario de la Scraper	159
B.1. Ejecución	159
B.2. Partes de la GUI	160
B.3. Obtención de datos	160
B.3.1. Productos	160
B.3.2. Categorías	162
B.4. Programación de obtención de datos	162
C. Contenido del CD	165

Índice de figuras

1.2.1. Penetración de smartphones en el mercado por países	3
2.1.1. Estadísticas de mercado de sistemas operativos en smartphones.	6
2.1.2. Evolución de tasa de mercado de SOs.	7
2.1.3. Pantalla inicial de los diferentes sistemas operativos. De izquierda a derecha: Blackberry OS, iOS, Windows Phone y Android.	9
2.2.1. Interfaz web de comparadores de Internet	11
2.2.2. Vistas de la aplicación de Carritus para Android.	12
2.2.3. Vistas de la aplicación Supertruper para Android.	13
2.2.4. Vistas de la aplicación Yibril para Android.	14
2.2.5. Vistas de la aplicación Shopsavvy para Android.	15
2.2.6. Vistas de la aplicación Google Shopper para Android.	16
2.2.7. Vistas de la aplicación Grocery King para Android.	17
3.1.1. Diagrama de casos de uso del cliente	33
3.1.2. Diagrama de casos de uso del cliente	52
3.1.3. Diagrama de casos de uso del servidor	58
3.1.4. Diagrama de estado Scraper	59
3.1.5. Diagrama secuencia CU-1: obtener productos de categorías.	60
3.1.6. Diagrama secuencia CU-8: buscar producto por categorías.	61
3.2.1. Arquitectura general del sistema.	62
3.2.2. Diagrama entidad-relación de la base de datos.	64
3.2.3. Atributos de entidades de tipo categoría	65
3.2.4. Atributos de la entidad Producto	65
3.2.5. Modelo relacional de la base de datos	67
3.2.6. Diseño de la interfaz gráfica de usuario del scraper	72
3.2.7. Diagrama ER base de datos del cliente	76
3.2.8. Modelo relacional base de datos del cliente	77
3.2.9. GUI cliente: pantalla inicial.	78
3.2.10. GUI cliente: pantalla listas.	79
3.2.11. GUI cliente: pantalla lista productos.	79
3.2.12. GUI cliente: pantalla comparación precios.	80
3.2.13. GUI cliente: pantalla comparación precios.	80
3.2.14. GUI cliente: pantalla opciones búsqueda.	81
3.2.15. GUI cliente: pantallas búsqueda por categorías.	81
3.2.16. GUI cliente: pantalla búsqueda por código de barras.	82
3.2.17. GUI cliente: pantallas búsqueda por nombre.	82
3.2.18. GUI cliente: pantalla favoritos.	83
3.2.19. GUI cliente: pantalla detalle producto.	83
3.2.20. Servidor: estructura servidor Web.	84

3.2.21. Servidor: estructura servicio RESTful.	88
3.3.1. Estructura de directorios y ficheros del programa scraper.	103
3.3.2. Scraper: ejemplo diseño interfaz de usuario mediante la librería Profligacy.	105
3.3.3. Estructura de directorios y ficheros del programa cliente.	108
3.3.4. Cliente: ejemplo diseño de pantalla de la aplicación móvil.	110
4.2.1. Vista pantalla principal test aplicación móvil.	121
4.2.2. Resultados tests cliente móvil: PhoneGap API.	122
4.2.3. Resultados tests cliente móvil: comprador.	123
4.2.4. Resultados tests cliente móvil: comprador (1).	124
4.2.5. Resultados tests cliente móvil: comprador (2).	124
4.2.6. Resultados tests cliente móvil: comprador (3).	125
4.2.7. Test scraper: captura ejecución test (1).	126
4.2.8. Test scraper: captura ejecución test (2).	126
4.2.9. Test scraper: captura ejecución test (3).	127
4.2.10. Test scraper: fichero de log para el test.	127
4.2.11. Test scraper: fichero log de errores.	128
5.1.1. Planificación: gráfico de tiempo dedicado por tareas.	133
A.1.1. Manual cliente móvil: inicio programa.	141
A.1.2. Manual cliente móvil: navegación.	142
A.2.1. Manual cliente móvil: búsqueda de productos.	143
A.2.2. Manual cliente móvil: búsqueda de productos mediante categorías (1).	144
A.2.3. Manual cliente móvil: búsqueda de productos mediante categorías (2).	144
A.2.4. Manual cliente móvil: búsqueda de productos mediante categorías (3).	145
A.2.5. Manual cliente móvil: búsqueda de productos mediante categorías (4).	145
A.2.6. Manual cliente móvil: búsqueda de productos mediante categorías (5).	146
A.2.7. Manual cliente móvil: búsqueda de productos mediante categorías (6).	146
A.2.8. Manual cliente móvil: búsqueda de productos mediante código de barras (1).	147
A.2.9. Manual cliente móvil: búsqueda de productos mediante código de barras (2).	148
A.2.10. Manual cliente móvil: búsqueda de productos mediante texto (1).	148
A.2.11. Manual cliente móvil: búsqueda de productos mediante texto (2).	149
A.3.1. Manual cliente móvil: gestión de listas.	149
A.3.2. Manual cliente móvil: creación de listas.	150
A.3.3. Manual cliente móvil: agregar producto a lista (1).	151
A.3.4. Manual cliente móvil: agregar producto a lista (2).	151
A.3.5. Manual cliente móvil: eliminar productos de listas.	152
A.3.6. Manual cliente móvil: eliminar lista (1).	152
A.3.7. Manual cliente móvil: eliminar lista (2).	153
A.3.8. Manual cliente móvil: modo compra.	153
A.4.1. Manual cliente móvil: comparación de productos.	154
A.5.1. Manual cliente móvil: favoritos.	155
A.5.2. Manual cliente móvil: agregar producto a favoritos (1).	156
A.5.3. Manual cliente móvil: agregar producto a favoritos (2).	157
A.5.4. Manual cliente móvil: eliminar favoritos.	157
A.6.1. Manual cliente móvil: filtrado.	158

ÍNDICE DE FIGURAS

B.2.1. Manual scraper: pantalla aplicación.	160
B.3.1. Manual scraper: inicio obtención de datos de productos.	161
B.3.2. Manual scraper: detención obtención de datos de productos.	161
B.3.3. Manual scraper: inicio obtención de datos de menú.	162
B.4.1. Manual scraper: programación de la obtención de datos (1).	163
B.4.2. Manual scraper: programación de la obtención de datos (2).	163
B.4.3. Manual scraper: programación de la obtención de datos (3).	164
C.0.1. Contenido del CD: carpeta principal	165
C.0.2. Contenido del CD: directorio Aplicaciones	165
C.0.3. Contenido del CD: directorio Documentos	166
C.0.4. Contenido del CD: directorio Vídeos demostrativos	166

Índice de tablas

3.1.1.	RF-1: obtener categorías	20
3.1.2.	RF-2: obtener categorías	20
3.1.3.	RF-3: programar obtención productos	21
3.1.4.	RF-4: programar obtención categorías	21
3.1.5.	RF-5: visualizar categorías	21
3.1.6.	RF-6: buscar productos	21
3.1.7.	RF-7: gestionar listas de productos	22
3.1.8.	RF-8: gestionar la pertenencia de productos a listas	22
3.1.9.	RF-9: comprobar listas de productos	22
3.1.10.	RF-10: visualizar listas en modo compra	23
3.1.11.	RF-11: gestionar favoritos	23
3.1.12.	RNF-1: sistema operativo del cliente móvil	24
3.1.13.	RNF-2: conectividad a Internet	24
3.1.14.	RNF-3: plataforma del servidor	24
3.1.15.	RNF-4: puertos habilitados del servidor	25
3.1.16.	RNF-5: MySQL	25
3.1.17.	RNF-6: puertos habilitados de la base de datos	25
3.1.18.	RNF-7: Scraper multiplataforma	25
3.1.19.	RNF-8: Idioma	26
3.1.20.	RNF-9: Acceso protegido a la base de datos	26
3.1.21.	RNF-10: Accesibilidad del cliente	26
3.1.22.	RNF-11: Facilidad de uso	26
3.1.23.	RNF-12: Tiempo de respuesta cliente	27
3.1.24.	RNF-13: Tolerancia a fallos	27
3.1.25.	RNF-14: UTF-8	27
3.1.26.	RNF-15: disponibilidad del servidor y la base de datos	27
3.1.27.	RNF-16: EAN-13	28
3.1.28.	CU-1: obtener categorías	29
3.1.29.	CU-2: programar obtención productos de categorías	30
3.1.30.	CU-3: obtener menú	31
3.1.31.	CU-4: programar obtención de menú	32
3.1.32.	CU-5: mostrar categorías	34
3.1.33.	CU-6: mostrar subcategorías	35
3.1.34.	CU-7: mostrar subsubcategorías	36
3.1.35.	CU-8: buscar producto por categoría	37
3.1.36.	CU-9: buscar producto por nombre	38
3.1.37.	CU-10: buscar producto por código de barras	39
3.1.38.	CU-11: mostrar listas	40
3.1.39.	CU-12: mostrar productos de una lista	41
3.1.40.	CU-13: crear una lista	42

3.1.41. CU-14: eliminar lista	43
3.1.42. CU-15: comprobar lista	44
3.1.43. CU-16: ver lista en modo compra	45
3.1.44. CU-17: agregar producto a lista	46
3.1.45. CU-18: eliminar producto de lista	47
3.1.46. CU-19: mostrar favoritos	48
3.1.47. CU-20: mostrar producto favorito	48
3.1.48. CU-21: agregar producto a favoritos	49
3.1.49. CU-22: eliminar producto de favoritos	50
3.1.50. CU-23: agregar producto favorito a lista	51
3.1.51. CU-24: guardar producto en la base de datos	53
3.1.52. CU-25: guardar categoría/subcategoría/subsubcategoría en la base de datos	54
3.1.53. CU-26: obtener producto de la base de datos	55
3.1.54. CU-27: obtener categoría/subcategoría/subsubcategoría de la base de datos	56
3.1.55. CU-28: obtener precios de la base de datos	57
3.2.1. MOD-SCR-1: GUI	69
3.2.2. MOD-SCR-2: navegador	69
3.2.3. MOD-SCR-3: scraper	70
3.2.4. MOD-SCR-4: log	70
3.2.5. MOD-SCR-5: comunicación	71
3.2.6. MOD-CLI-1: GUI	73
3.2.7. MOD-CLI-2: Notificaciones	74
3.2.8. MOD-CLI-3: Controlador	74
3.2.9. MOD-CLI-4: Comunicaciones	75
3.2.10. MOD-CLI-5: Acceso a datos	75
3.2.11. MOD-SERV-WEB-1: Servidor Web	85
3.2.12. MOD-SERV-WEB-2: Acceso a base de datos	85
3.2.13. MOD-SERV-WEB-3: Comprobar	85
3.2.14. MOD-SERV-WEB-4: Obtener categorías	86
3.2.15. MOD-SERV-WEB-5: Obtener subcategorías	86
3.2.16. MOD-SERV-WEB-6: Obtener subsubcategorías	86
3.2.17. MOD-SERV-WEB-7: Obtener producto	87
3.2.18. MOD-SERV-WEB-8: Guardar categoría	87
3.2.19. MOD-SERV-WEB-9: Guardar subcategoría	87
3.2.20. MOD-SERV-WEB-10: Guardar subsubcategoría	87
3.2.21. MOD-SERV-WEB-11: Guardar producto	88
3.2.22. MOD-SERV-REST-1: Servidor de aplicaciones	89
3.2.23. MOD-SERV-REST-2: Gestor de peticiones	89
3.2.24. MOD-SERV-REST-3: Lógica de negocio	89
3.2.25. MOD-SERV-REST-4: Acceso a base de datos	90
3.2.26. COM-WEB-1: Petición obtención categorías	91
3.2.27. COM-WEB-2: Petición obtención subcategorías	91
3.2.28. COM-WEB-3: Petición obtención subsubcategorías	91
3.2.29. COM-WEB-4: Petición obtención productos categoría	92
3.2.30. COM-WEB-5: Petición obtención producto por identificador	92
3.2.31. COM-WEB-6: Petición obtención producto por nombre	92
3.2.32. COM-WEB-7: Petición obtención producto por código de barras	93

3.2.33. COM-WEB-8: Petición comparación de precios.	93
3.2.34. COM-WEB-9: Petición guardado de categoría.	93
3.2.35. COM-WEB-10: Petición guardado de subcategoría.	94
3.2.36. COM-WEB-11: Petición guardado de subsubcategoría.	94
3.2.37. COM-REST-1: Petición obtención categorías.	96
3.2.38. COM-REST-2: Petición obtención subcategorías.	96
3.2.39. COM-REST-3: Petición obtención subsubcategorías.	96
3.2.40. COM-REST-4: Petición obtención productos categoría.	96
3.2.41. COM-REST-5: Petición obtención producto por identificador.	97
3.2.42. COM-REST-6: Petición obtención producto por nombre.	97
3.2.43. COM-REST-7: Petición obtención producto por código de barras.	97
3.2.44. COM-REST-8: Petición comparación de precios.	97
3.2.45. COM-REST-9: Petición guardado de categoría.	98
3.2.46. COM-REST-10: Petición guardado de subcategoría.	98
3.2.47. COM-REST-11: Petición guardado de subsubcategoría.	98
3.2.48. Matriz de trazabilidad de requisitos-módulos.	99
4.3.1. Medición rendimiento: Gestión de listas.	129
4.3.2. Medición rendimiento: Gestión de favoritos.	129
4.3.3. Medición rendimiento: Búsqueda de productos.	129
5.2.1. Presupuesto del proyecto.	134

Capítulo 1

Introducción

Para comenzar, en este capítulo, se hace un breve repaso histórico de la evolución de los dispositivos electrónicos y de la informática en general. Seguidamente se procede a exponer los hechos y creencias que han propiciado la elaboración de este proyecto así como qué objetivos se persiguen.

1.1. Contexto

Es indudable el hecho de que, desde que se empezaron a utilizar los primeros ordenadores de válvulas de vacío durante la II Guerra Mundial, la industria de la informática ha seguido un desarrollo imparable, incrementándose éste a partir de finales de la década de 1960 cuando se introdujeron los primeros circuitos integrados. Además, el uso de la informática ha traído consigo grandes cambios en la industria y áreas de conocimiento, haciendo que en la actualidad prácticamente cualquier actividad industrial esté estrechamente vinculada con la informática, lo que ha provocado un desarrollo de la sociedad inédito hasta el momento en los miles de años de historia del ser humano.

Son dos hechos fundamentales los que han provocado que la sociedad actual esté marcada por la huella de la informática. Por un lado se encuentra la introducción de los ordenadores en la vida cotidiana de las personas a partir de la creación de los primeros ordenadores personales en la década de 1970 permitiendo que el acceso a la informática no fuera exclusivo para empresas y grandes corporaciones. Por otro lado está la liberación de Internet en 1990 para su uso civil y de forma gratuita, haciendo incluso las formas relacionarse entre las personas hayan cambiado a la par que el uso de Internet se extiende.

Desde la creación de los primeros dispositivos electrónicos los objetivos han sido los mismos que los de la actualidad: construir máquinas con mayor capacidad y menor coste y tamaño. Las mejoras conseguidas en los últimos años han originado que los dispositivos electrónicos hayan podido integrarse en una gran cantidad de productos y sistemas, haciendo posible que éstos hayan sufrido a su vez una revolución paralela a la de la informática.

Como no podía ser de otra manera, el teléfono, uno de los primeros sistemas electrónicos de la historia, se presenta como uno de los grandes beneficiarios de los avances surgidos en el campo de la electrónica. Como consecuencia de la evolución sufrida, en 1979 se presentaba la primera generación de teléfonos móviles, la cual permitía establecer comunicaciones entre dos teléfonos sin necesidad de que éstos dispusieran de cableado alguno.

De una forma lógica, a la vez que las prestaciones de la electrónica aumentaban y el tamaño de los componentes se reducía, las funciones de los teléfonos móviles crecían enormemente. El culmen de la telefonía móvil ha sido la llegada de los llamados smartphones y de sus hermanos mayores, los conocidos como tablets. Estos nuevos dispositivos móviles han abierto nuevas líneas para los usos y aplicaciones informáticas que, hasta hace poco, estaban limitadas a ordenadores personales. Así, los usuarios de estas plataformas pueden acceder ubicua e instantáneamente a múltiples recursos disponibles a través de Internet, así como utilizar numerosas aplicaciones específicas que se están desarrollando para estos dispositivos y que aprovechan sus características tecnológicas.

1.2. Motivación

La creación de este proyecto ha estado basada desde un principio en la idea de que tanto la informática como los avances científicos deben tener un objetivo principal: facilitar al ser humano el desarrollo de las actividades. Bajo esta premisa, el abanico de posibilidades que se abre es prácticamente infinito.

En concreto, el contexto de crisis económica en el que España está inmersa en la actualidad supone un hecho fundamental a la hora de realizar una elección. Actualmente la tasa de paro del país es superior al 20 %, con gran cantidad de familias que tienen graves dificultades para llegar a fin de mes. En este sentido, suponen una gran ayuda para los consumidores los estudios realizados por organizaciones como la Organización de Consumidores y Usuarios (OCU)[43], que ponen de relieve que la diferencia de precio para un mismo producto puede llegar a ser de hasta un 25 % en función del establecimiento donde sea adquirido. Ésto implica la posibilidad de un ahorro superior a los 1000€ al año para un consumidor estándar si elige con acierto el supermercado en el que realizar sus compras.

Por otro lado, en nuestro país existe un número de líneas telefónicas móviles superior al de habitantes, por lo que el uso de este tipo de dispositivos está ampliamente extendido. En concreto, el número de smartphones está creciendo a un ritmo elevado en detrimento de los teléfonos móviles tradicionales, situándose la penetración en el mercado español de éstos en una cifra que, a finales de 2011, ronda el 50 %, la segunda más alta de Europa y América tal y como muestran los estudios llevados a cabo por comScore[72], cuyos resultados se reflejan en la figura 1.2.1. Además, según los estudios realizados por Gartner[73], en el último trimestre de 2011 se vendieron 149 millones de smartphones en todo el mundo, con un incremento en las ventas de un 47,3 % con respecto al año anterior, crecimiento que puede extenderse al comportamiento del mercado en España.

De esta forma, los argumentos anteriores fundamentan la creación de un aplicación para smartphones y tablets que permita al usuario comparar los precios de los productos que comercializan los supermercados más importantes del país y que aconseje al usuario el establecimiento idóneo para realizar la compra. Adicionalmente, se aprecia una carencia de aplicaciones de este tipo para smartphones en España, existiendo únicamente aplicaciones parecidas fuera de nuestras fronteras, con el consiguiente vacío y amplia posibilidad de demanda a priori.

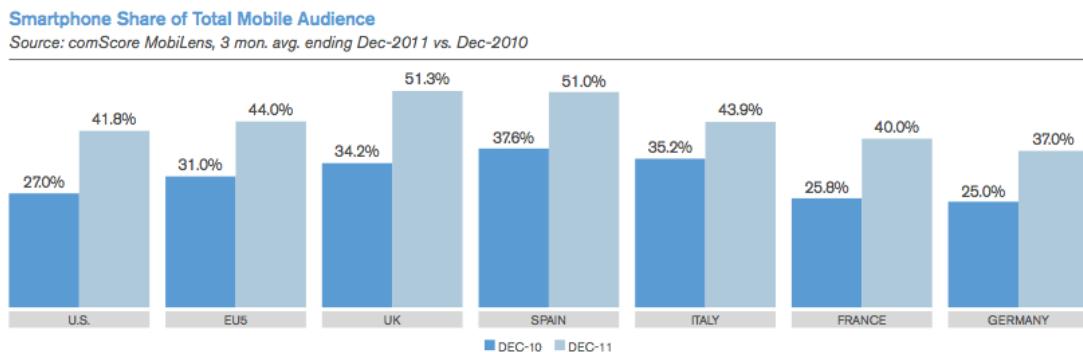


Figura 1.2.1: Penetración de smartphones en el mercado por países

1.3. Objetivos

El objetivo principal de este proyecto es el desarrollo de un sistema que permita al usuario final realizar una elección de supermercado óptima para realizar la compra de una serie de artículos que habrán sido escogidos previamente a través de una tablet o teléfono tipo smartphone. En este sentido, la aplicación que utilizará el usuario final dispondrá de una lista de productos seleccionables organizados en diferentes categorías y de la capacidad de realizar búsquedas de los mismos para facilitar la elección. Con objeto de acelerar el acceso a aquellos artículos que se consulten de una manera más frecuente, será posible marcar productos con la etiqueta de “favoritos”.

Asimismo, podrán almacenarse diferentes listas de productos de forma que, en cualquier momento, el usuario pueda acceder a las mismas para agregar o eliminar productos y/o realizar la comparación de precios de la lista al completo. Además, estas listas supondrán una ayuda adicional para el usuario a la hora de realizar la compra una vez que éste se encuentre dentro del supermercado, siendo posible tachar los productos que ya hayan sido depositados en el carro o cesto de la compra de forma similar a la que se ha venido realizando tradicionalmente mediante papel y lápiz.

En conclusión, el sistema pretende servir de ayuda al usuario final en la tarea cotidiana de realizar la compra de comida y productos de supermercado y permitir a éste un ahorro sustancial en la misma.

Capítulo 2

Estado del arte

En esta sección se realiza un estudio de las tecnologías que guardan relación con este proyecto. Asimismo, también se analizan diferentes alternativas web y para móviles que ofrecen servicios y capacidades relacionadas con el sistema que se desea desarrollar.

2.1. Dispositivos móviles

No existe un consenso claro acerca de la definición de dispositivo móvil. No obstante, todos aquellos aparatos agrupados bajo este término suelen tener ciertas características en común: tamaño reducido, capacidad de procesamiento y memoria limitados, posible conexión a Internet u otra red y elementos de entrada - salida que permitan al usuario interactuar con el dispositivo. Por lo general, suelen ser diseñados para realizar una función específica, aunque adicionalmente pueden llevar a cabo otras tareas. En consecuencia, es habitual hacer uso de este término para referirse a un teléfono móvil, independientemente de las características de éste, aunque también es posible recoger bajo esta definición a aparatos como cámaras de fotos, reproductores de música, PDAs, ordenadores portátiles, etc.

En lo que se refiere a los teléfonos móviles, en la actualidad existe una enorme diversidad en cuanto a capacidades y funcionalidades de los mismos, provocando que se trate de un segmento de alta heterogeneidad. A pesar de que las primeras generaciones de teléfonos móviles únicamente permitían realizar llamadas y mandar mensajes de texto (SMS), en la actualidad las posibilidades son prácticamente ilimitadas. Los nuevos terminales incluyen diversos dispositivos que incrementan drásticamente las posibles funcionalidades de los mismos, siendo posible disponer de pantalla táctil, cámaras de fotos, conexión a Internet, GPS, detectores de campos magnéticos, brújulas, almacenamiento extraíble, acelerómetro y un sinfín de avances tecnológicos embebidos en un mismo aparato además de la capacidad para comunicarse con otros dispositivos que aumenten sus capacidades y/o funciones aun más.

En consecuencia, ha sido necesaria la creación de una nuevo término para hacer referencia a teléfonos tecnológicamente avanzados que poseen gran parte de las características citadas anteriormente, el anglicismo *smartphone*.

Asimismo, a partir del reciente lanzamiento en el mercado del producto denominado iPad[24] por parte de la compañía norteamericana Apple[10], ha aparecido un nuevo dispositivo móvil, el conocido como *tablet PC* o simplemente *tablet*. Este concepto está intimamente relacionado con el de smartphone puesto que, a excepción de las

capacidades telefónicas y la disponibilidad de una pantalla táctil de mayor tamaño, poseen funcionalidades y posibilidades similares, haciendo uso incluso de un mismo sistema operativo en la mayoría de los casos.

2.1.1. Sistemas operativos para dispositivos móviles

Lógicamente, la evolución de las capacidades hardware y funcionales de smartphones y tablets ha propiciado que el problema de la gestión de los mismos aumente drásticamente, dando lugar a una nueva generación de sistemas operativos para estos dispositivos con una naturaleza más compleja y cercana a la de un ordenador personal. En este sentido hay que apuntar que la robustez, estabilidad y fiabilidad de un sistema operativo destinado a ser utilizado en un dispositivo móvil deben ser mucho mayores que las de un ordenador personal. Ésto se debe a que los dispositivos móviles rara vez son reiniciados o apagados y, mucho menos, formateados. A lo anterior hay que sumar que existen grandes limitaciones en cuanto a capacidad de procesamiento, memoria y almacenamiento y que estos dispositivos suelen ser alimentados a través de baterías, con el consiguiente problema de eficiencia en el consumo de la misma que acarrea.

Los estudios realizados por la compañía Gartner[73] señalan que, actualmente, el mercado de los sistemas operativos para smartphones y tablets está dominado por Android[1], iOS(Apple)[22], Symbian[3], RIM[2], Bada[13] y Microsoft[5] tal y como refleja la figura 2.1.1.

**Worldwide Smartphone Sales to End Users by Operating System in 4Q11
(Thousands of Units)**

Operating System	4Q11 Units	4Q11 Market Share (%)	4Q10 Units	4Q10 Market Share (%)
Android	75,906.1	50.9	30,801.2	30.5
iOS	35,456.0	23.8	16,011.1	15.8
Symbian	17,458.4	11.7	32,642.1	32.3
Research In Motion	13,184.5	8.8	14,762.0	14.6
Bada	3,111.3	2.1	2,026.8	2.0
Microsoft	2,759.0	1.9	3,419.3	3.4
Others	1,166.5	0.8	1,487.9	1.5
Total	149,041.8		100.0101,150.3	100.0

Source: Gartner (February 2012)

Figura 2.1.1: Estadísticas de mercado de sistemas operativos en smartphones.

A pesar de que Symbian ha sido un férreo líder durante años, Nokia, desarrollador de éste, anunció a principios de 2011 su paulatina retirada del mercado hasta su definitiva desaparición en 2016[17]en beneficio de Windows Phone, que ha sido el elegido por la marca para ser utilizado en sus próximos lanzamientos. Por el contrario, muy diferente ha sido la evolución de Android. Desde que apareciera el primer teléfono equipado con el sistema operativo apadrinado por Google en octubre de 2008, su ascenso ha sido imparable hasta alcanzar el liderato de ventas durante el primer

cuatrimestre de 2011, situándose en la actualidad con una cuota de mercado superior al 50 % y una media de activación de 850.000 terminales diarios tal y como el propio Andy Rubin (CEO de Google) afirmó con motivo de la la Barcelona WMC [75]. La progresión de Android frente a sus rivales en porcentaje de ventas se muestra en la figura 2.1.2.

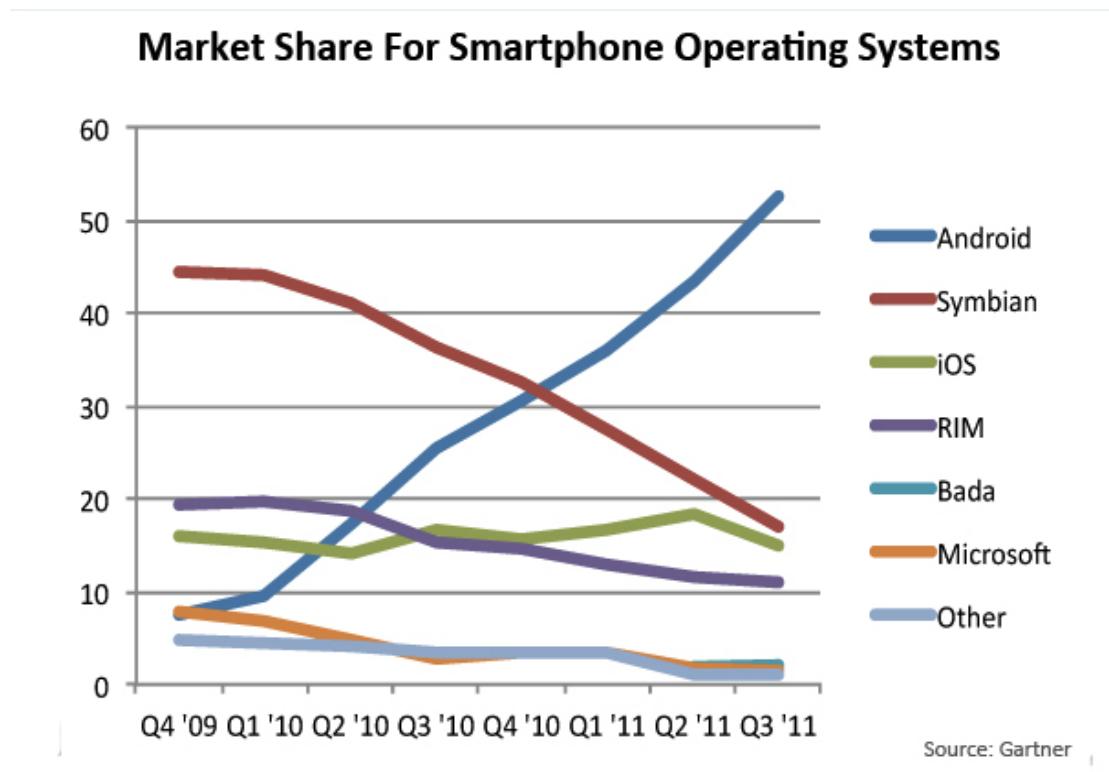


Figura 2.1.2: Evolución de tasa de mercado de SOs.

A continuación se muestra un pequeño análisis de las características de los sistemas operativos destinados a móviles que tienen mayor cuota de mercado.

- **Blackberry OS**[53]: es un sistema operativo propietario desarrollado por la compañía canadiense Research In Motion (RIM) para ser utilizado en sus smartphones conocidos mediante la denominación comercial de Blackberry[14]. Entre las características más notables que tiene este sistema destaca la implementación de la multitarea y, sobre todo, las posibilidades de comunicación y sincronización de correo electrónico y mensajería que ofrece. Puesto que se trata de un sistema propietario, ofrece un excelente soporte para los dispositivos específicos que integran los terminales como, por ejemplo, trackpad[62], trackball[63], trackwheel[64] o pantalla táctil.

En cuanto a lo que a programación se refiere, las aplicaciones que corren sobre este sistema operativo son implementadas en la actualidad utilizando el lenguaje Java. De esta forma, Blackberry dispone de una máquina virtual de Java propia que convierte el código en bytecode apropiado para sus dispositivos. Aunque la programación se puede llevar a cabo con cualquier herramienta, para facilitar la tarea del desarrollador se dispone un plugin para el entorno de desarrollo Eclipse [19] y de un entorno específico llamado BlackBerry JDE, además de un kit de desarrollo Java propio (BlackBerry® Java® SDK v6.0).

- **iOS:** se trata del sistema operativo propietario desarrollado por Apple para sus dispositivos móviles. Fue originalmente desarrollado para el smartphone iPhone[25] siendo, posteriormente, extendido su uso al tablet iPad y a otros dispositivos de la marca. El desarrollo de aplicaciones para esta plataforma se realiza mediante el lenguaje objective-c[42] y viene impuesto por el uso obligatorio de el entorno de desarrollo XCode[69], propio también de la marca norteamericana. Consecuentemente, es indispensable disponer de un ordenador con sistema operativo Mac OS puesto que dicho entorno de desarrollo puede ejecutarse únicamente sobre éste. XCode dispone de todas las herramientas necesarias para la programación de aplicaciones para iOS, facilitando la tarea al programador.
- **Windows Phone:** creado por Microsoft, ha sido el último en aparecer el mercado, siendo el sustituto de Windows Mobile. A diferencia de los dos sistemas anteriores, éste no está orientado hacia un dispositivo en concreto ya que puede encontrarse corriendo en productos de diferentes fabricantes. Por este motivo, para facilitar la compatibilidad con los diferentes dispositivos se establecen una serie de requisitos mínimos que éstos deben cumplir como son la pantalla táctil, un procesador ARM[12] y un mínimo de memoria RAM de 256 MB entre otros. La programación de aplicaciones para este sistema propietario debe realizarse utilizando utilizando las herramientas que proporciona el kit de desarrollo Windows Phone SDK, el cual se integra dentro del entorno Visual Basic IDE[4] de Microsoft. De esta forma, se plantea como indispensable el hecho de disponer de un ordenador con sistema operativo Windows para la creación de aplicaciones. Además, es posible crear aplicaciones haciendo uso de los frameworks XNA[71] y Silverlight[58] desarrollados también por Microsoft.
- **Android:** se trata del sistema operativo líder del mercado actualmente y cuyo desarrollo está impulsado principalmente por el gigante de Internet Google. Está basado en el kernel 2.6 de linux y sigue la filosofía de código abierto (open source)[45]. La programación de aplicaciones para este sistema puede realizarse en cualquier plataforma, proponiéndose el entorno de desarrollo Eclipse junto con el plugin creado para éste como principal alternativa. De esta forma, la programación se lleva a cabo principalmente en lenguaje Java a través del kit de desarrollo SDK de Android. No obstante, existe la posibilidad de hacer uso de otro kit de desarrollo, el NDK (Native Development Kit), que permite la creación de código nativo para procesadores ARM a través del uso de librerías en lenguaje C no estándares. De la misma forma que Windows Phone, Android no ha sido planteado para ser utilizado en un dispositivo en concreto, pudiendo ser utilizado tanto en tablets como en smartphones de diferentes fabricantes y con diferentes características.

Tras el breve análisis expuesto anteriormente hay que destacar que, a pesar de las particularidades propias de cada sistema operativo, todos albergan grandes similitudes en la medida de que permiten controlar la gran cantidad de dispositivos electrónicos que se encuentran embebidos en un smartphone a través de excelentes APIs, disponen de multitarea y ofrecen la posibilidad de utilizar uno o varios entornos de desarrollo que integran multitud de herramientas como depuradores o emuladores que facilitan considerablemente la labor del desarrollador. Adicionalmente, todos los sistemas analizados disponen de una plataforma que facilitan la descarga a través de Internet e



Figura 2.1.3: Pantalla inicial de los diferentes sistemas operativos. De izquierda a derecha: BlackBerry OS, iOS, Windows Phone y Android.

instalación de aplicaciones, gratuitas o de pago. Entre ellos destaca la AppStore[11] de iOS con más de 600.000 aplicaciones disponibles, seguida por el Google Play (anteriormente conocido como Android Market)[37] de Android con un número de aplicaciones superior a las 500.000 y que destaca por ser el que mayor número de aplicaciones gratuitas oferta. En este sentido es destacable el hecho de que existen gran cantidad de aplicaciones, muchas de ellas de renombrado éxito, que, aunque son de pago en iOS, Windows Phone y BlackBerry, se ofrecen de forma gratuita para Android.

2.2. Software y servicios relacionados

Los sistemas dedicados a la comparación de artículos llevan cosechando un gran éxito desde hace tiempo. Ejemplos de ésto pueden ser Trivago[65] o Kayak[34], comparadores de precios de hoteles y de vuelos respectivamente que operan a través de Internet y que obtienen miles de visitas diarias. El caso de la comparación de artículos de supermercados es más reciente, existiendo en España muy pocas alternativas serias y que funcionen correctamente. A continuación se van a detallar brevemente algunos de los sistemas más logrados que permiten realizar comparaciones de precios de productos de supermercado a través de Internet y de dispositivos móviles.

2.2.1. Comparadores de Internet

- **www.mysupermarket.co.uk:** puede que se trate del comparador de productos de supermercado con mayor número de usuarios y de mayor importancia de la red. Opera únicamente en Reino Unido y dispone de una interfaz web amigable y desenfadada en la que los productos están organizados mediante categorías, lo que facilita el acceso a los mismos. Permite ver el precio de productos que se venden en 4 de los mayores supermercados del país y ofrece una gran cantidad de información sobre los productos como precio, peso, calorías y otras características nutricionales. Además, permite realizar la compra directamente a través de la web, siendo ésta enviada al domicilio del usuario. Las excelentes capacidades citadas anteriormente se fundamentan en el hecho de que la web opera

gracias a convenios con los diferentes supermercados, los cuales posibilitan a la plataforma el acceso a la información necesaria para su funcionamiento.

- **www.carritus.com:** puede decirse que es lo más parecido a mysupermarket que puede encontrarse en España. Desde su creación no ha parado de crecer, consolidándose en poco tiempo como el mejor servicio de este tipo en nuestro país. Ofrece una estupenda interfaz interactiva que permite acceder al precio de los productos así como crear listas de éstos y realizar la compra en el supermercado elegido a través de la propia web. Asimismo, es capaz de ofrecer los precios y los supermercados disponibles según el código postal del usuario y de proponer productos similares en caso de que el que se busca no pueda encontrarse en un supermercado. Por si fuera poco, es posible importar las listas confeccionadas a través de la web del supermercado habitual del usuario para realizar la comparación de precios en Carritus. También dispone de un planificador de menús con el que se propone al usuario una serie de comidas con receta incluida y la posibilidad de realizar la compra de los productos necesarios.
- **www.yibril.com:** es una red social orientada a la publicación de precios de productos por parte de los usuarios de la misma. De esta forma, son los propios usuarios o empresas los que agregan nuevos establecimientos y precios a la plataforma. Es necesario disponer de una cuenta de usuario para acceder a los servicios aunque también se permite el acceso mediante cuenta de Facebook. Asimismo, como red social que es, permite crear redes de amigos y publicar comentarios en la web.

Una de las principales ventajas de esta plataforma es la posibilidad de interacción de los usuarios. De esta forma, el número de establecimientos es muy elevado, encontrándose muchos de ellos que son pequeñas tiendas y que no disponen siquiera de página web donde publicar sus productos. Además, permite que los usuarios alerten a los demás de nuevas ofertas u oportunidades encontradas. Por otro lado, es posible confeccionar listas de productos y buscar productos para su consulta de precio. Asimismo, también ofrece un mapa basado en la tecnología de Google Maps[36] que muestra la localización de todos los establecimientos, cerca de 400 en el momento de esta redacción, que se encuentran en la base de datos de la plataforma. Por último, hay que destacar que dispone de una aplicación para Android descargable gratuitamente desde el Market de Android.

En lo que respecta a los aspectos negativos de la plataforma, hay que destacar el hecho de que, debido a que no existe un completo control sobre los productos que agregan los usuarios, es habitual encontrar productos repetidos, lo que hace más complicado la localización de los mismos. Además, no se realiza una ordenación de los productos disponibles, sino que es necesario hacer uso del buscador que incorpora la web para acceder al precio y características de cada uno.

- **www.ahorrarte.com:** permite la comparación del precio de productos y la consulta de precios. Opera únicamente con supermercados de la comunidad de Canarias y con los establecimientos asociados a Carrefour, Mercadona, Hiperdino y Dia. Para hacer uso del servicio es necesario disponer previamente de una cuenta de usuario, la cual puede obtenerse de forma gratuita a través de la web.

En cuanto a los aspectos positivos hay que decir que permite crear listas de productos para su posterior comparación y que existe la posibilidad de seleccionar el

establecimiento concreto pudiendo escoger uno de entre los diferentes que haya en una ciudad pertenecientes a una misma cadena de supermercados.

Los aspectos más negativos encontrados son el limitado número de productos de los que se ofrece el precio, la desactualización de los mismos (no han sido actualizados desde 2010) y la limitación establecida a un número máximo de 40 productos por lista de la compra.

Por último, cabe señalar que existen dos plataformas adicionales de comparación de precios como son www.comparasuper.com y www.supercomprador.es aunque se ha decidido no analizarlos en este documento puesto que parece que ninguno de los dos se consolidó como una alternativa seria, datando las actualizaciones de sus precios de principios de 2010. No obstante, a pesar de que ambos tienen una interfaz poco cuidada y elaborada, disponen de una funcionalidad no apreciada en ninguno de los servicios analizados anteriormente como es la posibilidad de visualizar gráficamente la progresión sufrida por los precios durante un periodo de tiempo.

(a) mysupermarket.co.uk

(b) carritus.com

(c) yibril.com

(d) ahorrarte.com

Figura 2.2.1: Interfaz web de comparadores de Internet

2.2.2. Comparadores para dispositivos móviles

La aparición en el mercado de los dispositivos móviles y la gran aceptación de éstos por parte de los usuarios han provocado que el desarrollo de aplicaciones de comparación de productos haya proliferado velozmente. En este sentido hay que destacar que las tendencias han sido dos. La primera de ellas se basa en la creación de aplicaciones para dispositivos móviles que acceden a servicios que previamente se ofertaban a través de la web, es decir, se trata de clientes para dichos servicios enfocados para ser utilizados mediante dispositivos móviles. La segunda de ellas es más radical y está

basada en el desarrollo de aplicaciones que no se sustentan en servicios disponibles anteriormente, sino que todo el sistema utilizado es implementado desde cero.

De esta forma, las aplicaciones más conocidas, útiles y con mayores capacidades que facilitan al usuario la comparación de precios de productos de supermercado se estudian a continuación.

- **Carritus:** aplicación desarrollada para iOS que puede descargarse gratuitamente desde el AppStore. Es la aplicación oficial de la página web www.carritus.com, anteriormente analizada. Por tanto, posee funciones similares a las de la citada web, entre las que destacan:

- Búsqueda de productos mediante texto y navegación por categorías.
- Permite comparar entre una gran cantidad de supermercados.
- Muestra información sobre los productos, con imagen incluída.
- Es posible crear listas de productos
- Permite realizar la compra de los artículos a través del teléfono.
- Ofrece un histórico con las últimas compras realizadas

De entre todas las aplicaciones analizadas, ésta parece ser la más completa y útil, además, de la más fiable, puesto que los precios de los productos no son actualizados por los usuarios de la aplicación. No obstante, debido a que se trata de una aplicación desarrollada únicamente para iOS, no ha podido ser analizada en detalle puesto que no se dispone de ningún dispositivo capaz de ejecutar la aplicación.

Productos		Mis compras	
ANIMALES	>	Compra en Mercadona del 31...	
APERITIVOS Y FRUTOS SECOS	>	MERCADONA Precio TOTAL 126.15 €	>
ARROZ, PASTAS, HARINAS Y LEGUMBRES	>	0:59 - 31/01/12	
BAZAR	>	Compra en Mercadona del 25...	
BEBE	>	MERCADONA Precio TOTAL 119.80 €	>
BEBIDAS	>	11:13 - 25/10/11	
CALDOS, SOPAS Y PURÉS	>	Compra en Caprabo del 17/10...	
		caprabo Precio TOTAL 39.97 €	>
		10:12 - 17/10/11	
		Compra en Mercadona del 10...	
		MERCADONA Precio TOTAL 36.46 €	>
		11:47 - 10/10/11	

(a) Categorías de productos

(b) Histórico de compras

Figura 2.2.2: Vistas de la aplicación de Carritus para Android.

- **Supertruper:** funciona tanto en iOS como en Android y se ofrece de forma gratuita a través del AppStore y Google Play. Inicialmente fue lanzada sólo para iOS y posteriormente para Android en junio de 2011, contando en la actualidad con un número de descargas comprendido entre 100.000 y 500.000 en Android. Las funciones más relevantes de la aplicación se muestran a continuación

- Búsqueda de productos mediante código de barras y texto.
- Permite la creación de varias listas de productos.
- Muestra información acerca de los productos con imagen del mismo incluida en la mayoría de las ocasiones.
- Ofrece la posibilidad al usuario de agregar productos a la base de datos.
- Elevado número de supermercados.
- Sugiere productos similares al usuario.

El servicio ofrecido por esta aplicación es realmente bueno, siendo capaz de encontrar una gran cantidad de productos. No obstante, en las pruebas realizadas se ha podido comprobar que no encuentra muchos de los productos habituales que se venden en un supermercado. Además, a pesar de que los casos de éxito son elevados, éstos se producen de forma parcial debido a que, aunque encuentra el producto deseado, lo hace únicamente en algunos supermercados, teniendo la certeza de que ese mismo producto está disponible para su venta en otros establecimientos que, se supone, la aplicación incluye. Ésto es debido a que la aplicación basa su funcionamiento en que son los propios usuarios los que suben los productos y los precios de los mismos al servidor de la aplicación, por tanto, el precio obtenido es poco fiable, pues depende de la participación activa de éstos.

Asimismo, como aspecto negativo también hay que decir que el funcionamiento completo del programa está basado en peticiones a Internet, por lo que no es posible utilizarlo sin conexión a excepción de la visualización de las listas de productos ya que se realiza un guardado en caché de la información necesaria para su consulta sin conexión. Por otro lado, las sugerencias de productos que realiza, bajo mi punto de vista, no son del todo útiles ya que sugiere productos de la misma marca que el objetivo de la búsqueda y no del mismo tipo. Por último hay que destacar que dispone de una interfaz bastante lograda que permite una navegación fácil aunque no permite una navegación por categorías de productos.



Figura 2.2.3: Vistas de la aplicación Supertruper para Android.

- **Yibril:** se trata del cliente para dispositivos Android que permite acceder a los servicios ofertados por el comparador web analizado anteriormente www.yibril.com. La aplicación se puede obtener de forma gratuita a través del Market de Android, algo que han hecho ya entre 1.000 y 5.000 usuarios. Entre sus funciones principales se encuentran las siguientes:

- Búsqueda de productos mediante código de barras y texto.
- Creación de una única lista de productos.
- Modificación de precios de productos.
- Envío de comentarios a la comunidad de usuarios.

La aplicación se encuentra aún en fase de desarrollo, de igual forma que la plataforma web, por lo que el funcionamiento de la misma es pobre. Su interfaz de usuario está poco cuidada y es demasiado simple. Además, la información que ofrece acerca de los productos es, en la mayoría de los casos, muy escasa, destacando el bajo número de productos disponibles para su consulta.



Figura 2.2.4: Vistas de la aplicación Yibril para Android.

- **Shopsavvy:** es una aplicación gratuita disponible para Android e iOS que cuenta con un número de descargas superior a las 250.000 (en Android) y que permite buscar el precio e información acerca de un producto cualquiera. No obstante, puesto que se trata de una aplicación desarrollada fuera de España, no obtiene buenos resultados para productos que se venden en nuestro país y, en especial, para artículos de alimentación. Sus características más reseñables son:

- Búsqueda de productos mediante lectura de códigos de barras y texto.
- Obtiene los precios tanto de tiendas online como de tiendas físicas.
- Permite crear listas y exportalas a formato CSV.
- Sugiere ofertas al usuario.

- Muestra en un mapa los establecimientos en los que se puede comprar el artículo buscado.
- Dispone de historial de búsquedas.
- Muestra las últimas búsquedas realizadas por otros usuarios del sistema.
- Permite a los usuarios agregar nuevos productos a la base de datos.
- Posibilita compartir la información obtenida en diferentes redes sociales y a través de correo electrónico.

Se trata sin duda de una aplicación realmente completa y de buen comportamiento, destacando entre sus cualidades la calidad y rapidez de su función de escaneo de códigos de barras. La interfaz de usuario es muy completa provocando quizás que, en algunas ocasiones, se muestre excesivamente compleja para usuarios poco hábiles. Es resaltable el hecho de la capacidad de exportación y posibilidades de compartición que ofrece, estimándose éstas como realmente útiles. Por último, como aspecto negativo hay que hacer referencia al hecho de que la aplicación está disponible únicamente en inglés.

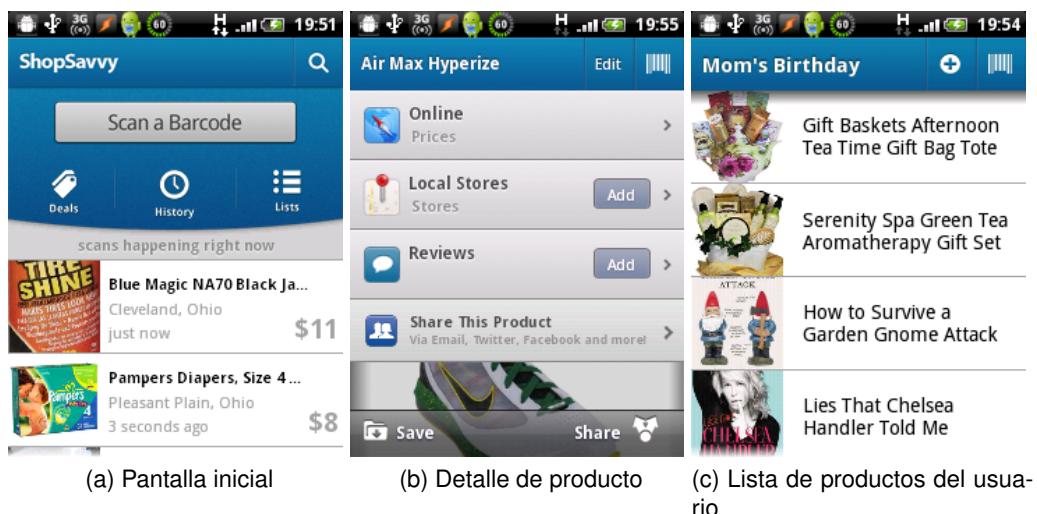


Figura 2.2.5: Vistas de la aplicación Shopsavvy para Android.

- **Google Shopper:** se trata de un servicio de búsqueda de precios e información relativa a un producto ofrecido de forma gratuita por Google a través de la web y de su aplicación para Android. El acceso a este servicio está limitado únicamente para usuarios de Estados Unidos y Reino Unido. De igual forma que la aplicación anterior, los resultados obtenidos en el campo de los supermercados son nulos ya que está orientada a la búsqueda de libros, discos, y videojuegos principalmente. Las características principales de esta aplicación son las siguientes:

- Además del precio e información del producto obtiene los comentarios realizados por los usuarios acerca de éste.
- La búsqueda puede realizarse a través del escaneo del código de barras, mediante una fotografía del producto, texto y voz.

- Almacena un historial de búsquedas que puede ser consultado sin conexión a Internet.
- Recomienda al usuario productos que están en oferta.
- Permite dar puntuaciones a los artículos.

Es de rigor en este caso destacar las posibilidades novedosas que ofrece en cuanto a posibilidades de búsqueda como son la búsqueda mediante fotografía do del producto en cuestión y mediante el uso de la voz. Asimismo, es notable el hecho de que muestra una cantidad de información acerca de los productos muy superior al resto de las aplicaciones, siendo muy positivo el hecho de que es posible visualizar las experiencias que otros usuarios han escrito acerca del producto. Además, cuenta a su favor con la posibilidad de consultar búsquedas anteriores sin necesidad de conexión a Internet, algo que puede ser de gran utilidad para usuarios que no disponen de una conexión permanente. En lo que respecta a su interfaz gráfica, a pesar de su simplicidad, es directa, útil y práctica. Como punto negativo es destacable que no es posible utilizarlo en España, al menos de forma oficial, y que no pueden elaborarse listas de artículos.

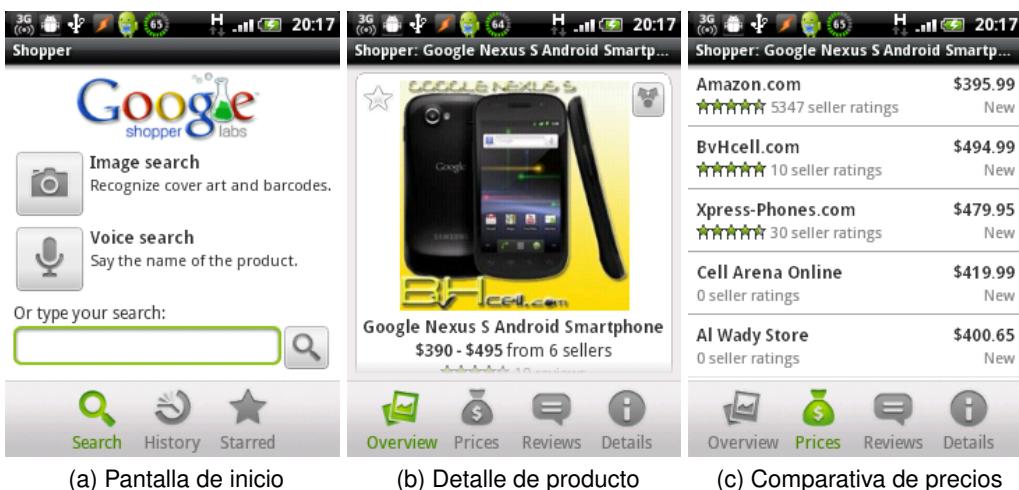


Figura 2.2.6: Vistas de la aplicación Google Shopper para Android.

Como puede comprobarse, el número de alternativas disponibles que sean capaces de proporcionar el precio de los productos es muy reducido, por lo que las posibilidades de mercado se atisban como amplias y generosas. Además, todas las aplicaciones encontradas están disponibles únicamente para Android y/o iOS, por lo que se deja a un lado a sistemas operativos como Windows Phone, RIM (Blackberry), Symbian y Bada cuya implantación conjunta supera el 20 % del mercado.

Por último, cabe destacar que existen una gran cantidad de aplicaciones para dispositivos móviles que permiten almacenar listas de productos como mero recordatorio a la hora de realizar la compra de forma física en una tienda. De esta forma, muchas de ellas permiten llevar un seguimiento del gasto realizado y/o del precio de los productos pero, en todo caso, tiene que ser introducido manualmente.

Entre este grupo destaca por su conseguida interfaz y prestaciones así como por su número de descargas la aplicación **Grocery King**, la cual ha sido descargada entre 10.000 y 50.000 veces desde el Market y tiene un precio de 4.99€. Entre las bondades de este programa destaca la excelente navegación que permite para seleccionar productos debido a la categorización que realiza de los mismos así como la posibilidad de establecer 3 tipos de listas: lista de productos genéricos, lista de un producto de un supermercado específico y lista de productos en base a una receta (incluye la posibilidad de almacenar la propia receta). Además, permite establecer productos dentro de la categoría de favoritos para posibilitar un posterior acceso a los mismos más rápido y fácil. Por último, hay que destacar la elaborada interfaz de usuario que posee.



Figura 2.2.7: Vistas de la aplicación Grocery King para Android.

Capítulo 3

Proyecto Comprador

Una vez realizado el análisis previo que nos ayuda a tener una visión general del ecosistema que enblogaba al sistema, se pasa a la fase en la que se define cómo se va a desarrollar la aplicación.

Primeramente se realiza un análisis de los requisitos que deberá de cumplir el sistema, estableciendo una serie de casos de uso. A continuación, se procede al diseño de las diferentes aplicaciones que compondrán el sistema. Por último, se detallarán los aspectos más relevantes de la implementación de las aplicaciones.

3.1. Análisis

En este punto se definen el propósito del sistema, los requisitos funcionales y no funcionales a los que éste ha de ajustarse y los casos de uso.

Por otro lado, aunque se hará una exposición más detallada en la sección dedicada al diseño, para facilitar la descripción de los requisitos y los casos de uso de uso es necesario adelantar que, arquitectónicamente, el sistema estará compuesto por un servidor con acceso a una base de datos, un cliente móvil que será la aplicación que manejará el usuario y, por último, un programa que tendrá como objetivo extraer de Internet, en concreto de la web www.carritus.com, la información relativa a los productos y categorías que permitan clasificar a éstos.

3.1.1. Propósito y alcance

El propósito principal de la aplicación será permitir comprobar el precio de productos de venta en supermercados así como aconsejar al usuario sobre el establecimiento en el que podrá realizar la compra de uno o varios artículos a un precio más bajo.

En cuanto a los supermercados que participarán en la comparación de precios, el proyecto se limitará a cuatro establecimientos del territorio español: Carrefour, Mercadona, El Corte Inglés e Hipercor.

Por otro lado, el alcance del proyecto está limitado de una forma muy estricta por el hecho de que los datos relativos a los productos así como las categorías que, a la postre, permitirán clasificarlos de forma ordenada serán extraídos de la web www.carritus.com. De esta forma, habrá que ceñirse a la información que dicha web proporciona para cada producto así como al número de artículos de que ésta dispone y las categorías de las que hace uso. Aunque lo anterior supone que el proyecto tenga que ceñirse a una

web ajena, facilita en gran medida la extracción de datos, ya que será necesario construir un único programa que obtenga la información necesaria para la implementación del proyecto, en lugar de una aplicación para la web de cada supermercado.

Por tanto, se desarrollarán todas las funcionalidades que sean necesarias para la consecución del propósito anterior de la mejor manera posible teniendo en cuenta los requisitos que se definirán en los siguientes puntos.

3.1.2. Requisitos funcionales

En este apartado se van a describir los requisitos funcionales del sistema, es decir, lo que éste tiene que hacer bajo la perspectiva del usuario. Para ello se hará uso, como soporte, de tablas que contendrán los siguientes conceptos:

- Identificador: se trata de un nombre único que tendrá el formato RF-X donde el valor X será un número que comenzará en 1 y se irá incrementando progresivamente para cada uno de los requisitos que se definen.
- Descripción: explicación detallada del requisito.
- Necesidad: indica la exigencia de implementación del requisito en el proyecto. Toma valores entre 1 y 3, siendo el segundo valor el nivel de necesidad más alto.
- Prioridad: establece el nivel de preferencia del requisito dentro del proyecto. Toma valores entre 1 y 3, siendo el segundo valor el nivel de prioridad más alto.

A continuación se detallan todos y cada uno de los requisitos funcionales propuestos para el sistema.

RF-1: Obtener productos	
Descripción:	El administrador del sistema podrá extraer la información de los productos de la web carritus.com
Necesidad:	3
Prioridad:	3

Tabla 3.1.1: RF-1: obtener categorías

RF-2: Obtener categorías	
Descripción:	El administrador del sistema podrá extraer la información de las categorías que clasifican a los productos de la web carritus.com
Necesidad:	3
Prioridad:	3

Tabla 3.1.2: RF-2: obtener categorías

RF-3: Programar obtención productos
Descripción:
El administrador del sistema deberá de disponer de la capacidad de programar una obtención automatizada de los productos de la web carritus.com estableciendo una frecuencia de obtención en días y una hora de inicio.
Necesidad: 2
Prioridad: 1

Tabla 3.1.3: RF-3: programar obtención productos

RF-4: Programar obtención categorías
Descripción:
El administrador del sistema podrá programar una obtención automatizada de las categorías de productos de la web carritus.com estableciendo una frecuencia de obtención en días y una hora de inicio.
Necesidad: 2
Prioridad: 1

Tabla 3.1.4: RF-4: programar obtención categorías

RF-5: Visualizar categorías
Descripción:
El usuario será capaz de ver en la pantalla de su dispositivo móvil el conjunto de categorías, subcategorías y subsubcategorías que facilitan la clasificación de los productos.
Necesidad: 3
Prioridad: 2

Tabla 3.1.5: RF-5: visualizar categorías

RF-6: Buscar productos
Descripción:
El usuario podrá realizar búsquedas de productos, proporcionandosele varias alternativas, para visualizar los detalles del artículo.
Necesidad: 3
Prioridad: 3

Tabla 3.1.6: RF-6: buscar productos

RF-7: Gestionar listas de productos
Descripción: El usuario tendrá la posibilidad de gestionar listas de productos, es decir, podrá: <ul style="list-style-type: none">○ Crear listas○ Visualizar las listas creadas○ Eliminar listas○ Ver los productos de una lista
Necesidad: 3
Prioridad: 2

Tabla 3.1.7: RF-7: gestionar listas de productos

RF-8: Gestionar la pertenencia de productos a listas
Descripción: El usuario dispondrá de las capacidades necesarias para: <ul style="list-style-type: none">○ Agregar productos a una lista, incluyendo la posibilidad de añadir el producto agitando el dispositivo móvil.○ Eliminar productos de una lista
Necesidad: 3
Prioridad: 2

Tabla 3.1.8: RF-8: gestionar la pertenencia de productos a listas

RF-9: Comprobar listas de productos
Descripción: El usuario podrá comprobar una lista de productos, es decir, comparar el precio conjunto de los artículos que pertenecen a la lista en los diferentes supermercados que soporta la aplicación.
Necesidad: 3
Prioridad: 3

Tabla 3.1.9: RF-9: comprobar listas de productos

RF-10: Visualizar listas en modo compra
Descripción:
El usuario tendrá la posibilidad de visualizar una lista en modo compra, de forma que pueda tachar y/o destachar los productos de una lista, simulando el comportamiento de una lista de la compra tradicional confeccionada mediante papel y lápiz.
Necesidad: 1
Prioridad: 1

Tabla 3.1.10: RF-10: visualizar listas en modo compra

RF-11: Gestionar favoritos
Descripción:
El usuario podrá gestionar una lista de productos de productos favoritos, siendo posible:
<ul style="list-style-type: none"> ○ Agregar productos a favoritos ○ Ver los productos favoritos ○ Eliminar productos de favoritos ○ Agregar un producto favorito a una lista
Necesidad: 1
Prioridad: 3

Tabla 3.1.11: RF-11: gestionar favoritos

3.1.3. Requisitos no funcionales

Seguidamente se van a exponer los requisitos no funcionales del sistema, definiendo éstos las restricciones que afectan a los servicios y funcionalidades que ofrece el sistema. De forma similar al punto anterior, los requisitos se muestran en forma de tablas con los mismos campos utilizados para los requisitos funcionales pero con la excepción de que el identificador tomará ahora el formato RNF-X.

RNF-1: Sistema operativo del cliente móvil
Descripción: La aplicación móvil correrá sobre un dispositivo móvil que disponga de uno de los siguientes sistemas operativos: <ul style="list-style-type: none">○ Android 2.2 o superior○ Blackberry OS 5.0 o superior o Blackberry Playbook OS○ iOS 3.0 o superior○ Symbian S60 o superior○ WebOS 3.0 o superior○ Windows Phone 7 o superior○ Bada 1.2 o superior
Necesidad: 3
Prioridad: 3

Tabla 3.1.12: RNF-1: sistema operativo del cliente móvil

RNF-2: Conectividad a Internet
Descripción: La aplicación deberá de disponer de conectividad a Internet para la consulta de productos y la comprobación de listas. El servidor y el software para obtener los productos y categorías deberán de tener una conexión permanente a Internet.
Necesidad: 3
Prioridad: 3

Tabla 3.1.13: RNF-2: conectividad a Internet

RNF-3: Plataforma del servidor
Descripción: La aplicación que actuará como servidor deberá de implementarse sobre un servidor HTTP Apache o sobre un servidor de aplicaciones Glassfish.
Necesidad: 3
Prioridad: 3

Tabla 3.1.14: RNF-3: plataforma del servidor

RNF-4: Puertos habilitados del servidor
Descripción:
El servidor aceptará las peticiones en el puerto 80.
Necesidad: 2
Prioridad: 1

Tabla 3.1.15: RNF-4: puertos habilitados del servidor

RNF-5: MySQL
Descripción:
La base de datos se implementará utilizando como base MySQL.
Necesidad: 2
Prioridad: 2

Tabla 3.1.16: RNF-5: MySQL

RNF-6: Puertos habilitados de la base de datos
Descripción:
La base de datos hará uso del puerto 3306 para recibir las peticiones.
Necesidad: 2
Prioridad: 1

Tabla 3.1.17: RNF-6: puertos habilitados de la base de datos

RNF-7: Scraper multiplataforma
Descripción:
La aplicación encargada de la obtención de los datos relativos a los productos deberá de ser multiplataforma, pudiendo ser ejecutada en Windows, Linux y MacOS.
Necesidad: 2
Prioridad: 3

Tabla 3.1.18: RNF-7: Scraper multiplataforma

RNF-8: Idioma
Descripción:
Las aplicaciones que conforman el sistema estarán disponibles en castellano.
Necesidad: 1
Prioridad: 1

Tabla 3.1.19: RNF-8: Idioma

RNF-9: Acceso protegido a la base de datos
Descripción:
El acceso a la base de datos se realizará a través de un usuario y contraseña que únicamente conocerá la aplicación servidora.
Necesidad: 2
Prioridad: 2

Tabla 3.1.20: RNF-9: Acceso protegido a la base de datos

RNF-10: Accesibilidad del cliente
Descripción:
La aplicación cliente deberá de disponer de las capacidades necesarias para facilitar su uso a personas con discapacidad, es decir, deberá ser una aplicación accesible.
Necesidad: 1
Prioridad: 2

Tabla 3.1.21: RNF-10: Accesibilidad del cliente

RNF-11: Facilidad de uso
Descripción:
En la medida de lo posible las aplicaciones serán fáciles de utilizar, evitando cualquier complejidad innecesaria.
Necesidad: 2
Prioridad: 3

Tabla 3.1.22: RNF-11: Facilidad de uso

RNF-12: Tiempo de respuesta cliente	
Descripción:	La aplicación cliente tendrá un tiempo de respuesta en las consultas al servidor lo más bajo posible siempre teniendo en cuenta las limitaciones de la conexión a Internet disponible en el dispositivo.
Necesidad:	3
Prioridad:	1

Tabla 3.1.23: RNF-12: Tiempo de respuesta cliente

RNF-13: Tolerancia a fallos	
Descripción:	Las aplicaciones que componen el sistema deberán ser tolerantes a los fallos que se produzcan en las comunicaciones, evitando el cierre de la aplicación en cuestión por problemas en el diálogo con otro de los componentes.
Necesidad:	2
Prioridad:	3

Tabla 3.1.24: RNF-13: Tolerancia a fallos

RNF-14: UTF-8	
Descripción:	Se utilizará el conjunto de caracteres UTF-8 tanto para la elaboración del software como para los mensajes involucrados en las comunicaciones.
Necesidad:	1
Prioridad:	2

Tabla 3.1.25: RNF-14: UTF-8

RNF-15: disponibilidad del servidor y la base de datos	
Descripción:	Tanto el servidor como la base de datos del sistema habrán de estar disponibles para su acceso a través de Internet el máximo tiempo posible, siendo deseable una disponibilidad 24/7.
Necesidad:	2
Prioridad:	2

Tabla 3.1.26: RNF-15: disponibilidad del servidor y la base de datos

RNF-16: EAN-13
Descripción: Los códigos de barras que la aplicación cliente podrá escanear serán de tipo EAN-13
Necesidad: 2
Prioridad: 2

Tabla 3.1.27: RNF-16: EAN-13

3.1.4. Casos de uso

A continuación se van a definir, en forma de tablas, los diferentes casos de uso que se van a dar en el sistema. Para facilitar la identificación de cada uno de ellos, se asigna el código CU-X donde las siglas CU hacen referencia a *caso de uso* y X será un número que comenzará en 1 y se irá incrementando de uno en uno. En cuanto a los actores del sistema se consideran dos: el primero de ellos será el usuario final de la aplicación mientras que el segundo será el usuario administrador del sistema. De esta forma, para simplificar la elaboración de los diagramas de casos de uso, se va realizar una clasificación de éstos en función del sistema con el que se relaciona el caso de uso, perteneciendo aquellos iniciados por el usuario administrador a la aplicación scraper y los iniciados por el usuario final al sistema cliente. Por último, también se van a establecer los casos de uso que conciernen al servidor y la base de datos conjuntamente, siendo éstos iniciados siempre por una de las aplicaciones del sistema, nunca por un usuario.

a) Casos de uso de aplicación scraper:

CU-1: Obtener productos de categorías	
Descripción:	Permite obtener los productos de las categorías de la web www.carritus.com
Actores:	Usuario administrador, servidor.
Precondiciones:	La máquina debe estar conectada a Internet y la web accesible. El programa debe estar en estado <i>detenido</i> , es decir, no puede estar realizando un scraping.
Flujo normal:	<ol style="list-style-type: none"> 1. El actor selecciona la/s categoría/s que desea obtener. 2. El actor presiona el botón obtener ahora de la sección de categorías. 3. El programa pasa a estar en estado iniciado y se informa del inicio y la hora. 4. Se obtienen los productos 5. Los productos obtenidos se envían al servidor para su almacenamiento en la base de datos. 6. El servidor confirma el correcto almacenamiento de los productos. 7. Se vuelve al punto 5 hasta finalizar la obtención de los productos de la categoría. 8. El programa pasa a estar en estado detenido. 9. Se informa de la finalización de la tarea y la hora a la que sucede.
Flujo alternativo:	<p>4.a. El actor presiona el botón detener scraping de la sección de categorías.</p> <ol style="list-style-type: none"> 1. La obtención de información finaliza. 2. El programa pasa a estar en estado detenido 3. Se informa de la finalización de la tarea y la hora a la que sucede. <p>5.a. La inserción en la base de datos falla.</p> <ol style="list-style-type: none"> 1. El servidor informa del fallo al scraper 2. El scraper almacena el fallo en un log. 3. Se salta al punto 4 del flujo normal.
Postcondiciones:	La información relativa a los productos de las categorías seleccionadas han sido almacenadas en la base de datos.

Tabla 3.1.28: CU-1: obtener categorías

CU-2: Programar obtención productos de categorías**Descripción:**

Permite programar una frecuencia en días y una hora de inicio para obtener la información de los productos de forma continua y automática.

Actores:

Usuario administrador, servidor.

Precondiciones:

La máquina debe estar conectada a Internet y la web accesible. El programa debe estar en estado *detenido*, es decir, no puede estar realizando un scraping.

Flujo normal:

1. El actor selecciona la/s categoría/s que desea obtener.
2. El actor presiona el botón programar de la sección de categorías.
3. El programa permanece a la espera de que se llegue a la fecha y hora programada.
4. El programa pasa a estar en estado iniciado y se informa del inicio y la hora.
5. Se obtienen los productos.
6. Los productos obtenidos se envían al servidor para su almacenamiento en la base de datos.
7. El servidor confirma el correcto almacenamiento de los productos.
8. Se vuelve al punto 5 hasta finalizar la obtención de los productos de la categoría.
9. La obtención de información finaliza.
10. El programa pasa a estar en estado detenido.
11. Se informa de la finalización de la tarea y la hora a la que sucede.
12. Vuelve al punto 3 hasta que el actor presiona el botón detener scraping de la sección de categorías.

Flujo alternativo:

- 5.a. El actor presiona el botón detener scraping de la sección de categorías.

1. La obtención de información finaliza.
2. El programa pasa a estar en estado detenido.
3. Se informa de la finalización de la tarea y la hora a la que sucede.
4. La programación del scraping se desactiva

- 6.a. La inserción en la base de datos falla.

1. El servidor informa del fallo al scraper
2. El scraper almacena el fallo en un log.
3. Se salta al punto 5 del flujo normal.

Postcondiciones:

La información relativa a los productos de las categorías seleccionadas han sido almacenadas en la base de datos.

CU-3: Obtener menú	
Descripción:	Permite obtener las categorías del menú de la web www.carritus.com, las cuales van a realizar la clasificación de los productos.
Actores:	Usuario administrador, servidor.
Precondiciones:	La máquina debe estar conectada a Internet y la web accesible. El programa debe estar en estado <i>detenido</i> , es decir, no puede estar realizando un scraping.
Flujo normal:	<ol style="list-style-type: none"> 1. El actor presiona el botón obtener ahora de la sección de menú. 2. El programa pasa a estar en estado iniciado y se informa del inicio y la hora. 3. Se obtienen las categorías. 4. Las categorías obtenidas se envían al servidor para su almacenamiento en la base de datos. 5. El servidor confirma el correcto almacenamiento de las categorías. 6. La obtención de información finaliza. 7. El programa pasa a estar en estado detenido. 8. Se informa de la finalización de la tarea y la hora a la que sucede.
Flujo alternativo:	<p>3.a. El actor presiona el botón detener scraping de la sección de menú.</p> <ol style="list-style-type: none"> 1. La obtención de información finaliza. 2. El programa pasa a estar en estado detenido 3. Se informa de la finalización de la tarea y la hora a la que sucede. <p>4.a. La inserción en la base de datos falla.</p> <ol style="list-style-type: none"> 1. El servidor informa del fallo al scraper 2. El scraper almacena el fallo en un log. 3. Se salta al punto 3 del flujo normal.
Postcondiciones:	La información relativa a las categorías del menú ha sido almacenada en la base de datos.

Tabla 3.1.30: CU-3: obtener menú

CU-4: Programar obtención de menú**Descripción:**

Permite programar una frecuencia en días y una hora de inicio para obtener las categorías del menú de la web www.carritus.com, las cuales van a realizar la clasificación de los productos, de forma continua y automática.

Actores:

Usuario administrador, servidor.

Precondiciones:

La máquina debe estar conectada a Internet y la web accesible. El programa debe estar en estado *detenido*, es decir, no puede estar realizando un scraping.

Flujo normal:

1. El actor presiona el botón programar de la sección de menú.
2. El programa permanece a la espera de que se llegue a la fecha y hora programada.
3. El programa pasa a estar en estado iniciado y se informa del inicio y la hora.
4. Se obtienen las categorías.
5. Las categorías obtenidas se envían al servidor para su almacenamiento en la base de datos.
6. El servidor confirma el correcto almacenamiento de las categorías.
7. Se vuelve al punto 5 hasta finalizar la obtención de los productos de la categoría.
8. La obtención de información finaliza.
9. El programa pasa a estar en estado detenido.
10. Se informa de la finalización de la tarea y la hora a la que sucede.
11. Vuelve al punto 2 hasta que el actor presiona el botón detener scraping de la sección de menú.

Flujo alternativo:

- 4.a. El actor presiona el botón detener scraping de la sección de menú.
 1. La obtención de información finaliza.
 2. El programa pasa a estar en estado detenido.
 3. Se informa de la finalización de la tarea y la hora a la que sucede.
 4. La programación del scraping se desactiva
- 5.a. La inserción en la base de datos falla.
 1. El servidor informa del fallo al scraper
 2. El scraper almacena el fallo en un log.
 3. Se salta al punto 4 del flujo normal.
- 2-4.a. El actor presiona el botón *detener scraping* de la sección de menú.

Postcondiciones:

La información relativa a las categorías del menú ha sido almacenada en la base de datos.

Detallados los casos de uso para el scraper, se confecciona el diagrama de casos de uso, el cual puede visualizarse en la figura 3.1.1 y que permite hacerse una idea de las relaciones que existen entre actores y casos de uso de una forma simple y rápida.

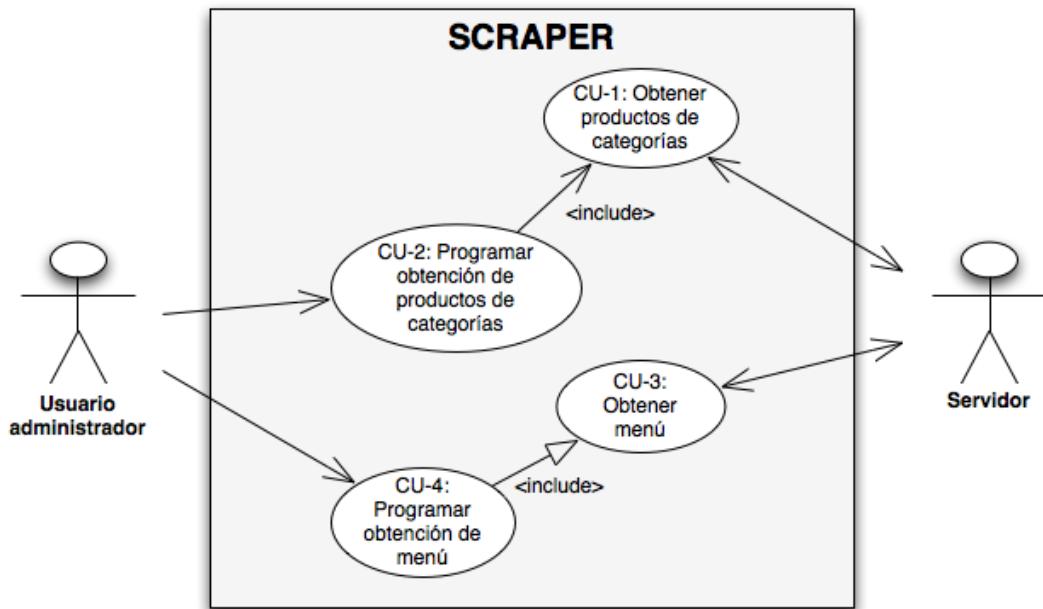


Figura 3.1.1: Diagrama de casos de uso del cliente

b) Casos de uso de aplicación cliente:

CU-5: Mostrar categorías
Descripción: Se muestran las categorías en la pantalla del dispositivo móvil del usuario.
Actores: Usuario final, servidor.
Precondiciones: El dispositivo móvil debe de disponer de conexión a Internet.
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la opción Búsqueda de productos.2. El actor pulsa sobre la opción Lista de categorías.3. El programa solicita las categorías al servidor.4. El servidor devuelve las categorías al cliente.5. Se muestran las categorías en forma de lista.
Flujo alternativo: <ol style="list-style-type: none">3.a. El dispositivo no dispone de conexión a Internet.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.4.a. El dispositivo no recibe respuesta.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.
Postcondiciones: El actor visualiza las categorías de los productos en la pantalla de su dispositivo.

Tabla 3.1.32: CU-5: mostrar categorías

CU-6: Mostrar subcategorías	
Descripción:	Se muestran las subcategorías asociadas a una categoría de productos en la pantalla del dispositivo móvil del usuario.
Actores:	Usuario final, servidor.
Precondiciones:	El dispositivo móvil debe de disponer de conexión a Internet.
Flujo normal:	<ol style="list-style-type: none">1. El actor accede a la visualización de las categorías a partir del caso de uso CU-5.2. El actor pulsa sobre la categoría a la que pertenecen las subcategorías que desea mostrar.3. El programa solicita las subcategorías al servidor.4. El servidor devuelve las subcategorías al cliente.5. Se muestran las subcategorías en forma de lista.
Flujo alternativo:	<ol style="list-style-type: none">3.a. El dispositivo no dispone de conexión a Internet.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.4.a. El dispositivo no recibe respuesta.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.
Postcondiciones:	El actor visualiza las subcategorías asociadas a una categoría de productos en la pantalla de su dispositivo.

Tabla 3.1.33: CU-6: mostrar subcategorías

CU-7: Mostrar subsubcategorías

Descripción:

Se muestran las subsubcategorías asociadas a una subcategoría de productos en la pantalla del dispositivo móvil del usuario.

Actores:

Usuario final, servidor.

Precondiciones:

El dispositivo móvil debe de disponer de conexión a Internet.

Flujo normal:

1. El actor accede a la visualización de las subcategorías a partir del caso de uso CU-6.
2. El actor pulsa sobre la subcategoría a la que pertenecen las subsubcategorías que desea mostrar.
3. El programa solicita las subsubcategorías al servidor.
4. El servidor devuelve las subsubcategorías al cliente.
5. Se muestran las subsubcategorías en forma de lista.

Flujo alternativo:

- 3.a. El dispositivo no dispone de conexión a Internet.
 1. Se muestra un mensaje informando de la situación.
- 4.a. El dispositivo no recibe respuesta.
 1. Se muestra un mensaje informando de la situación.
- 4.b. La subcategoría padre no tiene subsubcategorías asociadas
 1. Se muestra la lista de productos que pertenecen a la subcategoría.

Postcondiciones:

El actor visualiza las subsubcategorías asociadas a una subcategoría de productos en la pantalla de su dispositivo.

Tabla 3.1.34: CU-7: mostrar subsubcategorías

CU-8: Buscar producto por categoría
Descripción: Se muestran la información relativa a un producto perteneciente a una categoría, subcategoría y subsubcategoría en la pantalla del dispositivo móvil del usuario.
Actores: Usuario final, servidor.
Precondiciones: El dispositivo móvil debe de disponer de conexión a Internet.
Flujo normal: <ol style="list-style-type: none"> 1. El actor accede a la subsubcategoría a la que pertenece el producto a partir del caso de uso CU-7. 2. El actor pulsa sobre la subsubcategoría a la que pertenecen los productos que desea ver. 3. El programa solicita los productos al servidor. 4. El servidor devuelve los productos al cliente. 5. Se muestran los productos en forma de lista. 6. El actor pulsa sobre el producto que desea visualizar. 7. El programa solicita la información del producto al servidor. 8. El servidor devuelve la información requerida al cliente. 9. Se muestra la información relativa del producto.
Flujo alternativo: <p>3-7.a. El dispositivo no dispone de conexión a Internet.</p> <ol style="list-style-type: none"> 1. Se muestra un mensaje informando de la situación. <p>4-8.a. El dispositivo no recibe respuesta.</p> <ol style="list-style-type: none"> 1. Se muestra un mensaje informando de la situación.
Postcondiciones: El actor visualiza la información relativa a un producto al que ha accedido a través de la navegación por las categorías.

Tabla 3.1.35: CU-8: buscar producto por categoría

CU-9: Buscar producto por nombre
Descripción: Se realiza una búsqueda de productos a partir de una cadena de texto introducida por el actor.
Actores: Usuario final, servidor.
Precondiciones: El dispositivo móvil debe de disponer de conexión a Internet.
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la opción Búsqueda de productos.2. El actor pulsa sobre la opción Nombre.3. El actor introduce la cadena de texto a buscar.4. El programa envía la búsqueda al servidor.5. El servidor devuelve el resultado de la búsqueda al cliente.6. Se muestran, en forma de lista, los productos que contienen la cadena de texto en su nombre o descripción.7. El usuario pulsa sobre uno de los productos de la lista de coincidencias.8. Se muestra la información relativa al producto seleccionado.
Flujo alternativo: <ol style="list-style-type: none">4.a. El dispositivo no dispone de conexión a Internet.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.5.a. El dispositivo no recibe respuesta.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.
Postcondiciones: Se muestran la información relativa a un producto encontrado mediante una búsqueda de texto.

Tabla 3.1.36: CU-9: buscar producto por nombre

CU-10: Buscar producto por código de barras**Descripción:**

Se realiza una búsqueda de productos a partir del escaneo del código de barras de un producto mediante la cámara del dispositivo.

Actores:

Usuario final, servidor.

Precondiciones:

El dispositivo móvil debe de disponer de conexión a Internet y debe tener cámara de fotos. Esta opción estará disponible únicamente para sistemas operativos Android y será necesario que tenga instalada la aplicación de código libre y gratuita *Barcode Scanner*.

Flujo normal:

1. El actor accede a la opción Búsqueda de productos.
2. El actor pulsa sobre la opción Código de barras.
3. Se abre una nueva pantalla que muestra la imagen que capta, en tiempo real, la cámara de fotos del dispositivo. Ésta pertenece a la aplicación Barcode Scanner.
4. El actor apunta al código de barras del producto mediante la cámara del dispositivo.
5. La aplicación Barcode Scanner reconoce el código de barras y el número asociado a éste devolviéndolo a la aplicación Comprador.
6. El programa solicita al servidor la información del producto asociado al código de barras.
7. El servidor devuelve la información del producto.
8. Se muestra la información relativa al producto buscado.

Flujo alternativo:

- 3.a. El dispositivo no dispone de la aplicación Barcode Scanner.
 1. Se redirige a la página de la aplicación en Google Play para que se proceda a su instalación.
- 5.a. El código escaneado no tiene el formato EAN-13.
 1. Se muestra un mensaje informando de la situación.
- 6.a. El dispositivo no dispone de conexión a Internet.
 1. Se muestra un mensaje informando de la situación.
- 7.a. El dispositivo no recibe respuesta.
 1. Se muestra un mensaje informando de la situación.

Postcondiciones:

El actor visualiza los detalles del producto cuyo código de barras ha sido escaneado en la pantalla de su dispositivo.

Tabla 3.1.37: CU-10: buscar producto por código de barras

CU-11: Mostrar listas
Descripción: Se muestra el conjunto de listas de productos creadas por el usuario en la pantalla del dispositivo.
Actores: Usuario final.
Precondiciones: El actor debe haber creado una o más listas de productos con anterioridad.
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la opción Listas.2. Se muestran las listas almacenadas.
Flujo alternativo: <ol style="list-style-type: none">2.a. No existen listas que mostrar.<ol style="list-style-type: none">1. La pantalla se muestra vacía.
Postcondiciones: El actor visualiza las listas de productos en la pantalla de su dispositivo.

Tabla 3.1.38: CU-11: mostrar listas

CU-12: Mostrar productos de una lista	
Descripción:	Se muestra el conjunto de productos que pertenecen a una lista creada por el usuario en la pantalla del dispositivo.
Actores:	Usuario final.
Precondiciones:	El actor debe haber creado la lista a visualizar y haber agregado productos a la misma.
Flujo normal:	<ol style="list-style-type: none">1. El actor accede a la visualización de las listas de productos a partir del caso de uso 11.2. El actor pulsa sobre la lista cuyos productos desea mostrar.3. Se muestran los productos de la lista.
Flujo alternativo:	<ol style="list-style-type: none">3.a. La lista no contiene productos.<ol style="list-style-type: none">1. La pantalla se muestra vacía.
Postcondiciones:	El actor visualiza los productos de la lista en la pantalla de su dispositivo.

Tabla 3.1.39: CU-12: mostrar productos de una lista

CU-13: Crear una lista
Descripción: Se crea una lista a la que el usuario podrá agregar productos para una posterior comparación de precios.
Actores: Usuario final.
Precondiciones:
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la opción Listas.2. El actor pulsa sobre la opción Crear nueva lista.3. Se solicita al actor que introduzca el nombre de la lista.4. El actor introduce el nombre que desea que tenga la lista.5. El actor pulsa sobre el botón Crear.6. Se muestra el conjunto de listas de productos creadas por el usuario, incluyendo la nueva lista creada.
Flujo alternativo: <ol style="list-style-type: none">5.a. El actor pulsa sobre el botón Cancelar<ol style="list-style-type: none">1. Se vuelve a mostrar el conjunto de listas de productos.
Postcondiciones: Se crea una nueva lista de productos que queda almacenada en el dispositivo.

Tabla 3.1.40: CU-13: crear una lista

CU-14: Eliminar lista
Descripción: Se elimina crea una lista de productos.
Actores: Usuario final.
Precondiciones: La lista a eliminar debe haber sido creada anteriormente por el usuario.
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la opción Listas.2. El actor pulsa sobre el botón de opciones asociado a la lista que desea eliminar.3. Se muestra la lista de opciones posibles a realizar con la lista.4. El actor pulsa sobre el botón Eliminar5. Se pide al usuario que confirme su intención de eliminar la lista.6. El actor pulsa sobre la opción Aceptar.7. Se elimina la lista.8. Se notifica al usuario de que la lista ha sido eliminada mediante un mensaje por pantalla y una vibración del dispositivo.9. Se muestran las listas de productos.
Flujo alternativo: <ol style="list-style-type: none">5.a. El actor pulsa sobre el botón Cancelar<ol style="list-style-type: none">1. Se vuelve a mostrar el conjunto de listas de productos.
Postcondiciones: La lista ha sido eliminada del dispositivo.

Tabla 3.1.41: CU-14: eliminar lista

CU-15: Comprobar lista

Descripción:

Se realiza una comparación del precio de los productos de una lista.

Actores:

Usuario final, servidor.

Precondiciones:

El dispositivo debe de estar conectado a Internet y la lista a comprobar debe haber sido creada con anterioridad.

Flujo normal:

1. El actor accede a la opción Listas.
2. El actor pulsa sobre el botón de opciones asociado a la lista que desea comprobar.
3. Se muestra la lista de opciones posibles a realizar con la lista.
4. El actor pulsa sobre la opción Comprobar.
5. El programa envía la lista de productos al servidor.
6. El servidor devuelve el precio total de la lista para cada uno de los supermercados así como el número de aciertos de cada uno de ellos.
7. Se muestra una gráfica con los precios de los productos así como el precio total de éstos para cada uno de los supermercados.

Flujo alternativo:

- 5.a. El dispositivo no dispone de conexión a Internet.
 1. Se muestra un mensaje informando de la situación.
- 6.a. El dispositivo no recibe respuesta.
 1. Se muestra un mensaje informando de la situación.

Postcondiciones:

Se visualiza en la pantalla del dispositivo una comparativa de precios de los productos de la lista que permite al usuario ver qué supermercado es el más económico para realizar la compra de los artículos.

Tabla 3.1.42: CU-15: comprobar lista

CU-16: Ver lista en modo compra	
Descripción:	Se visualizan los productos de una lista permitiendo al usuario tacharlos y destacharlos de forma que éste pueda realizar un seguimiento de los artículos que ya ha comprado y los que no.
Actores:	Usuario final.
Precondiciones:	La lista a visualizar debe haber sido creada con anterioridad.
Flujo normal:	<ol style="list-style-type: none">1. El actor accede a la opción Listas.2. El actor pulsa sobre el botón de opciones asociado a la lista que desea visualizar en modo compra.3. Se muestra la lista de opciones posibles a realizar con la lista.4. El actor pulsa sobre la opción Modo compra.5. Se muestran los productos en forma de lista.6. El actor puede tachar o destachar los productos pulsando sobre ellos.
Flujo alternativo:	
Postcondiciones:	Se visualiza en la pantalla del dispositivo una lista de productos que podrán ser tachados o destachados.

Tabla 3.1.43: CU-16: ver lista en modo compra

CU-17: Agregar producto a lista
Descripción: Se agrega un producto a una lista.
Actores: Usuario final.
Precondiciones: La lista a la que se desea agregar el producto debe haber sido creada con anterioridad.
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la visualización de la información del producto a agregar a través de los casos de uso UC-8, UC-9, UC-10 o UC-20.2. El actor pulsa sobre el botón de Agregar a lista.3. Se muestra una pantalla con un desplegable que contiene las listas de productos creadas por el usuario.4. El actor selecciona la lista a la que desea agregar el producto.5. El actor pulsa sobre el botón Seleccionar.6. Se notifica mediante un mensaje por pantalla y una vibración del dispositivo de que el producto ha sido agregado a la lista.7. Se muestra de nuevo la pantalla que contiene la información relativa al producto.
Flujo alternativo: <ol style="list-style-type: none">2.a. El actor agita el dispositivo hacia un lado y el otro varias veces.<ol style="list-style-type: none">1. Se salta al punto 3 del flujo normal.5.a. El actor pulsa sobre el botón Atrás.<ol style="list-style-type: none">1. Se salta al punto 1 del flujo normal.
Postcondiciones: Se ha agregado un producto a una lista.

Tabla 3.1.44: CU-17: agregar producto a lista

CU-18: Eliminar producto de lista	
Descripción:	Se elimina un producto de una lista.
Actores:	Usuario final.
Precondiciones:	El producto a eliminar debe pertenecer a una lista.
Flujo normal:	<ol style="list-style-type: none">1. El actor accede a la visualización de los productos que pertenecen a la lista de la que se desea eliminar el producto a partir del caso de uso CU-12.2. El actor pulsa sobre el botón de opciones asociado al producto de la lista.3. Se muestran las opciones disponibles.4. El actor selecciona la opción Eliminar.5. El sistema solicita al actor que confirme su deseo de eliminar el producto de la lista.6. El actor pulsa sobre el botón Aceptar.7. Se elimina el producto de la lista.8. Se notifica al usuario de que el producto ha sido eliminado mediante un mensaje que se muestra por pantalla y una vibración del dispositivo.9. Se muestran los productos de la lista.
Flujo alternativo:	<p>6.a. El actor pulsa sobre el botón Cancelar.</p> <ol style="list-style-type: none">1. Se salta al punto 1 del flujo normal.
Postcondiciones:	Se ha eliminado el producto de la lista.

Tabla 3.1.45: CU-18: eliminar producto de lista

CU-19: Mostrar favoritos
Descripción: Se muestran los productos marcados como favoritos por el usuario.
Actores: Usuario final.
Precondiciones: El actor debe haber marcado algún producto como favorito.
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la opción Favoritos.2. Se muestran los productos favoritos en forma de lista.
Flujo alternativo: <ol style="list-style-type: none">6.a. No hay productos en favoritos.<ol style="list-style-type: none">1. La pantalla se muestra vacía.
Postcondiciones: El actor visualiza los productos favoritos.

Tabla 3.1.46: CU-19: mostrar favoritos

CU-20: Mostrar producto favorito
Descripción: Se muestra la información relativa a un producto marcado como favorito por el usuario.
Actores: Usuario final.
Precondiciones: El actor debe haber incluído el producto a visualizar en favoritos.
Flujo normal: <ol style="list-style-type: none">1. El actor accede a la visualización de los productos favoritos a partir del caso de uso CU-19.2. El actor pulsa sobre el producto que desea visualizar.3. Se muestra la información relativa al producto seleccionado en el paso anterior.
Flujo alternativo:
Postcondiciones: El actor visualiza la información relativa a un producto favorito en la pantalla de su dispositivo.

Tabla 3.1.47: CU-20: mostrar producto favorito

CU-21:Agregar producto a favoritos	
Descripción:	Se marca a un producto como favorito.
Actores:	Usuario final.
Precondiciones:	
Flujo normal:	<ol style="list-style-type: none">1. El actor accede a la visualización de la información del producto a agregar a través de los casos de uso CU-18, CU-19 o CU-10.2. El actor pulsa sobre el botón de Agregar a favoritos.3. Se agrega el producto a favoritos.4. Se notifica mediante un mensaje por pantalla y una vibración del dispositivo de que el producto ha sido agregado a favoritos.5. Se muestra de nuevo la pantalla que contiene la información relativa al producto.
Flujo alternativo:	
Postcondiciones:	Se ha agregado el producto a favoritos.

Tabla 3.1.48: CU-21: agregar producto a favoritos

CU-22: Eliminar producto de favoritos	
Descripción:	Se elimina un producto de la lista de favoritos.
Actores:	Usuario final.
Precondiciones:	El producto a eliminar debe haber sido agregado a favoritos con anterioridad.
Flujo normal:	<ol style="list-style-type: none">1. El actor accede a la visualización de los productos que pertenecen a favoritos a partir del caso de uso CU-19.2. El actor pulsa sobre el botón de opciones asociado al producto de la lista de favoritos.3. Se muestran las opciones disponibles.4. El actor selecciona la opción Eliminar.5. El sistema solicita al actor que confirme su deseo de eliminar el producto de favoritos.6. El actor pulsa sobre el botón Aceptar.7. Se elimina el producto de favoritos.8. Se notifica al usuario de que el producto ha sido eliminado mediante un mensaje que se muestra por pantalla y una vibración del dispositivo.9. Se muestran de nuevo los productos marcados como favoritos.
Flujo alternativo:	<p>6.a. El actor pulsa sobre el botón Cancelar.</p> <ol style="list-style-type: none">1. Se salta al punto 1 del flujo normal.
Postcondiciones:	Se ha eliminado el producto de favoritos

Tabla 3.1.49: CU-22: eliminar producto de favoritos

CU-23: Agregar producto favorito a lista	
Descripción:	Se agrega un producto favorito a una lista.
Actores:	Usuario final.
Precondiciones:	La lista a la que se desea agregar el producto debe haber sido creada con anterioridad y el producto debe haber sido marcado como favorito.
Flujo normal:	<ol style="list-style-type: none"> 1. El actor accede a la visualización de la información del producto a agregar a través del caso de uso CU-20. 2. El actor pulsa sobre el botón de Agregar a lista. 3. Se muestra una pantalla con un desplegable que contiene las listas de productos creadas por el usuario. 4. El actor selecciona la lista a la que desea agregar el producto. 5. El actor pulsa sobre el botón Seleccionar. 6. Se muestra de nuevo la pantalla que contiene la información relativa al producto.
Flujo alternativo:	<p>2.a. El actor agita el dispositivo hacia un lado y el otro varias veces.</p> <ol style="list-style-type: none"> 1. Se salta al punto 3 del flujo normal. <p>5.a. El actor pulsa sobre el botón Atrás.</p> <ol style="list-style-type: none"> 1. Se salta al punto 1 del flujo normal.
Postcondiciones:	Se ha agregado un producto favorito a una lista.

Tabla 3.1.50: CU-23: agregar producto favorito a lista

Una vez confeccionados los casos de uso, se procede a mostrar el diagrama de la figura 3.1.2, que los relaciona entre ellos y con los actores que intervienen para el cliente.

Proyecto Comprador

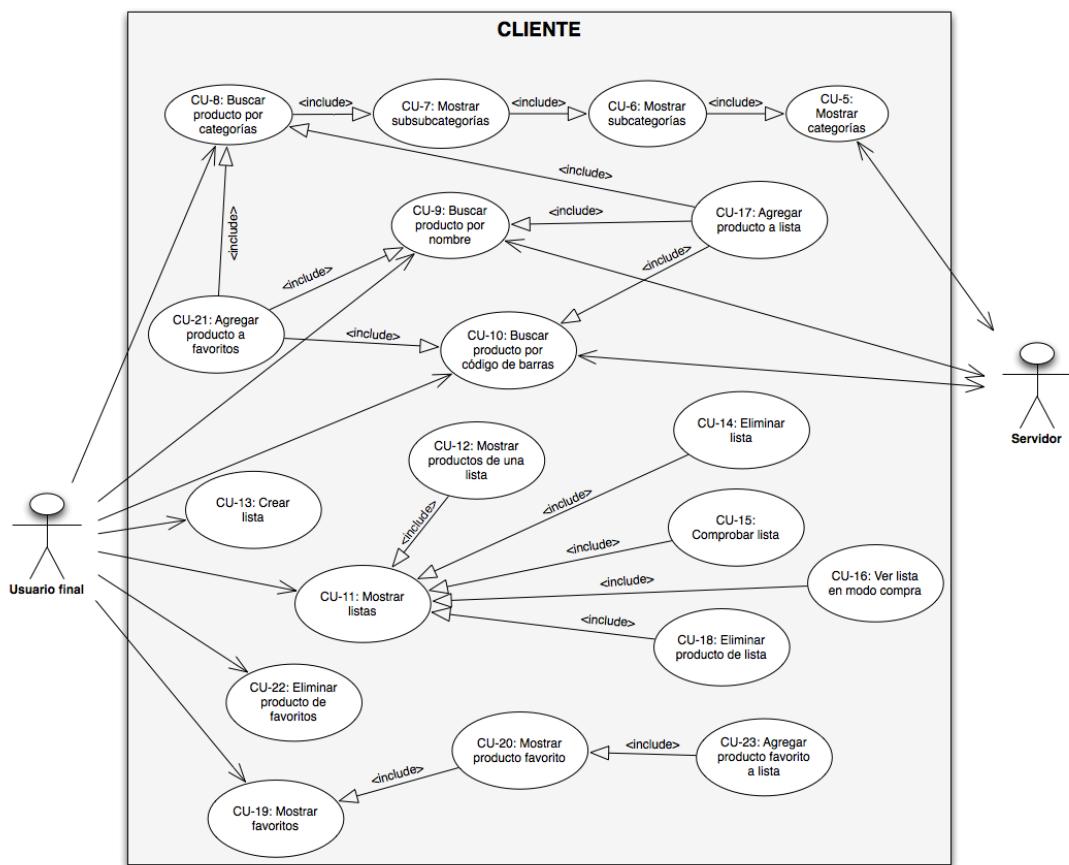


Figura 3.1.2: Diagrama de casos de uso del cliente

c) Casos de uso del servidor:

CU-24: Guardar producto en base de datos	
Descripción:	Se almacena un producto en la base de datos
Actores:	Scraper, servidor, base de datos
Precondiciones:	Todos los sistemas deberán de tener una comunicación directa entre si.
Flujo normal:	<ol style="list-style-type: none"> 1. El actor scraper envía la información del producto al servidor. 2. El servidor inserta la información en la base de datos. 3. El servidor informa al scraper de que la operación se ha completado con éxito.
Flujo alternativo:	<ol style="list-style-type: none"> 1.a. El scraper no dispone de conexión a Internet. <ol style="list-style-type: none"> 1. Se muestra un mensaje informando de la situación. 2.a. La inserción en la base de datos falla. <ol style="list-style-type: none"> 1. El servidor informa del fallo al scraper 2. El scraper almacena el fallo en un log. 3.a. El scraper no recibe respuesta. <ol style="list-style-type: none"> 1. Se muestra un mensaje informando de la situación.
Postcondiciones:	Se ha almacenado un producto en la base de datos.

Tabla 3.1.51: CU-24: guardar producto en la base de datos

CU-25: Guardar categoría/subcategoría/subsubcategoría en base de datos	
Descripción:	Se almacena una categoría/subcategoría/subsubcategoría en la base de datos
Actores:	Scraper, servidor, base de datos
Precondiciones:	Todos los sistemas deberán de tener una comunicación directa entre si.
Flujo normal:	<ol style="list-style-type: none">1. El actor scraper envía la información de la categoría/subcategoría/subsubcategoría al servidor.2. El servidor inserta la información en la base de datos.3. El servidor informa al scraper de que la operación se ha completado con éxito.
Flujo alternativo:	<ol style="list-style-type: none">1.a. El scraper no dispone de conexión a Internet.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.2.a. La inserción en la base de datos falla.<ol style="list-style-type: none">1. El servidor informa del fallo al scraper2. El scraper almacena el fallo en un log.3.a. El scraper no recibe respuesta.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.
Postcondiciones:	Se ha almacenado una categoría/subcategoría/subsubcategoría en la base de datos.

Tabla 3.1.52: CU-25: guardar categoría/subcategoría/subsubcategoría en la base de datos

CU-26: Obtener producto de la base de datos	
Descripción:	Se obtiene la información de un producto de la base de datos
Actores:	Cliente, servidor, base de datos
Precondiciones:	Todos los sistemas deberán de tener una comunicación directa entre si.
Flujo normal:	<ol style="list-style-type: none">1. El actor cliente envía la petición al servidor.2. El servidor realiza la consulta a en la base de datos para obtener la información.3. El servidor devuelve al cliente la información.
Flujo alternativo:	<ol style="list-style-type: none">1.a. El cliente no dispone de conexión a Internet.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.3.a. El dispositivo no recibe respuesta.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.
Postcondiciones:	Se ha obtenido la información de un producto de la base de datos.

Tabla 3.1.53: CU-26: obtener producto de la base de datos

CU-27: Obtener categoría/subcategoría/subsubcategoría de la base de datos

Descripción:

Se obtiene la información de una categoría/subcategoría/subsubcategoría de la base de datos

Actores:

Cliente, servidor, base de datos

Precondiciones:

Todos los sistemas deberán de tener una comunicación directa entre si.

Flujo normal:

1. El actor cliente envía la petición al servidor.
2. El servidor realiza la consulta a en la base de datos para obtener la información.
3. El servidor devuelve al cliente la información.

Flujo alternativo:

- 1.a. El cliente no dispone de conexión a Internet.
 1. Se muestra un mensaje informando de la situación.
- 3.a. El dispositivo no recibe respuesta.
 1. Se muestra un mensaje informando de la situación.

Postcondiciones:

Se ha obtenido la información de una categoría/subcategoría/subsubcategoría de la base de datos.

Tabla 3.1.54: CU-27: obtener categoría/subcategoría/subsubcategoría de la base de datos

CU-28: Obtener precios de la base de datos	
Descripción:	Se obtienen los precios de uno o varios productos de la base de datos para todos los supermercados.
Actores:	Cliente, servidor, base de datos
Precondiciones:	Todos los sistemas deberán de tener una comunicación directa entre si.
Flujo normal:	<ol style="list-style-type: none">1. El actor cliente envía la petición al servidor.2. El servidor realiza la consulta a en la base de datos para obtener la información.3. El servidor devuelve al cliente la información.
Flujo alternativo:	<ol style="list-style-type: none">1.a. El cliente no dispone de conexión a Internet.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.3.a. El dispositivo no recibe respuesta.<ol style="list-style-type: none">1. Se muestra un mensaje informando de la situación.
Postcondiciones:	Se han obtenido los precios de uno o varios productos de la base de datos para todos los supermercados.

Tabla 3.1.55: CU-28: obtener precios de la base de datos

Tras detallar los casos de uso, se confecciona el diagrama de la figura 3.1.3, el cual muestra las relaciones existentes entre actores y casos de uso para el servidor.

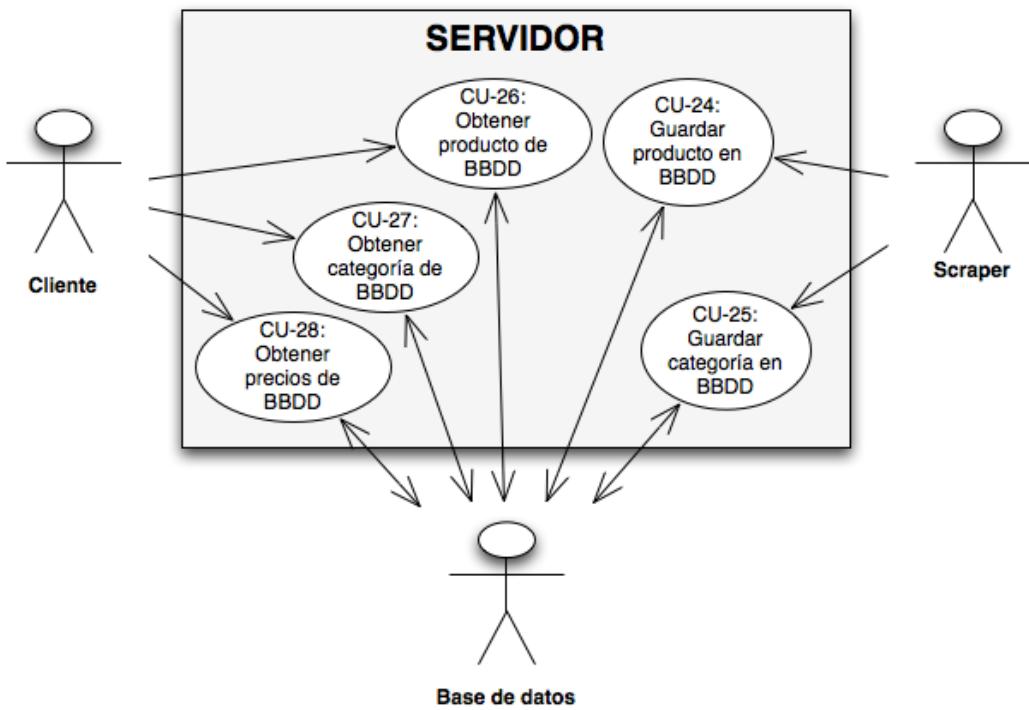


Figura 3.1.3: Diagrama de casos de uso del servidor

3.1.5. Otros diagramas

En esta última sección dedicada al análisis se van a incluir otros diagramas basados en UML que ayudan a un mejor entendimiento del problema y, por tanto, a un mejor análisis. No obstante, puesto que éstos se han considerado como de una menor importancia, no se van a elaborar de una forma exhaustiva y detallada sino que se mostrarán aquellos que se estima que tienen una relevancia mayor dentro del análisis y que aportan ideas no triviales.

3.1.5.1. Diagramas de estado

En este apartado se plasma de forma gráfica los posibles estados de la aplicación Scraper, puesto que es la única de las que componen el sistema que va a disponer de varios estados. Tal y como puede verse en el diagrama 3.1.4, esta aplicación podrá estar en tres estados: iniciado, programado y detenido. El primero de ellos sucederá cuando la aplicación esté trabajando en una obtención de datos, ya sean categorías o productos. No obstante, hay que apuntar que ambas extracciones trabajarán por separado pudiendo cada una de las extracciones estar en un estado distinto. El segundo de los estados se dará cuando el scraping para categorías o productos haya sido programado para su ejecución automática, permaneciendo en éste hasta que comience el scraping a la hora programada, momento en el que se pasará al estado iniciado. Por otro lado, el estado detenido tendrá lugar cuando la aplicación no esté ociosa, es decir, cuando no se estén obteniendo datos.

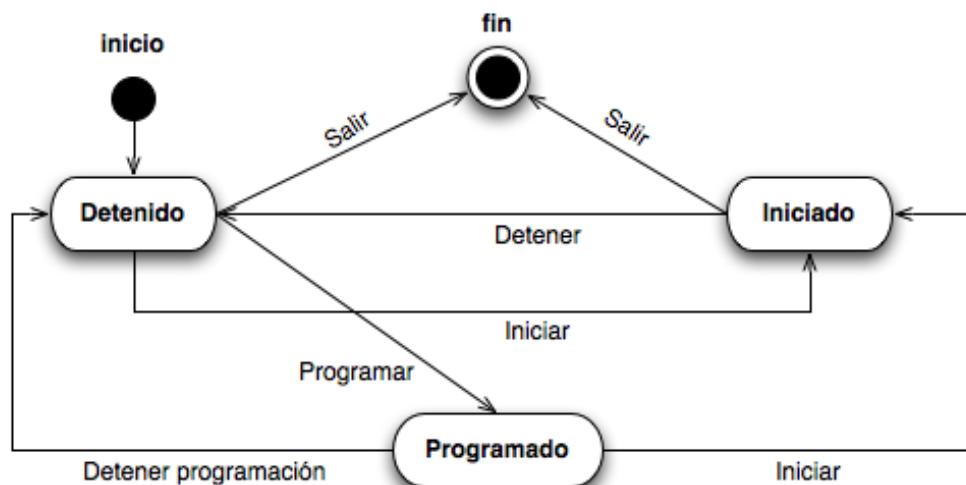


Figura 3.1.4: Diagrama de estado Scraper

3.1.5.2. Diagramas de secuencia

Para finalizar con la sección dedicada al análisis del sistema, se elaboran los diagramas de secuencia para los casos de uso CU-1: Obtener productos de categorías y CU-8: Buscar producto por categorías. Éstos dos casos de uso, además de ser unos de los más importantes para desarrollo de las funciones propias del sistema, son, probablemente, los más complejos. Es por ello que se decide, para facilitar aún más la comprensión de los mismos, plasmar el diagrama de secuencia correspondiente a cada uno de ellos. Es de especial interés el caso CU-8, puesto que éste incluye a varios de los casos de uso del cliente, de manera que, en un mismo diagrama, es posible mostrar una gran cantidad de información de una sola vez. Adicionalmente, hay que decir que en ambos casos de uso participan varios actores, siendo algunos de ellos programas del sistema, por lo que los hace aún más interesantes si cabe. Por otro lado, con estos diagramas se pretende escenificar la mecánica de trabajo que seguirá la aplicación cliente para solicitar y enviar información al servidor para que sea éste quién provea el acceso a la base de datos, tal y como se mostrará más adelante en la sección dedicada a la arquitectura del sistema.

Diagrama secuencia CU-1: Obtener productos de categorías:

El diagrama se muestra en la figura 3.1.5. Comparándolo con el flujo del CU-1, es posible observar que, aunque se han seguido los mismos pasos, se ha intentado profundizar un poco más en el funcionamiento interno que el Scraper seguirá para extraer la información de los productos. De esta forma, éste irá recorriendo las subcategorías y subsubcategorías que pertenecen a las categorías seleccionadas por el usuario y navegando a través de las páginas de productos que pertenecen a éstas de una forma iterativa hasta obtener la información de todos los productos. Una vez obtenidos los productos de una subsubcategoría, éstos se enviarán, uno a uno, al servidor para que realice el guardado de la información en la base de datos.

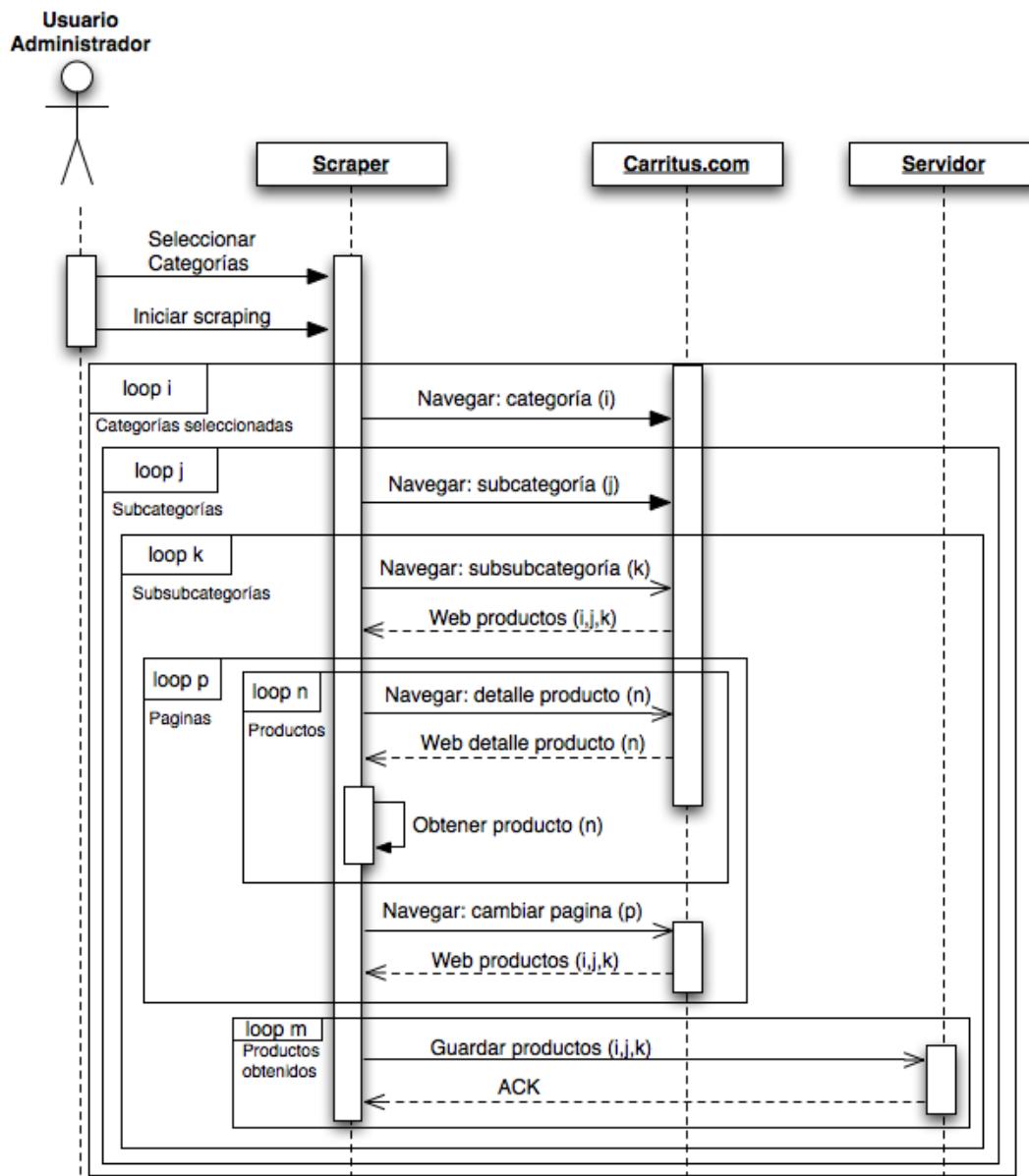


Figura 3.1.5: Diagrama secuencia CU-1: obtener productos de categorías.

Diagrama secuencia CU-8: Buscar producto por categorías:

Este diagrama se muestra en la figura 3.1.6 y de nuevo éste se construye siguiendo los pasos del flujo normal descritos en el caso de uso. No obstante, en esta ocasión, la elaboración del diagrama tiene la intención de resaltar el hecho de que no todas las subcategorías contienen subsubcategorías, estando clasificados algunos productos en 3 niveles y otros en 2. De esta forma, para el primer caso, al realizar la petición al servidor para obtener las subsubcategorías, éste devolverá las subsubcategorías solicitadas mientras en el segundo caso devolverá directamente los productos que pertenecen a la subcategoría pulsada por el usuario.

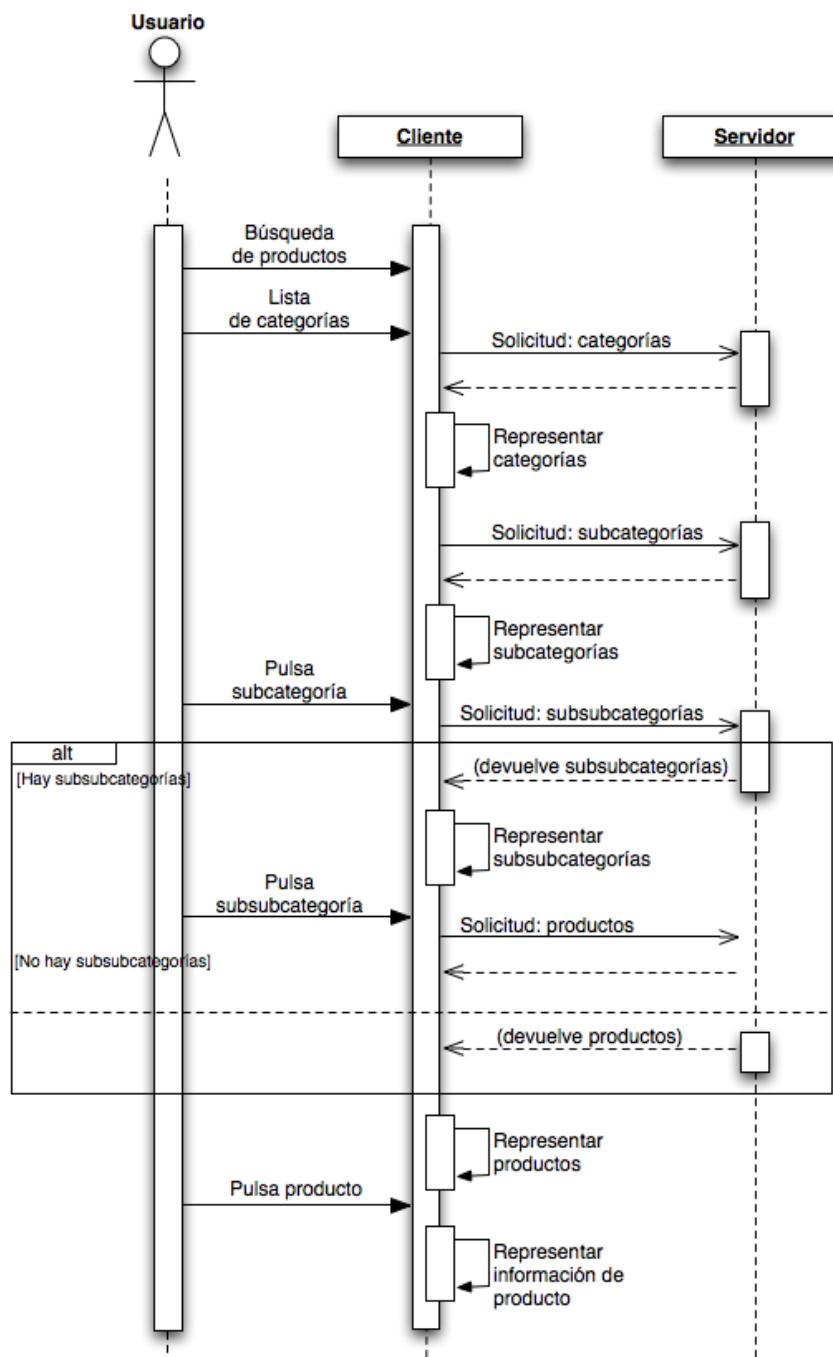


Figura 3.1.6: Diagrama secuencia CU-8: buscar producto por categorías.

3.2. Diseño del sistema

3.2.1. Descripción general y arquitectura del sistema

Como se ha comentado en la sección anterior, el sistema tiene como función principal asistir al usuario en la comparación de precios de productos de supermercado. Con

este objetivo, se propone la arquitectura que se muestra en la figura 3.2.1 como base de partida para el desarrollo de las funciones del sistema. El sistema propuesto está basado en el paradigma arquitectónico cliente-servidor, de forma que será la aplicación alojada en el servidor la que realice, en la medida de lo posible, los esfuerzos computacionales más costosos así como la encargada de la gestión del almacenamiento y recuperación de los datos.

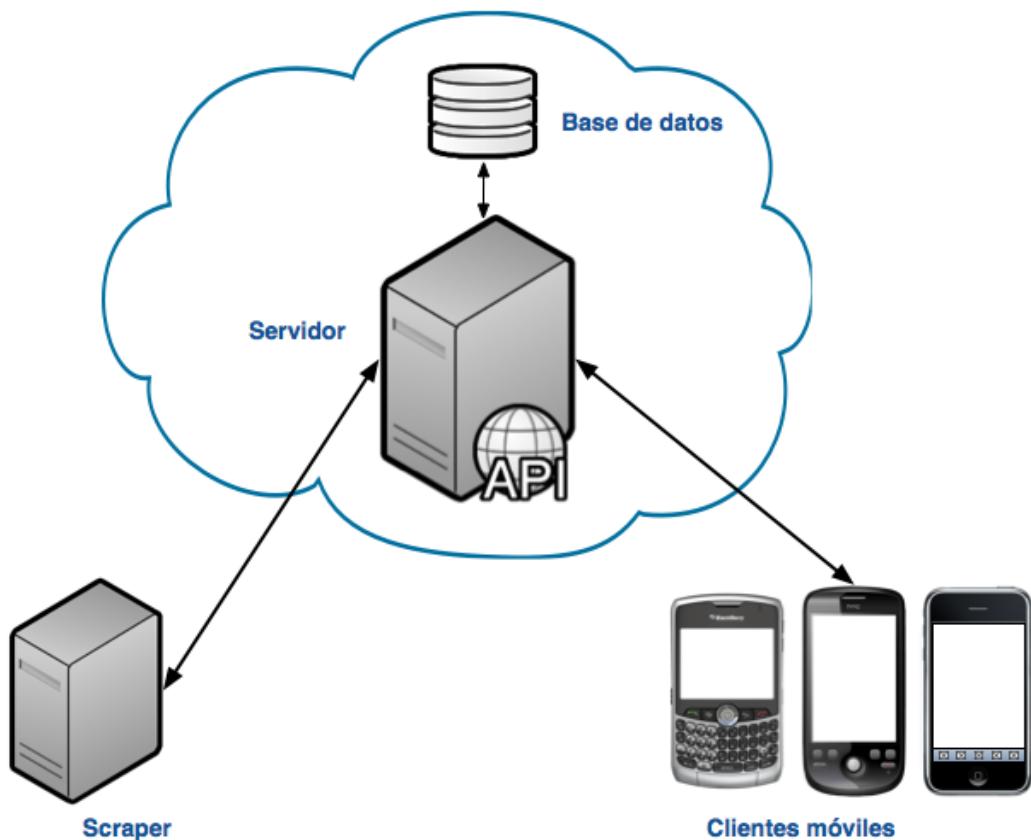


Figura 3.2.1: Arquitectura general del sistema.

De esta forma, la arquitectura del sistema a muy alto nivel puede dividirse en las siguientes partes o aplicaciones autónomas, cada una con un objetivo claramente definido y propio:

- **Servidor:** se presenta como la pieza angular del sistema, siendo el encargado de gestionar acceso, tanto para escritura como para lectura, a la base de datos. Adicionalmente tendrá como cometido el almacenamiento de las imágenes de los productos, actuando a su vez como servidor de archivos. De esta forma, el servidor, junto con la base de datos, actuará de pasarela entre la aplicación scraper y los clientes móviles, aislando así a los segundos de posibles fallos en el funcionamiento del primero y mejorando, por consiguiente, la estabilidad del sistema. Para poder proveer de los servicios que ofrece es necesario que disponga de una conexión remota con las demás aplicaciones del sistema que le permita establecer una comunicación con éstas. En concreto, deberá de estar conectado a

Internet dado que, por lo general, los dispositivos móviles no se encontrarán en la misma red física que el servidor.

- **Base de datos:** será el software encargado de almacenar toda la información de los productos de forma ordenada. Deberá de disponer de una comunicación directa con la aplicación servidora, no siendo necesario que tenga un acceso directo desde Internet. En ese sentido, será aconsejable que no pueda accederse de una forma directa a ella desde Internet, sino que el acceso siempre se realice a través del servidor, aumentando de esta forma el grado de seguridad del sistema y, en particular, de los datos dado que éstos son la base sobre la que se sustenta el proyecto. Por consiguiente, la situación ideal es que tanto la base de datos como el servidor estén siendo ejecutados en la misma máquina, lo que, además de facilitar la premisa anterior, reducirá el tiempo de acceso del servidor a la base de datos provocando un mejor rendimiento generalizado del sistema.
- **Aplicación scraper:** tiene como finalidad principal la obtención de los datos relativos a los productos a partir del análisis y consulta de una página web que contenga la información deseada, de ahí el término anglosajón utilizado para darle nombre, el cual hace referencia a este tipo concreto de aplicaciones. Por tanto, el software desarrollado será capaz de obtener los datos de los productos de una forma autónoma, pudiendo o no ser ejecutado en la misma máquina que el servidor.
- **Aplicaciones cliente:** como se ha comentado anteriormente, la aplicación cliente será ejecutada en dispositivos móviles, siendo la encargada de acceder, a través de peticiones al servidor, a la información que alberga la base de datos y mostrarla de forma correcta y ordenada al usuario. De esta forma, para establecer la conexión con el servidor, será necesario que dispongan de una conexión a Internet, ya sea mediante red WI-FI[68] o conexión de datos inalámbrica.

Por último, como parte también fundamental de la arquitectura del sistema se presenta el mecanismo de comunicación entre las diferentes aplicaciones, el cual será detallado más adelante.

3.2.2. Diseño de la base de datos

Tal y como se apuntó en el apartado anterior, la base de datos es una de las partes sobre las que se sustenta el sistema, siendo la encargada del almacenamiento de toda la información necesaria para el correcto funcionamiento de las aplicaciones. En este caso se ha optado por una base de datos de tipo relacional dado que éste cumple a la perfección con las características necesarias del sistema en cuanto a almacenado de datos y las relaciones existentes entre los mismos. Además, es un tipo de base de datos que se ha mostrado, desde sus primeras implementaciones surgidas en los años 70, como fiable y consistente para su uso en sistemas similares al que se desea desarrollar.

3.2.2.1. Diagrama entidad - relación

Atendiendo a uno de los principios básicos del diseño de bases de datos, inicialmente se procede a la creación del diagrama entidad-relación (ER), sobre el que se

cimentará la implementación de la base de datos y que permite, a muy alto nivel, tener un perspectiva conceptual de la misma. Puesto que este diagrama servirá como punto de partida para el diseño de la base de datos, es fundamental que se trate de un diseño que se ajuste a las necesidades del sistema y que sea elaborado con el mayor acierto posible. De esta forma, se facilitará la escalabilidad de la base de datos y, en consecuencia, la introducción de posibles nuevas capacidades y mejoras que se deseen agregar en un futuro al sistema.

En la figura 3.2.2 se muestra el diagrama ER propuesto para el sistema. Como puede verse, se trata de un diagrama bastante simple dado que, aunque se manejará una cantidad considerable de datos, se trabajará con un número reducido de entidades o tipos de dato.

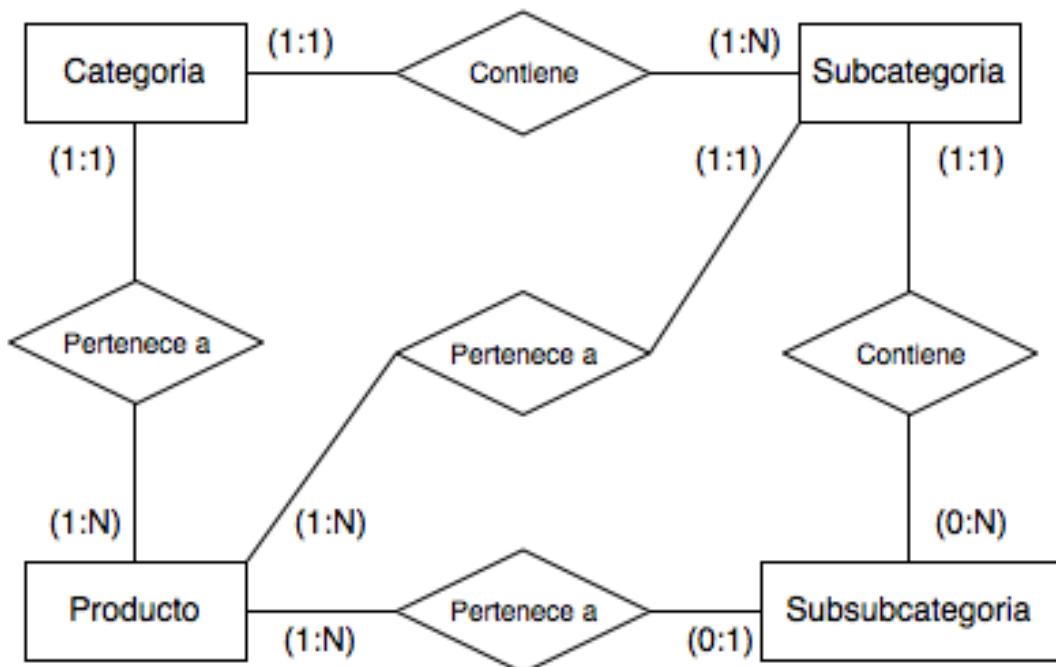


Figura 3.2.2: Diagrama entidad-relación de la base de datos.

Durante esta fase de diseño se han considerado cuatro entidades, siendo posible hacer una distinción entre tipos dos de entidades ya que, por un lado, la entidad Producto representa a un producto mientras que las tres restantes definen categorías a diferentes niveles a las que va a pertenecer un producto. Así, las categorías estarán relacionadas entre ellas mismas formando una estructura en árbol de manera que una subsubcategoría pertenecerá a una subcategoría, que, a su vez, pertenecerá a una categoría. En este punto, cabe resaltar como hecho destacado que, tal y como indican las cardinalidades representadas en el diagrama, una subcategoría puede tener o no subsubcategorías en un nivel inferior del árbol. En tal caso, un producto pertenecería únicamente a una subcategoría y a una categoría. Por otro lado, las entidades que han sido utilizadas para el diseño del diagrama ER son simples, no habiendo en ningún caso entidades relativas.

En la figura 3.2.3 pueden visualizarse las entidades **Categoría**, **Subcategoría** y **Subsubcategoría** junto con los atributos que tienen cada una:

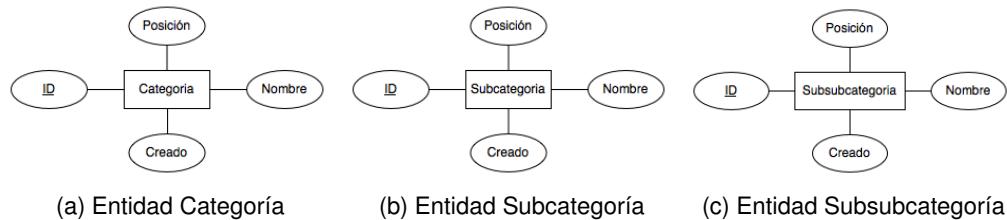


Figura 3.2.3: Atributos de entidades de tipo categoría

- **ID**: permite identificar únicamente a la categoría.
- **Nombre**: define el nombre de la categoría.
- **Creado**: establece la fecha en la que la categoría ha sido creada/almacenada.
- **Posición**: identifica la posición que tiene la categoría en el menú de la página web de la que se obtienen los productos y categorías.

En cuanto a la entidad **Producto**, se presenta como una entidad mas compleja que las anteriores, disponiendo de una mayor cantidad de atributos, los cuales pueden visualizarse en forma de diagrama en la figura 3.2.4.

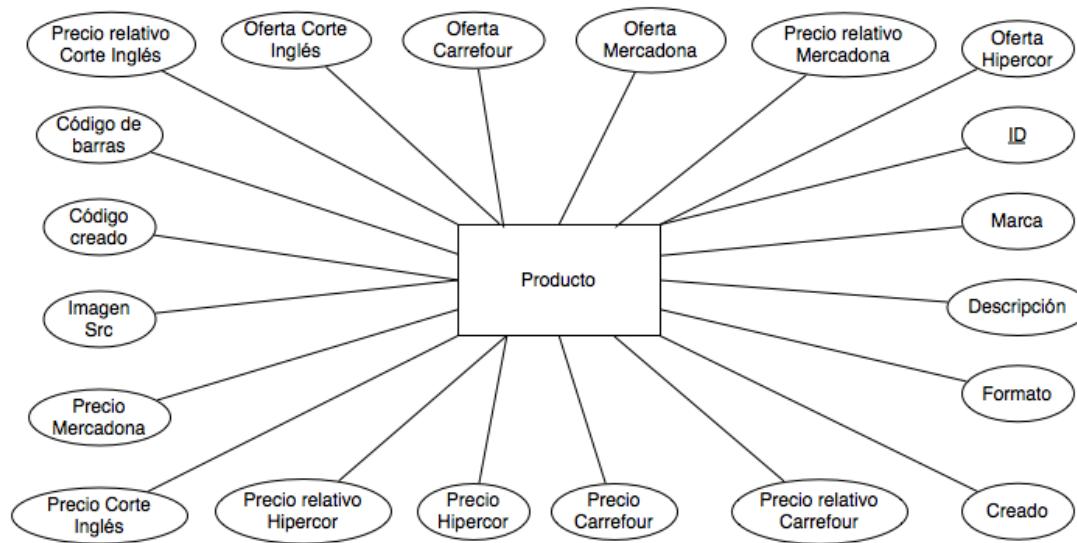


Figura 3.2.4: Atributos de la entidad Producto

- **ID**: permite identificar únicamente al producto.

- **Marca:** especifica la marca del producto.
- **Descripción:** contiene la descripción del producto.
- **Formato:** indica el formato en el que se comercializa el producto. Por ejemplo, una botella de agua de medio litro tendrá formato 1x50cl.
- **Creado:** almacena la fecha en la que se creó o se modificó alguno de los atributos del producto en la base de datos.
- **Precio XXXX:** contiene el precio del producto en el supermercado que indica la palabra XXXX.
- **Precio relativo XXXX:** representa el precio relativo al formato del producto en el supermercado que indica la palabra XXXX. Por ejemplo, si el producto en cuestión tiene un precio de 1€ y se comercializa en un envase de 0,5 Kg el precio relativo será 2€/Kg.
- **Oferta XXXX:** guarda la oferta asociada al producto en el supermercado indicado por XXXX.
- **Imagen Src:** almacena la URL de la imagen correspondiente al producto.
- **Código de barras:** representa el código de barras asociado al producto en formato EAN-13[18]
- **Código creado:** especifica la fecha en la que se agregó o modificó el código de barras al producto.

Es importante resaltar que todos los atributos que han sido propuestos para las diferentes entidades son de tipo monovaluados y no derivados. Además, para todas las entidades, se ha incluido un atributo ID que permite identificar únicamente a cada entidad. Por otro lado, las relaciones establecidas entre las diferentes entidades que conforman el diagrama ER son simples y sin atributos. Estas decisiones de diseño van a permitir, a posteriori, que la implementación de la base de datos y del resto del software del sistema sea más simple, eficiente y rápida.

3.2.2.2. Modelo relacional

Seguidamente, partiendo de la base proporcionada por el diagrama ER elaborado en la sección anterior, se procede al diseño del modelo relacional sobre el que trabajará la base de datos. Éste se representa en la figura 3.2.5 y muestra las tablas resultantes junto con los campos de cada una, tipo de dato que se utilizará para representar a éstos y las relaciones existentes entre las tablas.

En lo que refiere a los campos de las tablas, todos se corresponden con los atributos expuestos en el modelo ER aunque algunos han sido suprimidos ya que es posible obtenerlos como a partir de las relaciones entre las tablas. Así, el campo *categoría* de la tabla subcategorías ha sido eliminado, puesto que la relación de esta tabla y la de subcategorías permite su obtención. Por otro lado, en todos los casos se ha intentado utilizar el tipo de dato que mejor se adapta a la representación del campo, haciendo

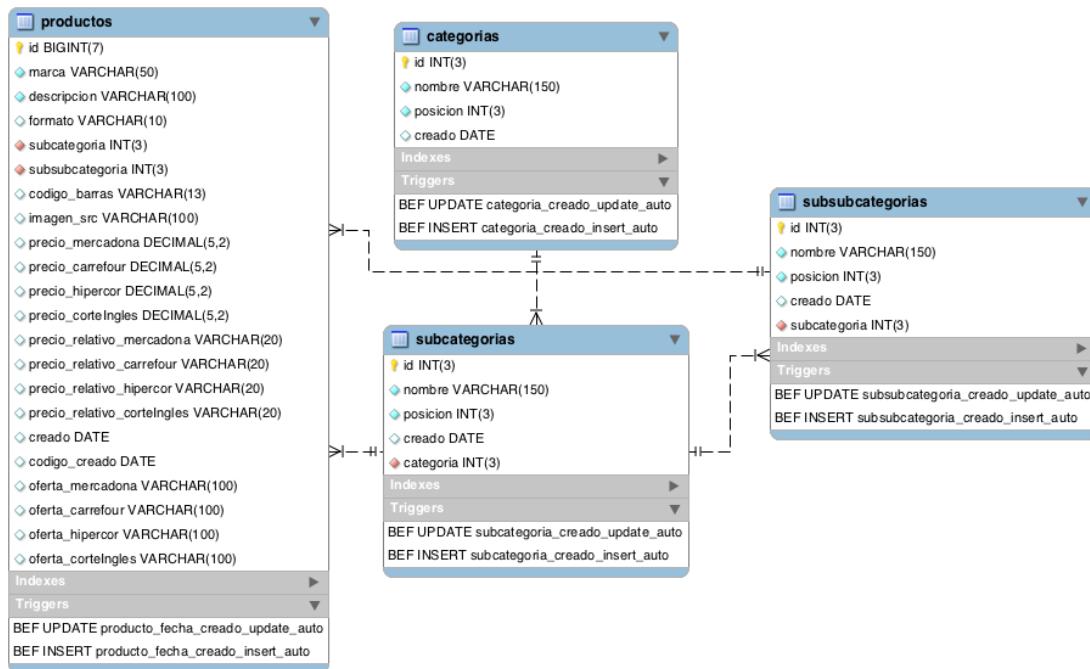


Figura 3.2.5: Modelo relacional de la base de datos

uso siempre de tipos de dato VARCHAR para aquellos campos que contengan texto. De esta manera, el espacio requerido para el almacenamiento de dichos campos será el menor posible ya que sólo se utilizará el espacio necesario y no un espacio fijo, lo que se refleja en un menor tamaño de la base de datos.

A nivel de diseño, lo más lógico hubiera sido identificar a las diferentes filas de cada una de las tablas utilizando claves primarias formadas por varios de los atributos. Por ejemplo, en la tabla subcategorías se podría haber utilizado como clave primaria la unión de los campos *posición* y *categoría*. No obstante, se toma la decisión de dejar a un lado esta idea y se proporciona un identificador numérico único a cada una de las filas. En consecuencia, se utiliza un número menor de columnas en las tablas provocando que complejidad de la base de datos sea más baja, lo que, una vez realizada la implementación, se traducirá en un mejor rendimiento de ésta. De esta forma, las relaciones entre las tablas se realizan mediante la migración de claves foráneas (atributos con rombo rojo) resultando éstas en la inclusión en la tabla correspondiente de una única columna, el *id*, en lugar de varias.

Continuando en el ámbito de las relaciones existentes entre las tablas, es notorio el hecho de que la tabla producto está relacionada con las tablas subcategorías y subsubcategorías, estando éstas relacionadas a su vez. Ésto es consecuencia de que todos los productos pertenecen obligatoriamente a una categoría y una subcategoría pero se dan casos en los que no pertenecen a una subsubcategoría. Por tanto, si únicamente se estableciera la relación del producto con la subsubcategoría, en caso de que el primero no pertenezca a ninguna subsubcategoría no sería posible poder relacionarlo con la subcategoría y categoría.

Por último, se ha hecho uso de ciertos *trigger* en cada una de las tablas. Éstos tienen un comportamiento similar en todos los casos, insertar de forma automática el valor de los campos de *creado* al agregar o modificar una fila. Adicionalmente, en el caso de la tabla producto, el *trigger* se encarga también de insertar el valor del campo

codigo_creado.

3.2.3. Diseño del scraper

Tal y como se apuntó anteriormente, el *scraper* será el software encargado de la extracción de la información relativa a los productos y las categorías. En concreto, la obtención de los datos se hará de la página web www.carritus.com, la cual fue analizada anteriormente y que reúne la toda la información necesaria para el desarrollo de la aplicación en una misma página web. La elección de esta web como base para la extracción de datos se fundamenta en el hecho de que dispone de una gran cantidad de productos y que posibilita la obtención de los precios de un producto en varios supermercados a partir de una sola web. De este modo, no será necesario consultar el precio e información de un producto en la web de cada uno de los supermercados, haciendo que este proceso sea mucho más rápido, siendo posible utilizar un único software en lugar de uno adaptado para cada establecimiento en concreto.

Hay que aclarar que un profundo análisis de la web www.carritus.com es fundamental para el desarrollo de todo el sistema y, principalmente, el del scraper. Así, la categorización de productos utilizada y la información ofrecida por dicha web es la que fundamenta el diseño de la base de datos realizado anteriormente.

El software a diseñar presenta grandes retos puesto que extraerá la información de forma automatizada no siendo posible, a priori, conocer qué información se encuentra en la web debido al enorme volumen de datos de que ésta dispone. Es por ello que se trata de una aplicación que se realizará específicamente para esta tarea, pudiendo presentar fallos en el momento en que la web cambie su estructura o la forma en la que se comporta. Por tanto, será imprescindible contar con herramientas que permitan la detección de fallos con el objetivo de que éstos sean subsanados a la mayor brevedad posible.

3.2.3.1. Estructura

La aplicación Scraper sigue un modelo arquitectónico basado en capas, disponiendo de las siguientes capas:

- Capa de presentación: permite al usuario tener un acceso rápido y fácil a las funcionalidades que ofrece el software a través del ratón. Además, será la encargada de mostrar la información al usuario.
- Capa de negocio: es la encargada de obtener los datos de la web [carritus.com](http://www.carritus.com) y dispone de comunicación directa con la capa de presentación, obteniendo de ésta las acciones indicadas por el usuario y ordenándole que muestre la información pertinente en cada momento de la ejecución del programa. Adicionalmente, proporciona las capacidades necesarias para almacenar ficheros de log que permitan al usuario realizar un seguimiento del funcionamiento del programa, con el objetivo de detectar fallos.
- Capa de datos: la persistencia de los datos obtenidos se realizará de forma remota, siendo éstos almacenados en la base de datos a través del servidor. Por tanto, en la aplicación Scraper esta capa estará dedicada al envío de los datos al servidor para su almacenamiento.

A continuación se describen de una forma más detallada los módulos de los que consta la aplicación mediante tablas. Éstas dispondrán de los siguientes campos:

- Identificador: permite distinguir únicamente al módulo y utiliza un código que seguirá el patrón MOD-SCR-X, donde X será un número que comenzará en 1.
- Propósito: define el objetivo del módulo.
- Capa: indica la capa de la estructura a la que pertenece el módulo.
- Funciones: lista las capacidades que desarrolla el módulo.
- Requisitos: contiene el código identificativo de los requisitos funcionales y no funcionales que se aplican al módulo.

MOD-SCR-1: GUI

Propósito:

Su objetivo es permitir la interacción del usuario con la aplicación y mostrarle la información relevante.

Capa: presentación.

Funciones:

1. Informar del estado del scraper.
2. Mostrar y seleccionar categorías.
3. Seleccionar los valores de frecuencia y hora de inicio para la programación de la obtención de datos de categorías y productos.
4. Iniciar y detener la obtención de datos de categorías y productos.

Requisitos:

RF-1, RF-2, RF-3, RF-4, RNF-7, RNF-8, RNF-11, RNF-14.

Tabla 3.2.1: MOD-SCR-1: GUI

MOD-SCR-2: Navegador

Propósito:

Tiene como finalidad permitir la navegación en la página web www.carritus.com

Capa: negocio.

Funciones:

1. Obtener las páginas web.
2. Navegar a través de las categorías, subcategorías y subsubcategorías.
3. Navegar entre supermercados.
4. Realizar el login en la web.

Requisitos:

RF-1, RF-2, RNF-2, RNF-7, RNF-13, RNF-14.

Tabla 3.2.2: MOD-SCR-2: navegador.

MOD-SCR-3: Scraper
Propósito: El objetivo del módulo será extraer la información de las páginas web que le ofrece el navegador.
Capa: negocio.
Funciones: <ol style="list-style-type: none">1. Obtener producto.2. Obtener categoría.3. Obtener subcategoría.4. Obtener subsubcategoría.5. Almacena temporalmente la información extraída.
Requisitos: RF-1, RF-2, RNF-7, RNF-13, RNF-14.

Tabla 3.2.3: MOD-SCR-3: scraper.

MOD-SCR-4: Log
Propósito: Es el encargado de almacenar un registro con los eventos que ocurren en el programa en ficheros de texto.
Capa: negocio.
Funciones: <ol style="list-style-type: none">1. Guardar registro de errores producidos.2. Guardar registro de eventos ocurridos durante el scraping de categorías.3. Guardar registro de eventos ocurridos durante el scraping de productos.
Requisitos: RNF-7, RNF-8, RNF-14.

Tabla 3.2.4: MOD-SCR-4: log.

MOD-SCR-5: Comunicación.	
Propósito:	Tiene como objetivo enviar la información extraída al servidor para que éste la almacene.
Capa: datos.	
Funciones:	<ol style="list-style-type: none"> 1. Enviar productos al servidor. 2. Enviar categorías al servidor.
Requisitos:	RNF-2, RNF-7, RNF-13, RNF-14.

Tabla 3.2.5: MOD-SCR-5: comunicación.

3.2.3.2. Interfaz gráfica

Para facilitar el uso del scraper por parte del usuario administrador, el programa dispondrá de una interfaz gráfica de usuario que permita seleccionar las categorías a las que pertenecen los productos que se desean obtener así como botones que faciliten el inicio y automatización de los scraping de productos y categorías. Asimismo, también se mostrará información al usuario.

Al tratarse de una aplicación que no está destinada al usuario final, se considera que el aspecto visual de la misma no es prioritario, tratando de que prevalezca la usabilidad por encima de éste. Por ello se propone una interfaz gráfica cuyo diseño se muestra en la figura 3.2.6.

3.2.4. Diseño del cliente

3.2.4.1. Estructura

De forma similar al Scraper, la aplicación cliente será construída de nuevo siguiendo el paradigma de arquitectura por capas:

- Capa de presentación: permite la interacción del usuario con la aplicación y le muestra la información que éste solicita. En este caso conlleva una importancia mayor si cabe puesto que será la única parte de todo el sistema que visualizará el usuario final y, además, puesto que se trata de una aplicación para dispositivos móviles de última generación, permitirá la interacción mediante toques y gestos táctiles.
- Capa de lógica de negocio: es la encargada de proporcionar las capacidades necesarias para que se desarrollen las funciones de la aplicación, gestionando las peticiones del usuario recibidas a través de la capa de presentación y solicitando información a la capa de acceso a datos.
- Capa de acceso a datos: permite almacenar y extraer la información con la que trabaja la aplicación. En este caso, la información sobre listas de productos y favoritos se almacenará de forma remota, siendo esta capa la encargada de llevar

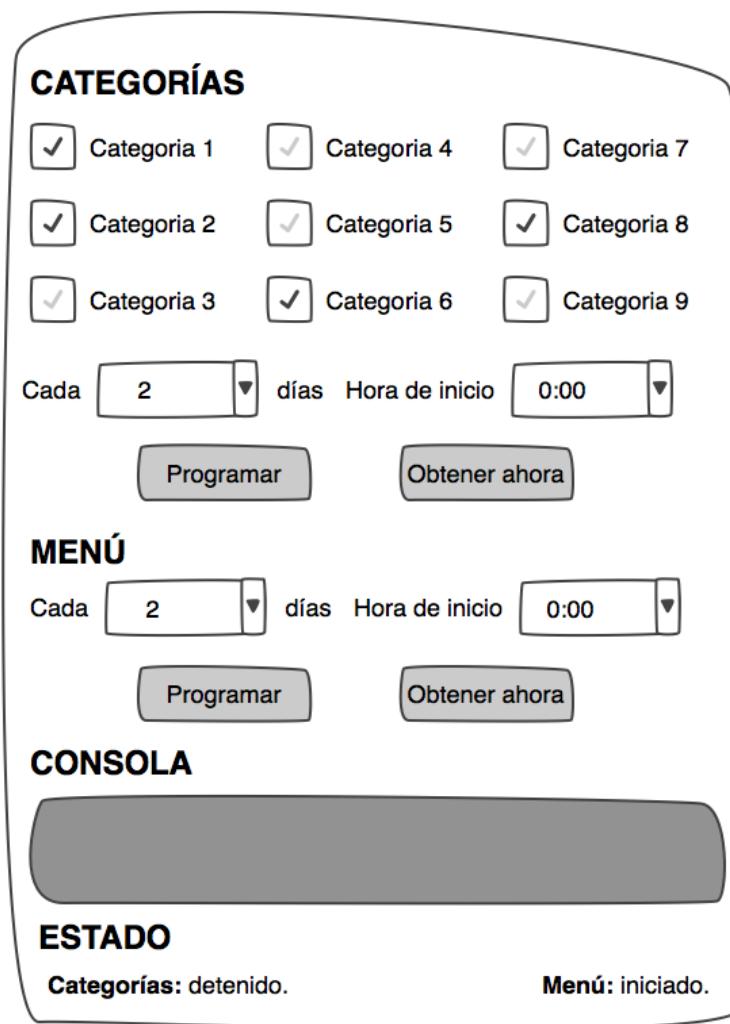


Figura 3.2.6: Diseño de la interfaz gráfica de usuario del scraper

a cabo la persistencia de los datos. Por otro lado, el acceso a los datos sobre los productos y categorías se realiza, nuevamente, de forma remota puesto que dicha información se encuentra en la base de datos.

A continuación, se muestran de forma detallada los módulos de que se compondrá la aplicación, siguiendo mismo patrón utilizado anteriormente para la definición de los módulos del Scraper pero utilizando un identificador de la forma MOD-CLI-X.

MOD-CLI-1: GUI	
Propósito:	Su objetivo es permitir la interacción del usuario con la aplicación y mostrarle la información relevante.
Capa:	presentación.
Funciones:	<ol style="list-style-type: none">1. Mostrar listas de productos.2. Mostrar productos de una lista.3. Iniciar eliminación de lista.4. Iniciar la creación de listas.5. Iniciar la comparación de una lista de productos.6. Iniciar visualización de lista en modo compra.7. Visualización de categorías, subcategorías y subsubcategorías.8. Visualización de productos.9. Iniciar proceso de inclusión de un producto en una lista.10. Iniciar proceso de inclusión de un producto en favoritos.11. Recoger el texto introducido por el usuario e iniciar la búsqueda de productos por texto.12. Iniciar el escaneo del código de barras de productos.13. Mostrar favoritos14. Iniciar inclusión de producto favorito en lista.15. Iniciar eliminación de producto favorito.
Requisitos:	RF-5, RF-6, RF-7, RF-8, RF-9, RF-10, RF-11, RNF-1, RNF-8, RNF-10, RNF-11, RNF-14.

Tabla 3.2.6: MOD-CLI-1: GUI

MOD-CLI-2: Notificaciones	
Propósito:	Tiene como finalidad mostrar notificaciones al usuario para informarle de los eventos ocurridos.
Capa:	presentación.
Funciones:	<ol style="list-style-type: none"> 1. Mostrar notificación al agregar producto a lista. 2. Mostrar notificación al agregar producto a favoritos. 3. Mostrar notificación al agregar eliminar lista. 4. Mostrar notificación al eliminar producto de lista. 5. Mostrar notificación al eliminar producto de favoritos. 6. Mostrar mensajes de error. 7. Mostrar información sobre ofertas en ventanas emergentes.
Requisitos:	RF-5, RF-6, RF-7, RF-8, RF-9, RF-10, RF-11, RNF-1, RNF-8, RNF-10, RNF-11, RNF-14

Tabla 3.2.7: MOD-CLI-2: Notificaciones

MOD-CLI-3: Controlador	
Propósito:	Su función es construir las vistas de la interfaz gráfica de usuario (GUI) a partir de los datos que se reciben del servidor y gestionar los eventos que se generan a partir de las interacciones del usuario con el dispositivo.
Capa:	negocio.
Funciones:	<ol style="list-style-type: none"> 1. Gestionar eventos de pulsación sobre elementos de la GUI. 2. Gestionar eventos de acelerómetro. 3. Gestionar cambios de vista. 4. Construcción de vistas.
Requisitos:	RF-5, RF-6, RF-7, RF-8, RF-9, RF-10, RF-11, RNF-1, RNF-2, RNF-12, RNF-13, RNF-14, RNF-16.

Tabla 3.2.8: MOD-CLI-3: Controlador

MOD-CLI-4: Comunicaciones	
Propósito:	Su objetivo es realizar las peticiones oportunas al servidor.
Capa:	acceso a datos.
Funciones:	<ol style="list-style-type: none"> 1. Obtener categorías del servidor. 2. Obtener subcategorías del servidor. 3. Obtener subsubcategorías del servidor. 4. Obtener productos del servidor. 5. Obtener comparación de productos del servidor.
Requisitos:	RF-5, RF-6, RNF-1, RNF-2, RNF-12, RNF-13, RNF-14.

Tabla 3.2.9: MOD-CLI-4: Comunicaciones

MOD-CLI-5: Acceso a datos	
Propósito:	Tiene como función gestionar el acceso y persistencia de los datos que se guardan de forma local en el cliente.
Capa:	acceso a datos.
Funciones:	<ol style="list-style-type: none"> 1. Obtener listas. 2. Obtener productos de lista. 3. Obtener favoritos. 4. Guardar lista. 5. Guardar favorito. 6. Guardar producto en lista. 7. Eliminar lista. 8. Eliminar producto de lista. 9. Eliminar favorito.
Requisitos:	RF-7, RF-8, RF-10, RF-11, RNF-1, RNF-12, RNF-13, RNF-14.

Tabla 3.2.10: MOD-CLI-5: Acceso a datos

3.2.4.2. Base de datos

Tal y como se ha apuntado en el apartado anterior, el cliente tendrá la capacidad de guardar información de forma local, de manera que el usuario final pueda consultarla

sin necesidad de disponer de conexión a Internet. De esta forma, los datos que serán almacenados serán los relativos a listas de productos y favoritos y el acceso a los mismos será gestionado por el módulo de acceso a datos del cliente.

Al igual que para la base de datos general del sistema, se hace uso de una base de datos de tipo relacional. Por tanto, se procede al diseño del diagrama entidad-relación (ER) que proporciona una primera visión general de las entidades que modelan la información a almacenar así como las relaciones entre las mismas. Este diagrama se muestra en la figura 3.2.7. De nuevo, siguiendo las mismas pautas que para el diseño de la base de datos del sistema, se utilizan relaciones simples entre entidades y atributos simples monovaluados, así como un atributo ID que identifica únicamente a la entidad, siempre teniendo como objetivo obtener un diseño sencillo y eficiente.

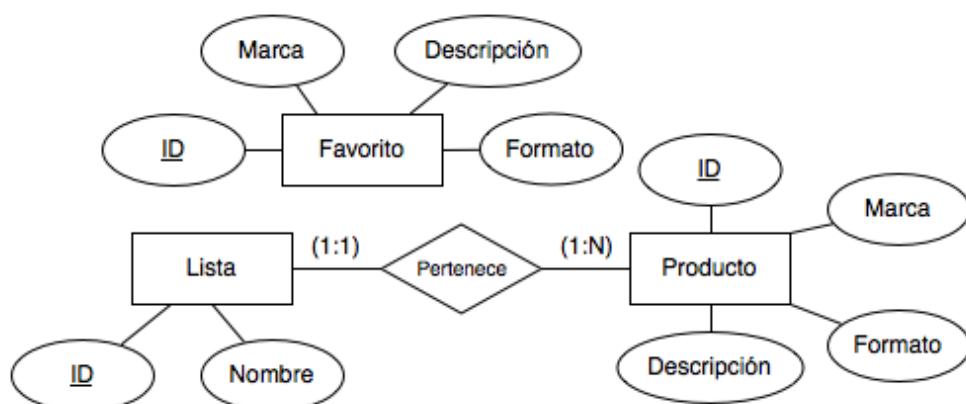


Figura 3.2.7: Diagrama ER base de datos del cliente

Tal y como puede comprobarse, se dispone de tres entidades: lista, producto y favorito. De esta manera, las entidades *Lista* y *Producto* mantienen una relación de pertenencia, es decir, las listas contendrán o estarán formadas por productos. Por otro lado, se encuentra la entidad favorito, que no guarda relación con ninguna de las otras entidades aunque si que comparte sus campos con la entidad *Producto*. Ésto es debido a que, en realidad, ambas entidades representan a un producto, con la particularidad de que los favoritos incluyen una connotación de realce sobre el producto.

Una vez elaborado el diagrama ER, se procede a la elaboración del modelo relacional, que plasma en forma de tablas las entidades y relaciones de las mismas diseñadas anteriormente. Este modelo puede verse en la figura 3.2.8 y, al igual que diagrama ER, es bastante simple. En este caso, todas las tablas disponen de una clave primaria (campo id) que permite identificar a las filas de forma única. Por otro lado, la relación existente entre las tablas *Lista* y *Producto* se plasma como una migración de la clave primaria de la primera (campo id) a la segunda (campo lista).

3.2.4.3. Interfaz gráfica

El diseño de la interfaz gráfica de usuario (GUI) de aplicación cliente supone uno de los mayores retos en el desarrollo de la aplicación dado que será la parte que el usuario

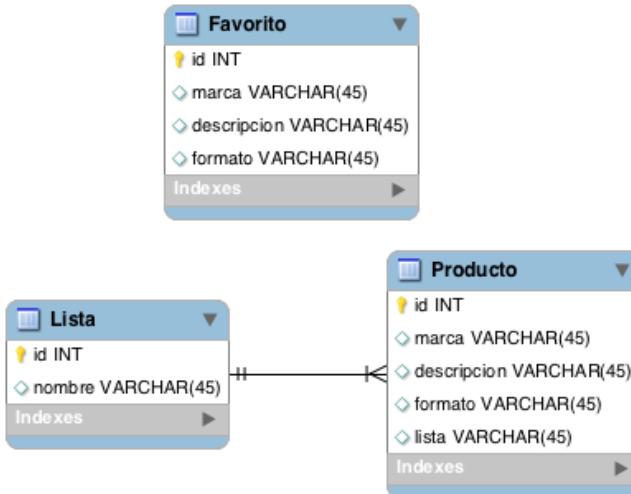


Figura 3.2.8: Modelo relacional base de datos del cliente

pueda ver, como es habitual, pero adicionalmente, en este caso, el usuario final podrá tocarla. Ésto permite nuevas posibilidades, dado que el usuario ya no está limitado al uso del ratón sino que puede acceder a las diferentes funcionalidades tocando partes de la pantalla e incluso tiene la posibilidad de arrastrar o mover elementos.

El diseño de la GUI se considera esencial ya que un correcto diseño de la navegación entre pantallas y de la colocación de los diferentes elementos que las componen provocará que el uso de la aplicación por parte del usuario sea más ágil y le permita acceder a las funcionalidades que desea de una forma más rápida y sencilla. Además, la interfaz gráfica en las aplicaciones, especialmente en aquellas destinadas a las aplicaciones móviles, supone un elemento de distinción y un factor clave en la elección del usuario de un software u otro de entre los miles disponibles, primando, en muchos de los casos, un diseño vistoso y eficaz de la interfaz gráfica por encima las funcionalidades que ofrece la aplicación.

Por tanto, con el objetivo de conseguir un diseño acertado y, de esta forma, intentar atraer a un mayor número de usuarios se seguirán los siguientes patrones de diseño:

- La pantalla inicial cumplirá el patrón *Dashboard*¹, es decir, mostrará un mosaico con las principales funciones que permite realizar la aplicación.
- En todo momento, excepto en la pantalla inicial, se dispondrá de una barra superior que permita una navegación directa hacia la pantalla anterior y hacia la pantalla de inicio y que mostrará el nombre de la pantalla, cumpliendo con lo establecido por el patrón *ActionBar*. Además, cuando se estime oportuno, se incluirá una segunda barra en la parte inferior de la pantalla que mostrará los botones que permitan un acceso instantáneo a las funciones más habituales que realiza el usuario en la pantalla que se muestra. Ésto se traducirá en una navegación más sencilla, rápida y efectiva.
- La información se mostrará en forma de lista, puesto que ésta representación es útil y fácil de usar, a la vez que amigable y natural para el usuario. Éstas a su vez

¹Tanto el patrón Dashboard como el ActionBar son recomendaciones, basadas en la experiencia en el diseño de interfaces gráficas de usuario para aplicaciones móviles, realizadas por Google. Para más información, consultar la referencia bibliográfica Google [74]

podrán moverse manteniéndola pulsada y deslizando el dedo sobre la pantalla en la dirección deseada.

- La navegación se hará a través de pulsaciones cortas sobre los botones que se encuentren en la pantalla, tratando siempre de no cargar demasiado la pantalla con un exceso de elementos.
- Todos los botones dispondrán de un ícono que escenifique el comportamiento de mismo.
- Se harán diseños simples de las pantallas, utilizando colores básicos que ofrezcan un alto contraste entre ellos para los elementos y los textos.

Una vez establecidas las bases sobre las que se elaborará el diseño de la interfaz gráfica, a continuación se esbozan las diferentes pantallas que la compondrán junto con las posibles transiciones que se realizan tras pulsar los diferentes botones que se muestran en la pantalla.

- **Pantalla inicial:**

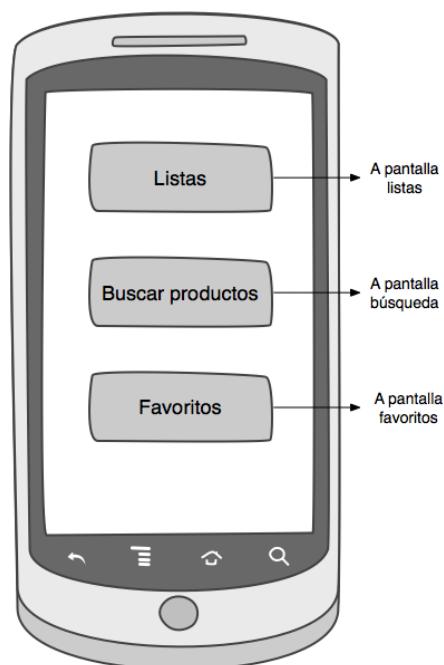


Figura 3.2.9: GUI cliente: pantalla inicial.

- **Pantalla listas:**

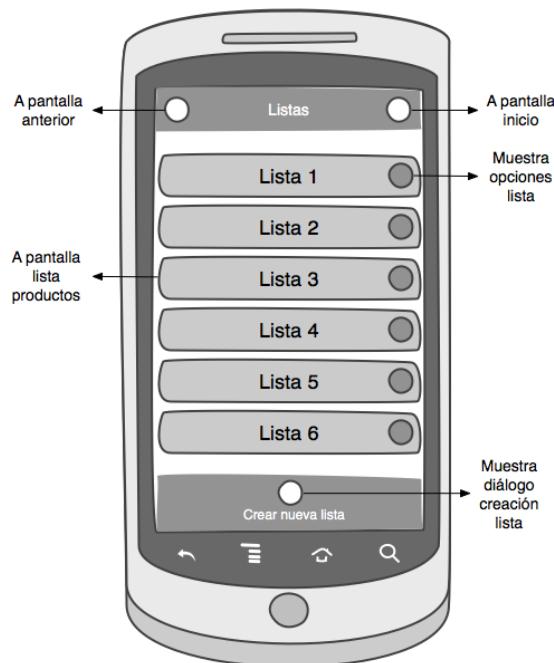


Figura 3.2.10: GUI cliente: pantalla listas.

- **Pantalla lista productos:**

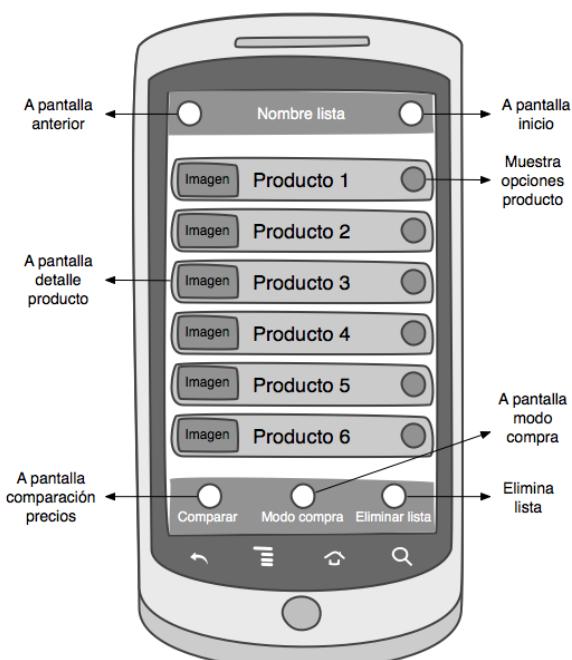


Figura 3.2.11: GUI cliente: pantalla lista productos.

- **Pantalla comparación de precios:**



Figura 3.2.12: GUI cliente: pantalla comparación precios.

- **Pantalla modo compra:**

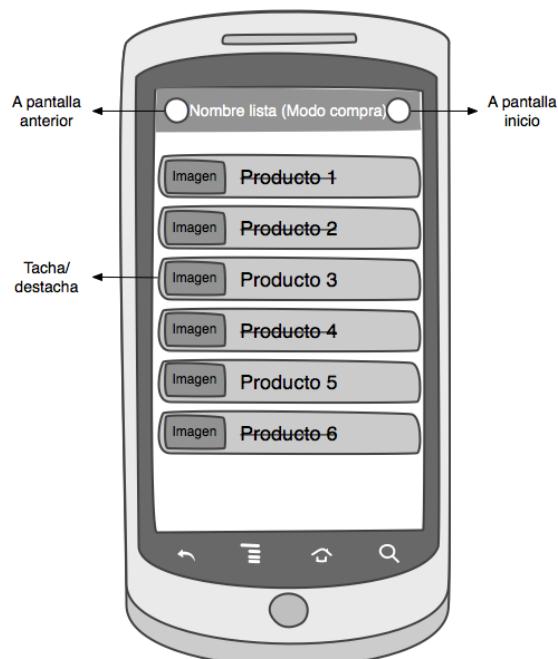


Figura 3.2.13: GUI cliente: pantalla comparación precios.

- Pantalla opciones de búsqueda:

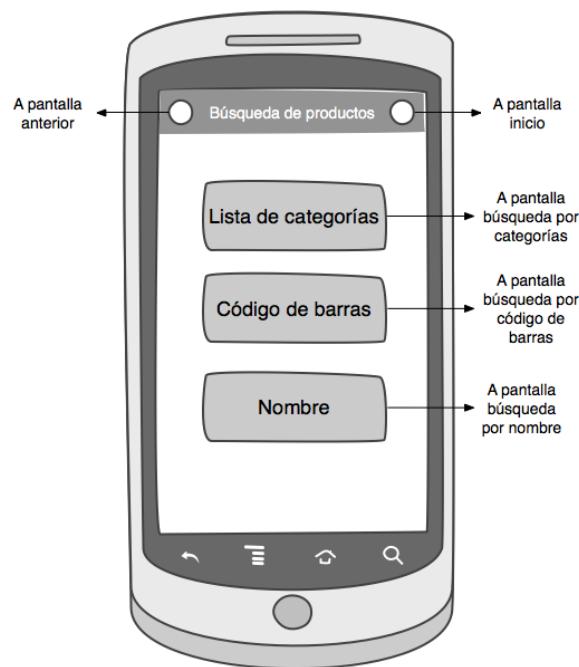


Figura 3.2.14: GUI cliente: pantalla opciones búsqueda.

- Pantallas búsqueda por categorías:

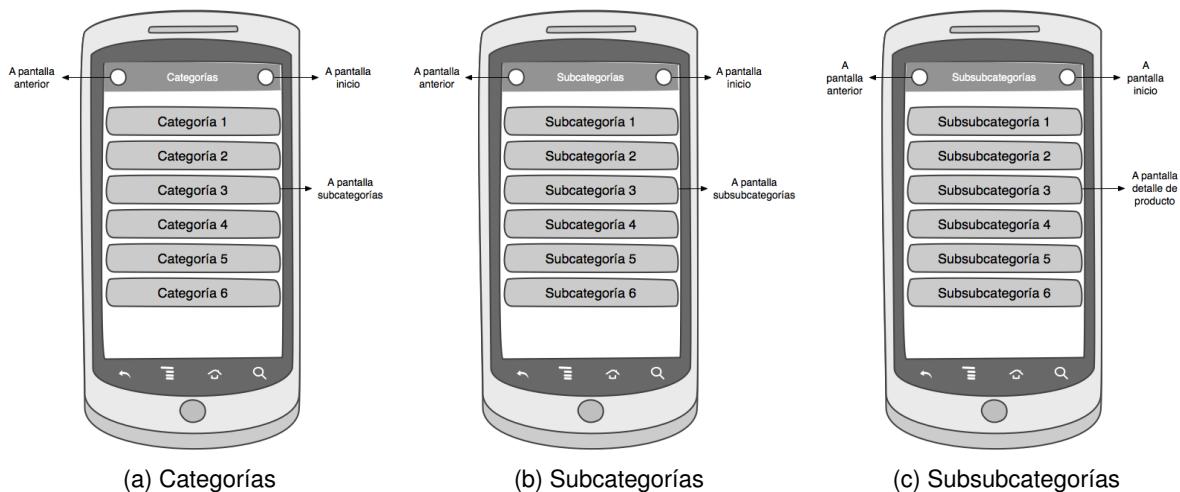


Figura 3.2.15: GUI cliente: pantallas búsqueda por categorías.

- **Pantalla búsqueda por código de barras:**

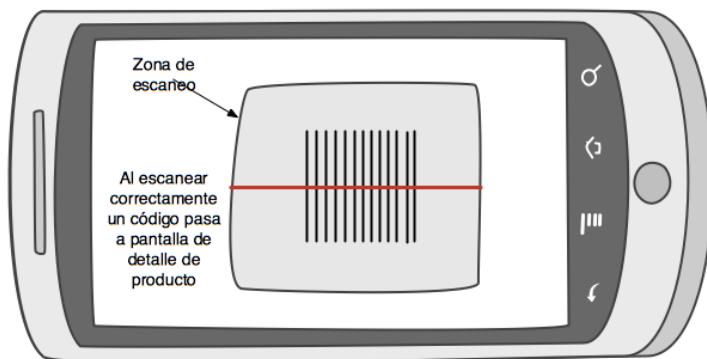


Figura 3.2.16: GUI cliente: pantalla búsqueda por código de barras.

- **Pantallas búsqueda por nombre:**

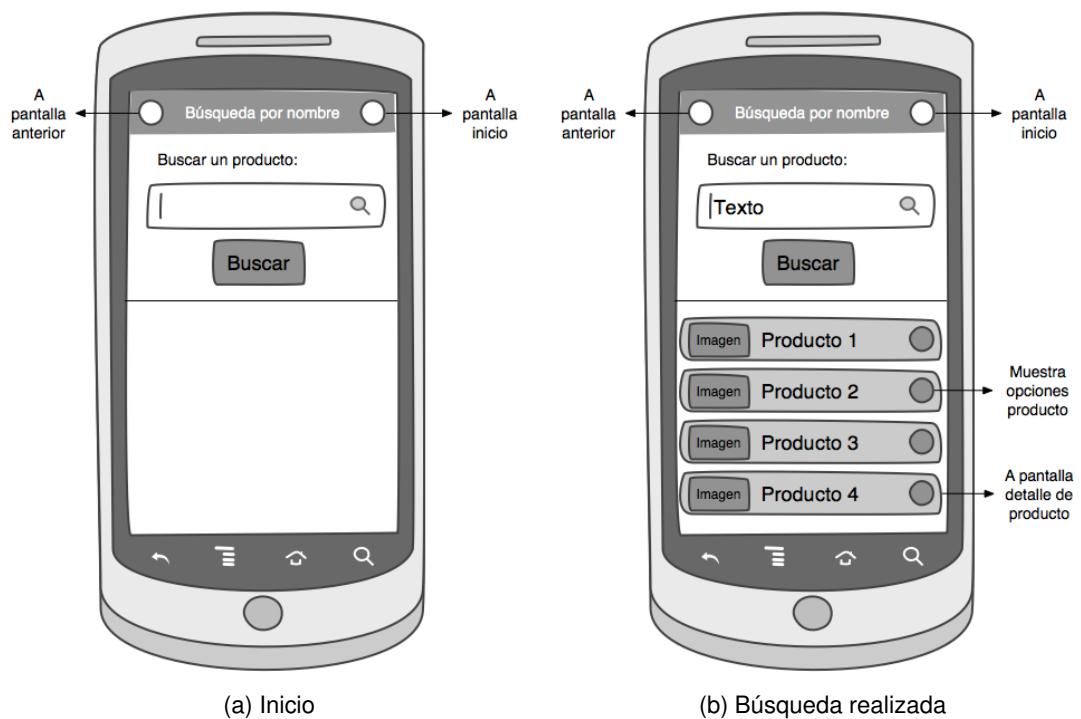


Figura 3.2.17: GUI cliente: pantallas búsqueda por nombre.

- Pantalla favoritos:



Figura 3.2.18: GUI cliente: pantalla favoritos.

- Pantalla detalle producto:

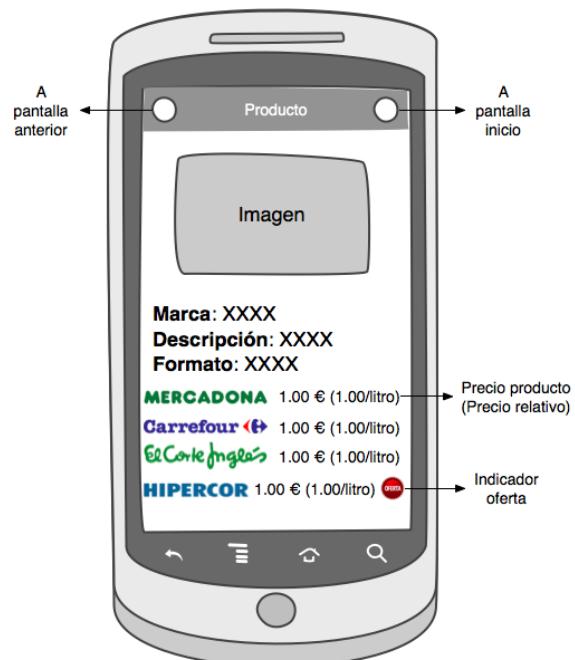


Figura 3.2.19: GUI cliente: pantalla detalle producto.

3.2.5. Diseño del servidor

El servidor se presenta como uno de los ejes principales del sistema puesto que va a ser el encargado de la gestión del acceso a la base de datos. De esta forma, recibirá las peticiones del cliente y del scraper para obtener datos y almacenarlos correspondientemente.

Tanto el scraper como el cliente móvil podrían acceder directamente a la base de datos para realizar las operaciones que necesiten pero se decide la inclusión del servidor para mantener una mayor separación entre las diferentes aplicaciones, aumentando tanto la seguridad como la escalabilidad del sistema completo. Así, el servidor actúa como pasarela entre las aplicaciones y la base de datos, escuchando las peticiones HTTP que recibe y actuando en consecuencia.

En principio, el servidor debe ejecutarse en la misma máquina que la base de datos de manera que el acceso a ésta se produzca de una forma rápida sin necesidad de comunicación a través de ningún tipo de red. Además, así podría cerrarse el acceso a la base de datos desde Internet, dificultando, por tanto, el acceso a intrusos. No obstante, si fuera necesario, ambos programas podrían ejecutarse en máquinas distintas aunque el rendimiento del sistema decaería.

3.2.5.1. Estructura

Las posibilidades para la implementación del servidor son muy amplias y por tanto, habrá que decidir una. Así, para este caso se establecen dos tipos de estructura posibles a implementar:

3.2.5.1.1. Servidor Web

Puesto que el servidor recibirá peticiones HTTP a través de Internet, es posible utilizar un servidor Web para tal fin. De esta forma, el servidor tendrá acceso a la base de datos a través de programas que corran sobre él mismo y que realicen las operaciones que se requieren, en este caso almacenamiento y obtención de datos. Por tanto, la implementación se realizaría mediante un modelo por capas tal y como se muestra en la figura 3.2.20. De esta forma, en la parte superior se encuentra el servidor web, que será el módulo al que llegarán las peticiones de los clientes. Éste trasladará, hacia abajo, la petición al módulo encargado de la gestión de la misma. Posteriormente, la capa de acceso a base de datos realizará la operación necesaria en la base de datos. Por último, la respuesta será generada por el módulo encargado de la gestión de la petición y será enviada de vuelta al cliente por el servidor Web.



Figura 3.2.20: Servidor: estructura servidor Web.

A continuación, se van a definir las funciones que realiza cada uno de los módulos anteriores en formato tabla de forma similar a como se ha hecho anteriormente e identificando a cada uno de los módulos con la nomenclatura MOD-SERV-WEB-X.

MOD-SERV-WEB-1: Servidor Web	
Propósito:	Tiene como objetivo gestionar la comunicación con los sistemas que realizan las peticiones.
Funciones:	<ol style="list-style-type: none"> 1. Escucha las peticiones HTTP de los clientes. 2. Envía los mensajes de respuesta a los clientes.
Requisitos:	RF-1, RF-2, RF-5, RF-6, RF-9, RNF-2, RNF-3, RNF-4, RNF-12, RNF-13, RNF-15.

Tabla 3.2.11: MOD-SERV-WEB-1: Servidor Web

MOD-SERV-WEB-2: Acceso a base de datos	
Propósito:	Es el encargado de realizar las consultas a la base de datos.
Funciones:	<ol style="list-style-type: none"> 1. Obtener datos de la base de datos. 2. Insertar datos en la base de datos.
Requisitos:	RF-1, RF-2, RF-5, RF-6, RF-9, RNF-4, RNF-9, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.12: MOD-SERV-WEB-2: Acceso a base de datos

MOD-SERV-WEB-3: Comprobar	
Propósito:	Tiene como objetivo gestionar las peticiones de comparación de precio de productos de una lista.
Funciones:	<ol style="list-style-type: none"> 1. Obtener la comparación de precios de una lista de productos y generar la respuesta que se enviará al cliente.
Requisitos:	RF-9, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.13: MOD-SERV-WEB-3: Comprobar

MOD-SERV-WEB-4: Obtener categorías

Propósito:

Su objetivo es gestionar las peticiones de obtención de categorías

Funciones:

1. Obtener la lista de categorías de la base de datos y generar la respuesta que se envía al cliente.

Requisitos:

RF-5, RF-6, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.14: MOD-SERV-WEB-4: Obtener categorías

MOD-SERV-WEB-5: Obtener subcategorías

Propósito:

Su objetivo es gestionar las peticiones de obtención de subcategorías

Funciones:

1. Obtener, de la base de datos, la lista de subcategorías que pertenecen a una categoría y generar la respuesta que se envía al cliente.

Requisitos:

RF-5, RF-6, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.15: MOD-SERV-WEB-5: Obtener subcategorías

MOD-SERV-WEB-6: Obtener subsubcategorías

Propósito:

Su objetivo es gestionar las peticiones de obtención de subsubcategorías

Funciones:

1. Obtener, de la base de datos, la lista de subsubcategorías que pertenecen a una subcategoría y generar la respuesta que se envía al cliente.

Requisitos:

RF-5, RF-6, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.16: MOD-SERV-WEB-6: Obtener subsubcategorías

MOD-SERV-WEB-7: Obtener producto**Propósito:**

Su objetivo es gestionar las peticiones de obtención de producto

Funciones:

1. Obtener, de la base de datos, la información relativa a un producto y generar la respuesta que se envía al cliente.

Requisitos:

RF-5, RF-6, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.17: MOD-SERV-WEB-7: Obtener producto

MOD-SERV-WEB-8: Guardar categoría**Propósito:**

Su objetivo es gestionar las peticiones de guardado de categorías

Funciones:

1. Gestionar el almacenamiento de categorías en la base de datos.

Requisitos:

RF-2, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.18: MOD-SERV-WEB-8: Guardar categoría

MOD-SERV-WEB-9: Guardar subcategoría**Propósito:**

Su objetivo es gestionar las peticiones de guardado de subcategorías

Funciones:

1. Gestionar el almacenamiento de subcategorías en la base de datos.

Requisitos:

RF-2, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.19: MOD-SERV-WEB-9: Guardar subcategoría

MOD-SERV-WEB-10: Guardar subsubcategoría**Propósito:**

Su objetivo es gestionar las peticiones de guardado de subsubcategorías

Funciones:

1. Gestionar el almacenamiento de subsubcategorías en la base de datos.

Requisitos:

RF-2, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.20: MOD-SERV-WEB-10: Guardar subsubcategoría

MOD-SERV-WEB-11: Guardar producto	
Propósito:	Su objetivo es gestionar las peticiones de guardado de productos
Funciones:	<ol style="list-style-type: none">1. Gestionar el almacenamiento de productos en la base de datos.
Requisitos:	RF-2, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.21: MOD-SERV-WEB-11: Guardar producto

3.2.5.1.2. Servicio Web RESTful

Para la implementación del servidor se propone esta segunda alternativa, la cual supone una arquitectura diferente basada en el uso de un servicio web RESTful, por lo que se hace necesario el uso de un servidor de aplicaciones en lugar de un servidor Web. Esta opción se presenta como más avanzada, robusta, escalable, compacta y simple que la anterior. Al igual que en el caso anterior, los datos son transmitidos a través del protocolo HTTP.

La arquitectura propuesta para esta solución es la que se refleja en la figura 3.2.21 y está basada en 4 capas formadas por un módulo por capa. En este caso, será el servidor de aplicaciones el encargado de recibir y enviar las peticiones HTTP, las cuales se trasladarán de arriba hacia abajo al ser recibidas y desde abajo hacia arriba de nuevo para devolver una respuesta al cliente.



Figura 3.2.21: Servidor: estructura servicio RESTful.

De la misma forma que se hizo anteriormente, a continuación, se van a definir con mayor detalle las funcionalidades propias de cada uno de los módulos que intervienen en la implementación del servicio web RESTful pero utilizando el identificador MOD-SERV-REST-X.

MOD-SERV-REST-1: Servidor de aplicaciones	
Propósito:	Tiene como fin servir de soporte para la ejecución de los demás módulos.
Funciones:	<ol style="list-style-type: none"> 1. Ejecutar los demás módulos. 2. Recibir las peticiones y enviar los mensajes de respuesta a los clientes.
Requisitos:	RF-1, RF-2, RF-5, RF-6, RF-9, RNF-2, RNF-3, RNF-4, RNF-12, RNF-13, RNF-15.

Tabla 3.2.22: MOD-SERV-REST-1: Servidor de aplicaciones

MOD-SERV-REST-2: Gestor de peticiones	
Propósito:	Su objetivo es realizar el tratamiento de las peticiones que llegan desde los clientes así como confeccionar los mensajes de respuesta.
Funciones:	<ol style="list-style-type: none"> 1. Tratar las peticiones y adaptarlas al módulo de la lógica de negocio. 2. Elaborar el mensaje que se devuelve al cliente a partir de la respuesta que genera la capa dedicada a la lógica de negocio.
Requisitos:	RF-1, RF-2, RF-5, RF-6, RF-9, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.23: MOD-SERV-REST-2: Gestor de peticiones

MOD-SERV-REST-3: Lógica de negocio	
Propósito:	Su objetivo será realizar las operaciones necesarias para ejecutar la funcionalidad solicitada por el cliente.
Funciones:	<ol style="list-style-type: none"> 1. Realizar las operaciones para satisfacer la petición del cliente.
Requisitos:	RF-1, RF-2, RF-5, RF-6, RF-9, RNF-3, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.24: MOD-SERV-REST-3: Lógica de negocio

MOD-SERV-REST-4: Acceso a base de datos	
Propósito:	Tiene como finalidad proveer de un acceso a la base de datos transparente para las demás capas.
Funciones:	<ol style="list-style-type: none">1. Obtener información de la base de datos.2. Insertar datos en la base de datos.
Requisitos:	RF-1, RF-2, RF-5, RF-6, RF-9, RNF-3, RNF-5, RNF-6, RNF-9, RNF-12, RNF-13, RNF-14, RNF-15.

Tabla 3.2.25: MOD-SERV-REST-4: Acceso a base de datos

3.2.6. Diseño de las comunicaciones

Tal y como se expresa en la figura 3.2.1, el sistema está compuesto por diferentes aplicaciones, las cuales se encuentran separadas físicamente. Por tanto, como ya se ha apuntado anteriormente, será necesaria la implementación de una forma de comunicación que permita el entendimiento de las mismas. Esta comunicación se realizará sobre Internet, haciendo uso del protocolo HTTP [21].

Basándose en la arquitectura definida para el sistema, las aplicaciones scraper y cliente móvil tendrán la capacidad de comunicarse con el servidor pero no podrán establecer una comunicación entre ellas. Por tanto, el servidor se presenta como el centro de recepción y envío de mensajes procedentes de las anteriores. Es por ello que, puesto que se definen dos alternativas de diseño para el servidor, el sistema de comunicación será diferente para cada caso.

3.2.6.1. Servidor tipo servidor Web

En este caso, las comunicaciones serán diferentes según el sentido en el que se realicen debido a las limitaciones que presenta el servidor Web. A continuación se detallan las diferentes posibilidades:

3.2.6.1.1. Peticiones de scraper y cliente móvil al servidor

El envío de datos desde las aplicaciones cliente hacia el servidor se realizará mediante peticiones de tipo HTTP GET. De esta forma, las peticiones se dirigirán hacia el módulo encargado de la gestión de ésta e incluirán en la cabecera los valores necesarios para su gestión por medio de parámetros. Por ejemplo, las peticiones se realizarán enviando un mensaje HTTP GET a una url con aspecto similar a `http://direccióndelservidor/modulo?parametro1=X¶metro2=Y`. A continuación se detallan todos los posibles mensajes clasificados por el módulo encargado de recibir la petición. Para ello se utilizan tablas con el siguiente formato:

- Identificador comunicación: tiene el formato COM-WEB-X, donde X será un número entero que comenzará en 1 y se irá incrementando sucesivamente.
- Origen: especifica la aplicación que genera el mensaje.

- Destino: indica la aplicación a la que se envía el mensaje.
- Tipo: hace referencia al tipo de petición HTTP que se utiliza para el envío.
- Módulo: muestra el módulo encargado de la gestión de la petición.
- Propósito: define la intención de la petición.
- Formato: indica la forma de envío del mensaje. En caso de que se utilicen parámetros se incluyen qué parámetros se utilizan.
- URL: muestra la dirección de destino del mensaje. Ésta será orientativa, no siendo la dirección real a la que se realizarán las peticiones una vez implementado el servidor.

COM-WEB-1: Petición obtención categorías	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: obtener categorías
Propósito: obtener todas las categorías.	
Formato: sin parámetros.	
URL: http://direccionservidor/obtenerCategorias	

Tabla 3.2.26: COM-WEB-1: Petición obtención categorías.

COM-WEB-2: Petición obtención subcategorías	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: obtener subcategorías
Propósito: obtener las subcategorías que pertenecen a una categoría.	
Formato: parámetros.	
1. cat: identificador de la categoría padre de las subcategorías.	
URL: http://direccionservidor/obtenerSubcategoria?cat=X	

Tabla 3.2.27: COM-WEB-2: Petición obtención subcategorías.

COM-WEB-3: Petición obtención subsubcategorías	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: obtener subsubcategorías
Propósito: obtener las subsubcategorías que pertenecen a una subcategoría y categoría.	
Formato: parámetros.	
1. cat: identificador de la categoría padre de las subcategorías. 2. subcat: identificador padre de las subsubcategorías.	
URL: http://direccionservidor/obtenerSubsubcategoria?cat=X&subcat=Y	

Tabla 3.2.28: COM-WEB-3: Petición obtención subsubcategorías.

COM-WEB-4: Petición obtención productos categoría

Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: obtener producto
Propósito: obtener los productos de una categoría	
Formato: parámetros.	
1. cat: identificador de la categoría a la que pertenece el producto	
2. subcat: identificador de la subcategoría a la que pertenece el producto.	
3. subsubcat: identificador de la subsubcategoría a la que pertenece el producto	
URL:	
http://direccionservidor/obtenerProducto?cat=X&subcat=Y&subsubcat=Z	

Tabla 3.2.29: COM-WEB-4: Petición obtención productos categoría.

COM-WEB-5: Petición obtención producto por identificador

Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: obtener producto
Propósito: obtener un producto a partir de su identificador	
Formato: parámetros.	
1. id: identificador del producto.	
URL: http://direccionservidor/obtenerProducto?id=X	

Tabla 3.2.30: COM-WEB-5: Petición obtención producto por identificador.

COM-WEB-6: Petición obtención producto por nombre

Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: obtener producto
Propósito: obtener un producto a partir de un texto.	
Formato: parámetros.	
1. texto: texto a buscar para obtener el producto	
URL: http://direccionservidor/obtenerProducto?texto=X	

Tabla 3.2.31: COM-WEB-6: Petición obtención producto por nombre.

COM-WEB-7: Petición obtención producto por código de barras	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: obtener producto
Propósito: obtener un producto a partir de su código de barras.	
Formato: parámetros.	
1. codigo: número que indica el código de barras del producto en formato EAN-13.	
URL: http://direccionservidor/obtenerProducto?codigo=X	

Tabla 3.2.32: COM-WEB-7: Petición obtención producto por código de barras.

COM-WEB-8: Petición comparación de precios	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: comprobar
Propósito: obtener la comparación de precios de una lista de productos.	
Formato: parámetros.	
1. ids[]: array que contiene los identificadores de productos a comparar.	
URL: http://direccionservidor/comprobar?ids[]=%20X&ids[]=%20Y&ids[]=%20Z	

Tabla 3.2.33: COM-WEB-8: Petición comparación de precios.

COM-WEB-9: Petición guardado de categoría	
Origen: scraper	Destino: servidor
Tipo: GET	Módulo: guardar categoría
Propósito: almacenar una categoría en la base de datos.	
Formato: parámetros.	
1. nombre: nombre de la categoría	
2. posición: posición de la categoría en el menú de carritus.com	
URL: http://direccionservidor/guardarCategoria?nombre=%20X&posicion=%20Y	

Tabla 3.2.34: COM-WEB-9: Petición guardado de categoría.

COM-WEB-10: Petición guardado de subcategoría	
Origen: scraper	Destino: servidor
Tipo: GET	Módulo: guardar subcategoría
Propósito: almacenar una subcategoría en la base de datos.	
Formato: parámetros. 1. nombre: nombre de la subcategoría 2. posición: posición de la subcategoría en el menú de carritus.com 3. categoría: posición de la categoría a la que pertenece la subcategoría en el menú.	
URL: http://direccionservidor/guardarSubcategoria?nombre=X&posicion=Y&categoria=Z	

Tabla 3.2.35: COM-WEB-10: Petición guardado de subcategoría.

COM-WEB-11: Petición guardado de subsubcategoría	
Origen: scraper	Destino: servidor
Tipo: GET	Módulo: guardar subcategoría
Propósito: almacenar una subsubcategoría en la base de datos.	
Formato: parámetros. 1. nombre: nombre de la subsubcategoría 2. posición: posición de la subsubcategoría en el menú de carritus.com 3. categoría: posición de la categoría a la que pertenece la subsubcategoría en el menú. 4. subcategoría: posición de la subcategoría a la que pertenece la subsubcategoría en el menú.	
URL: http://direccionservidor/guardarSubsubcategoria?nombre=X&posicion=Y&categoria=Z&subsubcategoria=A	

Tabla 3.2.36: COM-WEB-11: Petición guardado de subsubcategoría.

3.2.6.1.2. Respuestas del servidor al scraper y cliente móvil

Las respuestas que genera el servidor a las peticiones que le llegan serán diferentes según vayan dirigidas al scraper o al cliente móvil. Para el primero, se devolverá una cadena de texto que contendrá un mensaje explicativo del resultado de la operación dado que los mensajes procedentes del scraper tienen como objetivo el almacenamiento de datos. De esta forma, el servidor informará de si el almacenado se ha producido correctamente o no.

Las peticiones que genera el cliente móvil tienen como objetivo la obtención de datos y, en consecuencia, las respuestas del servidor contendrán los datos solicitados. Para ello se hará uso del formato XML [70], definiéndose entidades que hacen referencia al elemento a obtener y cuyos elementos hijo representarán las propiedades del

elemento. A su vez, podrán anidarse entidades con atributos propios. Por ejemplo, a continuación se muestra el mensaje en formato XML que devuelve el servidor tras una petición de obtención de producto por identificador del cliente móvil:

```
<producto>
    <id>Identificador del producto</id>
    <nombre>Nombre del producto </nombre>
    <descripcion>Descripción del producto</descripcion>
    <formato>Formato del producto</formato>
    <imagen>URL de la imagen del producto</imagen>
    <mercadona>
        <precio>Precio en Mercadona</precio>
        <relativo>Precio relativo en Mercadona</relativo>
        <oferta>Descripción de la oferta del producto en Mercadona</oferta>
    </mercadona>
    <carrefour>
        <precio>Precio en Carrefour</precio>
        <relativo>Precio relativo en Carrefour</relativo>
        <oferta>Descripción de la oferta del producto en Carrefour</oferta>
    </carrefour>
    <hipercor>
        <precio>Precio en Hipercor</precio>
        <relativo>Precio relativo en Hipercor</relativo>
        <oferta>Descripción de la oferta del producto en Hipercor</oferta>
    </hipercor>
    <corteIngles>
        <precio>Precio en El Corte Inglés</precio>
        <relativo>Precio relativo en El Corte Inglés</relativo>
        <oferta>Descripción de la oferta del producto
            en El Corte Inglés</oferta>
    </corteIngles>
</producto>
```

Estos mensajes tendrán por tanto un aspecto similar en todos los casos pero serán personalizados en función del elemento que se desea obtener.

3.2.6.2. Servicio web RESTful

En el caso de implementar el servidor haciendo uso de un servicio web tipo RESTful las comunicaciones se simplifican en gran medida dado que el servidor acepta la recepción de mensajes en formato XML. Además, por las propiedades que ofrecen este tipo de servicios web, para la obtención y almacenamiento de datos se aceptan peticiones en URLs específicas para cada recurso siguiendo un patrón tal y como se verá reflejado en las tablas que detallan los mensajes enviados y recibidos más abajo.

De este modo, para realizar el almacenamiento de una entidad, basta con realizar una petición POST que contenga dicha entidad en formato XML, mientras que para obtener datos basta con enviar una petición GET a una URL específica que identifica únicamente el dato. Tanto los mensajes XML que devuelve el servidor los clientes como respuesta a sus peticiones como los que éstos le envían para el almacenamiento

de datos tienen un aspecto similar a los definidos anteriormente en la alternativa de uso de un servidor Web.

En consecuencia, el uso de esta alternativa de implementación para el servidor simplifica en gran medida el diseño de las comunicaciones entre las diferentes aplicaciones que conforman el sistema. Además, las URLs a las que se realizan las peticiones son más cortas y sencillas y no contienen atributos.

A continuación se van a detallar, de la misma forma que se hizo para la versión con servidor Web, los mensajes enviados hacia el servidor:

COM-REST-1: Petición obtención categorías	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener todas las categorías.	
URL: http://direccionservidor/categorias/	

Tabla 3.2.37: COM-REST-1: Petición obtención categorías.

COM-REST-2: Petición obtención subcategorías	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener las subcategorías que pertenecen a una categoría.	
URL: http://direccionservidor/subcategorias/idCategoria/	

Tabla 3.2.38: COM-REST-2: Petición obtención subcategorías.

COM-REST-3: Petición obtención subsubcategorías	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener las subsubcategorías que pertenecen a una subcategoría y categoría.	
URL: http://direccionservidor/subsubcategorias/idCategoria/idSubcategoria	

Tabla 3.2.39: COM-REST-3: Petición obtención subsubcategorías.

COM-REST-4: Petición obtención productos categoría	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener los productos de una categoría	
URL: http://direccionservidor/productos/idCategoria/idSubcategoria/idSubsub/	

Tabla 3.2.40: COM-REST-4: Petición obtención productos categoría.

COM-REST-5: Petición obtención producto por identificador	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener un producto a partir de su identificador	
URL: http://direccionservidor/productos/idProducto/	

Tabla 3.2.41: COM-REST-5: Petición obtención producto por identificador.

COM-REST-6: Petición obtención producto por nombre	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener un producto a partir de un texto.	
URL: http://direccionservidor/productos/query/textoABuscar/	

Tabla 3.2.42: COM-REST-6: Petición obtención producto por nombre.

COM-REST-7: Petición obtención producto por código de barras	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener un producto a partir de su código de barras.	
Formato: parámetros. 1. codigo: número que indica el código de barras del producto en formato EAN-13.	
URL: http://direccionservidor/productos/code/codigoBarrasProducto/	

Tabla 3.2.43: COM-REST-7: Petición obtención producto por código de barras.

COM-REST-8: Petición comparación de precios	
Origen: cliente móvil	Destino: servidor
Tipo: GET	Módulo: manejador peticiones
Propósito: obtener la comparación de precios de una lista de productos.	
URL: http://direccionservidor/productos/check/idsProductos/	

Tabla 3.2.44: COM-REST-8: Petición comparación de precios.

COM-REST-9: Petición guardado de categoría

Origen: scraper	Destino: servidor
Tipo: POST	Módulo: manejador peticiones
Propósito: almacenar una categoría en la base de datos.	
Formato: XML.	
URL: http://direccionservidor/categorias/	

Tabla 3.2.45: COM-REST-9: Petición guardado de categoría.

COM-REST-10: Petición guardado de subcategoría

Origen: scraper	Destino: servidor
Tipo: POST	Módulo: manejador peticiones
Propósito: almacenar una subcategoría en la base de datos.	
Formato: XML.	
URL: http://direccionservidor/subcategorias/	

Tabla 3.2.46: COM-REST-10: Petición guardado de subcategoría.

COM-REST-11: Petición guardado de subsubcategoría

Origen: scraper	Destino: servidor
Tipo: POST	Módulo: manejador peticiones
Propósito: almacenar una subsubcategoría en la base de datos.	
Formato: XML.	
URL: http://direccionservidor/subsubcategorias/	

Tabla 3.2.47: COM-REST-11: Petición guardado de subsubcategoría.

3.2.7. Matriz de trazabilidad de requisitos-módulos

Para finalizar la sección dedicada al diseño del sistema, seguidamente se muestra la tabla 3.2.48 que relaciona los bloques que componen las diferentes aplicaciones del sistema con los requisitos funcionales y no funcionales en los que intervienen.

Como puede comprobarse, todos los requisitos son cumplidos por alguno de los módulos de las aplicaciones del sistema, por tanto, puede asegurarse que se satisfacen dichas obligaciones.

	MOD-SCR-1	MOD-SCR-2	MOD-SCR-3	MOD-SCR-4	MOD-SCR-5	MOD-CLI-1	MOD-CLI-2	MOD-CLI-3	MOD-CLI-4	MOD-CLI-5	MOD-SERV-WEB-1	MOD-SERV-WEB-2	MOD-SERV-WEB-3	MOD-SERV-WEB-4	MOD-SERV-WEB-5	MOD-SERV-WEB-6	MOD-SERV-WEB-7	MOD-SERV-WEB-8	MOD-SERV-WEB-9	MOD-SERV-WEB-10	MOD-SERV-WEB-11	MOD-SERV-REST-1	MOD-SERV-REST-2	MOD-SERV-REST-3	MOD-SERV-REST-4			
RF-1	X	X	X								X	X										X	X	X	X			
RF-2	X	X	X								X	X										X	X	X	X			
RF-3	X																					X						
RF-4	X																											
RF-5						X	X	X	X		X	X		X	X	X	X						X	X	X	X		
RF-6						X	X	X	X		X	X		X	X	X	X						X	X	X	X		
RF-7						X	X	X		X																		
RF-8						X	X	X		X																		
RF-9						X	X	X			X	X	X											X	X	X	X	
RF-10						X	X	X		X																		
RF-11						X	X	X		X																		
RNF-1						X	X	X	X	X																		
RNF-2	X					X		X	X	X														X				
RNF-3											X		X	X	X	X	X	X	X	X	X	X	X	X	X	X		
RNF-4											X	X											X					
RNF-5																											X	
RNF-6																											X	
RNF-7	X	X	X	X	X	X																						
RNF-8	X			X		X	X																					
RNF-9											X																X	
RNF-10						X	X																					
RNF-11	X					X	X																					
RNF-12								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
RNF-13	X	X		X		X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
RNF-14	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
RNF-15										X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
RNF-16								X																				

Tabla 3.2.48: Matriz de trazabilidad de requisitos-módulos.

3.3. Implementación

En esta sección se van a detallar algunas de las decisiones más importantes que se han tomado para la implementación del sistema que ha sido diseñado anteriormente junto con las justificaciones que nos han llevado a la toma de las mismas. No obstante, no se va a entrar en detalle de cómo se ha realizado específicamente el código de las aplicaciones puesto que se dispone del mismo para su consulta y ésto supondría una sobrecarga adicional en la extensión de éste documento. Únicamente se mostrarán aquellas partes del código que se consideren de relevancia superlativa o que sean de un especial interés.

Por otro lado, para todas las aplicaciones que conforman el sistema a desarrollar se ha intentado elegir las herramientas y tecnologías más apropiadas, teniendo en cuenta siempre el factor económico. Es por ello que se han utilizado únicamente herramientas y tecnologías que supongan un coste cero para el desarrollo tratando, siempre y cuando sea posible, de que éstas además sean de código libre. Adicionalmente, en la medida de lo posible, se intentará hacer lo que coloquialmente en la comunidad informática se conoce como “no reinventar la rueda”, es decir, utilizar librerías ya programadas para realizar las funciones que se estimen oportunas y que permitan acelerar el desarrollo de las aplicaciones y obtener, en la mayoría de los casos, mejores resultados.

Por último, hay que decir que la máquina utilizada para el desarrollo e implementación del software es un portátil Macbook fabricado por Apple que dispone del sistema

operativo MacOS 10.6.8 (Leopard Snow).

3.3.1. Base de datos

En el mercado existen una gran cantidad de bases de datos que permiten realizar una implementación de forma sencilla y rápida. Tal y como se apuntó en las secciones de análisis y diseño, la base de datos se regirá siguiendo el modelo relacional, por tanto, ha de utilizarse una que se adapte a esta restricción. De esta forma, entre las múltiples opciones posibles se escoge MySQL[38] como plataforma base de datos para el sistema. Esta decisión viene fomentada por las siguientes razones:

- Se distribuye de forma gratuita mediante la filosofía OpenSource².
- Ofrece buenas prestaciones en cuanto a rendimiento y fiabilidad así como una alta disponibilidad.
- Es fácil de administrar y dispone de documentación abundante y de calidad.
- Disfruta de un reconocido prestigio.
- Se dispone de buenas herramientas externas como PhpMyAdmin[49] y MySQL Workbench[39], ambas gratuitas.
- Está disponible para sistemas operativos Unix/Linux, MacOS y Windows.

Las razones y características anteriores son muy similares a las que ofrecen otras implementaciones de bases de datos de código libre como pueden ser SQLite o PostgreSQL pero, en este caso se decide el uso de MySQL en detrimento de las anteriores por una razón fundamental: MySQL viene instalada y configurada por defecto en la máquina que se realiza el desarrollo. Además, también se dispone de la plataforma MAMP[35], la cuál dispone de una versión gratuita y permite una instalación y gestión muy sencilla de MySQL, servidor web Apache y PhpMyAdmin.

De esta forma, para la implementación de la base de datos, una vez que se tiene instalada y configurada MySQL utilizando MAMP, se procede a la creación del modelo relacional de ésta, el cuál fue mostrado más arriba, mediante la herramienta MySQL Workbench. Una vez elaborado el modelo, la propia herramienta genera las sentencias SQL necesarias para la implementación de la base de datos e incluso permite ejecutarlas en MySQL.

3.3.2. Programa scraper

3.3.2.1. Lenguaje de programación y tecnologías

Existen gran cantidad de librerías ya implementadas para diversos lenguajes que permiten el scraping de páginas web, es decir, el análisis de las mismas y la extracción de datos de éstas de una forma sencilla. No obstante, dada la complejidad de la web de la que se desean extraer los datos, no todas serán válidas. En concreto, las mayores limitaciones vienen establecidas debido la cantidad de scripts de tipo javascript[28] que incluye la web para la generación de contenido dinámicamente y la navegación y a que se requiere el mantenimiento de una sesión HTTP entre navegador y servidor, por lo

²Open Source: filosofía basada en la compartición de el código y las técnicas empleadas en el desarrollo de diferentes aplicaciones tanto software como hardware.

que es necesario realizar una gestión de cookies. De esta forma, www.carritus.com se presenta como una web compleja en la que el contenido es generado de forma dinámica mediante javascript. En este sentido, se realiza un profundo estudio de qué peticiones web se realizan al servidor para la obtención del contenido, de manera que se reproducen éstas en el programa scraper para poder realizar la navegación por las páginas y extraer la información de los productos.

Por tanto, se antoja necesario el uso de un lenguaje de programación que permita realizar una navegación en la web a base de clicks virtuales sobre elementos de la página, tal y como lo haría un usuario cualquiera con un navegador web, que sea capaz de interpretar el lenguaje javascript y que pueda de gestionar sesiones HTTP.

Por otro lado, es necesario que el lenguaje permita, de una forma sencilla, la confección y envío de mensajes HTTP y de documentos XML de forma que pueda establecerse la comunicación necesaria con el servidor.

Asimismo, la tecnología a elegir deberá de ofrecer las herramientas necesarias para la creación de una interfaz gráfica de usuario sobre la que mostrar información al usuario y que permita la interacción de éste con el software mediante el uso del ratón. Aunque esta aplicación será utilizada por un usuario administrador y no el usuario final objeto de la implementación del sistema, es conveniente que la interfaz gráfica ofrezca un aspecto profesional y adaptado al sistema operativo sobre el que se ejecuta y que pueda ser diseñada sin muchas dificultades.

Adicionalmente, se prefiere que se trate de un lenguaje orientado a objetos, lo que permitirá una representación más sencilla de la información a extraer y un mejor entendimiento del código.

Otro aspecto a tener en cuenta para la implementación del scraper es la capacidad que éste debe tener de crear archivos de texto que se almacenen en el disco de la máquina sobre la que se ejecuta para poder realizar un seguimiento del funcionamiento de la aplicación, es decir, lo que habitualmente se conoce como generación de ficheros de log.

Por último, otro de los factores que se presenta clave a la hora de la elección es que la aplicación debe poderse ejecutar en el mayor número de plataformas posible sin necesidad de reescribir o recompilar el código, suponiendo ésto una gran restricción para la toma de decisiones al respecto.

De esta forma, la tecnología escogida para la implementación de la aplicación scraper es el lenguaje **JRuby**[33]. Este lenguaje es una implementación del lenguaje de scripting Ruby[54] sobre la plataforma Java[26] y que reúne, en gran medida, las bondades y defectos de ambos. Por tanto, se trata de un lenguaje de scripting interpretado orientado a objetos que puede ejecutarse en múltiples sistemas operativos, destacando entre ellos los más habituales: Windows, Unix/Linux y MacOS. Además, se trata de un lenguaje sencillo y muy rápido en cuanto a programación se refiere, acelerando mucho el desarrollo de las aplicaciones en comparación con otros lenguajes orientados a objetos como el propio Java o C++. Otro punto a su favor supone el hecho de que permite realizar una mezcla de los lenguajes Ruby y Java, es decir, dentro de un programa JRuby puede hacerse uso de clases Java así como de clases Ruby.

Por otro lado, dejando a un lado el hecho de que JRuby reúne las condiciones exigidas como lenguaje de programación, la razón determinante para la elección de éste lenguaje por encima de otros más conocidos y habituales se debe a la existencia de una gran cantidad de librerías que facilitan la implementación y, en concreto, a una de ellas: **celerity**[15]. Ésta es un wrapper o adaptación de la librería HtmlUnit[20], la cuál implementa un navegador sin interfaz gráfica con capacidad de ejecución de Javascript

basado en Java, permitiendo la navegación sobre páginas web mediante clicks sobre elementos de la misma y la inclusión de texto en los campos de texto de la web entre otras funcionalidades propias de un navegador web convencional. En este sentido cabe pensar que también podría haberse elegido el lenguaje Java para realizar la implementación del scraper, puesto que, al fin y al cabo, la clave de la elección es una librería para este lenguaje. No obstante también han sido claves en la elección a las demás bondades de JRuby frente a Java como son las facilidades de que éste dispone para generar y gestionar conexiones HTTP, generar documentos XML, almacenado de logs y creación de interfaces de usuario de una forma más rápida y sencilla que en Java y con el mismo aspecto, además del hecho de que son necesarias muchas menos sentencias JRuby para elaborar un programa que tenga exactamente las mismas funcionalidades que uno hecho en Java, con el consiguiente ahorro que supone en cuanto a tiempo y dinero.

Para finalizar, es necesario poner de manifiesto la mayor limitación encontrada del lenguaje JRuby: la rapidez en la ejecución, la cual es más lenta que Java o Ruby y aún más lenta que otros lenguajes orientados a objetos como C++ o de scripting como Perl. Sin embargo, ésto no supone un gran problema puesto que la frecuencia de actualización de precios de los productos en la web www.carritus.com no es muy elevada, haciendo que no sea necesario un gran rendimiento del programa scraper.

3.3.2.2. Entorno de desarrollo y librerías utilizadas

Puesto que el lenguaje utilizado para la implementación de esta aplicación está basado en Java, es necesario disponer tanto de la máquina virtual Java (JVM) como del kit de desarrollo propio de Java, ambos instalados de fábrica en la máquina utilizada con versiones Java 6 y JRE 1.6.

Por otro lado, para la ejecución de programas JRuby también es necesario el uso de la máquina virtual propia de este lenguaje, la cual ha sido obtenida de manera gratuita de la web oficial, y cuya versión escogida ha sido la estable más reciente en el momento del inicio del desarrollo, la 1.6.3. En este sentido, dado que se trata de un lenguaje interpretado, no es necesario ningún software adicional para compilar los programas.

Como se apuntó anteriormente, uno de los hechos determinantes a la hora de la elección del lenguaje JRuby como base para la implementación del scraper es la gran cantidad de librerías que éste ofrece y que pueden ser instaladas mediante la terminal del sistema operativo mediante comandos. De esta forma, éstas son descargadas e instaladas automáticamente para su uso posterior. A continuación se van a listar todas las librerías que han sido utilizadas para la implementación del scraper junto con la versión de las mismas y la función para que éstas han sido utilizadas.

- **Celerity (0.8.9):** se trata de un navegador web sin interfaz gráfica que dispone de soporte para JavaScript. Es utilizado para realizar la navegación por la web www.carritus.com y la obtención de las webs que se generan dinámicamente.
- **Net/Http:** librería incluída en la distribución de JRuby que facilita la creación de comunicaciones HTTP. Utilizada para el envío y recepción de mensajes del servidor.
- **Nokigiri (1.5.0)[41]:** es un parser HTML, XML y SAX con capacidades de búsqueda mediante XPath y selectores CSS. Usado para la búsqueda y extracción

de los datos en las páginas HTML proveídas por Celerity así como para la creación de los documentos XML que se envían al servidor.

- **Logger:** librería incluída en la distribución de JRuby utilizada que permite la creación de mensajes de log utilizando diferentes niveles de importancia y que dispone de las funcionalidades necesarias para exportar dichos mensajes a ficheros de texto. Utilizada para la elaboración de los ficheros log.
- **Profligacy (1.0)[50]:** librería que facilita la creación de interfaces de usuario mediante la tecnología Swing de Java. En concreto, se ha utilizado debido al mecanismo de diseño de layouts que ofrece denominado LEL (Layout Expression Language) y que evita gran parte de los inconvenientes que supone el uso de Swing. Este mecanismo hace que la definición de layouts para interfaces gráficas de usuario sea sorprendentemente sencilla y veloz de programar, siendo éstos definidos mediante texto y estando basados en el uso de celdas para acomodar a los diferentes elementos.
- **Rufus-scheduler (2.0.10)[55]:** librería para JRuby que facilita la automatización de ejecución de código mediante sentencias con sintaxis similar a la utilizada por cron³. Ha sido utilizada para programar la funcionalidad de automatización de los scrapings.

Por último, para la elaboración de los ficheros de texto que componen el programa se ha utilizado el software TextMate[60]. Este software es un editor de textos para MacOs orientado a la programación que provee de diversas funcionalidades para facilitar la tarea al desarrollado como resaltado y autocompletado de código o el lanzamiento de la ejecución del programa en la terminal del sistema operativo.

3.3.2.3. Otros aspectos

Para el desarrollo de la aplicación scraper se decide dividir el programa en diferentes ficheros de texto y directorios que aglutinen partes del código que guarden relación, de forma que éste pueda ser comprendido de manera más sencilla y rápida además de para que quede más ordenado y organizado. El árbol de ficheros y directorios de la aplicación utilizado se muestra en la figura 3.3.1

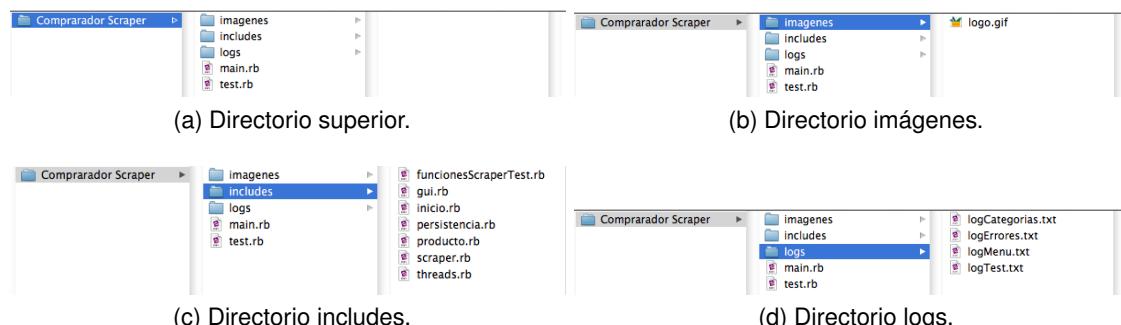


Figura 3.3.1: Estructura de directorios y ficheros del programa scraper.

³Cron: administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares en sistemas operativos basados en Unix.

La distribución realizada en directorios de los ficheros de texto .rb (propios de JRuby) es bastante simple e intuitiva, quedando a la vista en el nivel superior únicamente los archivos *main.rb* y *test.rb*, siendo el primero el que ha de ejecutarse para lanzar la aplicación y el segundo el dedicado a realizar el test para comprobar el correcto funcionamiento del código. En niveles inferiores encontramos los directorios *ímágenes*, que contiene las imágenes que se mostrarán en la aplicación; *includes*, en el que se encuentran los demás ficheros que son utilizados para la ejecución del programa; y *logs*, en el que se guardan los ficheros de log autogenerados por la aplicación y que serán diferentes según el módulo funcional que genera la información.

Por otro lado, hay que destacar que, puesto que se realizan dos implementaciones diferentes del servidor (servidor Web y servicio web RESTful) cuyas formas de comunicación con el scraper son distintas en ambos casos, el módulo dedicado para la persistencia (*includes/persistencia.rb*) contiene dos implementaciones distintas, una para cada servidor, permitiendo elegir el tipo de servidor a partir de la variable global *\$tipoServidor* que se configura al inicio de la ejecución del script *main.rb*.

Para finalizar con la sección dedicada a la implementación del scraper, seguidamente se muestra la parte del código dedicada a la elaboración del layout de la interfaz gráfica de la aplicación, que ha sido realizada utilizando la funcionalidad de diseño LEL proveída por la librería Profligacy y que pone de manifiesto la facilidad introducida para estas labores que ofrece ésta:

```
def crearLayoutPrincipal
    layout = "[controles] [consola] [estado]"
    layoutPrincipal = Swing::LEL.new(JPanel, layout) do |c,i|
        c.controles = crearLayoutControles
        c.controles.setBorder BorderFactory.createEtchedBorder
        c.consola = crearLayoutConsola
        c.consola.setBorder BorderFactory.createEtchedBorder
        c.estado = crearLayoutEstado
        c.estado.setBorder BorderFactory.createEtchedBorder
    end
    layoutPrincipal.build
end
```

Como puede verse en el cuadro anterior, la función *crearLayoutPrincipal* contiene en su interior la definición de las celdas principales que contienen los elementos de la interfaz gráfica. Éstas son *controles*, *consola* y *estado* y son definidas en una única línea. Posteriormente, se recorren las celdas anteriores, agregando los componentes que contendrán mediante las funciones *crearLayoutControles*, *crearLayoutConsola* y *crearLayoutEstado*, cuyas definiciones se realizan de forma similar a ésta. Seguidamente, se agregan los bordes a las celdas y, para finalizar se ordena que se muestre el layout principal mediante la llamada a la función *layoutPrincipal.build*.

Por tanto, en unas pocas líneas se ha definido un layout con las tres celdas celadas que se muestran sombreadas en la figura 3.3.2 y se ha creado la ventana de la aplicación, algo impensable si se utiliza Swing de la forma tradicional.

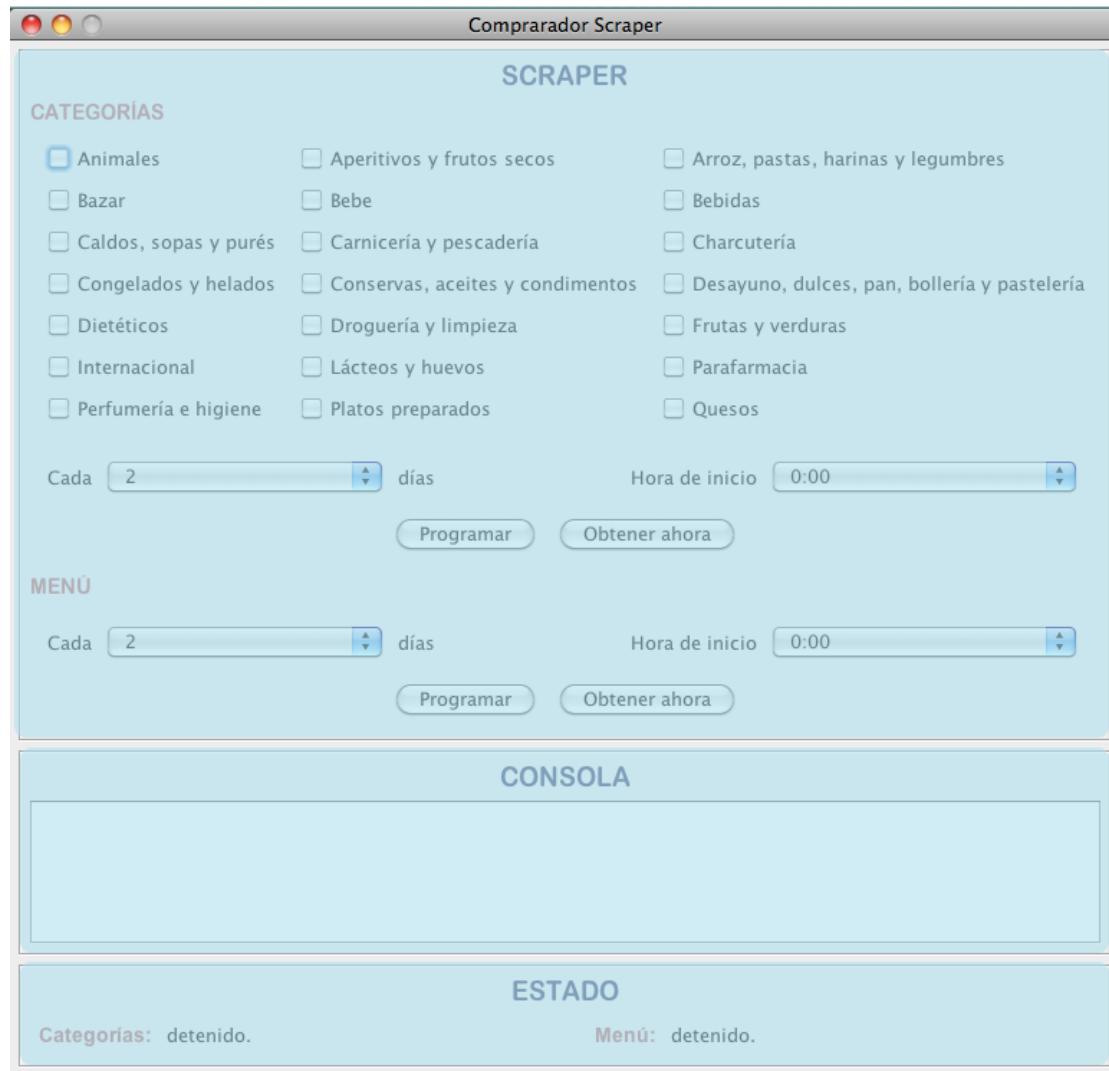


Figura 3.3.2: Scraper: ejemplo diseño interfaz de usuario mediante la librería Profligacy.

3.3.3. Programa cliente

3.3.3.1. Lenguaje de programación y tecnologías

El requisito no funcional RNF-1 establece que la aplicación cliente pueda ser ejecutada en diferentes plataformas móviles. Ésto representa el mayor reto para la implementación de esta aplicación ya que, cada sistema operativo móvil establece un lenguaje de programación diferente así como herramientas propias tanto para la implementación de los programas como para la simulación del funcionamiento y pruebas de los mismos. Con objeto de satisfacer dicho requisito se hace uso del framework OpenSource para desarrollo de aplicaciones móviles **PhoneGap(1.6.0)[47]**, que, tras el éxito cosechado, ha sido incluido dentro del proyecto Apache bajo el nombre de Cordova[16].

Este framework pretende ofrecer una solución a la heterogeneidad de las plataformas móviles, permitiendo el desarrollo de programas que puedan ser ejecutados en las plataformas móviles más actuales y con mayor número de usuarios, entre las que se encuentran Android, iOS, BlackBerry, Windows Phone, Bada y Symbian entre otros.

Para lograr dicho objetivo, basan la programación de las aplicaciones en los lenguajes estándar HTML, CSS y JavaScript, dedicados en principio al desarrollo de aplicaciones web. De esta forma, el framework proporciona un proyecto inicial específico para el entorno de desarrollo propio del sistema operativo sobre el que se desea que funcione la aplicación a desarrollar. Éste incluye la librería Phonegap, la cual contiene un navegador web basado en WebKit[67] de manera que al ejecutarse la aplicación, ésta muestra el contenido de una página web que, a su vez, se ejecuta de forma interna y transparente al usuario en dicho navegador. Adicionalmente, se provee de una API JavaScript que permite acceder de una manera unificada a las funciones y capacidades internas que ofrece el dispositivo móvil sobre el que se ejecuta la aplicación y el sistema operativo de éste como son la cámara, vibrador, micrófono, etc.

Por tanto, haciendo uso de este framework, la programación de una aplicación multiplataforma se limita a la implementación de una página web mediante HTML5, CSS y JavaScript que proporcione las funciones que se requieran. Una vez hecho ésto, basta con crear un proyecto con el entorno de desarrollo propio de cada uno de los sistemas operativos sobre los que se desea ejecutar la aplicación y, posteriormente, agregar la web y demás archivos necesarios para la implementación de ésta tal y como se indica en la documentación del framework. En consecuencia, finalmente se tendrán varios proyectos, uno por plataforma móvil, que contendrán una carpeta con nombre *www* en la que estará el código desarrollado. De esta forma, el desarrollo de estas aplicaciones es realmente rápido puesto que se elabora una vez únicamente el código y, por lo general, el desarrollo web se realiza de una forma más rápida que el nativo.

Por otro lado, tal y como se indicó anteriormente, es necesario que el cliente móvil disponga de una base de datos en la que almacenar las listas de productos creadas por el usuario y los productos marcados como favoritos. Para tal fin, se hace uso de la base de datos de tipo SQLite que incorpora el navegador WebKit integrado en PhoneGap, tal y como define el estándar HTML5. De esta forma, el acceso a la misma para la ejecución de sentencias se realiza mediante JavaScript de la forma especificada en el estándar.

Asimismo, es resaltable el hecho de que el framework PhoneGap puede aumentar sus funcionalidades a base de plugins que se agregan a éste y que han de ser desarrollados en el lenguaje nativo de cada uno de las plataformas móviles y que permiten acceder a nuevas capacidades no implementadas en el framework mediante funciones JavaScript. De esta forma, para satisfacer el requisito funcional RF-6, se hace uso del plugin oficial del framework *BarcodeScanner*. Éste está únicamente disponible para plataformas Android, y permite obtener el identificador asociado a un código de barras que es que detectado apuntando con la cámara del dispositivo hacia éste. Por tanto, gracias al uso de este plugin, la versión Android de la aplicación dispondrá de una funcionalidad extra que permitirá realizar búsquedas de productos a partir del escaneo de su código de barras.

Para finalizar, hay que apuntar que, como inconveniente principal de las tecnologías utilizadas para el diseño de la aplicación cliente móvil se encuentra el factor del rendimiento de la aplicación. Como es lógico, este tipo de aplicaciones presentan un rendimiento más bajo que las aplicaciones nativas y sus funcionalidades son más limitadas que las de estas últimas aunque, en este caso, el funcionamiento de la aplicación es correcto y suficiente para el objetivo que se persigue puesto que no se trata de una aplicación que requiera de una gran capacidad de procesamiento y velocidad.

3.3.3.2. Entorno de desarrollo y librerías utilizadas

Tal y como se apuntó más arriba, es necesario crear un proyecto diferente para cada una de las plataformas móviles sobre las que se desea ejecutar la aplicación. En este sentido, dado que se ha utilizado para el desarrollo una máquina con sistema operativo MacOS, únicamente se han creado proyectos para los sistemas operativos móviles Android, iOS y BlackBerry ya que son los únicos que ofrecen posibilidad de desarrollo en dicha máquina. En concreto, puesto que sólo se dispone de un dispositivo real que dispone de sistema operativo Android para realizar las pruebas, primeramente se realizó el desarrollo del programa destinado a Android y, posteriormente, se portó la web desarrollada a iOS y BlackBerry de una forma rápida y sin complicaciones resaltables.

Por tanto, los entornos de desarrollo utilizados son diferentes para cada una de las plataformas y en todos los casos se han utilizado aquellos recomendados de forma oficial. Éstos se detallan a continuación:

- **Android:** se ha hecho uso del entorno de desarrollo OpenSource Eclipse Indigo [19], junto con los plugins desarrollados por Google que permiten el manejo del SDK de Android: ADT (Android Developer Tools) v18. Además, para realizar la compilación del proyecto es necesario disponer del SDK de Android, en concreto, de la versión para Android 2.2 (Froyo) ya que ésta será la versión mínima sobre la que puede ejecutarse la aplicación.
- **iOS:** para el desarrollo de la aplicación para este sistema operativo se utiliza el entorno de desarrollo propio del fabricante, XCode 3.2.6 [69]. Éste se encuentra disponible para su descarga gratuita en la web oficial e incluye todas las herramientas para realizar el desarrollo. Puede ser ejecutado únicamente en sistemas operativos MacOS
- **BlackBerry:** para la versión PlayBook, que es para la que está disponible el framework PhoneGap, no existe un entorno de desarrollo integrado. Es por ello que, se utiliza TextMate para la edición de los ficheros de texto y Apache Ant 1.8.2 [7] para la compilación del proyecto. Los ficheros XML de configuración necesarios para la compilación se incluyen preconfigurados en el framework PhoneGap. Además, se utiliza la máquina virtual VMWare Fusion 3.0.0, la cual permite emular un dispositivo BlackBerry PlayBook mediante la imagen que puede descargarse de la web oficial.

Gracias al uso del framework PhoneGap, la resolución del problema se transforma en el desarrollo de una aplicación web basada en HTML5, CSS y JavaScript que haga uso de la API común que proporciona PhoneGap para el acceso a las funcionalidades nativas de los dispositivos móviles. Por tanto, se ha hecho uso de diferentes librerías para desarrollo web que faciliten la tarea que se detallan a continuación:

- **jQuery Mobile (1.0)[32]:** se trata de un framework destinado al desarrollo de aplicaciones web específicas para dispositivos móviles. Éste proporciona de una forma transparente al programador un aspecto y comportamiento para las páginas web similar al que tiene una aplicación nativa del sistema operativo iOS. Además, facilita la gestión de la memoria caché del navegador de forma que se pueda navegar hacia atrás para visualizar las pantallas mostradas anteriormente de igual manera que se realiza en las aplicaciones nativas. Asimismo, también integra transiciones y efectos CSS predefinidos que mejoran el aspecto visual de

la aplicación, acercándose e incluso mejorando en algunos casos las transiciones entre pantallas nativas de algunos sistemas operativos móviles. Por último, cabe destacar que, también proporciona las capacidades necesarias para que la aplicación sea accesible sin prácticamente esfuerzo dedicado a tal fin por parte del desarrollador.

- **jQuery (1.7.0)[31]**: librería JavaScript que proporciona múltiples funciones que facilitan el desarrollo de aplicaciones web y sobre la que se sustenta la anterior.
- **SimpleDialog (1.0)[59]**: plugin para jQuery Mobile que facilita el mostrado y personalización de ventanas emergentes, también conocidas como dialog.
- **RGraph (1.0)[52]**: librería basada en HTML5 y JavaScript que permite la creación y personalización de gráficas de una forma muy sencilla y con resultados visuales realmente sorprendentes. En concreto, para el desarrollo de la aplicación se ha utilizado únicamente el módulo que ofrece la posibilidad de creación de gráficos de barras, puesto que la librería dispone de una gran cantidad de tipos de gráfica posible para los que hay que agregar un módulo propio de cada uno.

3.3.3.3. Otros aspectos

De la misma forma que ocurre con el programa scraper, puesto que se desarrollan dos versiones diferentes del servidor, ambas con distintas formas de comunicación, se duplican los proyectos de la aplicación móvil, cambiando la forma en que se envían y reciben los datos del servidor.

Tal y como se ha afirmado con anterioridad, el desarrollo de la aplicación se reduce a la programación de una aplicación web, que deberá ser almacenada en la carpeta `www` de los diferentes proyectos generados. En la figura 3.3.3 se muestra la distribución de directorios seguida para la organización del proyecto así como los ficheros que lo componen.

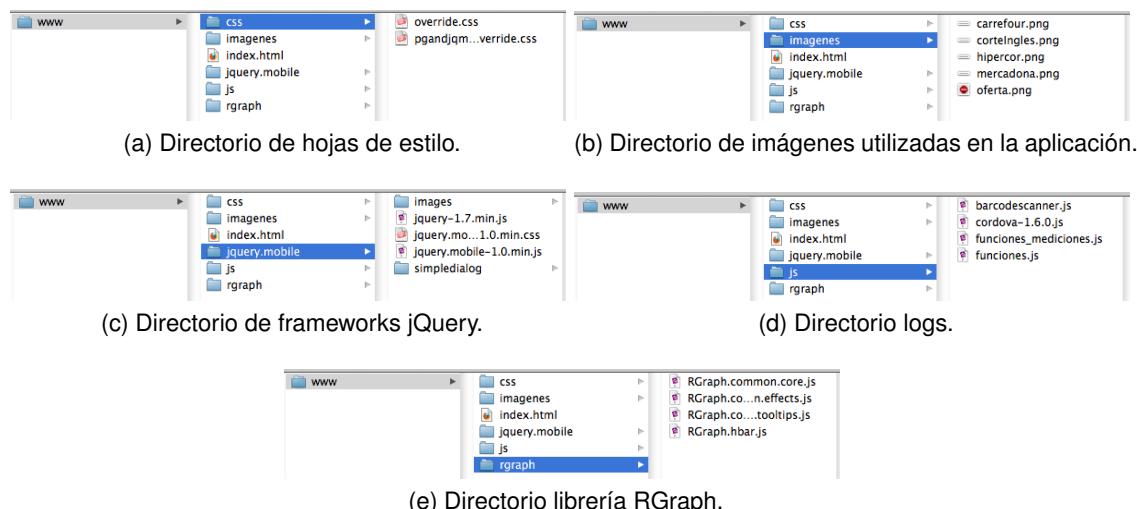


Figura 3.3.3: Estructura de directorios y ficheros del programa cliente.

A pesar de que el estilo de la aplicación viene dado por este mismo framework, en el directorio *css* se incluyen dos nuevas hojas de estilo. La hoja *pgandjqm-style-override.css* es incluida por PhoneGap y solventa pequeños problemas detectados en el uso conjunto de ambos frameworks. Por otro lado, la hoja de estilos *override.css* es de elaboración propia y tiene como objetivo modificar los estilos de jQuery Mobile para poder hacer uso de iconos propios y personalizar algunos otros detalles.

Como puede verse en la figura que muestra la estructura de directorios seguida, resulta curioso el hecho de que únicamente se dispone de un fichero HTML, el *index.html*, en el que se definen todas las pantallas que se van a mostrar. Posteriormente, el framework jQuery Mobile será el encargado de ejecutar la parte de dicho código correspondiente a la pantalla que se desea mostrar en cada caso. Valga como ejemplo de este comportamiento el código siguiente, mediante el que se define la página en la que se muestran las listas de productos creadas por el usuario y cuyo resultado se muestra en la figura 3.3.4.

```
<div data-role='page' id='paginaListas' data-theme='d'>
    <div data-role='header' data-position='fixed' data-theme='b'>
        <h1>Listas</h1>
        <a href='#' data-rel='back' data-icon='back'
            data-iconpos='notext'>Atras</a>
        <a href='#paginaPrincipal' data-icon='home'
            data-iconpos='notext'>Inicio</a>
    </div>
    <div data-role='content' id='contenidoPaginaListas'>
        <ol data-role='listview' data-inset='true'>
        </ol>
    </div>
    <div data-role='footer' data-position='fixed' data-theme='b'>
        <div data-role='navbar' data-iconpos='top' data-theme='b'>
            <ul>
                <li><a href='#paginaNuevaLista' data-rel='dialog'
                    data-icon='plus' data-iconpos='notext'>
                        Crear nueva lista
                    </a>
                </li>
            </ul>
        </div>
    </div>
</div>
```

Para la definición de las pantallas se hace uso de cuatro elementos div, los cuales tendrán un atributo llamado *data-role* que permitirá clasificarlos y cuyo valor será *page* para definir los límites del código propio de una pantalla, *header* para la cabecera o barra superior, *content* para la parte central de la pantalla en el que se muestra el contenido y *footer* para la parte inferior. Además, en esta página en concreto, el footer contiene otro div con *data-role navbar*, el cual permite definir una barra que contiene botones mediante listas, aunque en este caso se ha utilizado solo un botón.

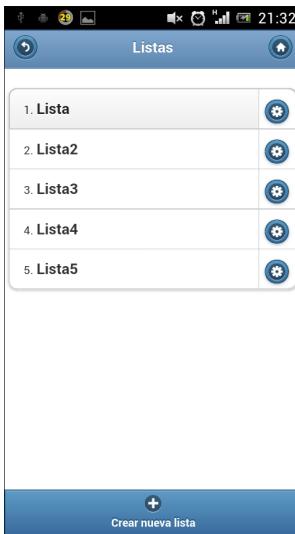


Figura 3.3.4: Cliente: ejemplo diseño de pantalla de la aplicación móvil.

Por tanto, el uso de jQueryMobile facilita mucho la implementación de las páginas pero es necesario tener un conocimiento de los tipos de atributos que se pueden usar en cada caso y que permiten que el framework muestre lo requerido con tan solo dar un valor a un atributo de una etiqueta HTML. No obstante, la documentación oficial es extensa y resulta sencillo encontrar ejemplos que facilitan enormemente la labor.

En este ejemplo, al igual que se ha hecho con casi todas las pantallas, la parte dedicada al contenido principal de la pantalla se define vacía de manera que éste es generado de forma dinámica mediante javascript.

Por último, en lo que se refiere al aspecto visual de la aplicación, los colores y formas de los elementos que se muestran son una combinación de los temas que se incluyen por defecto en jQueryMobile. Este framework dispone de 5 temas de colores que el programador puede aplicar a un elemento simplemente definiendo el atributo *data-theme*. No obstante, es posible crear nuevos estilos de una forma muy sencilla utilizando la herramienta web ThemeRoller [61], desarrollada por el equipo de jQuery-Mobile.

3.3.4. Programa servidor

Durante la fase de diseño se ofrecieron dos alternativas para la implementación del servidor. Ambas serán implementadas ya que, en principio, únicamente se contempló la opción de utilizar un servidor Web y, posteriormente, una vez desarrollada esta opción adquirí nuevos conocimientos que me permitieron conocer más a fondo la implementación de servicios web y decidí implementar la segunda opción para poder realizar una comparativa entre ambas. Es por ello que, los apartados que siguen serán subdivididos para mostrar las dos alternativas.

3.3.4.1. Lenguaje de programación y tecnologías

Servidor Web

En este caso se decide hacer uso del conocido servidor web OpenSource Apache[8] debido a que viene instalado de fábrica en la máquina utilizada para realizar los desarrollos. Además, tal y como ya se comentó más arriba, se hace uso de la herramienta

MAMP que facilita el uso y configuración de éste y de la base de datos MySQL y que incluye la versión 5.2.13 del lenguaje PHP[48]. Éste lenguaje será el utilizado para la gestión de las comunicaciones con los clientes a base de la creación de diferentes scripts que acceden a la base de datos y generan las respuestas para cada una de las peticiones. El servidor web Apache es el más utilizado y presenta grandes capacidades, ofreciendo un gran rendimiento en todos los casos. No obstante, el desarrollo del código en PHP se hace algo tedioso ya que es un lenguaje difícil de depurar. Además, la generación de los ficheros XML se hace “a mano”, sin la ayuda de librería alguna que facilite la tarea.

Servicio web RESTful

Esta alternativa es posible implementarla utilizando una gran cantidad de lenguajes de programación, siendo ésta una de las características principales de los servicios web y es que fueron creados con el objetivo de comunicar máquinas que utilizasen diferentes lenguajes de programación. En este caso, para la implementación se decide utilizar el lenguaje Java en su versión Enterprise Edition 6 (Java EE) [27] debido a que es un lenguaje ampliamente conocido por el desarrollador y que aporta grandes facilidades para el desarrollo de servicios web dado que incluye una gran cantidad de librerías que simplifican la tarea. De esta forma, a diferencia de la versión Java SE dedicada a aplicaciones de escritorio, la versión EE tiene como objetivo la programación de aplicaciones para la web, entre las que se incluyen los servicios web. Además, hace uso no solo de ficheros .java sino también de ficheros descriptores XML y anotaciones que facilitan enormemente la tarea. Para la ejecución de aplicaciones basadas en Java EE es necesario el uso de un servidor de aplicaciones, el cual ofrece todas las capacidades necesarias para ejecutar código Java y gestionar, de manera transparente para el programador, las conexiones HTTP con los clientes que realizan las peticiones.

El hecho de tener que utilizar un servidor de aplicaciones[56] para desplegar el servicio web puede parecer un inconveniente a priori pero realmente tiene ciertas ventajas sobre un servidor web para su uso de manera profesional.

Para poner en producción el sistema sería necesaria la contratación de un alojamiento o hosting en Internet que sirviera como soporte para el servidor ya que no se disponen de los medios suficientes para proveer de este servicio de forma propia. En este sentido, existen multitud de empresas dedicadas a estas tareas, las cuales suelen ofrecer buenos servicios a precios competitivos. Especialmente son destacables los servidores *compartidos* o *shared hosting*[57], que suelen ofrecer el uso de un servidor web, normalmente Apache, junto con bases de datos MySQL u otras tecnologías y PHP. Por tanto, éstos serían apropiados para la primera opción de implementación mostrada y supondrían un bajo desembolso económico para poner en funcionamiento el sistema. No obstante, esta opción del servidor compartido dispone de recursos limitados, por lo que para un uso más profesional debería de contratarse un servidor privado virtual (VPS)[66] u otro producto con características similares del mercado, los cuales tienen un coste superior y son, por lo general, más difíciles de gestionar y mantener.

Por el contrario, para el uso de servidores de aplicaciones en Internet también es necesario el uso de VPSs aunque existen alternativas excelentes y que eliminan las dificultades de gestión y mantenimiento como pueden ser Google App Engine (GAE)[9], Amazon Web Services[6] y OpenShift[44] de Red Hat. Las plataformas anteriores, conocidas como PaaS (Platform as a Service)[46] ofrecen servicios para el despliegue de aplicaciones Java EE usando la infraestructura de las grandes compañías que hay de-

trás de dichos servicios así como bases de datos para el almacenamiento de los datos, por lo que se asegura un rendimiento excelente en todos los casos. Mención especial merece la plataforma OpenShift, la cual además ofrece sus servicios de forma gratuita, sin límite de almacenamiento por lo que permitiría poner en funcionamiento el sistema de forma gratuita y ofreciendo un rendimiento excepcional. En el caso de Google App Engine y Amazon Web Services, el pago se realiza en función del uso del servicio que se hace, no estableciéndose un precio fijo.

Para finalizar, hay que hacer referencia al hecho de que el desarrollo del servidor mediante servicios web hace que el sistema sea más escalable, pudiendo ampliarse las funcionalidades ofrecidas de una forma más sencilla.

3.3.4.2. Entorno de desarrollo

Servidor Web

Para el desarrollo de los scripts PHP que se ejecutan en el servidor Web no es necesario el uso de ningún entorno de desarrollo particular. Únicamente habrá que escribir los scripts en ficheros de texto con extensión .php y situarlos en la carpeta de que el servidor dispone para ellos. Por tanto, en este caso, únicamente se ha utilizado Text-Mate para la elaboración de los scripts. Adicionalmente, para depurar y comprobar el correcto funcionamiento de los programas realizados se hace uso de un navegador web, en este caso Google Chrome. De esta manera, en la barra de direcciones se escribe la URL destinada para la obtención o almacenamiento de los datos, incluyendo los parámetros necesarios, y el propio navegador muestra el resultado obtenido.

Servicio web RESTful

En este caso, para el desarrollo del servicio web se utiliza NetBeans 7.0.1[40] como entorno de desarrollo. La elección de este IDE en lugar de otros con similares características y, para mi gusto mejor comportamiento, viene promovida por el hecho de que es el entorno de desarrollo recomendado de forma oficial e integra todas las herramientas necesarias para la creación de aplicaciones Java EE, entre las que se incluye el servidor de aplicaciones Glassfish v3.1.1 que es el que se ha utilizado para desplegar la aplicación. También que destacar que NetBeans ofrece de forma integrada las capacidades necesarias para el manejo del servidor de aplicaciones de una forma automatizada y sencilla. Además, permite la creación de servicios web de una forma muy simple, a través de un asistente, por lo que la tarea se simplifica en gran medida. Por otro lado, para la implementación del servicio web es necesario el uso de ciertas librerías, las cuales vienen integradas en NetBeans y que éste configura de forma automática al crear el proyecto. Éstas son las siguientes:

- **EclipseLink JPA 2.0[30]**: facilita la persistencia de los datos en la base de datos. Realiza un mapeo de la base de datos a objetos Java, lo que facilita tremenda- mente el acceso a ésta y la realización de consultas.
- **Jersey (JSR-311)[29]**: implementación oficial Java para el desarrollo de servicios RESTful. Provee de todo lo necesario para la creación del servicio, haciendo esta tarea transparente para el desarrollador. Entre la funcionalidades que incluye, destaca la conversión automática de objetos Java a documento XML y viceversa, algo que es realmente útil.

Por tanto, el uso de NetBeans y de las librerías anteriores facilita mucho la tarea de la creación del servicio web, quedando para el programador prácticamente la implementación del comportamiento del servicio ante la recepción de peticiones. Por otro lado, el uso de la tecnología Java EE, como se ha comentado anteriormente, maximiza la escalabilidad de la aplicación dado que es posible implementar una gran cantidad de funcionalidades en la capa dedicada a la lógica de negocio con un mínimo esfuerzo gracias a la impagable ayuda del IDE.

3.3.4.3. Otros aspectos

En esta sección se va a mostrar el código utilizado para atender a la petición de guardado de una categoría en la base de datos para las dos opciones de implementación. De esta forma, se podrá hacer una comparación entre ambas.

PHP (servidor Web)

```
<?php header('Content-Type: text/html; charset=UTF-8');
// se incluye el archivo que abre la conexión con la base de datos
include("includes/db_conectar.php");
//*********************************************************************
***** DEFINICIÓN DE FUNCIONES
//*********************************************************************
// verifica si existe la categoria en la base de datos
function buscarCategoria($nombre) {
    // confección de la consulta
    $sql = "SELECT * FROM categorias WHERE nombre='$nombre'";
    // se genera la consulta
    $result = mysql_query($sql);
    // se obtiene el resultado
    $row = mysql_fetch_array($result);
    // se liberan recursos
    mysql_free_result($result);
    return $row;
}
// actualiza los valores de la categoria
function actualizarCategoria($id,$posicion) {
    // confección de la consulta
    $sql = "UPDATE categorias SET posicion='$posicion' WHERE id='$id'";
    // se genera la consulta
    mysql_query($sql);
    echo "actualizada posicion";
}
// agrega una nueva categoria a la base de datos
function agregarCategoria($nombre,$posicion){
    // confección de la consulta
    $sql = "INSERT INTO categorias ('nombre', 'posicion') VALUES ('";
    $sql .= "'$nombre', '$posicion')";
    // se genera la consulta
    mysql_query($sql);
    echo "insertada";
}
```

```
(Continuación)
/********************* EJECUCIÓN
/******************
// Obtención de parámetros de URL
$nombre = $_GET['nombre'];
$posicion = $_GET['posicion'];
// se comprueba si la categoria existe o no
$busqueda = buscarCategoria($nombre);
if($busqueda){ // si la categoria existe
    if($posicion==$busqueda["posicion"]){
        // la categoria no ha cambiado
        echo "no modificada"
    }else{
        // la categoria ha cambiado de posición
        actualizarCategoria($busqueda["id"],$posicion);
    }
    // si la categoria no existe -> agregar categoria
    agregarCategoria($nombre,$posicion);
}
// se incluye el archivo que cierra la conexión con la base de datos
include("includes/db_desconectar.php"); ?>
```

Java (servicio web RESTful)

```
@POST
@Consumes({"application/xml"})
@Produces({"text/plain"})
public String crear(Categorias entity) {
    Categorias c = null;
    // Se busca verifica si el producto se encuentra en la base de datos
    try {
        c = (Categorias) getEntityManager().createQuery("SELECT c
            FROM Categorias c WHERE c.nombre = :nombre")
            .setParameter("nombre", entity.getNombre()).getSingleResult();
    } catch (Exception e) {
    }
    if (c != null && entity.getNombre().equals(c.getNombre())) {
        // LA CATEGORIA YA SE ENCUENTRA EN LA BASE DE DATOS
        if (c.getPosicion() != entity.getPosicion()) {
            // LA POSICIÓN ES DIFERENTE -> ACTUALIZAR
            c.setPosicion(entity.getPosicion());
            super.edit(c);
            return "actualizada posicion";
        }else{
            return "no modificada";
        }
    } else {
        // LA CATEGORIA NO SE ENCUENTRA EN LA BASE DE DATOS -> AGREGAR
        super.create(entity);
        return "insertada";
    }
}
```

Analizando los fragmentos de código anteriores puede verse que el código Java tiene una menor longitud y es mucho más sencillo. Esta simplicidad en el código se ve acentuada en aquellos casos en los que la petición procede del cliente móvil, mediante la que solicita que le sea devuelta un producto o una categoría, puesto que interviene la necesidad de la generación de un documento XML como respuesta. En el caso de Java, dado que se utiliza la librería Jersey que implementa la especificación JAX-RS, es suficiente con que la función encargada de proveer el servicio devuelva un objeto Java, encargándose dicha librería de la generación del documento XML a partir del objeto devuelto. Ésto puede verse en el siguiente fragmento de código, el cual tiene por objetivo devolver un producto a partir de su identificador.

PHP (servidor Web)

```
<?php header('Content-Type: text/xml; charset=UTF-8');
// se incluye el archivo que abre la conexión con la base de datos
include("includes/db_conectar.php");
$id = $_GET['id'];
$sql = "SELECT * FROM 'productos' WHERE id='$id'";
$consulta = mysql_query($sql);

// se crea el XML
$xml = '<?xml version="1.0"?>';
$xml .= '<producto>';
while($fila = mysql_fetch_array($consulta)){
    $xml .= '<marca>' . $fila['marca'] . '</marca>';
    $xml .= '<descripcion>' . $fila['descripcion'] . '</descripcion>';
    $xml .= '<formato>' . $fila['formato'] . '</formato>';
    $xml .= '<imagen>' . $fila['imagen_src'] . '</imagen>';
    $xml .= '<categoria>' . $fila['categoria'] . '</categoria>';
    $xml .= '<subcategoria>' . $fila['subcategoria'] . '</subcategoria>';
    $xml .= '<subsubcategoria>' . $fila['subsubcategoria']
        . '</subsubcategoria>';
    $xml .= '<creado>' . $fila['creado'] . '</creado>';
    $xml .= '<mercadona precio="'. $fila['precio_mercadona'] . '"'
        'relativo="'. $fila['precio_relativo_mercadona'] . '"'
        'oferta="'. $fila['oferta_mercadona'] . '" />';
    $xml .= '<carrefour precio="'. $fila['precio_carrefour'] . '"'
        'relativo="'. $fila['precio_relativo_carrefour'] . '"'
        'oferta="'. $fila['oferta_carrefour'] . '" />';
    $xml .= '<hipercor precio="'. $fila['precio_hipercor'] . '"'
        'relativo="'. $fila['precio_relativo_hipercor'] . '"'
        'oferta="'. $fila['oferta_hipercor'] . '" />';
    $xml .= '<corteIngles precio="'. $fila['precio_corteIngles'] . '"'
        'relativo="'. $fila['precio_relativo_corteIngles'] . '"'
        'oferta="'. $fila['oferta_corteIngles'] . '" />';
}
mysql_free_result($consulta);
$xml .= '</producto>';
// se genera el nuevo menú en formato XML echo $xml;
// se incluye el archivo que cierra la conexión con la base de datos
include("includes/db_desconectar.php"); ?>
```

Java (servicio web RESTful)

```
@GET  
@Path("{id}")  
@Produces({"application/xml"})  
public Productos find(@PathParam("id") Long id) {  
    Productos p=null;  
    try{  
        p = (Productos) getEntityManager().createQuery("SELECT p  
                FROM Productos p WHERE p.id = :id").setParameter("id", id)  
                .getSingleResult();  
    }catch (Exception e){  
    }  
    return p;  
}
```

De nuevo, el código Java es más fácil de entender, limpio y corto. Sin embargo, hay que hacer notar que en favor de la implementación mediante PHP juega el hecho de que el código mostrado es el código completo mientras que en el caso de Java únicamente se muestra el código que se ejecuta al recibir la petición del cliente. No obstante, en la implementación Java, el resto del código es autogenerado por NetBeans como es el caso, por ejemplo, de las clases que representan las entidades de la base de datos.

Capítulo 4

Pruebas y resultados

Este apartado va a estar dedicado a la realización de pruebas para confirmar el correcto funcionamiento de las diferentes aplicaciones. También se hacen ciertas mediciones de tiempos en la aplicación cliente para poder evaluar el rendimiento de ésta de forma que pueda corroborarse que la experiencia del usuario es todo lo satisfactoria que debiera.

4.1. Entorno de pruebas

Las pruebas de las diferentes aplicaciones se van a hacer utilizando los distintos dispositivos y máquinas de que se dispone, siendo éstas propiedad del desarrollador y las utilizadas para llevar a cabo el desarrollo del proyecto. Además, hay que puntualizar que, a pesar de que en un entorno real de producción tanto el servidor como la base de datos deberían estar accesibles a través de Internet, en este caso, para la realización de las pruebas éstos estarán conectados a los dispositivos mediante red local wireless.

A continuación se describen los diferentes elementos utilizados para la elaboración de las pruebas:

- **Servidor:** la aplicación servidora será ejecutada, en ambas versiones de implementación, en un ordenador portátil tipo MacBook, el cuál dispone de un procesador Core 2 Duo a 2 GHz y 4 GB de memoria RAM 667 MHz. En cuanto a lo que software se refiere, se dispone de los programas comentados en la sección dedicada al desarrollo.
- **Scraper:** este programa es ejecutado en la misma máquina que el servidor, por lo que el envío de mensajes al servidor se realiza de forma local, sin necesidad de comunicación mediante red.
- **Cliente móvil:** las pruebas de la aplicación móvil se realizan en dos dispositivos distintos, ambos con sistema operativo Android pero de diferente gama y, por tanto, distinta capacidad de procesamiento y características. Los teléfonos de que se dispone son los siguientes:
 - HTC Wildfire (A3333): teléfono de gama baja con procesador Qualcomm MSM7225 a 528 MHz, 384 MB de RAM y sistema operativo Android 2.2 (Froyo).

- Sony Ericsson Xperia Neo (MT15): teléfono de gama media con procesador Qualcomm Snapdragon MSM8255 a 1000 MHz, 512 MB de RAM y sistema operativo Android 4.0.4 (Ice Cream Sandwich).
- **Red:** el entorno de red utilizado para las comunicaciones entre aplicaciones está gestionado por un router ADSL LiveBox de Orange con capacidad de conexión a Internet de hasta 20 Mbps. Además, los dispositivos se conectan a éste mediante red local wireless siguiendo el protocolo 802.11 g que provee una velocidad de transferencia de 54 Mbps.

4.2. Pruebas de funcionamiento de las aplicaciones

Se realizan pruebas para el cliente móvil y para la aplicación scraper, de forma que se comprueba a su vez la correcta comunicación con el servidor así como la generación de las respuestas apropiadas de éste en base a las peticiones realizadas. En los apartados que siguen se detalla de forma más pormenorizada las pruebas realizadas en cada caso.

4.2.1. Pruebas en la aplicación cliente móvil

Las pruebas en la aplicación cliente se realizan a partir de la creación de un nuevo proyecto que utiliza las mismas librerías y entornos que en el desarrollo. Así, los tests a realizar son lanzados desde la nueva aplicación construida y están basados en el código utilizado para el cliente móvil. Por un lado, se comprueba el correcto funcionamiento de la API JavaScript que proporciona el framework PhoneGap, de manera que quede asegurado que el comportamiento de éste es el esperado. Adicionalmente, se realizan diferentes tests del código desarrollado para la aplicación y las comunicaciones con el servidor mediante el uso del framework para testeo JavaScript QUnit [51], el cual es el recomendado de forma oficial para realizar dicha labor por jQuery.

En la figura 4.2.1 se muestra una captura de pantalla de la aplicación desarrollada para lanzar los tests en la aplicación móvil. Desde dicha pantalla principal se tiene acceso, mediante los botones destinados a tal efecto que se encuentran en la parte inferior, a la ejecución de los diferentes tests programados. Adicionalmente, en la parte superior de la aplicación se muestran detalles del modelo de teléfono utilizado para la ejecución de los tests.

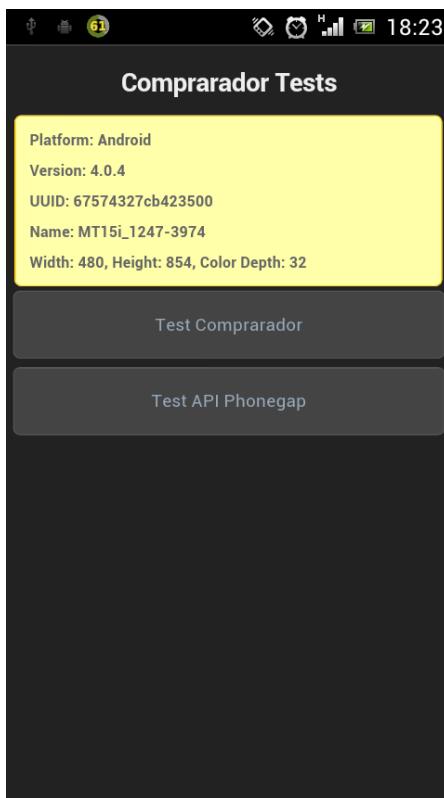


Figura 4.2.1: Vista pantalla principal test aplicación móvil.

Testeo del framework PhoneGap:

En este caso se realizan diferentes pruebas que vienen a corroborar que el funcionamiento de la API que provee el framework sobre el que se basa el desarrollo de la aplicación es el que se espera para las funciones que han sido utilizadas en el desarrollo de la aplicación. Las funciones de dicha API que han sido testadas han ofrecido todas un comportamiento correcto y son las siguientes:

- Obtención de características del dispositivo: nombre del modelo, número de serie y versión del sistema operativo.
- Acelerómetro.
- Acceso a red.
- Notificaciones mediante vibración y audio.
- Acceso a base de datos.

En la figura 4.2.2 se muestran las capturas realizadas como resultado de la ejecución de los tests 22 dedicados a framework PhoneGap. Adicionalmente, tal y como puede verse en la subfigura 4.2.2.b, dichos tests se componen a su vez de diferentes verificaciones, las cuales vienen indicadas entre paréntesis para cada uno de los test, haciendo un total de 53 comprobaciones, todas con resultado satisfactorio.

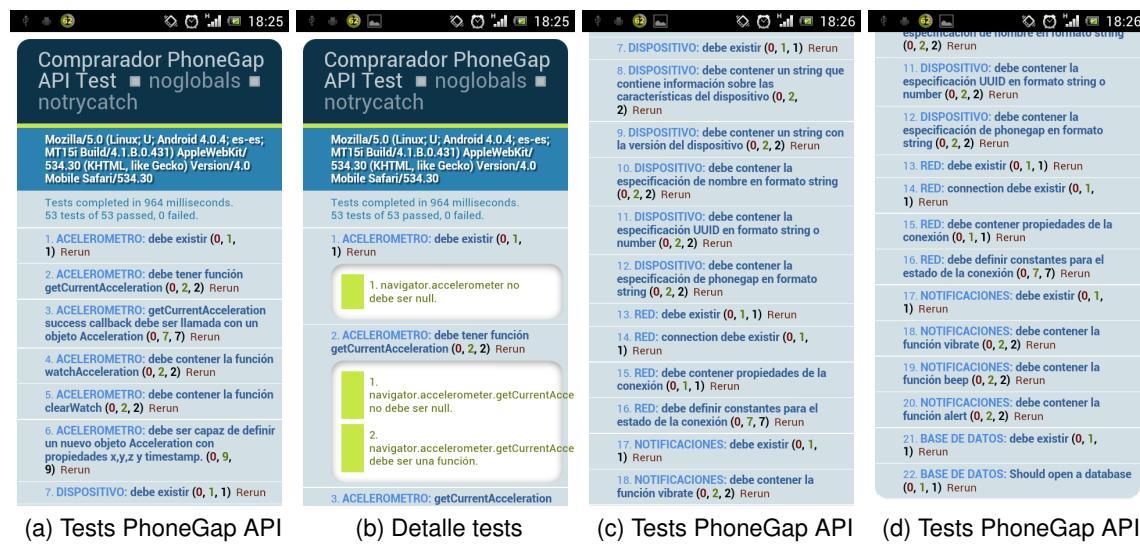


Figura 4.2.2: Resultados tests cliente móvil: PhoneGap API.

Testeo de la aplicación móvil:

Se programan una serie de tests para probar el correcto funcionamiento de la aplicación cliente móvil. Así, las pruebas realizadas tienen como objetivo verificar que las respuestas del servidor a las peticiones realizadas son las esperadas, pudiendo asegurar de esta forma que el servidor no produce ningún fallo. Los tests dedicados a la aplicación móvil tienen como objetivo la comprobación de las siguientes funcionalidades tal y como se puede observar en la figura 4.2.3.

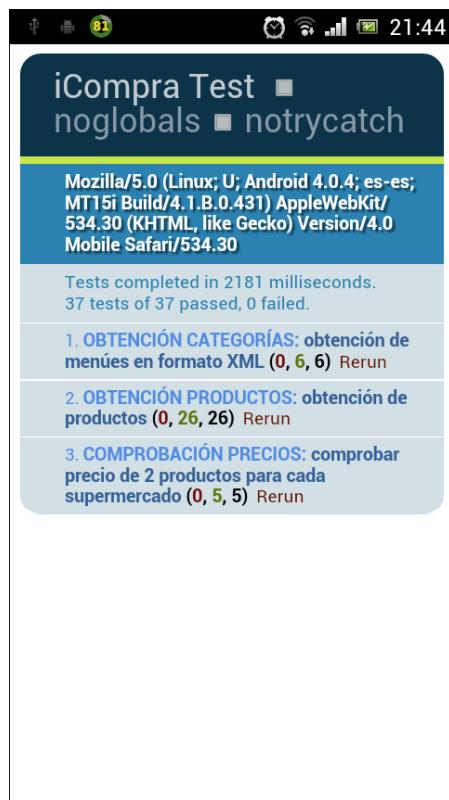


Figura 4.2.3: Resultados tests cliente móvil: comprador.

- Obtención de categorías: contiene 6 tests dedicados a la verificación de la correcta obtención de las categorías en formato XML.

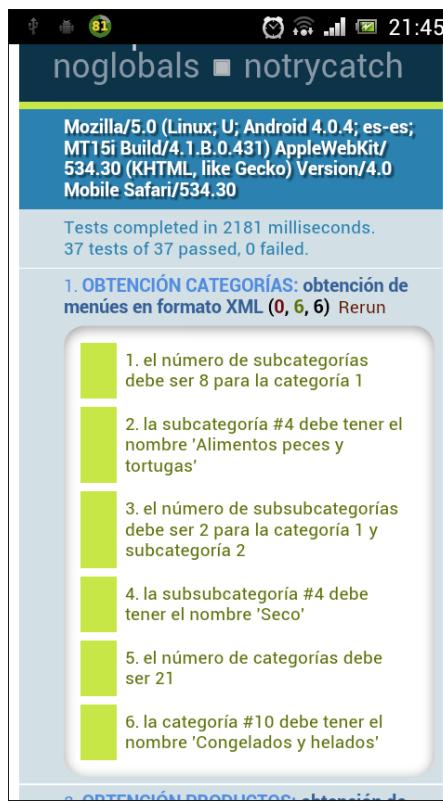


Figura 4.2.4: Resultados tests cliente móvil: comprador (1).

- Obtención de productos: consta de 26 pruebas cuyo objetivo es comprobar que se obtienen de forma correcta la información de productos del servidor a partir de diferentes métodos.

2. OBTENCIÓN PRODUCTOS: obtención de productos (0, 26, 26) Rerun	3. COMPROBACIÓN PRECIOS: comprobar precio de 2 productos para cada supermercado (0, 5, 5) Rerun
<p>1. el producto con código de barras 4902505154300 debe tener descripción 'Cerveza Abadía Carrefour - Lata 33 Cl'</p> <p>2. el producto con código de barras 4902505154300 debe tener marca 'Carlsberg'</p> <p>3. el producto con código de barras 4902505154300 debe tener formato '1x30ml'</p> <p>4. el producto con código de barras 4902505154300 debe id=114</p> <p>5. el producto con código de barras 4902505154300 debe pertenecer a la categoría 6</p> <p>6. el producto con código de barras 4902505154300 debe pertenecer a la subcategoría 5</p> <p>7. el producto con código de barras 4902505154300 debe pertenecer a la subsubcategoría 3</p> <p>8. el producto con id=881 debe tener descripción 'Cerveza Abadía Carrefour - 33 Cl'</p> <p>9. el producto con id=881 debe tener marca 'Carrefour'</p> <p>10. el producto con id=881 debe tener formato '1x30ml'</p> <p>11. el producto con id=881 debe pertenecer a la categoría 6</p> <p>12. el producto con id=881 debe pertenecer a la subcategoría 5</p> <p>13. el producto con id=881 debe pertenecer a la subsubcategoría 7</p> <p>14. el número de productos obtenidos para la búsqueda 'Agua Sabor' debe ser 3</p> <p>15. el producto #2 para la búsqueda 'Agua Sabor' debe tener descripción 'Agua Sabor Mandarina Font Vella - Botella De 1,25 Cl.'</p> <p>16. el producto #2 para la búsqueda 'Agua Sabor' debe tener marca 'Font Vella'</p> <p>17. el producto #2 para la búsqueda 'Agua Sabor' debe tener formato '1x1250ml'</p> <p>18. el producto #2 para la búsqueda 'Agua Sabor' debe id=28</p> <p>19. el producto #2 para la búsqueda 'Agua Sabor' debe pertenecer a la categoría 6</p> <p>20. el producto #2 para la búsqueda 'Agua Sabor' debe pertenecer a la subcategoría 1</p> <p>21. el producto #2 para la búsqueda 'Agua Sabor' debe pertenecer a la subsubcategoría 2</p> <p>22. el número de productos debe ser 30 para la categoría 6, subcategoría 2, subsubcategoría 1</p> <p>23. el producto #10 debe tener descripción 'El Cantero De Letur Batido De Cacao Procedente De Ganadería Ecológica Botella 1 L'</p> <p>24. el producto #10 debe tener marca 'El Cantero De Letur'</p> <p>25. el producto #10 debe tener formato '1x1000ml'</p> <p>26. el producto #10 debe tener id '123'</p>	<p>17. el producto #2 para la búsqueda 'Agua Sabor' debe tener formato '1x1250ml'</p> <p>18. el producto #2 para la búsqueda 'Agua Sabor' debe id=28</p> <p>19. el producto #2 para la búsqueda 'Agua Sabor' debe pertenecer a la subcategoría 6</p> <p>20. el producto #2 para la búsqueda 'Agua Sabor' debe pertenecer a la subsubcategoría 1</p> <p>21. el producto #2 para la búsqueda 'Agua Sabor' debe pertenecer a la subsubcategoría 2</p> <p>22. el número de productos debe ser 30 para la categoría 6, subcategoría 2, subsubcategoría 1</p> <p>23. el producto #10 debe tener descripción 'El Cantero De Letur Batido De Cacao Procedente De Ganadería Ecológica Botella 1 L'</p> <p>24. el producto #10 debe tener marca 'El Cantero De Letur'</p> <p>25. el producto #10 debe tener formato '1x1000ml'</p> <p>26. el producto #10 debe tener id '123'</p>

Figura 4.2.5: Resultados tests cliente móvil: comprador (2).

- Comprobación de precios: incluye 5 tests que verifican que la comparación de precios de productos se realiza con el resultado esperado.

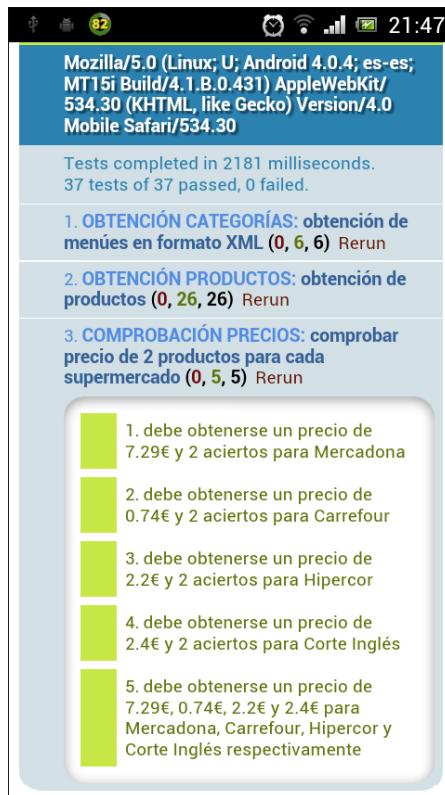


Figura 4.2.6: Resultados tests cliente móvil: comprador (3).

Hay que aclarar que la realización de los tests anteriores está basada en la realización de peticiones seleccionadas de forma aleatoria al servidor y cuya respuesta correcta se conoce a priori.

4.2.2. Pruebas en la aplicación scraper

Para el testeo del funcionamiento de la aplicación scraper se realiza un nuevo programa basado en el código utilizado para el scraper. De esta forma, el nuevo programa, haciendo uso de las funciones implementadas para el scraper, realiza un test interactivo de las funcionalidades principales como son el logueo en www.carritus.com, navegación a través del menú de la página, obtención de productos, etc.

Así, tras lanzar el test, el programa solicita al usuario que realice una serie de acciones en su navegador y que introduzca el resultado obtenido. Tras la realización de los diferentes tests, se muestra un resumen del funcionamiento del programa según se muestra en las figuras 4.2.7, 4.2.8 y 4.2.6:

Proyecto Comprador

```
Empollicass-MacBook:Comprador_Scraper albertomateos$ jruby test.rb
*****
Test: 1. INICIALIZACION DEL SCRAPER
*****  
  
ORDEN:  
Navegue hasta www.carritus.com/carrito, elija Mercadona como supermercado habitual y seleccione 18004 como código postal.  
  
*****  
*** TEST: 2. OBTENCION DE CATEGORIAS  
*****  
Animales  
Aperitivos y frutos secos  
Arroz, pastas, harinas y legumbres  
Bazar  
Bebe  
Bebidas  
Caldos, sopas y purés  
Carnicería y pescadería  
Charcutería  
Congelados y helados  
Conservas, aceites y condimentos  
Desayuno, dulces, pan, bollería y pastelería  
Dietéticos  
Droguería y limpieza  
Frutas y verduras  
Internacional  
Lácteos y huevos  
Parafarmacia  
Perfumería e higiene  
Platos preparados  
Quesos  
  
ACCION: Introduzca 1 (OK) o 0 (Falló) si las categorías mostradas son las mismas que las del menú de Carritus  
1
```

Figura 4.2.7: Test scraper: captura ejecución test (1).

```
*****
*** TEST: 3. PAGINA DE PRODUCTOS
*****  
  
ORDEN:  
Navegue a la categoría BEBIDAS - REFRESOS - COLA  
  
PRODUCTO OBTENIDO:  
---- Descripción: Cola Normal, Pepsi, Lata 330 Cc  
---- Imagen: http://static.carritus.com/images/images_pms/38/39938.jpg  
---- Precio Mercadona: 0.43 (1.30€ / l.)  
---- Precio Carrefour: 0.00 ()  
---- Precio Hipercor: 0.00 ()  
---- Precio Corte Inglés: 0.00 ()  
  
ACCION: Introduzca 1 (OK) o 0 (Falló) si el producto mostrado es el primero de la categoría para MERCADONA  
1  
*****  
*** TEST: 4. CAMBIO DE SUPERMERCADO
*****  
  
ORDEN:  
Cambio de supermercado a CARREFOUR  
  
PRODUCTO OBTENIDO:  
---- Descripción: Refresco Cola Light Carrefour - 33 Cl.  
---- Imagen: http://static.carritus.com/images/images_pms/08/15279968.gif  
---- Precio Mercadona: 0.00 ()  
---- Precio Carrefour: 0.21 (0.64€ / l.)  
---- Precio Hipercor: 0.00 ()  
---- Precio Corte Inglés: 0.00 ()  
  
ACCION: Introduzca 1 (OK) o 0 (Falló) si el producto mostrado es el PRIMERO de la categoría para CARREFOUR  
1
```

Figura 4.2.8: Test scraper: captura ejecución test (2).

```
*****
*** TEST: 5. PÁGINA SIGUIENTE
*****  

ORDEN:  

Navegue a la página 2  

PRODUCTO OBTENIDO:  

---- Descripción: Refresco De Cola Regular Coca-cola - Botella De 2,20l  

---- Imagen: http://static.carritus.com/images/images_pms/08/15251608.gif  

---- Precio Mercadona: 0.00 ()  

---- Precio Carrefour: 6.90 (0.57 - 3,14€ / l.)  

---- Precio Hipercor: 0.00 ()  

---- Precio Corte Inglés: 0.00 ()  

ACCIÓN: Introduzca 1 (OK) o 0 (Falló) si el producto mostrado es el PRIMERO de la categoría para CARREFOUR y la PÁGINA 2  

1  

*****
*** TEST: 6. INDEXADO DE PRODUCTOS
*****  

PRODUCTO OBTENIDO:  

---- Descripción: Bebida Refrescante De Cola Light Coca-cola - Pack De 6 Botellas De 20 Cl.  

---- Imagen: http://static.carritus.com/images/images_pms/58/15231468.jpg  

---- Precio Mercadona: 0.00 ()  

---- Precio Carrefour: 3.72 ($1.10€ / l.)  

---- Precio Hipercor: 0.00 ()  

---- Precio Corte Inglés: 0.00 ()  

ACCIÓN: Introduzca 1 (OK) o 0 (Falló) si el producto mostrado es el TERCERO de la categoría para CARREFOUR y la PÁGINA 2  

1  

*****
*** RESULTADOS DEL TEST ***
*****  

1. INICIALIZACIÓN DEL SCRAPER: OK  

2. OBTENCIÓN DE CATEGORIAS: OK  

3. PÁGINA DE PRODUCTOS: OK  

4. CAMBIO DE SUPERMERCADO: OK  

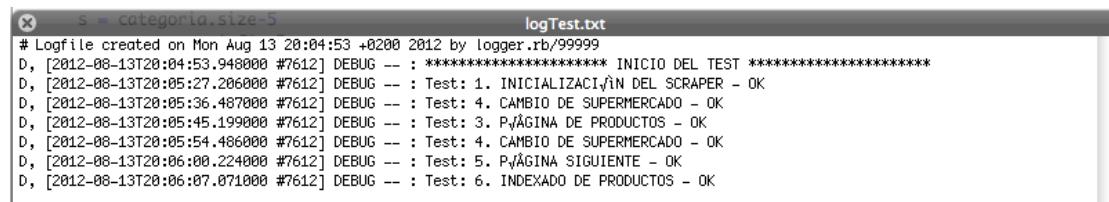
5. PÁGINA SIGUIENTE: OK  

6. INDEXADO DE PRODUCTOS: OK  

Empolliticas-MacBook:Comprador_Scraper albertomateos$
```

Figura 4.2.9: Test scraper: captura ejecución test (3).

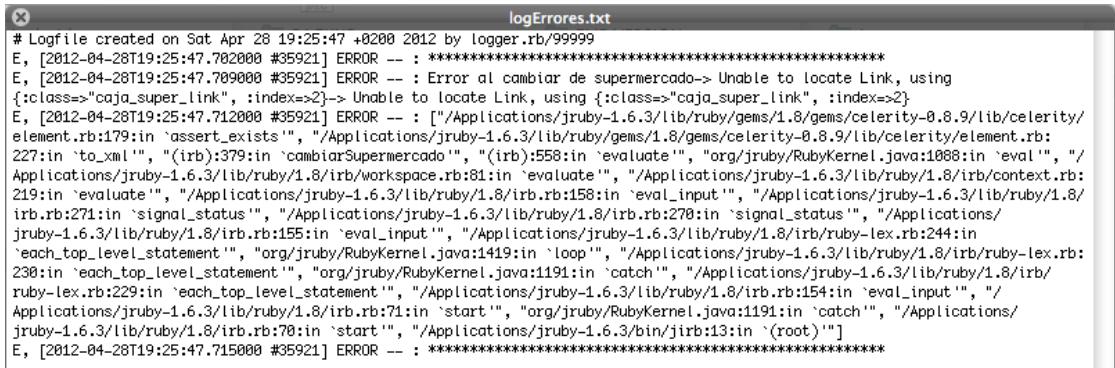
Asimismo, al finalizar la ejecución del test del scraper se crea un fichero de log que almacena el resultado del test tal y como se muestra en la figura 4.2.10:



```
s = categoria.size-5                                     logTest.txt
# Logfile created on Mon Aug 13 20:04:53 +0200 2012 by logger.rb/99999
D, [2012-08-13T20:04:53.948000 #7612] DEBUG -- : ****INICIO DEL TEST ****
D, [2012-08-13T20:05:27.206000 #7612] DEBUG -- : Test: 1. INICIALIZACIÓN DEL SCRAPER - OK
D, [2012-08-13T20:05:36.487000 #7612] DEBUG -- : Test: 4. CAMBIO DE SUPERMERCADO - OK
D, [2012-08-13T20:05:45.199000 #7612] DEBUG -- : Test: 3. PÁGINA DE PRODUCTOS - OK
D, [2012-08-13T20:05:54.486000 #7612] DEBUG -- : Test: 4. CAMBIO DE SUPERMERCADO - OK
D, [2012-08-13T20:06:00.224000 #7612] DEBUG -- : Test: 5. PÁGINA SIGUIENTE - OK
D, [2012-08-13T20:06:07.071000 #7612] DEBUG -- : Test: 6. INDEXADO DE PRODUCTOS - OK
```

Figura 4.2.10: Test scraper: fichero de log para el test.

Además, cabe destacar que, para la detección de fallos producidos durante la ejecución del programa es posible también verificar los diferentes archivos de log que son generados con cada ejecución del programa. De hecho, se crea un fichero de log específico para los errores producidos durante la ejecución del programa. A continuación se muestra, a modo de ejemplo, el contenido de uno del log de errores generado tras una ejecución fallida del programa:



```
# Logfile created on Sat Apr 28 19:25:47 +0200 2012 by logger.rb/99999
E, [2012-04-28T19:25:47.782000 #35921] ERROR -- : ****
E, [2012-04-28T19:25:47.789000 #35921] ERROR -- : Error al cambiar de supermercado-> Unable to locate Link, using
{:class=>"caja_super_link", :index=>2}-> Unable to locate Link, using {:class=>"caja_super_link", :index=>2}
E, [2012-04-28T19:25:47.712000 #35921] ERROR -- : [/Applications/jruby-1.6.3/lib/ruby/gems/1.8/gems/celerity-0.8.9/lib/celerity/
element.rb:179:in `assert_exists'", "/Applications/jruby-1.6.3/lib/ruby/gems/1.8/gems/celerity-0.8.9/lib/celerity/element.rb:
227:in `to_xml'", "(irb):379:in `cambiarSupermercado'", "(irb):558:in `evaluate'", "org/jruby/RubyKernel.java:1088:in `eval'", "/
Applications/jruby-1.6.3/lib/ruby/1.8/irb/workspace.rb:81:in `evaluate'", "/Applications/jruby-1.6.3/lib/ruby/1.8/irb/context.rb:
219:in `evaluate'", "/Applications/jruby-1.6.3/lib/ruby/1.8/irb.rb:158:in `eval_input'", "/Applications/jruby-1.6.3/lib/ruby/1.8/
irb.rb:271:in `signal_status'", "/Applications/jruby-1.6.3/lib/ruby/1.8/irb.rb:278:in `signal_status'", "/Applications/
jruby-1.6.3/lib/ruby/1.8/irb.rb:155:in `eval_input'", "/Applications/jruby-1.6.3/lib/ruby/1.8/irb/ruby-lex.rb:244:in
`each_top_level_statement'" "org/jruby/RubyKernel.java:1419:in `loop'", "/Applications/jruby-1.6.3/lib/ruby/1.8/irb/ruby-lex.rb:
230:in `each_top_level_statement'" "org/jruby/RubyKernel.java:1191:in `catch'", "/Applications/jruby-1.6.3/lib/ruby/1.8/irb/
ruby-lex.rb:229:in `each_top_level_statement'" "/Applications/jruby-1.6.3/lib/ruby/1.8/irb.rb:154:in `eval_input'", "/
Applications/jruby-1.6.3/lib/ruby/1.8/irb.rb:71:in `start'", "org/jruby/RubyKernel.java:1191:in `catch'", "/Applications/
jruby-1.6.3/lib/ruby/1.8/irb.rb:70:in `start'", "/Applications/jruby-1.6.3/bin/jirb:13:in `(root)'" ]
E, [2012-04-28T19:25:47.715000 #35921] ERROR -- : ****
```

Figura 4.2.11: Test scraper: fichero log de errores.

Por último, hay que indicar que junto a este documento se anexa un vídeo que muestra la ejecución del test para el scraper que puede ser visualizado para ver el funcionamiento en tiempo real del mismo.

4.3. Mediciones de rendimiento

Para finalizar el apartado dedicado al testeo del sistema, se realizan mediciones de tiempos para diferentes tareas que el cliente móvil deberá de realizar de forma habitual. Así, será posible ver cómo se comporta la aplicación en diferentes dispositivos así como poder hacernos una idea de cuál será la experiencia de usuario que se obtiene. Para las mediciones de rendimiento se hace uso de una versión modificada de la aplicación que permite anotar el tiempo de inicio y fin de una tarea, de forma que sea posible calcular la diferencia existente entre éstos para determinar cuál ha sido el tiempo empleado. Puesto que el tiempo que tarda la aplicación en realizar diferentes tareas depende de múltiples de factores ajenos a la propia aplicación, para poder establecer una medida más independiente se realizan tres mediciones de tiempo para una misma tarea y, posteriormente, se calcula la media de la terna.

Por otro lado, puesto que se desarrollan dos versiones diferentes de la aplicación servidor se duplican dichas mediciones para poder realizar una comparativa entre ambas implementaciones.

Las mediciones realizadas se corresponden a la ejecución de la aplicación cliente en los dos dispositivos Android que se poseen dado que las mediciones que se pueden realizar en los emuladores de las diferentes plataformas programadas no ofrecen tiempos reales. Además, la conexión establecida entre servidor y cliente móvil se realiza a través de la conexión WiFi del dispositivo, de forma que ambos sistemas están conectados a través de una red de área local a 54 Mbps.

A continuación se muestran las tablas que resumen los resultados obtenidos para los diferentes dispositivos e implementaciones del servidor. Para más detalle puede consultarse el archivo para Excel que se adjunta en el que pueden verse todas las mediciones de tiempos realizadas.

Funciones	RESTful		SERVIDOR WEB	
	HTC (ms)	Sony (ms)	HTC (ms)	Sony (ms)
Acceso vista listas	4517,67	914,00	4408,67	939,67
Acceso lista (5 items)	6959,33	1340,67	6550,00	1082,67
Acceso lista (10 items)	7706,33	1419,00	8041,00	1619,00
Eliminar lista (5 items)	495,67	115,33	578,33	116,33
Eliminar lista (10 items)	655,00	101,33	667,33	172,67
Comprobar lista (5 items)	5334,00	1014,33	5357,33	1339,67
Comprobar lista (10 items)	7744,33	1243,00	7888,00	1556,33
Agregar producto a lista	282,00	93,33	186,33	157,00

Tabla 4.3.1: Medición rendimiento: Gestión de listas.

Funciones	RESTful		SERVIDOR WEB	
	HTC (ms)	Sony (ms)	HTC (ms)	Sony (ms)
Acceso favoritos(5 items)	5243,67	936,00	5043,33	919,33
Acceso favoritos (10 items)	7643,00	1246,33	7505,67	1194,67
Eliminar favorito	557,67	189,33	541,67	193,00
Agregar favorito	89,67	29,00	94,33	45,33

Tabla 4.3.2: Medición rendimiento: Gestión de favoritos.

Funciones	RESTful		SERVIDOR WEB	
	HTC (ms)	Sony (ms)	HTC (ms)	Sony (ms)
Carga menú categorías	8469,67	1400,33	8711,67	1474,00
Carga menú aperitivos y frutos secos	6941,33	1177,33	4908,00	1456,00
Carga menú frutos secos	6445,33	1119,67	7277,00	1462,33
Carga almendras	39382,67	4424,33	16721,67	4409,67
Acceso a producto	6630,67	2225,67	6723,33	2421,00
Búsqueda "pila"	6413,00	931,00	10383,67	1042,67
Búsqueda código barras	11835,33	1381,67	11664,33	1284,33

Tabla 4.3.3: Medición rendimiento: Búsqueda de productos.

Si se analizan los datos anteriores detenidamente, puede verse que los tiempos de respuesta son similares para ambos tipos de implementación en el servidor aunque, en la mayoría de casos, la implementación RESTful ofrece mejores resultados aunque por escaso margen.

En cuanto a lo que se refiere a los tiempos de respuesta según el dispositivo vemos que existe una gran diferencia entre ambos terminales, resultando el Sony Ericsson Xperia Neo mucho más veloz en todas las mediciones realizadas. De hecho, la

experiencia de usuario es realmente pobre en el HTC Wildfire, lo que sugiere que para que la aplicación se ejecute de forma agradable para el usuario será necesario que éste disponga de un aparato relativamente moderno y potente. No obstante, hay que indicar que, por lo general, el teléfono HTC Wildfire es muy lento y todas las aplicaciones tienen un rendimiento muy bajo, por lo que es razonable pensar que parte del el problema de la lentitud radica en el dispositivo.

Por otro lado, los tiempos obtenidos son correctos, ofreciendo una experiencia al usuario aceptable. A pesar de eso, hay que apuntar que el uso se antoja algo pesado a diferencia de otras aplicaciones desarrolladas de forma nativa para Android. Este efecto era de esperar puesto que, tal y como ya se apuntó en la sección de diseño, éste es uno de los puntos flacos del framework PhoneGap.

Capítulo 5

Planificación y presupuesto

En este capítulo se va a tratar de hacer una estimación de las fases de las que constará el proyecto así como de la duración de las mismas. En base a éstas se realizará un presupuesto con el objetivo de conocer, de manera aproximada, el coste del desarrollo del proyecto.

5.1. Planificación

La planificación de un proyecto es una de las tareas más complejas existentes a la hora de llevar a cabo el desarrollo de un proyecto. Para realizar una buena planificación es necesario conocer el problema muy a fondo y entenderlo completamente. Asimismo, para estimar de forma correcta el tiempo a dedicar para el desarrollo del proyecto es imprescindible poseer un alto grado de experiencia en dicho ámbito, puesto que es muy difícil saber el tiempo necesario para el desarrollo de las diferentes tareas a realizar si se desconoce cómo hay que llevarlas a cabo y los problemas que es probable que surjan, los cuales harán que las estimaciones sean erróneas.

Por otro lado, el desarrollo del proyecto se realiza a tiempo parcial, puesto que paralelamente se desarrollan otras actividades que impiden que el desarrollador pueda dedicar la jornada completa a esta tarea. Adicionalmente, se cuenta con el hecho de que el desarrollo del proyecto se realiza en diferentes etapas en las que ya se sabe de antemano que el desarrollador no podrá dedicar tiempo alguno durante ciertos meses puesto que estará dedicado plenamente a otras actividades.

Por tanto, en este caso, la realización de una planificación correcta se hace ciertamente complicada. No obstante, para simplificar ésta, se realiza la planificación del proyecto suponiendo que el desarrollador estará dedicado a tiempo completo al desarrollo de este proyecto, obviando los impedimentos citados anteriormente.

5.1.1. Fases del proyecto

Para facilitar la elaboración de la planificación, se divide el proyecto en diferentes fases:

- **Análisis (80 horas):** durante esta fase el objetivo principal será comprender el problema, intentando plasmar por escrito los diferentes requisitos que deberá de cumplir el sistema así como los usos que se hará del mismo. Es imprescindible dedicar el tiempo que sea necesario a esta etapa, dado que será en ella en la que se sentarán las bases para realizar un correcto desarrollo, siendo más

complicado y costoso tener que volver a esta fase debido a fallos cometidos en la misma que analizar bien el problema en primera instancia.

- **Diseño e investigación (240 horas):** en esta etapa se establece la arquitectura del sistema y se buscan formas de acometer los diferentes desarrollos a realizar. La investigación realizada ha sido muy intensa dado que tanto el scraper como la aplicación móvil presentaban grandes retos para el desarrollador. La búsqueda de alternativas para la implementación de ambas ha sido muy costosa, puesto que, además, para la implementación de la aplicación móvil los frameworks utilizados ni siquiera ofrecían versiones estables cuando se comenzó el desarrollo del proyecto y la documentación en ese momento era escasa.
- **Desarrollo (400 horas):** se presenta como la fase más larga del proyecto, dado que, aunque se realizó un buen análisis y diseño, algo que facilita en gran medida la tarea, el alcance del proyecto es muy amplio, habiéndose implementado una gran cantidad de funcionalidades. Además, se ha hecho uso de algunas tecnologías que se desconocían por completo al inicio de la implementación, como por ejemplo JRuby para el scraper o los frameworks usados en el cliente móvil. Asimismo al inicio de la implementación del proyecto, tanto jQueryMobile como PhoneGap, frameworks utilizados para el desarrollo de la aplicación móvil se encontraban en fase inicial de desarrollo y la documentación era escasa y estaba incompleta. De hecho, jQueryMobile ni si quiera ofrecía una versión estable.
- **Pruebas (24 horas):** a pesar de que tras el desarrollo de cada fragmento de código se comprueba que éste se comporta de la manera deseada, una vez completada la fase de implementación de las aplicaciones se procede a la realización de diferentes tests para verificar que no se producen errores. Además, estas pruebas realizadas podrán realizarse más adelante, una vez el sistema esté en funcionamiento, para detectar errores que puedan surgir posteriormente.
- **Paso a producción (16 horas):** puesto que el proyecto no se ha puesto a disposición de los usuarios, esta fase es la única que no se ha llevado a cabo. En ella se realizarían las tareas necesarias para que el sistema funcione en un entorno real. Adicionalmente, una vez producido el paso a producción sería necesario realizar de nuevo las pruebas hechas en la fase anterior para verificar que el funcionamiento del sistema es el esperado antes de que esté disponible para su uso por parte de los usuarios.

A modo de resumen, en la figura 5.1.1 se muestra un gráfico que refleja el porcentaje de tiempo dedicado a cada una de las tareas. Como puede verse, se ha destinado un mayor esfuerzo a la fase de desarrollo, abarcando ésta más de la mitad de las horas dedicadas al proyecto, seguido de las fases de diseño y análisis respectivamente. Por último, con un porcentaje notablemente menor se encuentran las fases de pruebas y paso a producción.

Para el desarrollo del proyecto se sigue un modelo en cascada, de forma que, una vez completada una fase se procede a la realización de la siguiente fase. Sin embargo, hay que destacar que las fases de diseño e investigación, desarrollo y pruebas se desarrollan de forma iterativa, donde se considera que una iteración es la realización de las tres fases en cascada. Así, se tiene como objetivo que a la finalización de una iteración

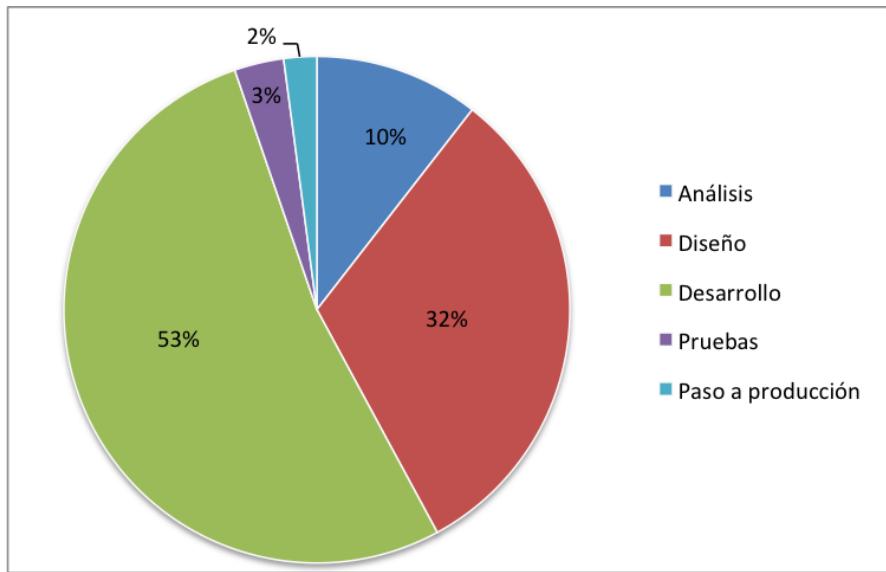


Figura 5.1.1: Planificación: gráfico de tiempo dedicado por tareas.

se disponga de un prototipo del sistema que sea completamente funcional pero que no incluya todas las funcionalidades requeridas. De esta manera, tras varias iteraciones se llegará a la consecución del producto final. Aplicando este método de desarrollo se pretende detectar y corregir los errores que se puedan producir lo más rápido posible, evitando que éstos se extiendan conforme el sistema crece. Adicionalmente, aunque no es el caso, esta forma de implementación facilita la inclusión de nuevos requisitos mientras que se está realizando el desarrollo, algo que es bastante frecuente en la implementación de aplicaciones.

El desarrollo iterativo mencionado anteriormente se lleva a cabo por separado para las diferentes aplicaciones que componen el sistema completo. De esta forma, una vez realizado el análisis del sistema, se procede a la realización de las iteraciones relativas a cada una de las aplicaciones siguiendo el siguiente orden: base de datos, servidor, scraper y cliente.

5.2. Presupuesto

Para la elaboración del presupuesto del coste del proyecto se van a tener en cuenta dos factores. Por un lado se considera el importe de las herramientas y dispositivos necesarios para la implementación y pruebas, mientras que por otro lado se tiene el coste de las horas destinadas al desarrollo del software. En este sentido, el segundo se computará en base a los dos perfiles necesarios para la elaboración del software:

- Analista: será el encargado de realizar las fases de análisis y diseño
- Programador: tendrá como tarea el desarrollo y las pruebas del software, así como del paso a producción.

Para este cálculo se hace uso de la herramienta InfoJobs Trends [23], que permite obtener, de manera aproximada, el salario medio de una profesión en base a las ofertas que se publican en dicha página Web. Además, para estimar el salario de cada uno de los perfiles se establece que el número de jornadas laborales es de 229 por año y

que se trabajan 8 horas diarias. De esta forma, el salario establecido para el perfil de analista es de 16,38 €/ hora, mientras que para el programador es de 14,20 €/hora.

En base a lo anterior, en la tabla 5.2.1 se muestran los diferentes cálculos realizados para la elaboración del presupuesto:

Hardware		Concepto		Coste (€)
		Ordenador portátil Macbook 13"		1000,00 ¹
		Teléfono móvil HTC Wildfire (A3333)		190,00 ²
		Teléfono móvil Sony Ericsson Xperia Neo (MT15)		240,00 ³
Software	Concepto	Horas	Precio (€)/hora	Coste (€)
	Análisis	80	16,38	1310,40
	Diseño	240		3931,20
	Desarrollo	400	14,20	5680,00
	Pruebas	24		340,80
	Paso a producción	16		227,20
TOTAL				12919,60 €

Tabla 5.2.1: Presupuesto del proyecto.

Como puede verse en la tabla anterior, el coste estimado total del proyecto asciende a la cantidad de **12919,60€**.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

Una vez finalizados los capítulos anteriores, llega el momento de hacer balance y evaluar las experiencias vividas durante el desarrollo del proyecto así como los resultados obtenidos. Antes de iniciar este proyecto se estudiaron diferentes posibilidades acerca del tema a desarrollar como proyecto final de carrera pero siempre con un objetivo claro, realizar una aplicación para dispositivos móviles Android dado que se disponía de un teléfono con las características necesarias y que, bajo mi punto de vista, este tipo de aplicaciones tienen un gran mercado en la actualidad y presentan peculiaridades que les hacen estar a la vanguardia de la tecnología. No obstante, tras muchas horas de investigación, posteriormente, este objetivo cambió y se decidió ampliar la aplicación a desarrollar a múltiples plataformas, haciendo aun mayor el nicho de negocio. De esta cuestión me siento particularmente orgulloso puesto que se presenta, a mi parecer, como una solución novedosa y que resuelve, en parte, uno de los mayores problemas a los que se enfrenta un desarrollador de aplicaciones móviles que no es otro que la gran cantidad de sistemas diferentes que existen, cada uno de ellos con sus propiedades, que hacen que se tengan que redoblar esfuerzos a la hora de desarrollar aplicaciones que puedan ser usadas en más de una plataforma.

Por otro lado, también era deseable que el proyecto constara de una parte servidor ya que se quería profundizar en los conocimientos adquiridos en este campo durante la carrera. En este aspecto influye sobremanera el que durante el desarrollo realizado también se estudiara un máster dedicado a estas tecnologías, en concreto el máster INFTEL de la universidad de Málaga que, dicho sea de paso, me ha ayudado de sobremanera para la realización del proyecto puesto que me ha permitido ampliar mis conocimientos de programación orientada a Internet y móviles así como otros aspectos propios del campo de la informática que, dado mi perfil de ingeniero de telecomunicación, desconocía. Es por esta razón que, a pesar de que la aplicación servidor ya había sido desarrollada en la versión con servidor web, se decide implementar una nueva versión que hace uso de servicios web RESTful aplicando parte de los conocimientos adquiridos. Adicionalmente, el haber cursado dicho máster me ha hecho sensibilizarme aún más sobre la necesidad de que las aplicaciones sean accesibles, es decir, que puedan ser utilizadas por personas con ciertas discapacidades, ya que éste es patrocinado por la Fundación Vodafone, la cual tiene como objetivo acercar la tecnología a este sector de la población. En consecuencia, se tratado de hacer que la aplicación móvil sea accesible, algo que se logra dadas las capacidades de que dispone el framework jQueryMobile.

En cuanto a la temática escogida, se barajaron diferentes posibilidades de diferentes ámbitos, siendo elegida ésta por su utilidad para el usuario en estos tiempos de crisis económica en los que vivimos. Personalmente, creo que ha sido un acierto ya que por lo que he podido observar durante este tiempo, este tipo de aplicaciones y servicios han proliferado sufriendo un gran aumento. Además, era difícil encontrar una aplicación que no hubiera sido desarrollada dado que existen miles de aplicaciones para móviles hoy en día. Esta premisa fue cumplida durante la mayor parte del tiempo dedicado al desarrollo de este proyecto hasta que aparecieron las primeras aplicaciones similares. De hecho, tanto Carritus (versión móvil) como Supertrupper, principales rivales, aparecieron mucho tiempo después de iniciarse este proyecto. No obstante, dada la naturaleza multisistema operativo de la solución propuesta, aun se presenta como la única aplicación de este estilo para dispositivos con un gran número de usuarios como pueden ser Symbian, BlackBerry, Bada o Windows Phone.

A lo largo de la elaboración del proyecto los problemas y complicaciones han sido muchos, la mayoría de ellos provocados limitaciones en cuanto a programación se refiere puesto que, como se ha comentado anteriormente, el perfil del autor es el de ingeniero de telecomunicación. A pesar de ello, ésto ha tenido como consecuencia una gran mejora en cuanto a conocimientos sobre ingeniería del software, bases de datos y otros campos más ligados con la programación como puede ser el uso de librerías o entornos de desarrollo así como de la programación orientada a objetos en diferentes lenguajes puesto que hasta el inicio de este desarrollo únicamente se tenían conocimientos de Java.

Dada la larga duración del desarrollo del proyecto, fundamentada en diferentes aspectos como el estudio del máster y la finalización de las asignaturas de la carrera, también se ha realizado un seguimiento del desarrollo de frameworks claves para el futuro de la programación web y de móviles y en los que están involucrados grandes empresas del sector como son PhoneGap y jQueryMobile. Mención especial merece el segundo ya que se comenzó a utilizar en la versión 0.8, la cual ni siquiera era estable y presentaba gran cantidad de errores. Además, la documentación en ese momento era escasa, lo que hizo que se tuviera que hacer un uso intensivo de herramientas como los foros oficiales, repositorio oficial informando de bugs y errores encontrados, así como de la web stackoverflow. De esta forma, el sentido de comunidad y de la filosofía del software libre se ha acrecentado durante este periodo, siendo ahora mucho más consciente de las ventajas que éstos aportan al sector.

Asimismo, he podido vivir desde dentro la evolución de un proyecto software con todas las consecuencias que ésta conlleva como la ampliación del alcance y de requisitos durante el desarrollo del proyecto. Éstos cambios han sido motivados en la mayoría de los casos por los nuevos conocimientos que se han ido adquiriendo y que hacían que moralmente fuera imprescindible para mi aplicarlos en este proyecto a pesar que supondría un mayor esfuerzo y tiempo. No obstante, aunque el alcance del proyecto se ha ido incrementando con el paso del tiempo, éste ha sido un hecho que no me ha importado en absoluto ya que este proyecto ha sido algo que realmente he hecho con gusto más que por obligación y además deseaba que el resultado final fuera de mi agrado, es decir, el mejor posible dentro de mis limitaciones. De hecho, he tenido que imponerme un límite para finalizarlo y no alargar más el desarrollo del mismo. En este aspecto ha sido de gran ayuda el hecho de que para la elaboración del proyecto me marqué como meta principal el aprender lo máximo posible, de forma que pudiera sacar rendimiento al proyecto y no limitarme a hacer lo que ya se. Es por ello que no me ha importado en absoluto que se haya alargado en el tiempo ni que haya tenido

que realizar un gran esfuerzo para el desarrollo de éste, especialmente en la última etapa en la que he tenido que compaginarlo con mi carrera profesional.

En lo que se refiere al resultado final obtenido, creo que se han cumplido con creces mis expectativas y objetivos marcados. El producto final es completamente funcional y, creo, realmente útil. Sin embargo, éste no va a ser puesto en funcionamiento para los usuarios, al menos de momento, ya que, posteriormente al desarrollo realizado, surge la duda de que es posible que existan problemas legales dado el hecho de que los datos utilizados no son propios.

Para finalizar, a modo de conclusión, creo que el desarrollo del proyecto me ha supuesto de gran ayuda en mi crecimiento como ingeniero, aumentando mis conocimientos y ayudándome a enfrentarme a la vida real ya que nunca me había abordado la resolución de un problema de estas dimensiones. Asimismo, ha incrementado mi interés, el cual ya era notable, por los dispositivos móviles e Internet a la vez que ha abierto nuevas puertas hacia un futuro profesional que no existían anteriormente y que espero aprovechar en un futuro.

6.2. Trabajos futuros

Cualquier proyecto de ingeniería puede mejorarse y éste no es menos. Existen gran cantidad de mejoras posibles muchas de ellas ya conocidas y que serán expuestas aquí y otras muchas que deberían de surgir una vez el sistema esté en fase de producción y a partir de la experiencia del usuario ya que son éstos los que hacen un uso más intensivo de las aplicaciones y los que suelen sufrir la mayoría de los fallos.

A continuación se detallan las diferentes líneas de trabajo a desarrollar en un futuro:

- **Ampliación del número de supermercados:** actualmente se dispone únicamente de cuatro supermercados sobre los que realizar las comparaciones de precios de los productos. Sería interesante incorporar nuevos establecimientos para ampliar el campo de comparación y ofrecer al usuario mejores resultados.
- **Obtención de productos y precios a partir de otra fuente:** como se ha comentado en capítulos anteriores, la extracción de productos y de los precios y características de los mismos se realiza a partir de la web www.carritus.com. Ésto a pesar de resultar cómodo a la hora de la programación de la aplicación scraper presenta impedimentos legales por lo que, para poder comercializar el proyecto, sería interesante hacer uso de otra fuente de información. La opción ideal es obtener los productos de las páginas webs oficiales de cada uno de los supermercados aunque esta solución presenta grandes problemas como es que cada uno de ellos ofrece diferentes datos acerca de los productos, pudiendo éstos variar incluso para un mismo producto por lo que es difícil proceder a un reconocimiento y clasificación satisfactorio de los mismos.
- **Mejora del rendimiento del scraper:** a pesar de que la aplicación scraper cumple las funcionalidades para las que fue diseñada, la ejecución de ésta es lenta. Ésto viene provocado por varios factores, entre ellos la tecnología escogida para su desarrollo. Adicionalmente, presenta problemas de consumo de memoria ya que la librería utilizada para la navegación, Celerity, ofrece un consumo muy elevado de memoria, efecto que se incrementa conforme se aumenta el número de páginas consultadas ya que no incluye mecanismos de recogida de basura. Este efecto también se observa en la versión Java de la librería, por lo que sería

necesario realizar una búsqueda de nuevas alternativas que permitan desarrollar las funcionalidades requeridas con un mejor rendimiento.

No obstante, la versión de la aplicación scraper que se entrega es fruto de diversas modificaciones realizadas a lo largo del desarrollo del sistema en los algoritmos programados para recorrer los elementos de la web carritus.com y extraer la información de la misma. En este ámbito se aprecia que la mejora de rendimiento es realmente difícil, a menos que se utilice una máquina más potente y una conexión a Internet más veloz, ya que la obtención de información está basada en llamadas a la propia API privada de carritus, de la misma forma que lo hacen las funciones JavaScript que implementan las funcionalidades de los diferentes botones y elementos de la web.

- **Uso de memoria caché para almacenado de las imágenes de productos en la aplicación móvil:** debería de implementarse la capacidad de almacenar las imágenes de productos que ya hayan sido consultados alguna vez. De esta manera, la aplicación mejoraría su rendimiento ya que no se tendría que descargar la imagen de un producto cada vez que éste se visualice a la vez que se reduce el uso de Internet necesario, algo que sería recomendable ya que lo habitual es que se disponga de un número de megas limitado para navegación mediante tecnologías 3G o GPRS en los dispositivos móviles.
- **Carga parcial de pantallas que incluyan listas en la aplicación móvil:** es habitual la visualización en la aplicación móvil de pantallas que contienen un gran número de elementos organizados en forma de lista, como pueden ser los productos que pertenecen a una categoría o tras la realización de una búsqueda. Puesto que en muchos casos el número de elementos de la lista es muy elevado, ésto provoca que sea necesario descargar una gran cantidad de información del servidor, lo que ralentiza la aplicación. Sería interesante en estos casos aplicar la filosofía del *lazy loading*, es decir, descargar una parte de los elementos de la lista de manera que éstos sean visualizados rápidamente en la pantalla para posteriormente descargar los demás elementos en caso de que así se requiera. Comunmente se suele descargar en primera instancia el número de items que caben en la pantalla y, conforme el usuario desciende o asciende sobre la lista, se van cargando nuevos items. De esta forma, se consigue que el usuario tenga una mejor experiencia ya que, aunque hay que realizar múltiples descargas de información, éstas son de menor longitud. Además, con esta técnica puede reducirse también el uso de la red ya que no es común que el usuario visualice por completo una lista con un número de elementos, siendo lo normal que encuentre lo que buscaba antes de llegar al final de la lista.
- **Extensión de la funcionalidad de búsqueda por código de barras a todas las plataformas móviles:** actualmente esta función está disponible únicamente para la aplicación Android debido básicamente a dos razones. La primera de ellas es que, como se dijo anteriormente, el plugin que permite la captura de los códigos sólo está disponible para la versión Android de PhoneGap por lo que sería necesario el desarrollo de plugins propios que ofrecieran esta posibilidad para las demás plataformas. La segunda se debe a que se dispone sólo de teléfonos Android para la realización de pruebas de la aplicación, por lo que, aun en caso de que el plugin necesario estuviera disponible para las demás plataformas, no sería posible probarlo ni comprobar el correcto funcionamiento de éste ya que no

tiene utilidad alguna en los emuladores de teléfonos móviles. Adicionalmente, esta funcionalidad también presenta el problema de que no se dispone de ninguna fuente de la que se pueda extraer cuál es el código de barras correspondiente a cada producto.

Apéndice A

Manual de usuario Cliente móvil

Aunque en el CD que acompaña a este documento se incluyen vídeos que muestran cómo utilizar el programa, en este capítulo se van a exponer las diferentes posibilidades que ofrece la aplicación cliente y cómo hacer uso de las mismas. Las imágenes que acompañan a este capítulo son capturas de pantalla del programa funcionando en un dispositivo Android real, en concreto, un Sony Ericsson Xperia Neo.

A.1. Conceptos básicos

A.1.1. Apertura del programa y pantalla inicial

Para iniciar el programa basta con pulsar sobre el ícono que se muestra en la pantalla de aplicaciones instaladas (ésto puede variar en función del sistema operativo, versión de éste y marca del dispositivo):



Figura A.1.1: Manual cliente móvil: inicio programa.

Como puede observarse, tras lanzar la ejecución se muestra la pantalla inicial de la aplicación. A partir de ésta se tendrá acceso a las tres funcionalidades más importantes: gestión de listas, búsqueda de productos y gestión de productos favoritos.

A.1.2. Navegación

Como en todo programa destinado a ser ejecutado en dispositivos móviles con pantallas táctiles, para accionar cada uno de los botones basta con pulsar con un dedo sobre el mismo. Además, tal y como se muestra en la siguiente imagen, en casi todas las pantallas se dispone de dos botones situados en la parte de arriba de la pantalla. Éstos permiten volver a la pantalla anterior o a la pantalla inicial de la aplicación respectivamente. Adicionalmente, para volver a visualizar la pantalla anterior puede pulsarse sobre la tecla física “BACK” (suele tener una flecha hacia la izquierda como icono) del dispositivo.

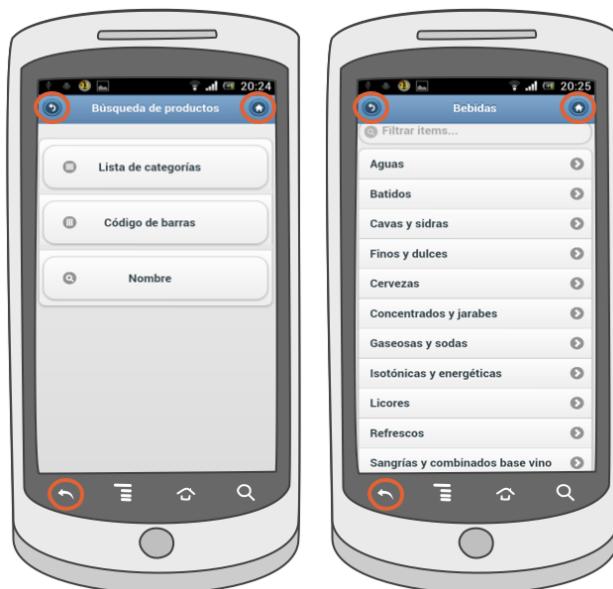


Figura A.1.2: Manual cliente móvil: navegación.

A.2. Búsqueda de productos

La búsqueda de productos es un elemento básico de la aplicación, por lo que puede accederse a través de la pantalla inicial:



Figura A.2.1: Manual cliente móvil: búsqueda de productos.

Se dispone de tres opciones en dispositivos Android para realizar dicha búsqueda (dos para los demás dispositivos puesto que no se implementa la opción de búsqueda mediante código de barras).

A.2.1. Búsqueda mediante categorías

Esta opción permite navegar a través de las diferentes categorías de productos hasta llegar a encontrar el producto que queremos ver. Para ello, se accede a dicha función a partir de la pantalla de selección de la opción de búsqueda y después se irá pulsando sucesivamente las diferentes categorías a las que pertenece dicho producto:

Proyecto Comprador



Figura A.2.2: Manual cliente móvil: búsqueda de productos mediante categorías (1).



Figura A.2.3: Manual cliente móvil: búsqueda de productos mediante categorías (2).



Figura A.2.4: Manual cliente móvil: búsqueda de productos mediante categorías (3).



Figura A.2.5: Manual cliente móvil: búsqueda de productos mediante categorías (4).

Siguiendo los pasos que se muestran en las imágenes anteriores, finalmente se accederá a una vista en la que se muestran los diferentes productos que pertenecen a las categorías seleccionadas. En este caso es posible realizar dos cosas. La primera de ellas es pulsar el producto que se deseé para acceder a una información más detallada:



Figura A.2.6: Manual cliente móvil: búsqueda de productos mediante categorías (5).

La segunda de las opciones es pulsar el botón que acompaña a cada uno de los productos y que está situado a la derecha de la pantalla. Éste hará que se muestren las opciones para este producto:



Figura A.2.7: Manual cliente móvil: búsqueda de productos mediante categorías (6).

A.2.2. Búsqueda mediante código de barras

Se trata de la opción más rápida para acceder a la información detallada de un producto. Lo primero será acceder a esta funcionalidad (únicamente en Android):

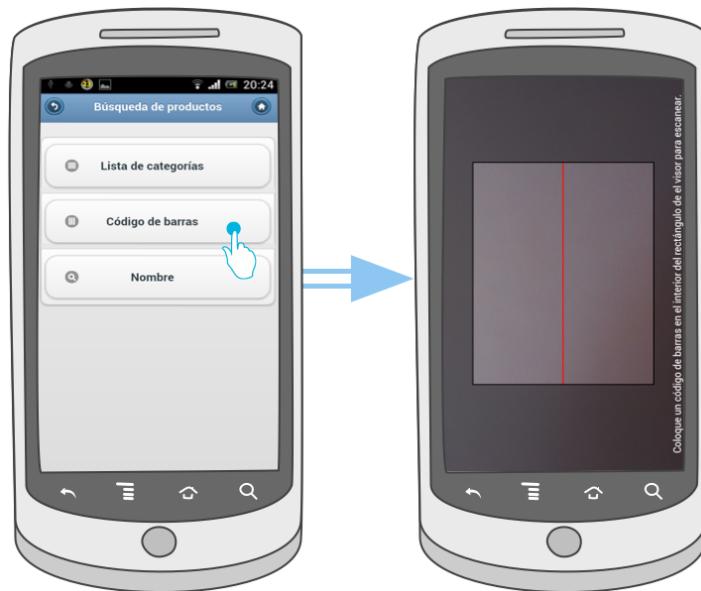


Figura A.2.8: Manual cliente móvil: búsqueda de productos mediante código de barras (1).

Una vez que se muestra la pantalla de captura de códigos de barras bastará con encuadrar con la cámara del dispositivo el código a escanear y, si éste pertenece a algún producto de la base de datos, tras ser detectado se procederá a mostrar la pantalla de detalle del producto:



Figura A.2.9: Manual cliente móvil: búsqueda de productos mediante código de barras (2).

A.2.3. Búsqueda mediante texto

Esta función permite realizar búsquedas en la base de datos a partir de un texto introducido por el usuario. En concreto, la cadena introducida se buscará en la marca y descripción del producto. Para ello será necesario acceder a esta funcionalidad a partir de la pantalla de selección del método de búsqueda. Posteriormente se introducirá el texto a buscar y se pulsará el botón “Buscar”:



Figura A.2.10: Manual cliente móvil: búsqueda de productos mediante texto (1).

En caso de que la búsqueda tenga éxito, se mostrará una lista con los productos encontrados. De nuevo, podremos acceder a la información detallada de uno de los productos o a las opciones asociadas a éstos:



Figura A.2.11: Manual cliente móvil: búsqueda de productos mediante texto (2).

A.3. Gestión de listas de productos

La aplicación permite crear listas de productos que quedarán almacenadas en el dispositivo y que podrán ser consultadas sin conexión. Para acceder a esta funcionalidad basta con pulsar sobre el botón “Listas” de la pantalla principal:



Figura A.3.1: Manual cliente móvil: gestión de listas.

A.3.1. Creación de listas

Para crear una lista de productos hay que pulsar sobre el botón “Crear nueva lista” que se muestra en la pantalla anterior. Posteriormente, habrá que introducir el nombre que se desea que tenga ésta:



Figura A.3.2: Manual cliente móvil: creación de listas.

A.3.2. Agregar producto a lista

Una vez que se dispone de listas es posible agregar productos a las mismas. Para ello, primeramente hay que buscar el producto que se desea agregar mediante las múltiples posibilidades que se detallan anteriormente o mediante el acceso a los favoritos que será explicado posteriormente:

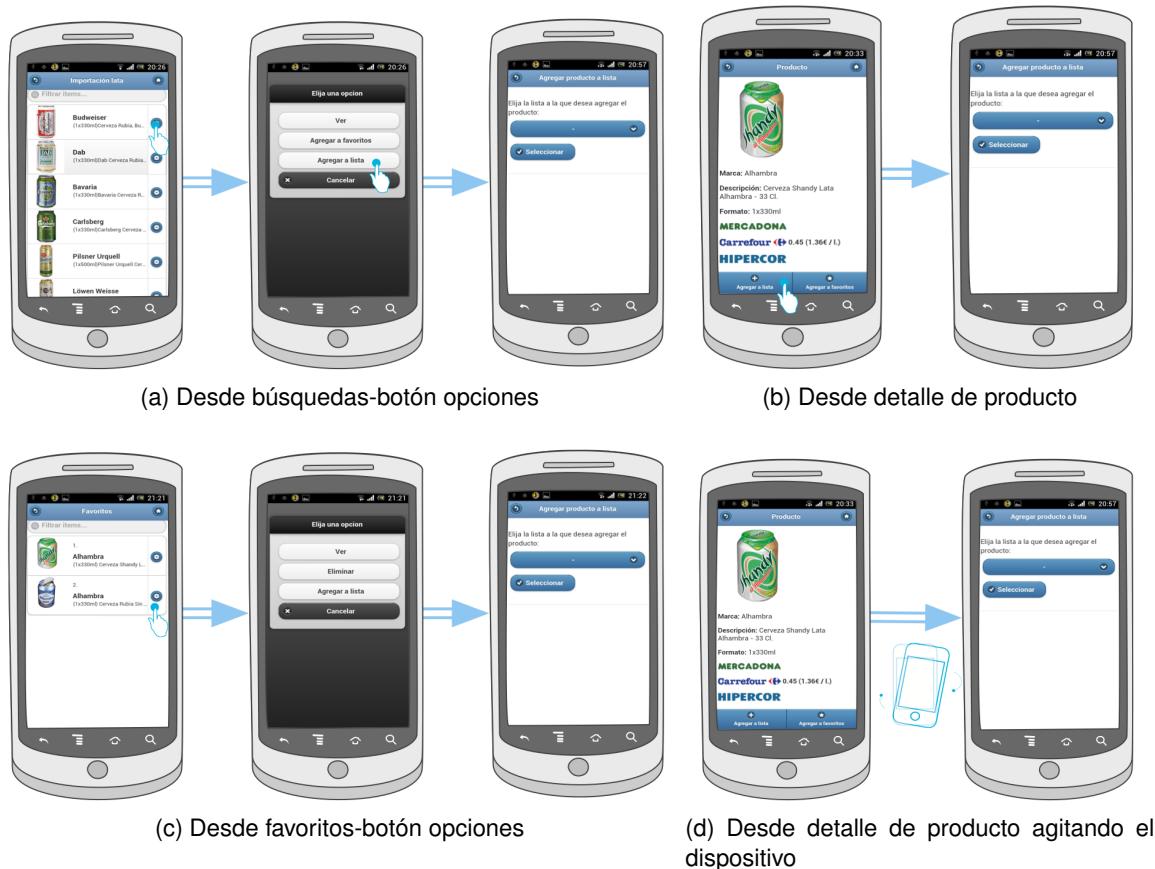


Figura A.3.3: Manual cliente móvil: agregar producto a lista (1).

Como puede verse, es posible acceder a esta función a través del botón de opciones que se muestra en las pantallas de búsqueda y favoritos o bien a través de la pantalla de detalle del producto haciendo click sobre el botón destinado a tal fin o agitando el dispositivo hacia los lados. De esta forma, se accede a la pantalla que permite elegir la lista a la que se desea agregar el producto:



Figura A.3.4: Manual cliente móvil: agregar producto a lista (2).

A.3.3. Eliminar producto de lista

Una vez que el producto haya sido agregado a una lista es posible eliminarlo de ésta. Para ello, se habrá que acceder a la lista en la que se encuentra el producto y acceder a la opción “Eliminar” que se encuentra en las opciones específicas del producto a eliminar:



Figura A.3.5: Manual cliente móvil: eliminar productos de listas.

A.3.4. Eliminar lista

Para eliminar una lista, así como todo lo que ésta contenga, se tienen dos opciones. La primera de ellas hacer click sobre el botón “Eliminar lista” dentro de la pantalla que muestra la lista de productos:



Figura A.3.6: Manual cliente móvil: eliminar lista (1).

También es posible eliminar una lista a través de la opción “Eliminar” que se muestra tras pulsar sobre el botón de opciones de la lista en cuestión:



Figura A.3.7: Manual cliente móvil: eliminar lista (2).

A.3.5. Modo compra

Esta función permite al usuario tachar y destachar los productos de una lista, lo que la hace especialmente útil para cuando el usuario se encuentra en el supermercado y quiere saber qué productos ha echado en su carro. Para acceder a ella es necesario haber creado una lista previamente y agregado productos a ésta.

Para activar el modo compra basta con entrar a la lista en cuestión y presionar el botón “Modo compra” o mediante el botón de opciones de la lista.

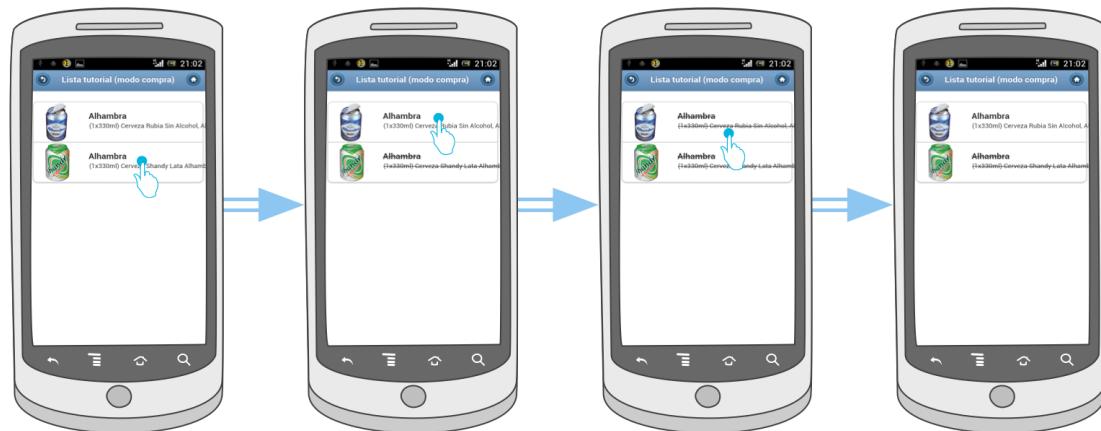


Figura A.3.8: Manual cliente móvil: modo compra.

Como puede verse en la imagen anterior, tras pulsar un elemento de la lista se tacha o destacha la descripción de éste.

A.4. Comparación de precios de productos

Se trata de la función principal del programa y en torno a la que gira todo el desarrollo realizado. Para comparar el precio de uno o varios productos es necesario, primeramente, crear una lista y agregar a ésta los productos que se quieran comparar. Posteriormente, mediante el botón de opciones de la lista o el botón dedicado para esta función el cual se muestra tras acceder a la lista se puede realizar la comparación requerida:



Figura A.4.1: Manual cliente móvil: comparación de productos.

Como se muestra en la imagen anterior, se genera un gráfico de barras según el precio obtenido para la lista junto con el valor en euros de la lista para cada uno de los supermercados. Adicionalmente, puesto que es posible que no todos los productos estén disponibles en todos los supermercados, si se pulsa sobre cualquiera de las barras se muestra el número de aciertos para el establecimiento correspondiente.

A.5. Gestión de favoritos

Esta función permite marcar productos como favoritos, de forma que posteriormente se pueda acceder a ellos más rápidamente. De este modo, los favoritos no son mas que una lista desde la que se puede consultar la información de los productos marcados como favoritos así como agregarlos a las diferentes listas de productos creadas por el usuario.

Para acceder a esta lista basta con pulsar sobre el botón “Favoritos” de la pantalla principal:



Figura A.5.1: Manual cliente móvil: favoritos.

A.5.1. Agregar producto a favoritos

Es posible agregar productos a favoritos a través del botón de opciones para el producto que se muestra en las diferentes pantallas de búsqueda de productos:

Proyecto Comprador



Figura A.5.2: Manual cliente móvil: agregar producto a favoritos (1).

También se puede agregar un producto a favoritos a partir de la vista de detalle de éste pulsando el botón “Agregar a favoritos”:



Figura A.5.3: Manual cliente móvil: agregar producto a favoritos (2).

A.5.2. Eliminar producto de favoritos

Para eliminar un producto de favoritos hay que acceder a la función “Eliminar” que se muestra tras pulsar el botón de opciones de un favorito:



Figura A.5.4: Manual cliente móvil: eliminar favoritos.

A.6. Filtrado

Como se ha podido observar en capturas anteriores, en las pantallas en las que se muestran una lista de items, en la parte superior se encuentra un campo de texto que

permite realizar un filtrado sobre la lista. De esta forma, es posible acelerar la búsqueda que queremos realizar:



Figura A.6.1: Manual cliente móvil: filtrado.

Apéndice B

Manual de usuario de la Scraper

En este apéndice se va a mostrar un pequeño manual que explica el uso de la aplicación scraper.

B.1. Ejecución

Para ejecutar la aplicación scraper, una vez que se dispone del entorno de ejecución correctamente configurado, basta con abrir una terminal del sistema operativo y ejecutar el comando:

```
jruby main.rb
```

Asimismo, en el fichero main.rb se incluye una sección al inicio en la que se definen varias variables de configuración del programa:

- **\$tipoServidor**: indica el tipo de implementación a utilizar para el servidor, tomando el valor 0 para la implementación RESTful y 1 para el servidor Web.
- **\$urlServidorWEB**: almacena la URL de acceso al servidor Web
- **\$urlServidorREST**: contiene la URL de acceso al servidor de aplicaciones sobre el que ha sido desplegado el servicio web RESTful

B.2. Partes de la GUI

La pantalla de la aplicación tiene el siguiente aspecto:

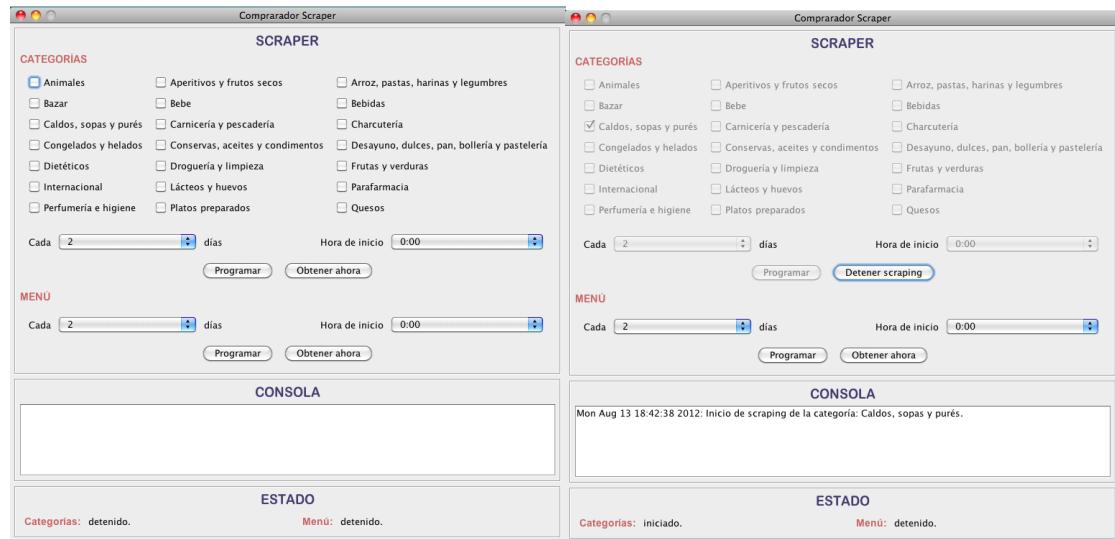


Figura B.2.1: Manual scraper: pantalla aplicación.

La pantalla de la aplicación puede dividirse en tres secciones verticales que engloban diferentes aspectos relativos a la aplicación:

- **Scraper (parte superior):** contiene todos los elementos para iniciar y parar la obtención de datos de carritus.com
- **Consola (parte central):** tiene como función ofrecer información al usuario sobre la ejecución del programa como puede ser la fecha y hora del inicio y finalización de los scrapings.
- **Estado (parte inferior):** indica el estado, iniciado o detenido, de los motores de scraping para las categorías y menú. De esta forma, el usuario podrá saber en cada momento si se están obteniendo datos.

B.3. Obtención de datos

A continuación se va a detallar como iniciar y detener la obtención de datos de la web carritus.com

B.3.1. Productos

Para iniciar la obtención de productos hay que seleccionar las categorías de las que se quieren extraer los datos de los productos marcando el checkbox correspondiente. Posteriormente, basta con pulsar el botón *Obtener ahora* de la sección de Categorías:



Figura B.3.1: Manual scraper: inicio obtención de datos de productos.

Tras la obtención de todos los productos de las categorías seleccionadas, el motor de scraping de categorías se detendrá automáticamente. No obstante, una vez que esté iniciado, siempre es posible detener la obtención de datos pulsando el botón *Detener scraping*:



Figura B.3.2: Manual scraper: detención obtención de datos de productos.

B.3.2. Categorías

Para iniciar la obtención de las categorías únicamente hay que pulsar sobre el botón *Obtener ahora* de la sección de Menú:

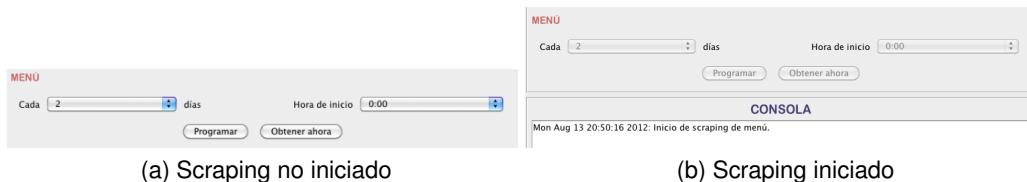


Figura B.3.3: Manual scraper: inicio obtención de datos de menú.

En este caso, no es posible detener la obtención de los datos de las categorías, es decir, del menú de carritos ya que se ha considerado que, puesto que la operación se realiza en poco tiempo, no es de utilidad detener dicho scraping.

B.4. Programación de obtención de datos

Puesto que la obtención de los datos deberá de realizarse de forma periódica, se incorpora una funcionalidad que permite programar la obtención de datos de forma de manera que ésta se inicia de forma automática en base a los parámetros seleccionados por el usuario.

En este caso, la programación de los scraping de productos y categorías puede realizarse a través de los combobox que se muestran en la GUI respectivamente:

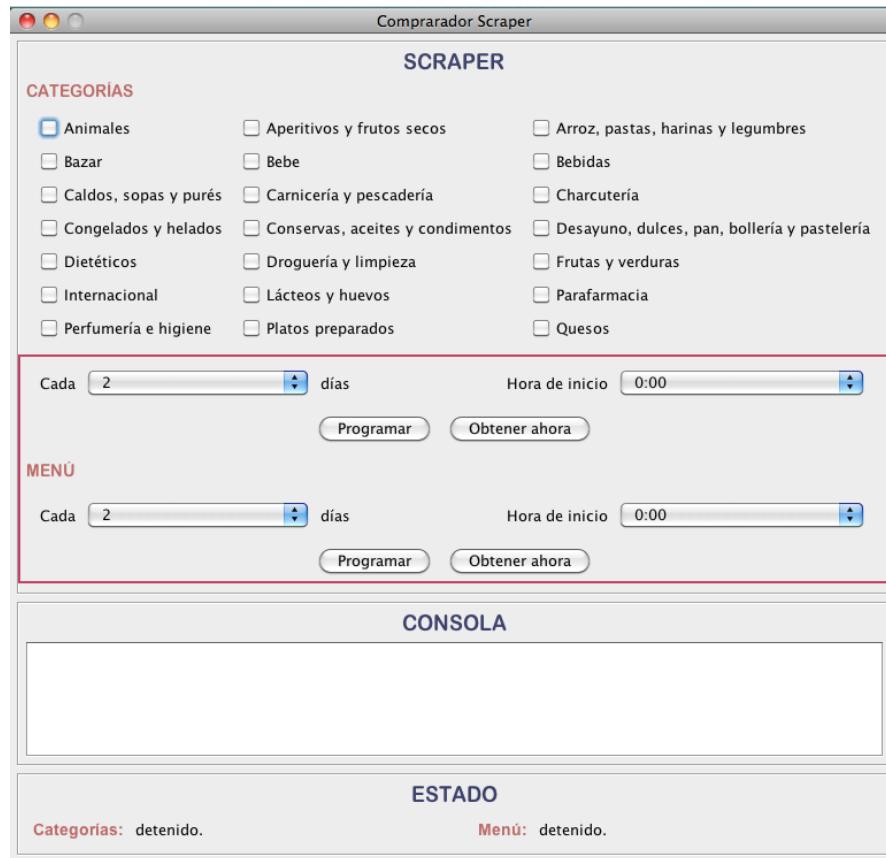


Figura B.4.1: Manual scraper: programación de la obtención de datos (1).

Tanto para los productos (sección superior) como para las categorías (sección inferior) se dispone de dos combobox:



Figura B.4.2: Manual scraper: programación de la obtención de datos (2).

El combobox de la izquierda permite seleccionar la frecuencia en días con la que se ejecutará la obtención de datos. Por otro lado, el combobox de la derecha permite establecer la hora en la que dará comienzo la obtención de los datos.

Una vez que haya sido seleccionada la configuración de frecuencia y hora de inicio deseada basta con pulsar el botón/es *Programar* y el/los motor/es de scraping para productos y/o categorías pasarán a estar en estado *programado*, de forma que estarán a la espera de que se llegue a la hora de inicio establecida para comenzar su ejecución.

Proyecto Comprador

Una vez terminada ésta, volverán a ejecutarse a la misma hora pero pasados el número de días que haya sido seleccionado mediante el combobox de frecuencia.

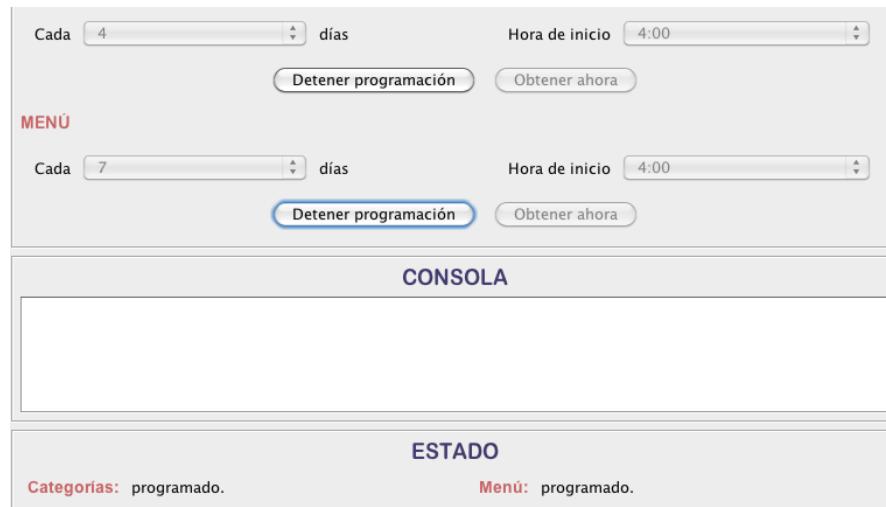


Figura B.4.3: Manual scraper: programación de la obtención de datos (3).

Por último, hay que indicar que es posible detener la programación de los scraping pulsando el botón *Detener programación*, lo que pondrá al motor de scraping en estado *detenido* y a la espera de órdenes para iniciar de nuevo su ejecución.

Apéndice C

Contenido del CD

Junto a esta memoria se entrega un CD que contiene el código al completo de todas las aplicaciones desarrolladas para este proyecto, los documentos creados y una serie de vídeos demostrativos en los que se puede ver el funcionamiento de las aplicaciones Scraper y Cliente. En la figura C.0.1 se puede ver el contenido de la carpeta principal del CD:

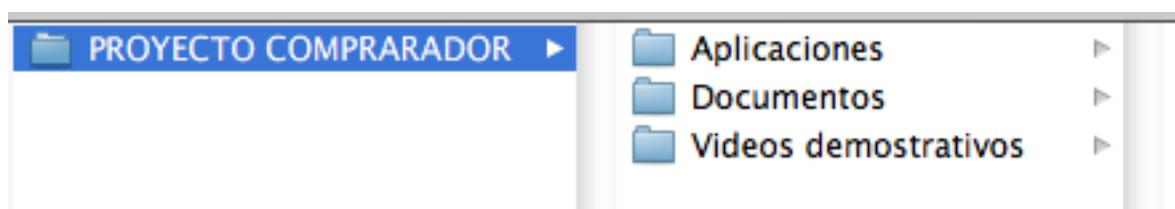


Figura C.0.1: Contenido del CD: carpeta principal

- **Directorio *Aplicaciones*:**

Contiene el software desarrollado: scripts de generación y de insercción de datos en la base de datos; proyectos de aplicaciones cliente móvil para Android, iOS y BlackBerry; aplicación Scraper y los proyectos de la aplicación servidor en sus dos versiones: servidor web y servicio web RESTful.

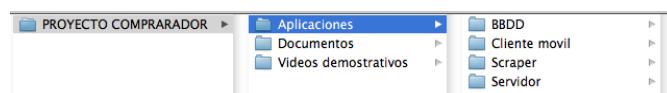


Figura C.0.2: Contenido del CD: directorio Aplicaciones

- **Directorio *Documentos*:**

Almacena esta memoria en formato PDF así como el conjunto de mediciones de tiempos obtenidos para realizar la medición de rendimiento de la aplicación Cliente móvil guardados en un fichero de Excel.

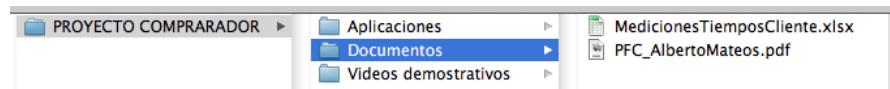


Figura C.0.3: Contenido del CD: directorio Documentos

○ **Directorio Vídeos demostrativos:**

En esta carpeta se encuentran todos los vídeos demostrativos de las aplicaciones Scraper y Cliente móvil, mediante los que se puede comprobar el funcionamiento correcto de todas las funcionalidades implementadas en ambos programas. En el caso del cliente, los vídeos muestran la ejecución del programa en un dispositivo móvil real con sistema operativo Android 4.0.2 (Sony Ericsson Xperia Neo) y en el emulador de iPhone 4.2.

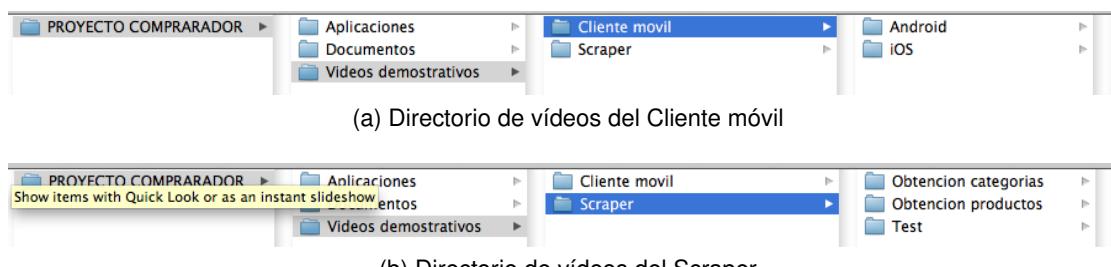


Figura C.0.4: Contenido del CD: directorio Vídeos demostrativos

Bibliografía

- [1] Android: sistema operativo desarrollado por Google. URL <http://www.android.com/>.
- [2] RIM: sistema operativo desarrollado para Blackberry. URL <http://www.rim.com>.
- [3] Symbian: sistema operativo desarrollado por Nokia. URL <http://symbian.nokia.com/>.
- [4] Visual Basic IDE: entorno de programación para aplicaciones que se ejecutan sobre sistemas operativos creados por Microsoft. URL <http://msdn.microsoft.com/en-us/vbasic/ms789056>.
- [5] Windows Phone: sistema operativo desarrollado por Microsoft. URL www.microsoft.com/windowsphone.
- [6] Amazon Web Services: plataforma para el despliegue de aplicaciones Java. URL <http://aws.amazon.com/es/>.
- [7] Apache Ant: herramienta de línea de comandos que permite la ejecución automatizada de procesos. URL <http://ant.apache.org/>.
- [8] Apache: servidor web. URL <http://httpd.apache.org/>.
- [9] Google App Engine: plataforma para el despliegue de aplicaciones Java., . URL <https://developers.google.com/appengine/?hl=es>.
- [10] Apple: compañía norteamericana que ha diseñado algunos de los dispositivos móviles con mayor éxito que existen hoy en día., . URL www.apple.com.
- [11] Appstore: plataforma que facilita la descarga e instalación de aplicaciones en dispositivos de Apple., . URL <http://www.apple.com/iphone/apps-for-iphone/>.
- [12] ARM: fabricante de procesadores para sistemas embebidos. URL www.arm.com.
- [13] Bada: sistema operativo desarrollado por Samsung. URL <http://www.bada.com>.
- [14] Blackberry: conocida marca fabricante de teléfonos de tipo inteligente. URL [http://www.blackberry.com/](http://www.blackberry.com).
- [15] Celerity: wrapper de la librería Java HtmlUnit para JRuby. URL [http://celerity.rubyforge.org/](http://celerity.rubyforge.org).
- [16] Cordova: framework Phonegap incluido bajo esta nomenclatura dentro del proyecto Apache. URL <http://incubator.apache.org/cordova/>.

- [17] Anuncio de la desaparición de Symbian por parte de Stephen Elop, presidente de Nokia. URL <http://conversations.nokia.com/2011/05/26/stephen-elop-in-china-time-for-a-challenger-mindset-video/>.
- [18] Código de barras EAN-13. URL <http://es.wikipedia.org/wiki/EAN>.
- [19] Eclipse: entorno de desarrollo Open Source multilenguaje. URL www.eclipse.org/.
- [20] HtmlUnit: librería Java. URL <http://htmlunit.sourceforge.net/>.
- [21] HyperText Transfer Protocol (HTTP). URL <http://www.w3.org/Protocols/rfc2616/rfc2616.html>.
- [22] iOS: sistema operativo desarrollado por Apple. URL <http://developer.apple.com/technologies/ios/>.
- [23] InfoJobs Trends: herramienta para estimación de salarios. URL <http://salarios.infojobs.net/>.
- [24] iPad: primer tablet PC desarrollado por Apple. URL www.apple.com/ipad/.
- [25] iPhone: smartphone desarrollado por Apple. URL www.apple.com/iphone.
- [26] Java: lenguaje de programación orientado a objetos., . URL <http://www.java.com/es/>.
- [27] Java Enterprise Edition (JEE): versión del conocido lenguaje destinada al desarrollo de aplicaciones web., . URL <http://www.oracle.com/technetwork/java/javaee/overview/index.html>.
- [28] Javascript: lenguaje de scripting aplicado a la web., . URL <http://es.wikipedia.org/wiki/JavaScript>.
- [29] Jersey (JSR-311): implementación Java de servicio web RESTful. URL <http://jsr311.java.net/>.
- [30] EclipseLink JPA: implementación de persistencia en Java. URL <http://www.eclipse.org/eclipselink/jpa.php>.
- [31] jQuery: framework JavaScript., . URL www.jquery.com.
- [32] jQueryMobile: framework para el desarrollo de aplicaciones web destinadas a dispositivos móviles., . URL <http://jquerymobile.com/>.
- [33] JRuby: implementación Java del lenguaje de scripting Ruby. URL <http://jruby.org/>.
- [34] Kayak: comparador de precios de vuelos. URL www.kayak.com.
- [35] Mamp: plataforma que instala y permite una gestión sencilla de Apache, MySQL y PhpMyAdmin para sistemas operativos MacOs. URL <http://www.mamp.info/en/index.html>.
- [36] Google Maps: sistema de mapas ofrecido de forma gratuita a través de Internet. URL <http://maps.google.com/>.

- [37] Google Play: plataforma que facilita la descarga e instalación de aplicaciones en dispositivos Android así como la venta y distribución de películas, libros y música. URL <https://play.google.com/store>.
- [38] Mysql, . URL <http://www.mysql.com/>.
- [39] MySQL WorkBench: herramienta que permite la gestión y diseño completo de bases de datos MySQL., . URL <http://www.mysql.com/downloads/workbench/>.
- [40] Netbeans: entorno de desarrollo oficial para Java. URL <http://www.netbeans.org>.
- [41] Nokogiri: parser HTML, XML y SAX para JRuby. URL <http://nokogiri.org/>.
- [42] Objective-c: lenguaje de programación utilizado para desarrollo de aplicaciones para iOS y Mac OS. URL <http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html>.
- [43] OCU: Organización de Consumidores y Usuarios. URL <http://www.ocu.org>.
- [44] OpenShift: plataforma creada por Red Hat para el despliegue de aplicaciones en Internet., . URL <https://openshift.redhat.com/app/>.
- [45] Open Source: filosofía basada en la compartición de el código y las técnicas empleadas en el desarrollo de diferentes aplicaciones tanto software como hardware., . URL http://en.wikipedia.org/wiki/Open_source.
- [46] Platform as a Service (PaaS): servicio de plataforma de computación en la nube. URL http://en.wikipedia.org/wiki/Platform_as_a_service.
- [47] Phonegap: framework para el desarrollo de aplicaciones móviles basado en HTML5, CSS y JavaScript. URL <http://www.phonegap.com/>.
- [48] PHP: lenguaje interpretado destinado a la creación de webs dinámicas., . URL <http://www.php.net/>.
- [49] PhpMyAdmin: herramienta que permite la gestión de bases de datos MySQL a través de un navegador web., . URL http://www.phpmyadmin.net/home_page/index.php.
- [50] Profligacy: librería para JRuby que facilita el diseño de GUIs mediante la tecnología Swing de Java. URL <http://ihate.rubyforge.org/profligacy/>.
- [51] QUnit: framework orientado al testeo de aplicaciones JavaScript. URL <http://docs.jquery.com/QUnit>.
- [52] RGraph: librería HTML5 y JavaScript que permite la creación de gráficas. URL <http://www.rgraph.net/>.
- [53] Descripción de las características del sistema operativo de Blackberry. URL <http://us.blackberry.com/developers>.
- [54] Ruby: lenguaje de scripting orientado a objetos. URL <http://www.ruby-lang.org/es/>.

- [55] Rufus-scheduler: librería para JRuby que permite la automatización de ejecución de código mediante sentencias tipo CRON. URL <https://github.com/jmettraux/rufus-scheduler/>.
- [56] Servidor de aplicaciones. URL http://es.wikipedia.org/wiki/Servidor_de_aplicaciones.
- [57] Shared hosting. URL http://en.wikipedia.org/wiki/Shared_web_hosting_service.
- [58] Silverlight: framework de Microsoft para desarrollo de aplicaciones ricas para Internet (RIA). URL <http://msdn.microsoft.com/en-us/vbasic/ms789056>.
- [59] SimpleDialog: plugin para jQuery Mobile que facilita el mostrado de ventanas emergentes. URL <http://dev.jtsage.com/jQM-SimpleDialog/>.
- [60] TextMate: editor de textos para MacOs orientado a la programación. URL <http://macromates.com/>.
- [61] ThemeRoller: herramienta web que permite crear estilos para jQueryMobile de forma rápida y sencilla. URL <http://jquerymobile.com/themeroller/>.
- [62] Descripción y definición de touchpad. URL en.wikipedia.org/wiki/Touchpad.
- [63] Descripción y definición de trackball,. URL es.wikipedia.org/wiki/Trackball.
- [64] Descripción y definición de trackwheel,. URL http://en.wikipedia.org/wiki/Scroll_wheel.
- [65] Trivago: comparador de precios de hoteles. URL www.trivago.es.
- [66] Virtual Private Server (VPS). URL http://en.wikipedia.org/wiki/Virtual_private_server.
- [67] WebKit: navegador web ligero OpenSource. URL <http://www.webkit.org/>.
- [68] Wi-Fi. URL <http://es.wikipedia.org/wiki/Wi-Fi>.
- [69] XCode: entorno de desarrollo propio de Apple. URL <http://developer.apple.com/xcode/>.
- [70] eXtensible Markup Language (XML). URL <http://www.w3.org/XML/>.
- [71] XNA: framework de Microsoft para desarrollo de juegos. URL <http://msdn.microsoft.com/en-us/library/bb200104.aspx>.
- [72] comScore. 2012 Mobile Future in Focus. URL http://www.comscore.com/Press-Events/Presentations_Whitepapers/2012/2012_Mobile_Future_in.Focus.
- [73] Inc Gartner. Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth, Febrero 2012. URL <http://www.gartner.com/it/page.jsp?id=1924314>.
- [74] Google. Android UI design patterns, 2010. URL <http://dl.google.com/googleio/2010/android-android-ui-design-patterns.pdf>.
- [75] Andy Rubin. Android@Mobile World Congress: Its all about the ecosystem, Febrero 2012. URL <http://googlemobile.blogspot.com.es/2012/02/androidmobile-world-congress-its-all.html>.

ANEXO IV: INFORME DE VALORACIÓN DE UN PROYECTO

El tribunal constituido para la evaluación del proyecto PFC titulado:

Realizado por el alumno: _____

Y dirigido por el tutor: _____

Ha resuelto asignarle la calificación de:

- SOBRESALIENTE (9 - 10 puntos)
- NOTABLE (7 – 8.9 puntos)
- APROBADO (5 – 6.9 puntos)
- SUSPENSO

Con la nota¹: puntos.

El Presidente: _____

El Secretario: _____

El Vocal: _____

Granada, a ____ de _____ de 200 ____

¹ Solamente con un decimal.