Desarrollo de un aplicativo *mobile* multiplataforma

Tomoki Kamo Mora

Enginyeria Informàtica

Carles Sanchez Rosa

06/06/2012

Resumen del proyecto

El principal objetivo de este proyecto final de carrera es aplicar de forma conjunta los conocimientos que se han adquirido durante las diferentes asignaturas cursadas. En esta memoria se recoge toda la documentación que se genera en las etapas de desarrollo de un proyecto *software*: recogida inicial de requerimientos, análisis, diseño, implementación y validación.

El *software* a desarrollar es un aplicativo *mobile* multiplataforma usando un *framework* de desarrollo ágil. El objetivo dicha aplicación es crear una red para compartir información entre usuarios sobre el precio y la localización de los mejores *souvenirs* en cualquier parte del mundo.

Índice

Índice de figuras

1.- Introducción

1.1.- Justificación del PFC y contexto: punto de partida y aportación del PFC

Gracias a las nuevas redes de información y terminales móviles cada vez tenemos más información a nuestro alcance y estamos conectados en todo momento. Disponemos de una ingente cantidad de información relacionada con nuestro día a día y con la sociedad en la que vivimos, pero cuando nos movemos a otros lugares del mundo en los que las costumbres o la información no son tan accesibles nos sentimos perdidos. Esto es lo que sienten por ejemplo los viajeros cuando se adentran en países que tienen en la cultura del regateo su forma de vida: no sabemos cuánto deberíamos pagar por algo tan sencillo como un imán para la nevera.

El objetivo del *software* **HowMuchDidYouPay** es crear una red para compartir este tipo de información, ¿cuánto ha pagado un usuario por un objeto?, y de esta forma poder ayudar a futuros viajeros en la ardua tarea que supone el regateo.

Como objetivo secundario y como consecuencia de compartir esta información otros viajeros podrán consultar cuáles son las mejores zonas en las que comprar *souvenirs*, saber éobjetos podrán encontrar y buscar nuevas ideas.

1.2.- Objetivo PFC

El objetivo de este proyecto es poner en práctica de forma conjunta los conocimientos adquiridos en las diferentes asignaturas del segundo ciclo de Ingeniería Informática, gestionando un proyecto *software* desde la toma de requerimientos hasta la entrega del *software* al cliente, mediante unas metodologías que garanticen la calidad del mismo.

Otro objetivo de este PFC es estudiar las diferentes alternativas que existen en el mercado para realizar desarrollos para dispositivos móviles que sean independientes de la plataforma en la que van a ser usados.

1.3.- Enfoque y método seguido

Para el desarrollo del PFC se ha seguido la metodología de desarrollo en cascada, en el que el inicio de cada etapa debe esperar a que la etapa anterior finalice ya que la documentación de partida de una etapa se genera en las etapas anteriores.

Se ha escogido esta metodología porque los objetivos del desarrollo están marcados inicialmente y no variarán a lo largo del desarrollo ya que el plazo de entrega es fijo y todas las etapas deben seguir una calendarización muy estricta.

Las fases del desarrollo son las siguientes:

- Toma de requerimientos
    - Documentación de los requisitos de la aplicación
    - Guiones
    - Casos de uso
    - Casos de uso detallados
- Diseño interficie gráfica
    - *Wireframes*
    - Selección *framework* / entorno de trabajo
    - Diseño prototipo
    - Pruebas de uso del prototipo con usuarios
- Diseño aplicativo
    - Diseño de la arquitectura
    - Diseño del modelo de datos
- Implementación
- Pruebas validación
- Líneas de futuro

Durante todos los pasos del desarrollo se irá generando documentación que será adjuntada en el presente documento.

1.4.- Planificación del proyecto

La planificación de las diferentes fases del proyecto es la siguiente:

Tarea	Fecha de inicio	Fecha de fin	Duración (días)
PAC 2			
Documentación de los requisitos de la aplicación	14/03	20/03	5
Selección *framework*	21/03	23/03	3
Diseño interficie gráfica	26/03	30/03	5
Diseño de prototipo	02/04	04/04	3
Test prototipo	05/04	11/04	5
PAC 3			
Diseño de la arquitectura	12/04	16/04	3
Diseño modelo de datos	17/04	19/04	3
Diseño *software*	20/04	09/05	14
Entrega final			
Desarrollo	10/05	29/05	14
Pruebas validación	30/05	01/06	3
Líneas de futuro	04/06	05/06	2
Total			60

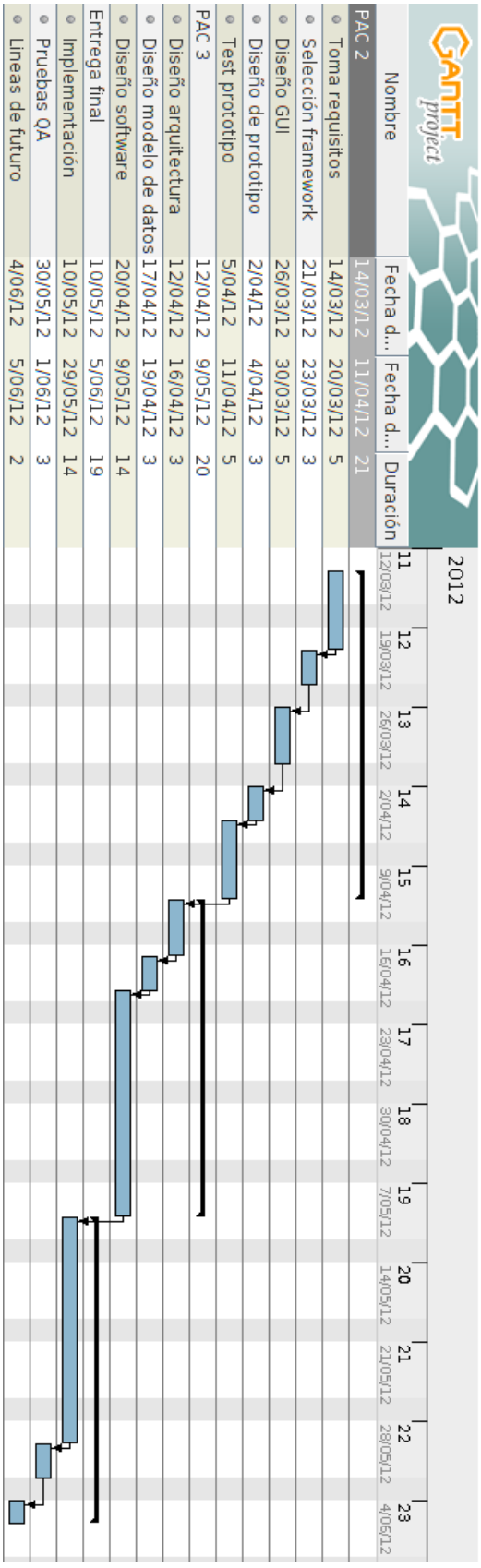


Ilustración 1: Planificación

1.5.- Productos obtenidos

Los productos obtenidos del desarrollo de este PFC son:

- Fichero a instalar en el dispostivo
- Código fuente de la aplicación
- Prototipo
- Memoria del PFC con toda la documentación técnica generada a lo largo del desarrollo

1.6.- Breve descripción de otros capítulos de la memoria

Los siguientes capítulos de esta memoria son las diferentes fases que componen el proyecto:

1.6.1.- Toma de requerimientos

Contacto con el cliente para la recogida de los requerimientos del *software* que se quiere desarrollar. Se analizarán las necesidades para determinar qué objetivos tiene que cumplir el aplicativo.

En esta fase se definirán claramente los flujos de información, formatos de los datos que se presentarán a los usuarios y la operativa en general.

1.6.2.- Diseño interficie gráfica

Durante esta fase se realizará un prototipo para validar que los requisitos recogidos son los correctos y que el diseño gráfico implementado se adapta a las características de los usuarios del aplicativo. Para poder realizar la interficie gráfica primero se tendrá que seleccionar el *framework* de trabajo ya que dependiendo del *framework* escogido se podrá hacer uso de elementos nativos del S.O. del terminal o se podrán usar elementos genéricos dentro de una interficie web.

De esta fase se obtiene un documento detallado con el prototipo y validación de la

interficie gráfica.

1.6.3.- Diseño de la arquitectura

En la fase de diseño de la arquitectura se realizará un análisis de la arquitectura del sistema. Se analizarán que sistemas intervendrán en los procesos, las interacciones y la comunicación que se realizará entre las mismas, así como la descomposición en componentes *software* a desarrollar.

Esta fase producirá un documento con el diseño de la arquitectura del sistema.

1.6.4.- Diseño del modelo de datos

En la fase de diseño del modelo de datos se realizará un análisis de los modelos que intervienen en el sistema y como éstos están estructurados.

Esta fase producirá un documento con el diseño del modelo de negocio.

1.6.5.- Análisis software del aplicativo

Durante esta fase del proyecto se definirán los actores que intervendrán y se crearán los diagramas de casos de uso y secuencia para cada componente *software* detectado durante el diseño de la arquitectura.

De esta fase se obtiene un documento con el diseño detallado de los casos de uso y de los diferentes diagramas con las especificaciones a implementar.

1.6.6.- Implementación

Durante esta fase se realizará la implementación del aplicativo partiendo del diseño detallado realizado en la etapa anterior.

De esta fase se obtiene un fichero ejecutable.

1.6.7.- Pruebas validación

Durante la fase de pruebas se validará que los requisitos iniciales y las especificaciones diseñadas en etapas anteriores han sido correctamente implementadas en el ejecutable desarrollado.

1.6.8.- Lineas de futuro

En este capítulo de la memoria se evaluarán posibles evolutivos que se podrían llevar a cabo para mejorar el sistema y ampliarlo.

2.- Toma de requerimientos

2.1.- Documentación de los requisitos de la aplicación

El objetivo principal de este *software* es compartir información. El usuario podrá añadir información o recibirla. La información a compartir hace referencia a tiendas o lugares y a objetos. Éstos pueden estar relacionados con una tienda o únicamente con una posición GPS.

- La aplicación debe tener compatibilidad con el mayor número de dispositivos, como mínimo Iphone/Ipad, Android y Blackberry.
- Se podrá valorar los establecimientos y añadir la siguiente información: nombre, dirección, descripción, posición GPS y valoración.
- Se podrán valorar objetos. Para cada objeto se podrá añadir la siguiente información: nombre, descripción, foto, relacionarlo con un establecimiento, precio inicial, precio pagado, categoría.
- Las categorías de los objetos estarán cargadas previamente en el sistema, solo tendrán un nivel de profundidad.
- Dentro de un mapa al estilo Google Maps se mostrarán los diferentes puntos con los establecimientos. Al hacer clic en un establecimiento se mostrará la información básica y un enlace hacia el listado de objetos relacionados con dicho establecimiento. En este primer nivel se mostrará el título del elemento, el tipo de elemento y una miniatura de la foto, con un enlace para mostrar la ficha del elemento donde se mostrará toda la información.
- Los establecimientos también serán accesibles desde un listado que mostrará los datos básicos y un acceso al detalle del establecimiento.
- Al usuario únicamente se le mostrarán las tiendas que estén a X kilómetros alrededor del centro del mapa. A medida que navegue se le cargarán nuevos objetos.

- La información que se muestra en el mapa se podrá filtrar según los siguientes criterios:
    - categoría o categorías
    - valoración mínima a mostrar
    - distancia X desde la posición en la que el mapa está centrado

2.2.- Guiones

2.2.1.- Buscar productos

Cuando el usuario abra la aplicación se le mostrará un mapa con los establecimientos que el sistema tiene registrados. Al hacer clic en los iconos de las tiendas se le abrirá una ventana con la información y un enlace para abrir la ficha de la tienda.

La ficha de la tienda muestra la información detallada así como todos los productos relacionados, desde esta pantalla podrá añadir nuevos objetos. A medida que el usuario se vaya moviendo por el mapa se le cargarán nuevos objetos.

2.2.2.- Mostrar listado de establecimientos

Desde la vista principal el usuario podrá acceder a un listado de los establecimientos que el sistema tiene cargados. Para cada establecimiento se mostrará la información básica y se podrá acceder a la ficha detalle del mismo.

2.2.3.- Añadir nuevo establecimiento

Desde la pantalla principal el usuario podrá añadir nuevos establecimientos. Al hacer clic en la opción se le mostrará un formulario con los diferentes atributos que deberá rellenar. Cuando el usuario envíe el formulario se le mostrará un mensaje de confirmación o error según corresponda. En caso de que el proceso se complete correctamente se enviará al usuario a la ficha de la tienda que acaba de añadir.

2.2.4.- Añadir nuevo producto

Esta opción estará disponible únicamente desde la ficha de la tienda. Se le pedirá al

usuario que rellene los diferentes atributos del producto a introducir y se le dará la opción de añadir una imagen. Al enviar el formulario se le mostrará un mensaje de confirmación o error según corresponda, en caso de que el proceso se complete correctamente se enviará al usuario a la ficha de la tienda.

2.2.5.- Filtrar establecimientos

Desde la vista principal del aplicativo el usuario podrá acceder a un formulario para filtrar las tiendas que se le están mostrando en el mapa. El usuario podrá filtrar por los siguientes criterios:

- precio final (varios intervalos)
- categoría o categorías
- valoración mínima a mostrar
- distancia X desde la posición en la que el mapa está centrado

2.3.- Casos de uso

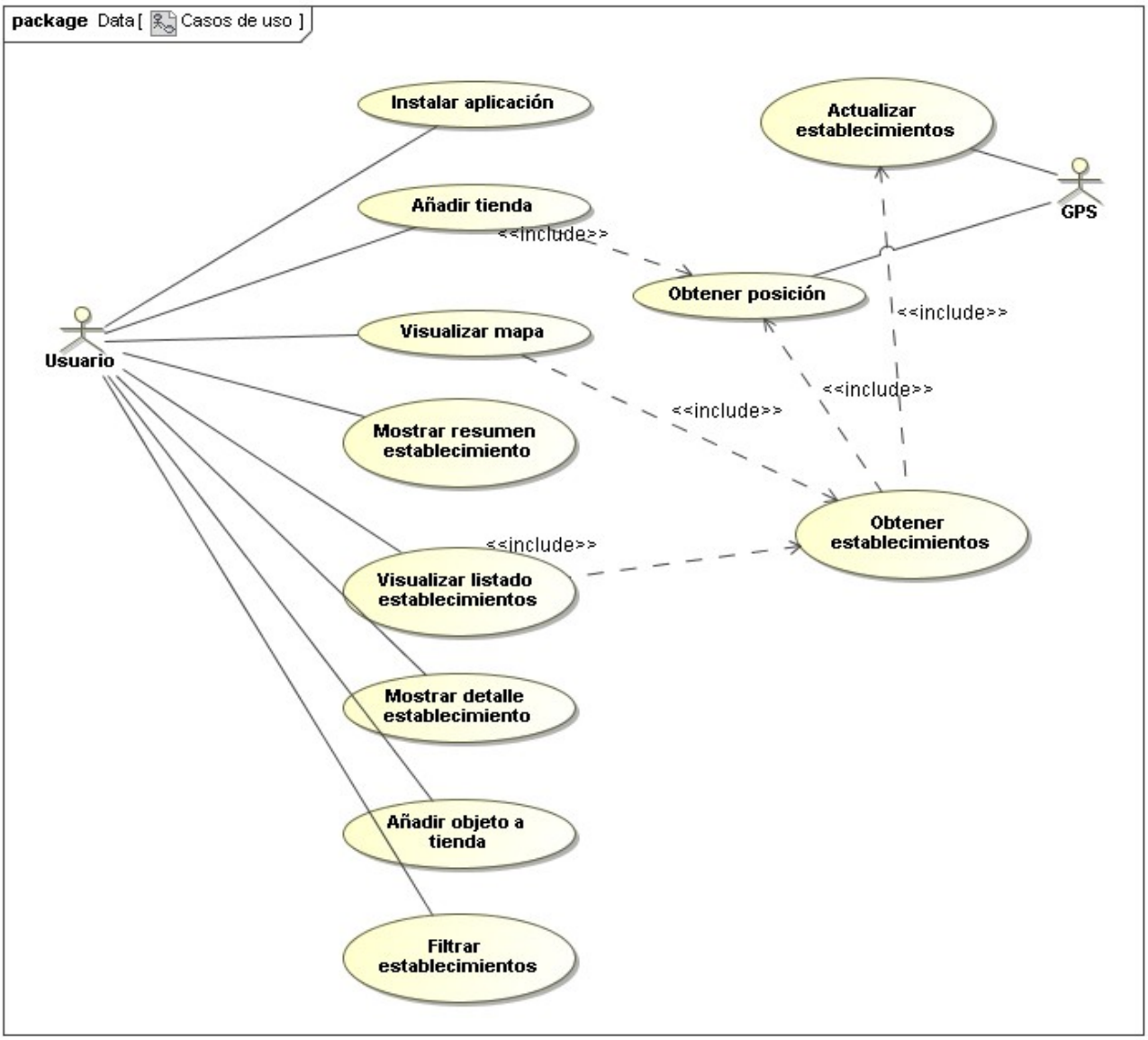


Ilustración 2: Diagrama casos de uso

Únicamente tenemos dos actores, el usuario y el GPS. Se ha considerado al GPS como actor ya que participa en los procesos de añadir nuevos establecimientos, obtención del listado inicial de establecimientos a mostrar y actualización de los elementos mostrados en el mapa si el centro del mapa ha variado X kilómetros respecto a la última actualización.

2.4.- Especificación detallada casos de uso

2.4.1.- Instalar aplicación

Resumen de la funcionalidad: El usuario instala la aplicación en su teléfono móvil. Puede que el usuario esté buscando la aplicación en el repositorio de aplicaciones o simplemente navegado por las diferentes opciones del repositorio la encuentre por casualidad.

Actores: Usuario

Casos de uso relacionados: ninguno

Flujo de acontecimientos principal:

1. El usuario accede al repositorio o 'market' donde la aplicación está disponible
2. El usuario busca la aplicación por su nombre
3. La aplicación pide los permisos para poder acceder a determinadas partes del sistema cliente
4. El usuario acepta
5. La aplicación se instala correctamente

Flujo de acontecimientos alternativos:

1. El usuario accede al repositorio o 'market' donde la aplicación está disponible
2. El usuario navega por las diferentes categorías del market y encuentra la aplicación
3. La aplicación pide los permisos para poder acceder a determinadas partes del sistema cliente
4. El usuario acepta
5. La aplicación se instala correctamente

2.4.2.- Añadir tienda

Resumen de la funcionalidad: El usuario podrá añadir nuevos establecimientos al sistema.

Actores: Usuario, GPS

Casos de uso relacionados: Obtener posición.

Flujo de acontecimientos principal:

1. El usuario hace clic en el botón añadir nuevo establecimiento
2. El sistema muestra un formulario con todos los atributos del establecimiento que el usuario debe rellenar
3. El sistema valida que los campos introducidos son válidos mostrando un error si algún campo ha sido rellenado incorrectamente o no está rellenado
4. El sistema envía el formulario al servidor y muestra un mensaje de error o confirmación al usuario, en caso que el proceso haya finalizado correctamente se muestra al usuario el detalle del establecimiento que ha introducido

2.4.3.- Visualizar mapa

Resumen de la funcionalidad: Cuando el usuario inicia la aplicación, la primera pantalla que se muestra es el mapa centrado en la posición GPS del usuario con los establecimientos que el sistema tiene registrados a X kilómetros a la redonda.

Actores: Usuario, GPS

Casos de uso relacionados: obtener posición, obtener establecimientos

Flujo de acontecimientos principal:

1. El usuario inicia la aplicación
2. El sistema solicita la posición GPS actual
3. El sistema solicita al servidor los establecimientos cercanos a la posición encontrada en el paso 2 y se los muestra al usuario

Flujo de acontecimientos alternativos

1. El usuario vuelve a la pantalla del mapa desde una pantalla secundaria
2. El sistema muestra los establecimientos que tiene cargados de solicitudes anteriores

2.4.4.- Mostrar resumen del establecimiento

Resumen de la funcionalidad: En la pantalla donde se muestran los establecimientos el usuario puede abrir y cerrar el resumen de los establecimientos mostrados. En cada resumen se muestra el nombre del establecimiento, la descripción, la cantidad de elementos asociados y un enlace para acceder a la pantalla con la información detallada.

Actores: Usuario

Casos de uso relacionados: ninguno

Flujo de acontecimientos principal:

1. El usuario hace clic en un establecimiento
2. Se abre una vista con el resumen del establecimiento

2.4.5.- Visualizar listado establecimientos

Resumen de la funcionalidad: desde el menú de la aplicación se puede acceder a un listado de los establecimientos. Se mostrarán los mismos que en el mapa pero en un formato que puede resultar más cómodo de usar en determinadas circunstancias.

Actores: Usuario

Casos de uso relacionados: obtener posición, obtener establecimientos

Flujo de acontecimientos principal:

1. El usuario hace clic en el botón para mostrar el listado de establecimientos
2. El sistema muestra los mismos establecimientos que están cargados en el mapa

2.4.6.- Mostrar detalle establecimiento

Resumen de la funcionalidad: desde el mapa o desde la lista de establecimientos se puede acceder a la vista que muestra el detalle del establecimiento. Se muestra toda la información así como el listado de objetos relacionados con el establecimiento.

Actores: Usuario

Casos de uso relacionados: ninguno

Flujo de acontecimientos principal:

1. El usuario hace clic en un enlace que lleva al detalle del establecimiento
2. El sistema muestra muestra toda la información así como el listado de objetos relacionados con el establecimiento

2.4.7.- Añadir objeto a tienda

Resumen de la funcionalidad: desde la ficha del detalle de un establecimiento el usuario podrá añadir nuevos objetos al establecimiento.

Actores: Usuario

Casos de uso relacionados: mostrar detalle establecimiento

Flujo de acontecimientos principal:

1. El usuario hace clic en el botón 'añadir nuevo objeto'
2. El sistema muestra un formulario con todos los atributos del objeto que el usuario debe rellenar
3. El sistema valida que los campos introducidos son válidos mostrando un error si algún campo ha sido rellenado incorrectamente o no está rellenado
4. El sistema envía el formulario al servidor y muestra un mensaje de error o confirmación al usuario, en caso que el proceso haya finalizado correctamente se muestra al usuario el detalle del establecimiento que ha introducido

2.4.8.- Filtrar establecimientos

Resumen de la funcionalidad: es posible que en determinadas zonas la cantidad de establecimientos que se muestran sean excesivos o que el usuario esté buscando algún tipo de producto en concreto.

Actores: Usuario

Casos de uso relacionados: ninguno

Flujo de acontecimientos principal:

1. El usuario hace clic en el botón filtrar
2. Rellena el formulario con los criterios que quiere aplicar a la búsqueda
3. El sistema muestra la vista del mapa con los establecimientos que cumplen los criterios solicitados por el usuario.

2.4.9.- Obtener establecimientos

Resumen de la funcionalidad: este caso de uso puede ser llamado desde los casos de uso 'Visualizar listado establecimientos' o desde 'Visualizar mapa'. Pide al servidor los establecimientos cercanos a la posición actual.

Actores: GPS

Casos de uso relacionados: 'Visualizar listado establecimientos', 'Visualizar mapa'

Flujo de acontecimientos principal:

1. El aplicativo solicita al dispositivo el listado de establecimientos cercanos

2.4.10.- Obtener posición

Resumen de la funcionalidad: este caso de uso puede ser utilizado por el caso de uso 'Obtener establecimientos' o 'Añadir tienda', devuelve la posición GPS actual del dispositivo.

Actores: GPS

Casos de uso relacionados: 'Obtener establecimientos', 'Añadir tienda'

Flujo de acontecimientos principal:

1. El aplicativo solicita al dispositivo la posición GPS actual

2.4.11.- Actualizar establecimientos

Resumen de la funcionalidad: inicialmente el sistema carga los establecimientos que estén a X kilómetros a la redonda de la posición actual, pero el usuario puede moverse físicamente o mover el centro del mapa para buscar otros establecimientos. En estos casos el aplicativo tendrá que cargar nuevos establecimientos cuya posición se ajuste al nuevo centro del mapa que el usuario está visualizando.

Actores:GPS

Casos de uso relacionados: visualizar mapa

Flujo de acontecimientos principal:

1. El usuario mueve en centro del mapa, ya sea porque físicamente se mueve o porque haciendo clic mueve el mapa
2. El sistema calcula que por la nueva posición debe solicitar al servidor un nuevo listado de establecimientos

3.- Diseño interficie gráfica

El diseño inicial de la interficie gráfica se hará mediante *wireframes* usando la versión de prueba de la herramienta http://wireframesketcher.com.

Para identificar las pantallas necesarias en el aplicativo desarrollaremos las historias de usuario y los casos de uso que se pueden dar según los requisitos recogidos en capítulos anteriores.

3.1.- Wireframes

El aplicativo se compondrá de las siguientes pantallas:

- Pantalla inicial que muestra el mapa de Google con las opciones para añadir nuevas tiendas y filtrar los resultados
- Listado de tiendas
- Detalle de una tienda con todos los elementos asociados a la tienda
- Formulario para filtrar los establecimientos encontrados
- Formulario para añadir un nuevo establecimiento
- Formulario para añadir un nuevo producto al establecimiento

Pantalla inicial

Ilustración 3: Wireframe. pantalla principal

Se muestra el mapa centrado en la posición actual capturada desde el GPS del dispositivo. En caso que no se pueda obtener esta posición por defecto se pondrá una localización de Barcelona.

Desde esta primera pantalla se podrá añadir una nueva tienda, acceder al listado de tiendas, acceder al detalle de un establecimiento o realizar una búsqueda.

Listado de tiendas

Ilustración 4: Wireframe, listado de tiendas

En esta pantalla se mostrará el listado de tiendas que el sistema tiene disponibles, con información básica como el nombre del establecimiento o su dirección.

Detalle de establecimiento

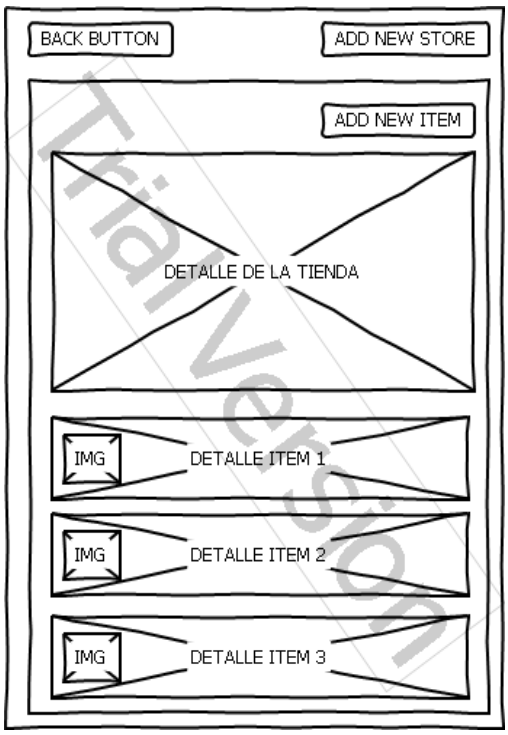


Ilustración 5: Wireframe, detalle tienda

A esta pantalla se podrá acceder o bien haciendo clic en los iconos del mapa inicial o haciendo clic en el listado de establecimientos. Se mostrará toda la información del establecimiento: nombre, dirección, descripción, valoración media y el listado de objetos asociados. Para cada objeto se mostrará también toda su información.

Formulario de búsqueda

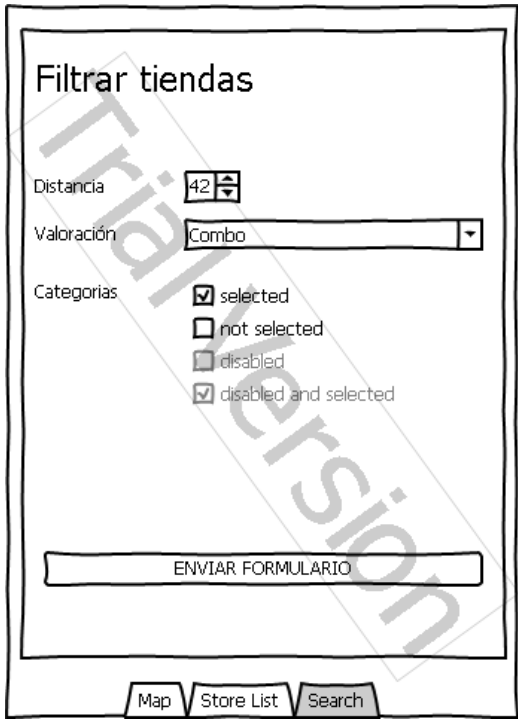


Ilustración 6: Wireframe, formulario búsqueda

A esta pantalla se accede desde el menú inferior. Se podrá filtrar la información listada ajustándola a los siguientes parámetros:

- Distancia del centro del mapa
- Valoración mínima del establecimientos
- Categorías de los productos
- Precio mínimo
- Precio máximo

Una vez aplicados los filtros, si un establecimiento no contiene ningún producto no se mostrará en el mapa principal ni en el listado.

Formulario para añadir nuevo establecimiento

Ilustración 7: Wireframe, formulario nueva tienda

En este formulario se le pedirá al usuario que introduzca los datos de la tienda.

Formulario para añadir nuevo producto al establecimiento

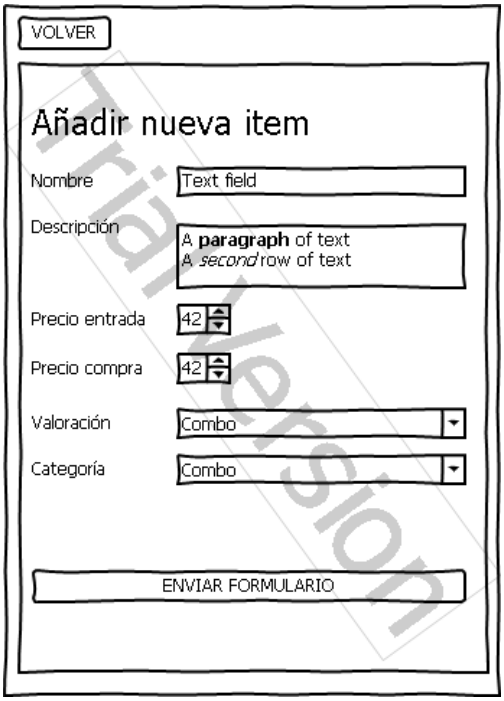


Ilustración 8: Wireframe, formulario nuevo objeto

Se le pedirá al usuario que introduzca los datos del producto y se le dará la opción de añadir una fotografía.

2.4.- Selección del framework / entorno de trabajo

Para el desarrollo de esta aplicación se valorarán los siguientes *frameworks*:

- WAC, http://www.wacapps.net/
- PhoneGap, http://phonegap.com/
- Titanium, http://www.appcelerator.com/
- OpenMeap, http://www.openmeap.com/

La valoración se hará en base a los siguientes puntos:

- Compatibilidad entre dispositivos móviles
- Velocidad del programa generado
- Comunidad *online*
- Documentación
- Confianza página web / documentación aportada
- Coste licencia
- Complejidad para desarrollar
- Movimiento corporativo (código al día y mantenido)

	WAC	PhoneGap	Titanium	OpenMeap
Compatibilidad	?	android ios blackberry windows	android ios blackberry (beta)	android ios
Velocidad	No nativo	No nativo	Código nativo	No nativo
Comunidad	Foro prácticamente vacío	Foro con actividad	Foro con actividad	Foro prácticamente vacío
Documentación	Documentación media	Buena documentación	Buena documentación	Escasa documentación
Confianza	Muy baja	Tanto la web como la documentación inspiran seriedad	Tanto la web como la documentación inspiran seriedad	baja
Coste	openSource	openSource	openSource	openSource
Complejidad	?	Html5 + javascript	javascript	Html5 + javascript
Movimiento	No se ha encontrado ni el descargable	Desarrollo muy activo, versiones nuevas cada varios meses	Desarrollo muy activo, varias versiones al año	Versión alpha

Después de un análisis rápido las opciones de WAC y OpenMeap han quedado descartadas por estar en un estado de madurez o de mantenimiento que no era aceptable. Se ha hecho un estudio más en profundidad de las otras dos opciones, PhoneGap y Titanium.

Ambas soluciones tienen una comunidad muy activa, como así queda demostrado en los foros y wikis de sus respectivas páginas web. Visitando los repositorios de código en gitHub se puede ver que el desarrollo es activo y se van liberando versiones nuevas cada poco tiempo.

La principal diferencia entre PhoneGap y Titanium es que la primera no genera código nativo, mientras que las aplicaciones desarrolladas en Titanium si que son convertidas a código nativo por lo que el *look & feel* será el nativo del dispositivo. De esta manera los usuarios podrán tener una experiencia de usuario más natural y la aplicación podría ser más rápida.

A pesar de esto el *framework* de trabajo escogido ha sido PhoneGap v1.4, ya que para esta aplicación no se cree necesario hacer un uso intensivo de los recursos del sistema. Otro punto a favor de PhoneGap es que el tiempo de desarrollo será limitado y resulta más sencillo desarrollar una interficie en HTML5 + Javascript que únicamente Javascript generando elementos nativos, y la aplicación creada con PhoneGap podría incluso servirse desde servidores web para ser usada desde Internet con un navegador web corriente.

Para facilitar el desarrollo de la aplicación usaremos un *framework* javascript que facilitará la tarea de crear la GUI, Sencha Touch v2.0 http://www.sencha.com/products/touch/. Sencha Touch en su última versión dispone de la funcionalidad para generar ejecutables empaquetados del mismo estilo que PhoneGap, pero esta funcionalidad se encuentra en estado Beta y solo da soporte a Android, por lo que se continuará usando PhoneGap como capa para generar los instalables para las diferentes plataformas.

El uso de Sencha Touch también nos permite aplicar el patrón MVC para generar un código más claro y fácil de mantener. PhoneGap como *framework* no tiene (o no está claramente documentado) la estructura para desarrollar bajo MVC, y si se quisiera aplicar este patrón se tendrían que desarrollar todas las clases auxiliares desde cero, mientras que Sencha Touch si que lo soporta de forma nativa teniendo únicamente que heredar y aplicar las clases y procedimientos que se especifican en su documentación.

El desarrollo principal se hará para la plataforma Android (versiones desde 2.3.3 hasta 3.3) ya que para desarrollar para IOs es necesaria una licencia de su SDK y usar el sistema operativo Mac OS. El IDE de desarrollo será Eclipse.

3.2.- Diseño de prototipo

Dado que en las aplicaciones móviles un punto crítico es la usabilidad se ha decidido desarrollar un prototipo de alta fidelidad usando las mismas herramientas que el producto final, PhoneGap y Sencha Touch, con esto conseguiremos que las pruebas con usuarios sean mucho más precisas y útiles.

Ilustración 10: Pantalla principal con el listado de establecimientos

Ilustración 9: Pantalla principal mostrando información de un establecimiento

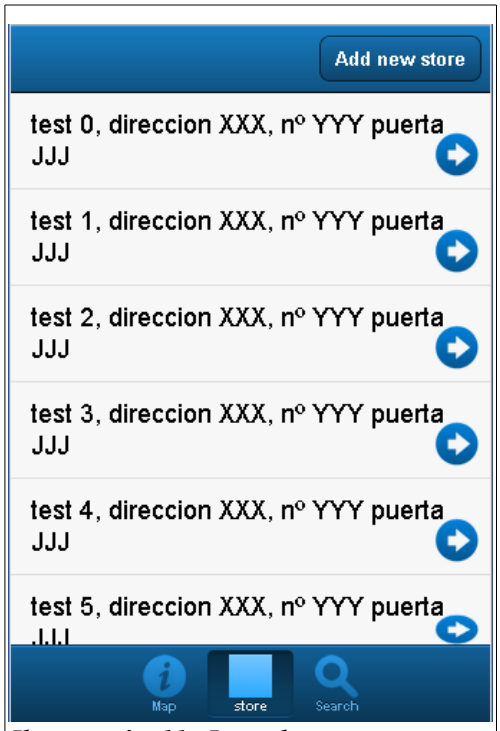


Ilustración 11: Listado establecimientos

Ilustración 12: Formulario para añadir nuevo establecimiento

3.3.- Test prototipo

Para realizar las pruebas se usarán dos dispositivos diferentes, un Samsumg Galaxy SII con pantalla de 4,3 pulgadas y un Samsung Galaxy Mini, con pantalla de 3,14 pulgadas.

Se han seleccionado dos perfiles de usuario diferentes, uno habituado a usar aplicaciones móviles y otro que aunque usa el móvil a diario no está muy acostumbrado a usar aplicaciones.

Las historias de usuarios que se probarán son las siguientes:

1. El usuario debe buscar establecimientos que estén cerca de su posición actual y llegar al detalle del producto para poder valorar si algún objeto le interesa
2. El usuario debe buscar establecimientos que estén en otra zona para encontrar nuevas zonas comerciales dentro de su misma ciudad
3. El usuario debe añadir un producto que acaba de adquirir

La prueba se realizará en una céntrica zona comercial de Barcelona para intentar reproducir una zona de uso real. Al usuario se le explicará brevemente la finalidad de la aplicación y los objetivos que tiene que alcanzar.

Usuario con experiencia

Prueba 1:

1. El usuario se mueve por el mapa y hace clic en algunos iconos de tiendas
2. Abre otro icono de tienda y hace clic en el enlace para acceder al detalle del producto

Prueba 2:

1. El usuario se mueve por el mapa
2. Intenta hacer zoom usando la acción de mover dos dedos pero no obtiene resultados

3. Usa los iconos +- del mapa de Google para hacer zoom out del mapa
4. Localiza otros establecimientos que están lejos de su posición actual y accede a uno de ellos

Prueba 3:

1. El usuario hace clic en el icono para añadir nueva tienda
2. Rellena el formulario
3. Accede a la página de detalle de la tienda
4. Hace clic en el icono de añadir nuevo producto
5. Rellena el formulario

Usuario novel

Prueba 1:

1. El usuario se mueve por el mapa pero no parece ver los iconos de los establecimientos
2. Al cabo de pocos segundos hace clic en algunos iconos de tiendas
3. Abre otro icono de tienda y hace clic en el enlace para acceder al detalle de producto

Prueba 2:

1. El usuario se mueve por el mapa haciendo clic y arrastrando el centro del mapa
2. Localiza otros establecimientos que están lejos de su posición actual y accede a uno de ellos

Prueba 3:

1. El usuario busca el botón para añadir un nuevo objeto pero no lo encuentra
2. Accede al detalle de un establecimiento y ve el botón de añadir objeto a ese

establecimiento

3. Vuelve al mapa principal y hace clic en el icono para añadir nueva tienda
4. Introduce los datos del producto en el formulario de la tienda
5. Envía el formulario pensando que ha introducido el producto

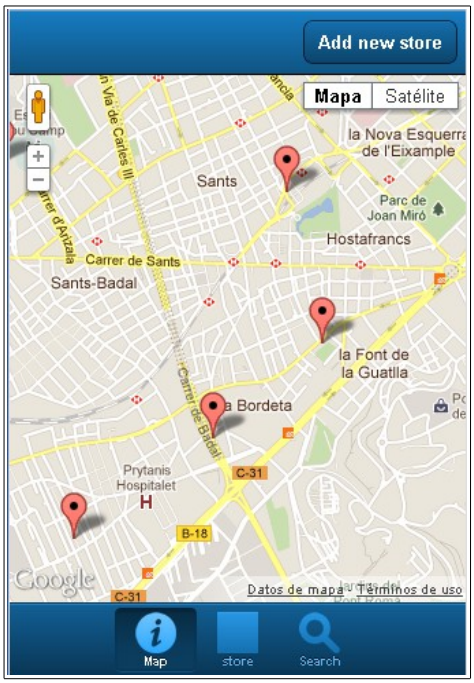
Conclusiones y comentarios de los usuarios

Después de las pruebas con los usuarios se han propuesto un conjunto de mejoras:

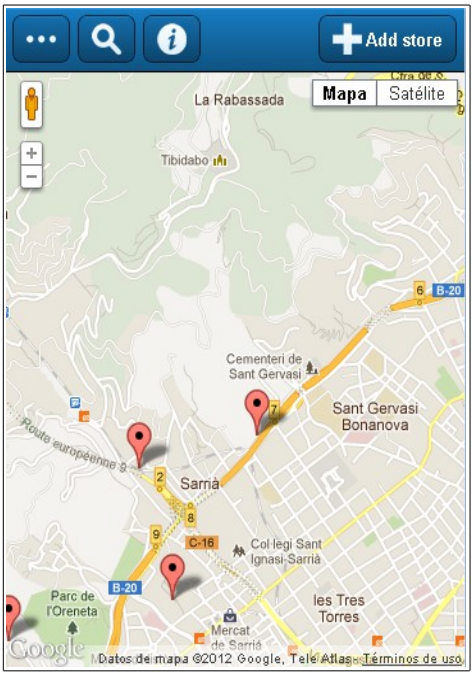
- Mostrar distancia de los puntos respecto a la posición actual
- Hacer opcionales algunos campos en los formularios para añadir establecimientos y productos
- Modificar la disposición de botones inferior para disponer de más espacio para el mapa ya que en dispositivos con la pantalla pequeña el área de visualización del mapa se queda un poco pequeño
- Añadir icono de ayuda en el menú general con explicaciones básicas
- Añadir la opción 'Como llegar' a un establecimiento que mostrará el camino desde la posición actual hasta el establecimiento indicado
- La navegación en el mapa no es del todo fluida, en ocasiones al moverse parece que el mapa no reacciona o tarda bastante en hacerlo
- Dificultad para añadir objetos nuevos si el establecimiento no está creado, demasiados pasos y los usuarios pueden perderse en el proceso
- Puede ser que los iconos de los establecimientos no sean del todo visibles

Diferencia entre prototipo inicial y prototipo revisado

Versión inicial:



Versión revisada:



El cambio más importante introducido es la eliminación de la barra de botones inferior, con este cambio se busca disponer de más espacio para la zona central de la aplicación.

El resto de cambios introducidos son nuevas funcionalidades necesarias para que los usuarios puedan usar la aplicación de forma más sencilla o mejoras en la usabilidad en algunos puntos. El principal cambio se ha hecho en el proceso para añadir un producto si el establecimiento no existía.

Nuevos guiones

Cómo llegar

Desde la pantalla principal que muestra la información básica de un establecimiento o desde la ficha detalle del mismo se podrá acceder a una nueva funcionalidad para que se muestre la ruta hacia el establecimiento.

Ayuda

Desde el menú principal se añadirá un nuevo icono 'Ayuda' que mostrará un pequeño tutorial para ayudar al usuario con el funcionamiento del aplicativo.

Guiones modificados

Buscar productos

Cuando el usuario acceda por primera vez al mapa principal automáticamente se mostrará el detalle del establecimiento más cercano. Con esto se quiere conseguir que los usuarios no familiarizados con Google Maps relacionen los iconos de los establecimientos con acciones.

Añadir tienda

El proceso será el mismo pero se mostrará un mensaje en la parte superior del formulario indicando que si se quiere añadir un objeto primero se tiene que crear el establecimiento. Si el proceso de creación del establecimiento finaliza correctamente al usuario se le llevará a la pantalla detalle del establecimiento y se le mostrará un mensaje preguntándole si quiere añadir el primer objeto al establecimiento.

Casos de uso definitivos

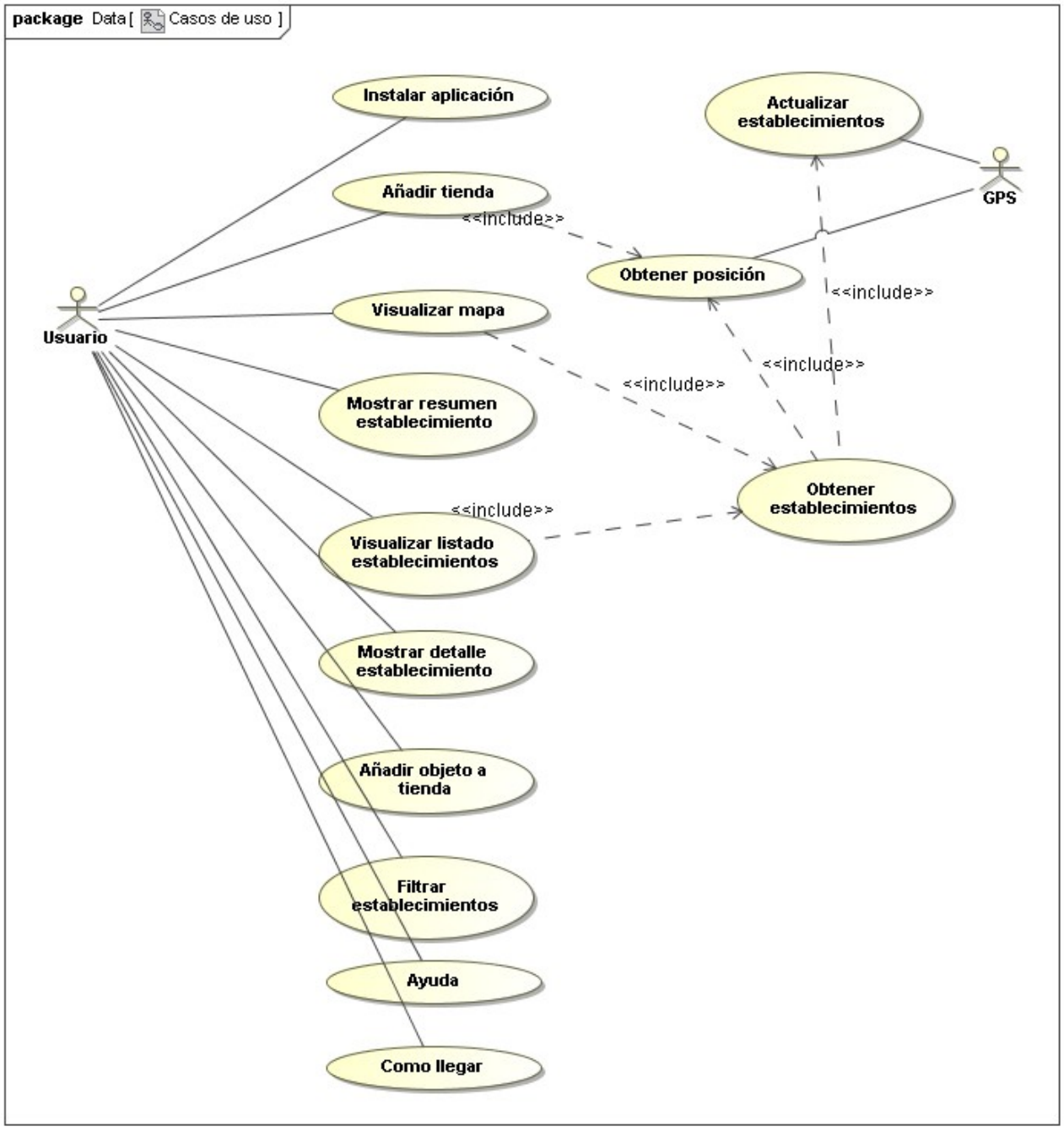


Ilustración 13: Diagrama definitivo casos de uso

4.- Diseño arquitectura

El aplicativo a desarrollar se divide en tres sistemas:

- Sistema cliente instalado en el dispositivo del usuario, es la parte del aplicativo encargada de representar la información e interactuar con el usuario para solicitar nueva información que será añadida al sistema.
- Sistema servidor, es el encargado de leer la información de la base de datos y servirla al sistema cliente, así como de procesar la nueva información y enviarla al sistema de almacenamiento.
- Sistema de almacenamiento, sistema gestor de base de datos, es el sistema donde persistirá la información de la aplicación.

Para los sistema cliente y servidor se usará una arquitectura en tres capas MVC (modelo, vista, controlador) y la comunicación entre los sistemas se hará mediante el protocolo REST/HTTP, la capa de vista del sistema servidor se compondrá de diferentes funciones a modo webservice que devolverá la información en formato JSON y la recibirá como POST para añadir nuevos elementos o GET HTTP cuando se realicen consultas.

Gracias a esta arquitectura conseguiremos lo siguiente:

- Clara separación entre el código que representa la interfaz, con el código que modela el negocio y el código que almacena los datos
- Reutilización de los componentes, al tener la capa de vista separada de la capa de negocio se pueden crear diferentes vistas adaptadas a los dispositivos sin modificar la capa de controlador
- Facilidad en el mantenimiento del sistema, ya que al tener el código separado es más fácil de entender
- El sistema cliente y servidor están totalmente desacoplados gracias a un formato de intercambio de mensajes estándar, se podría re implementar el sistema servidor en un lenguaje de programación totalmente distinto sin que el sistema cliente se vea afectado

El siguiente diagrama ilustra la arquitectura que se quiere implementar:

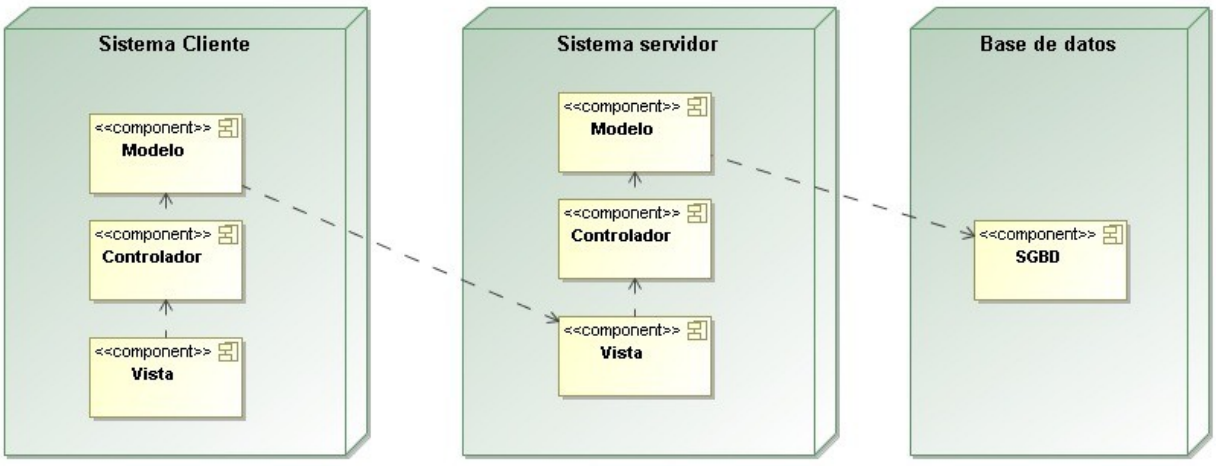


Ilustración 14: Arquitectura aplicación

Sistema cliente:

- PhoneGap 1.4.1
- MVC con Sencha Touch 2.0.1

Sistema servidor:

- MVC con Symphony2, PHP 5.3
- Doctrine 2.2 como capa de abstracción con la base de datos

Base de datos:

- MySQL 5,1

Esta arquitectura puede parecer sobredimensionada para un aplicativo que implemente los requisitos iniciales pero facilitará que pueda escalar y crecer sin problemas si la carga del sistema o las nuevas funcionalidades que se quieran implementar lo requieren.

Gracias a la capa de abstracción entre el sistema servidor y la base de datos, si fuera necesario se podría cambiar el tipo de base de datos por otro y no tendríamos que realizar ningún cambio en el resto del sistema, ya que doctrine es capaz de trabajar con diferentes sistemas gestores de bases de datos de forma transparente al resto de la implementación.

Como añadido a esta arquitectura, y si la carga del sistema fuera elevada, fácilmente se

podrían añadir sistemas adicionales que permitan a la arquitectura soportar más carga de trabajo, como añadir un sistema de cacheado (reserve proxy) entre las peticiones de consulta del sistema cliente y el sistema servidor para que únicamente las peticiones de escritura lleguen al sistema servidores; añadir un CDN para que sirva las imágenes asociadas a los objetos o modificar la capa de abstracción de datos para que pueda usar una base de datos distribuida o con una arquitectura Master-Slave.

El diagrama de los servidores de la aplicación podría ser el siguiente:

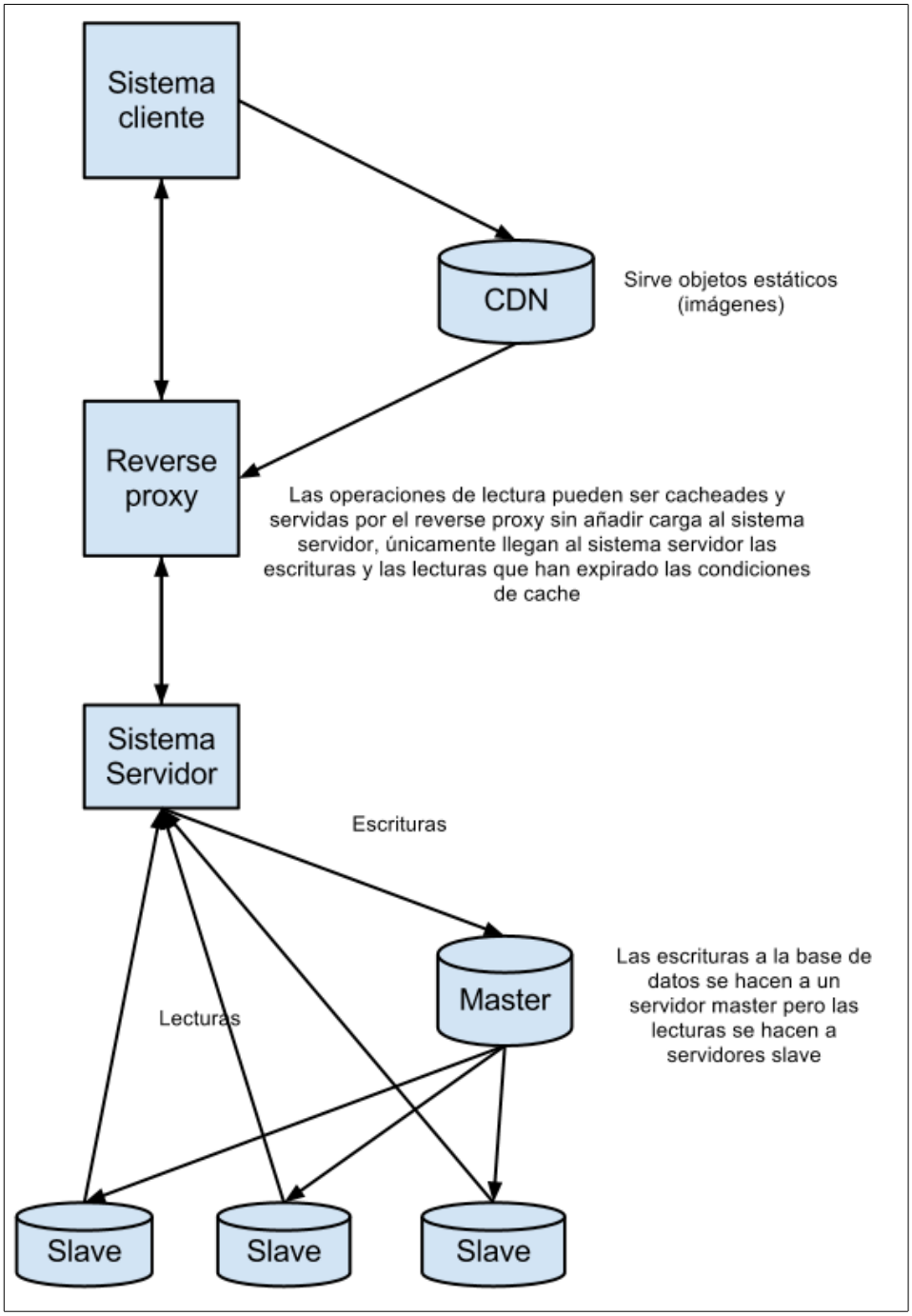


Ilustración 15: Arquitectura sistemas

5.- Diseño software

5.1.- Diagrama de clases

Analizando los requisitos y los casos de uso se identifican tres clases:

- Store
- Item
- Category

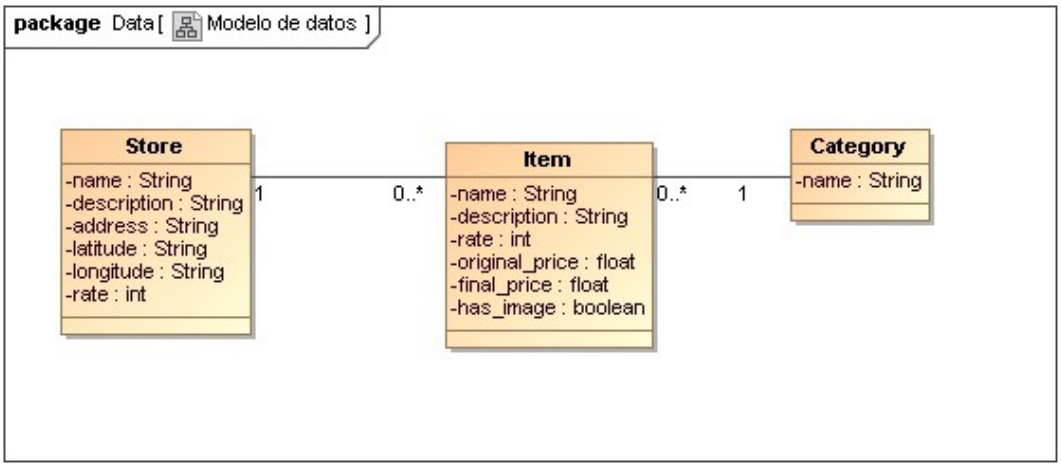


Ilustración 16: Diagrama de clases

La entidad Store representa los establecimientos, una store puede tener cero o varios items relacionados y un item siempre tiene que estar asociado a una Store. Los items siempre están relacionados con una categoría. Este diagrama de clases es común para el sistema servidor y para el sistema cliente.

5.2.- Patrones

Los patrones de diseño software que hemos usado son los siguientes:

- **MVC**, modelo vista controlador. La principal aportación de este patrón de diseño es la clara separación entre la capa de datos, presentación y lógica de la aplicación. Se usará tanto para desarrollar la parte cliente como la servidor.
- **Observador**, la mayor parte de acciones que se realizan en la capa de vista lanzan eventos que la capa de controlador captura y reacciona.
- **Factory**, centraliza en una clase la gestión de cada entidad de negocio desde la que se podrán realizar las acciones para crear, editar, borrar y actualizar.
- **Singleton** o instancia única, este patrón de diseño es usado por Sencha touch para compartir datos entre diferentes clases.
- **Decorator,** añadimos dinámicamente funcionalidades a una clase ya existente, esto es algo muy común en los aplicativos desarrollados en lenguaje javascript.
- **DAO**, data access object. Encapsulamos y abstraemos el acceso a bases de datos u otros mecanismos de persistencia.

5.3.- Diagrama componentes extendido

El diseño se dividirá en dos partes que representarán el diseño del sistema cliente y el diseño del sistema servidor.

5.3.1.- Sistema cliente

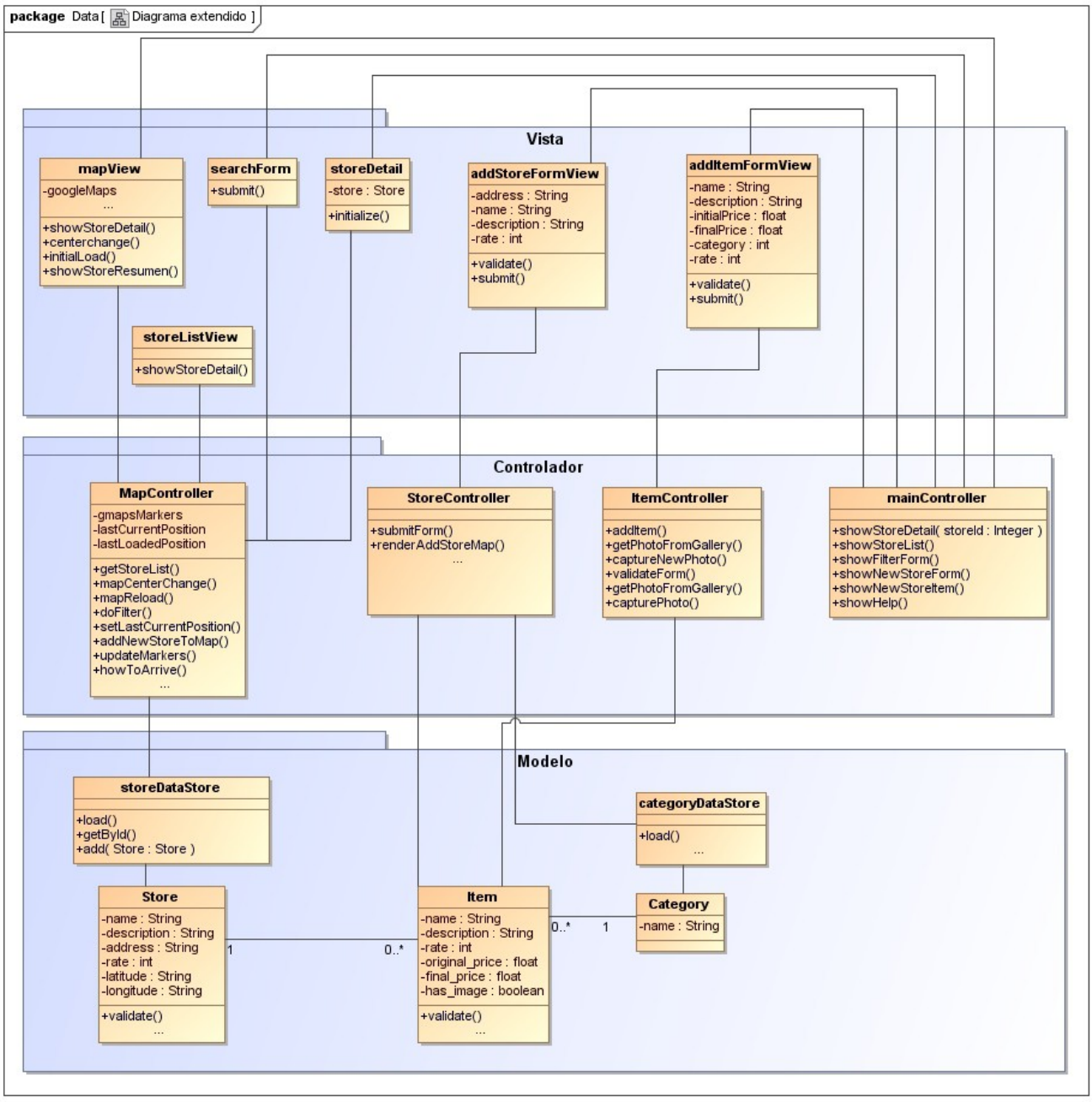


Ilustración 17: Diagrama clases extendido - sistema cliente

En la capa de modelo se han añadido dos clases que aunque no son modelos en si, son las capas de abstracción que tiene Sencha Touch para acceder a los datos usando un proveedor externo o proxy.

Dentro de los controladores de la aplicación se accede a sistemas externos, como a las librerías de Google Maps o a las de PhoneGap. Se definen las siguientes operaciones:

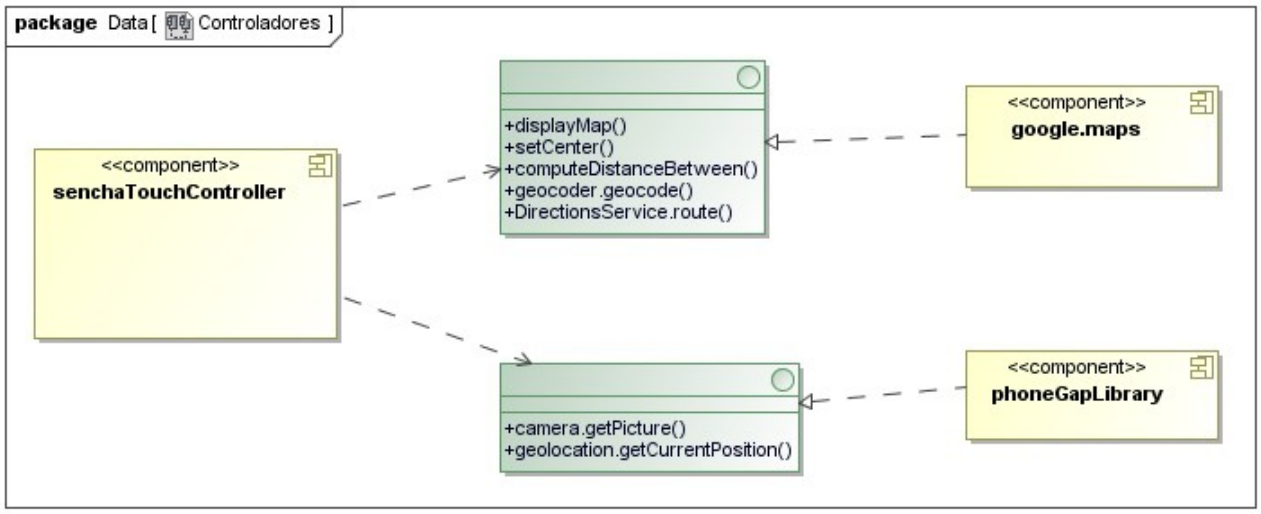


Ilustración 18: Componentes externos

Google.maps:

- DisplayMap y setCenter son las funciones de Google Maps para mostrar el mapa al usuario y centrarlo en una posición en concreto.
- ComputeDistanceBetween, es una función que sirve para calcular la distancia entre dos puntos. Se usará para calcular la distancia entre la posición del usuario y los diferentes establecimientos.
- geocoder.geocode: esta funcionalidad se usará al añadir un nuevo establecimiento, se le pedirá al usuario que sitúe en un mapa el establecimiento y mediante la latitud y la longitud obtendremos la dirección postal.
- DirectionsService.route(): con esta función implementaremos el 'como llegar', trazará una ruta desde la posición del usuario hasta el establecimiento.

PhoneGap:

- camera.getPicture(): se utilizará para acceder a la librería de imágenes del dispositivo móvil y a la cámara para obtener nuevas fotografías.
- geolocation.getCurrentPosition(): esta función devolverá la posición actual del usuario para poder centrar el mapa cuando la aplicación se inicie.

5.3.2.- Sistema servidor

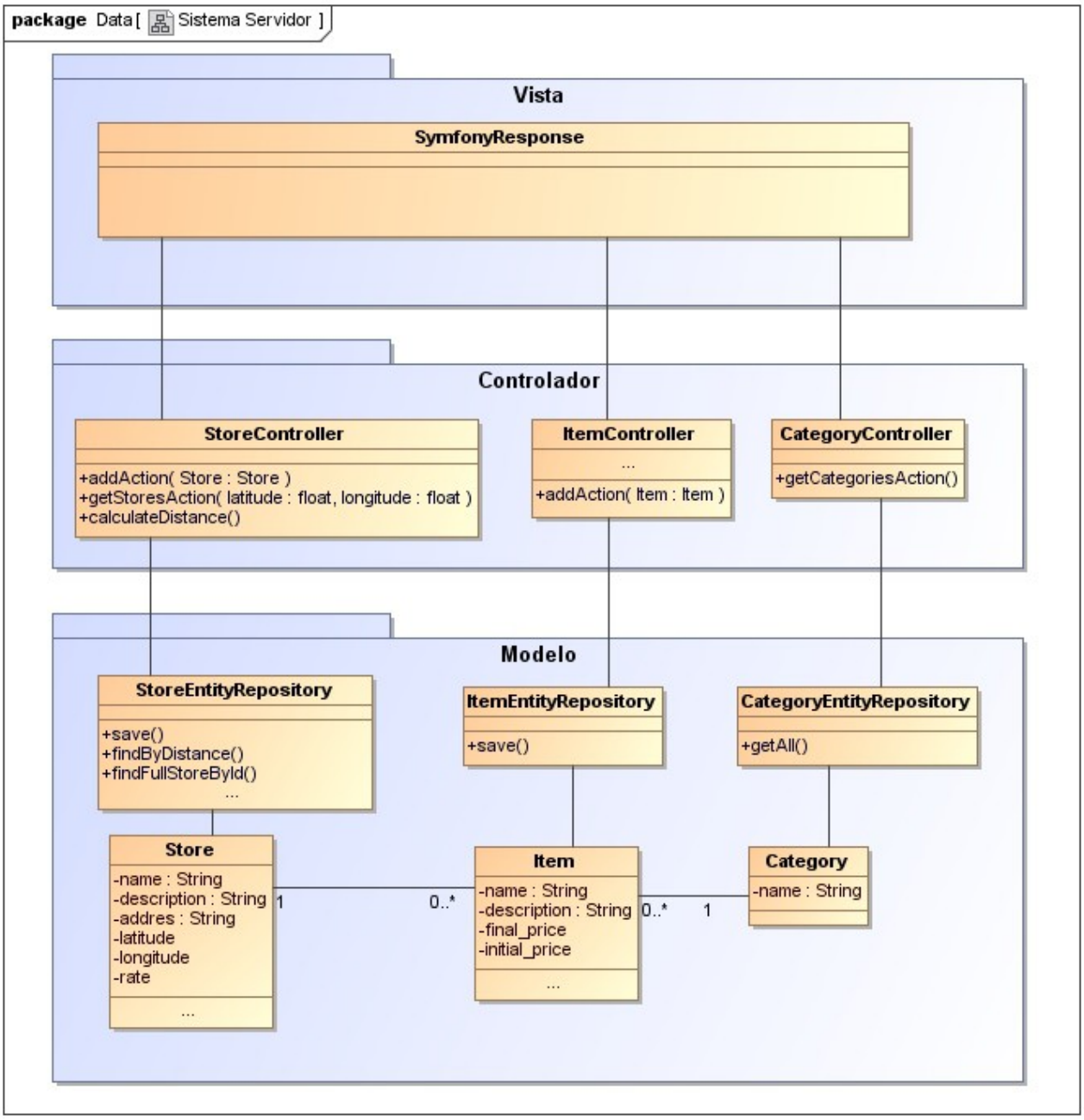


Ilustración 19: Diagrama clases entendido - sistema servidor

El diseño del sistema servidor es mucho más sencillo que el sistema cliente ya que únicamente se encargar de pasarle datos al sistema cliente y recoger lo que éste recibe del usuario y almacenarlo en la base de datos.

El sistema servidor se implementará usando el framework de desarrollo ágil Symfony2

que provee de diversas funcionalidades para agilizar el desarrollo. Al ser una aplicación REST/HTTP la capa de vista es inexistente, únicamente se codifica en formato JSON los datos y se imprimen en pantalla como texto plano.

El controlador Store es quien se encargará de pasarle a la vista todas las stores y los items relacionados a la vista El controlador item únicamente se encargará de añadir nuevos items a la base de datos.

Como capa de abstracción entre la base de datos y el servidor usaremos doctrine, que nos proporciona todas las clases y métodos necesarios para poder operar con la base de datos independientemente del SGBD que usemos.

5.4.- Diagramas de secuencia

A continuación se muestran los diagramas de secuencia de los principales casos de uso analizados en capítulos anteriores.

5.4.1.- Añadir tienda

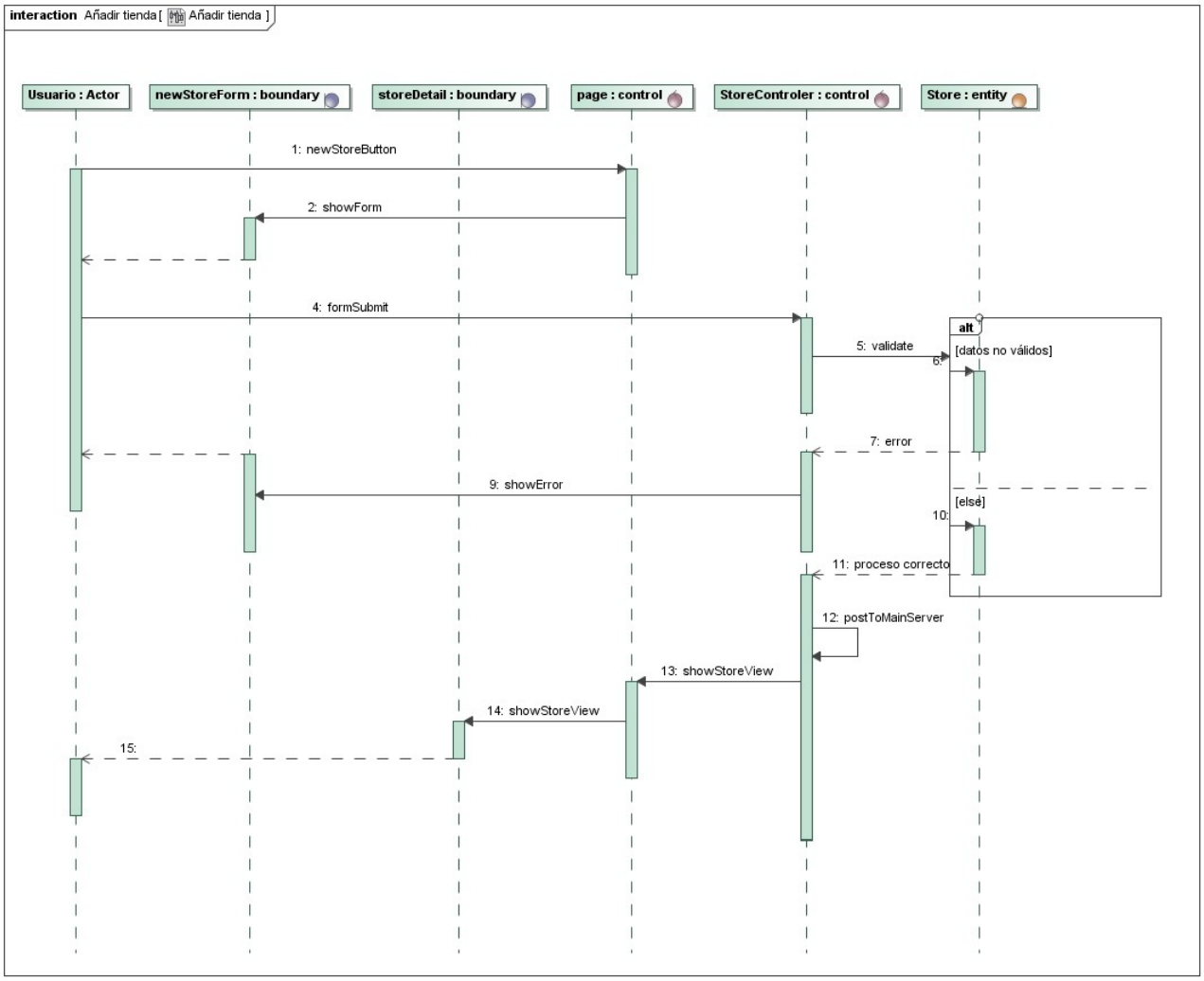


Ilustración 20: diagrama de secuencia, caso de uso añadir tienda

5.4.2.- Visualizar mapa

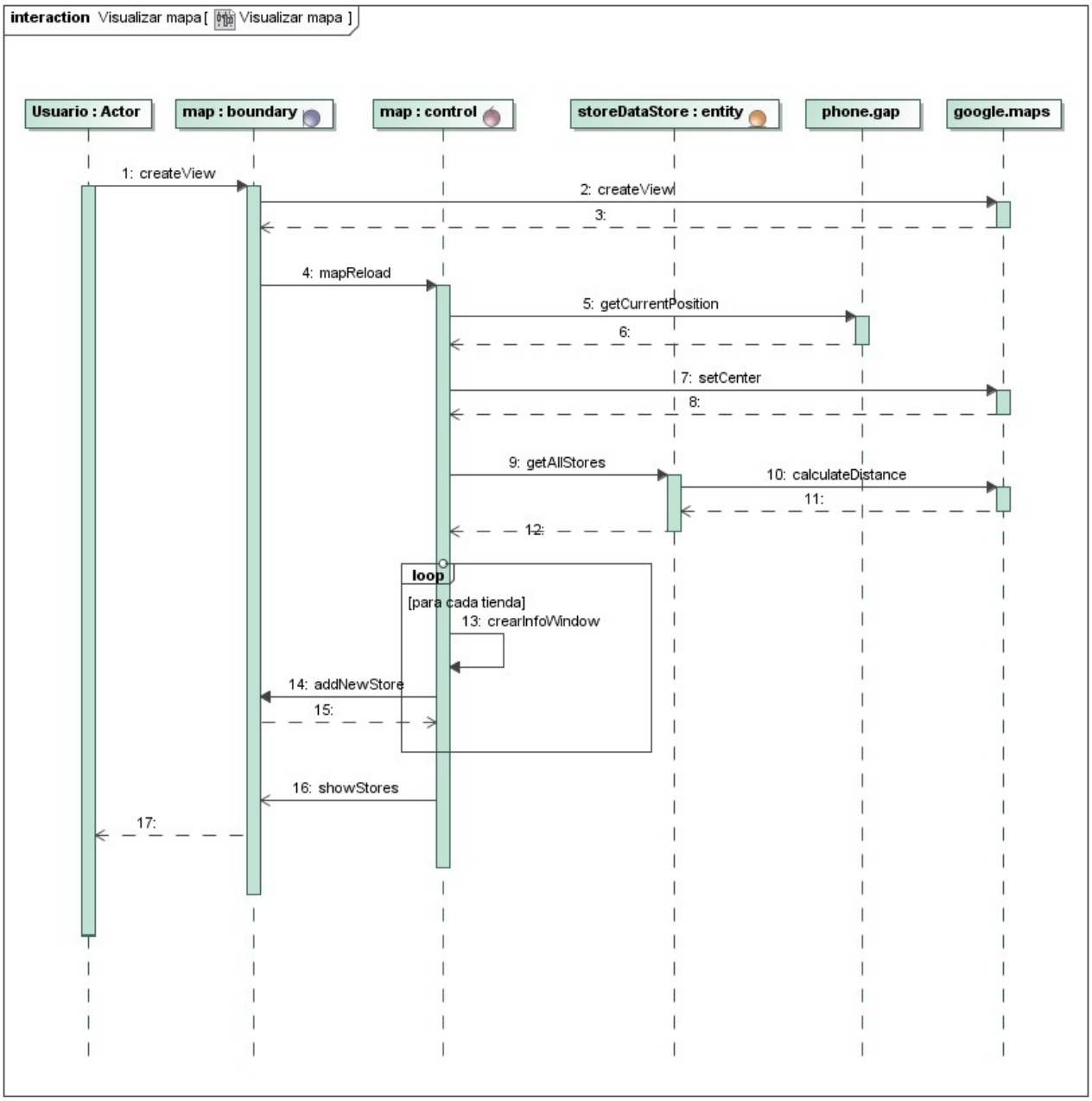


Ilustración 21: diagrama de secuencia, caso de uso visualizar mapa

5.4.3.- Visualizar listado establecimientos

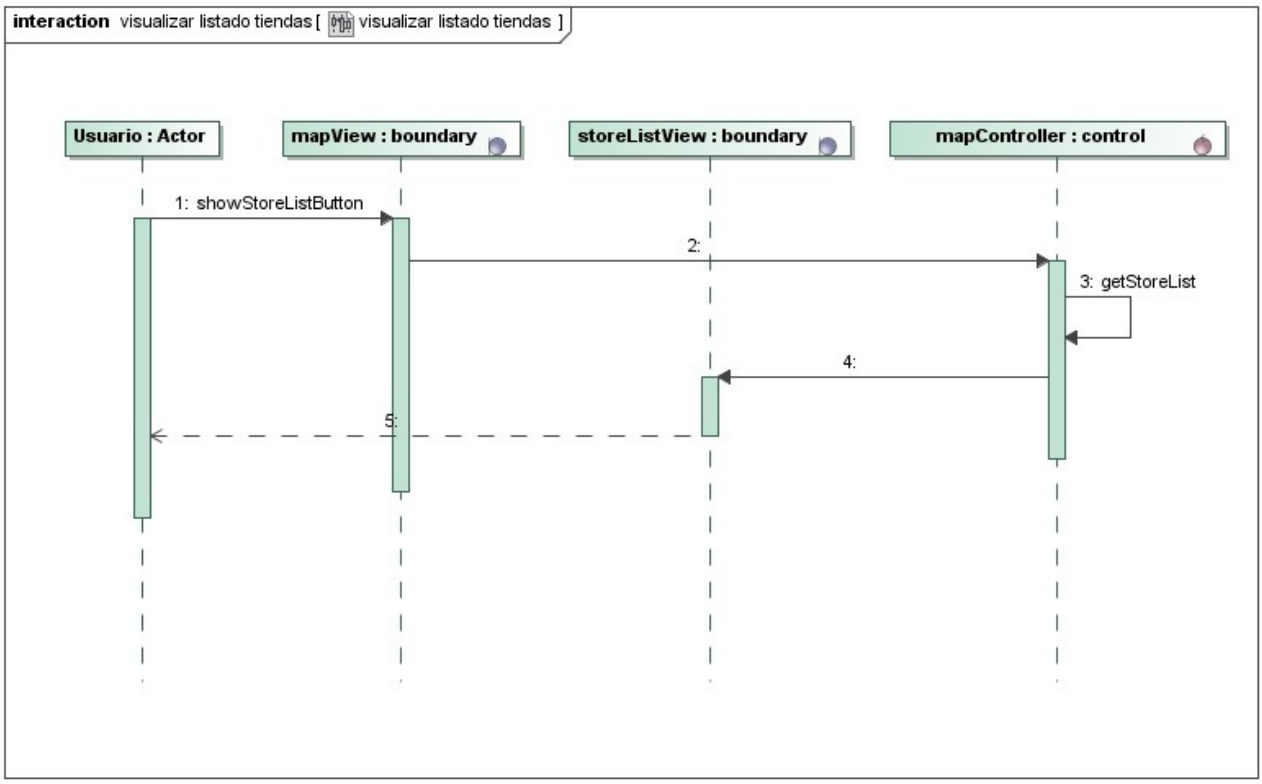


Ilustración 22: diagrama de secuencia, caso de uso visualizar listado establecimientos

5.4.4.- Mostrar detalle establecimiento

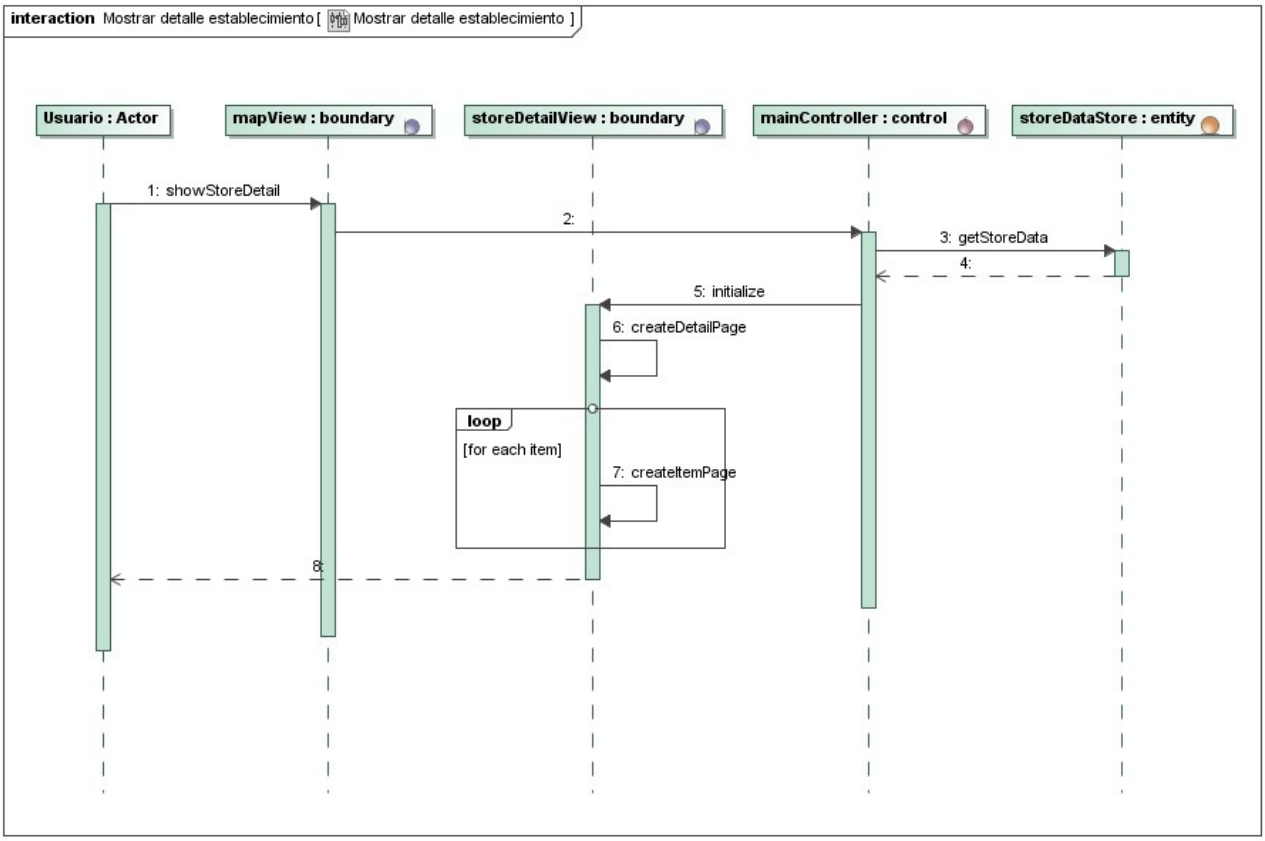


Ilustración 23: diagrama de secuencia, caso de uso mostrar detalle establecimiento

6.- Implementación

6.1.- Aspectos a destacar sistema cliente

PhoneGap y SenchaTouch

La versión de PhoneGap escogida para desarrollar esta aplicación ha sido la 1.4.1 ya que la versión 1.5 (la última versión al inicio del proyecto) daba bastante problemas y ejemplos sencillos de su documentación no funcionaban correctamente. Actualmente la última versión disponible es la 1.7 en la que únicamente se corrigen errores de versiones anteriores pero que no parece aportar ninguna mejora que pueda beneficiar al proyecto.

La versión de SenchaTouch usada es la 2.0.1 que se publicó a finales de abril 2012. Esta versión corrige numerosos fallos detectados en la versión 2.0.0, algunos de ellos encontrados durante el desarrollo de esta aplicación.

Problemas al iniciar la aplicación

PhoneGap se usa en este proyecto para contener la aplicación y para tener acceso a la cámara y el GPS. En las pruebas iniciales realizadas, dependiendo de la versión del sistema Android instalado en el dispositivo, la aplicación se iniciaba correctamente o únicamente mostraba una pantalla en blanco sin mostrar realmente la aplicación,

Este problema es debido a que, dependiendo de la versión del sistema operativo, a la aplicación no le daba tiempo a cargar el index.html que lanza la vista de usuario antes de que el navegador interno diera timeout, por lo que ampliando este tiempo se soluciona el problema:

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  //compatibilidad con Android 2.3.3, sino da timeout
  super.loadUrlTimeoutValue = 600000;
  super.loadUrl("file:///android_asset/www/index.html");

}
```

Reinicio al hacer fotos

En determinadas circunstancias en las que el dispositivo tiene poca memoria RAM disponible al obtener fotografías desde la cámara la aplicación se reinicia. Esto es debido a que al acceder al sistema de la cámara, la aplicación sale del primer plano y se pone en modo segundo plano, entonces se activa la aplicación nativa de la cámara, el dispositivo al necesitar más memoria RAM para sacar la fotografía ejecuta el garbage collector y reinicia el estado de la aplicación PhoneGap.

Para paliar este problema se ha bajado la calidad de las fotos que obtenemos y se ha bajado la versión mínima del SDK de Android a 7, tal y como se propone desde algunos foros.

```
navigator.camera.getPicture(this.onPhotoURISuccess, this.onFail, {
     quality: 25,
     destinationType: destinationType.DATA_URL,
     targetWidth: 256,
     targetHeight: 256,
     sourceType: Camera.PictureSourceType.PHOTOLIBRARY
});
```

```
<uses-sdk android:minSdkVersion="7" />
```

Gracias a estos cambios se ha conseguido paliar el problema aunque no solucionarlo.

Imagen codificada en base64

El API de PhoneGap para acceder a la cámara puede devolver la URL local de la imagen o la imagen codificada en base64 en función de unos parámetros de entrada.

```
 Camera.DestinationType = {
   DATA_URL : 0,                // Return image as base64 encoded string
   FILE_URI : 1                 // Return image file URI
 };
```

En la documentación de PhoneGap recomiendan trabajar con URIs ya que los dispositivos actuales tienen cámaras con una resolución muy alta que generan imágenes de gran tamaño, que al ser convertidas a base64 ocupan todavía más y ésto puede dar

lugar a problemas de memoria. El problema de trabajar con URIs es como subir la imagen al servidor remoto, ya que el envío de los formularios se hace mediante POST ajax, que no permiten subir objetos como imágenes. Se tendría que subir por una parte los datos del objeto y abrir otro tipo de conexión para subir la imagen.

Debido al tipo de aplicación que se está desarrollando la imagen que se muestra al usuario es de pequeño tamaño por lo que podemos trabajar con imágenes codificadas en base64 sin ningún problema. Para ahorrar tiempo en el envío y evitar problemas de memoria solicitamos al dispositivo que nos devuelva la imagen capturada por el usuario a una resolución bastante baja:

```
navigator.camera.getPicture(this.onPhotoURISuccess, this.onFail, {
     quality: 25,
     destinationType: destinationType.DATA_URL,
     targetWidth: 256,
     targetHeight: 256,
     sourceType: Camera.PictureSourceType.PHOTOLIBRARY
});
```

Google Maps

El aplicativo hace un uso intensivo de varios servicios que ofrece Google Maps:

- **maps**, genera la interficie principal de la aplicación donde se muestra la lista de establecimientos.
- **static maps**, se usa en la ficha detalle de un establecimiento para generar una imagen estática de un mapa centrada en unas coordenadas que se le pasan como parámetros en la URL.
- **directionsService**, es usado para implementar la funcionalidad 'como llegar'.
- **geoCoder**, es usado en el proceso de añadir un nuevo establecimiento. Se le pide al usuario que indique en un mapa la posición con lo que obtenemos la latitud y la longitud del punto y le pedimos a Google que nos convierta esas coordenadas en una dirección postal.

6.2.- Aspectos a destacar sistema servidor

Obtener listado de establecimientos

El objetivo de este aplicativo es mostrar establecimientos cercanos. Pero la base de datos podría contener millones de establecimientos, por lo que cuando se le muestra el mapa no se pueden cargar todos ya que la aplicación ocuparía demasiado en la memoria del dispositivo y el mensaje de transferencia entre el sistema cliente y el sistema servidor sería demasiado grande por lo que ralentizaría el aplicativo. Por este motivo solo se envían los establecimientos que estén a 25 kilómetros alrededor de la posición inicial del usuario. Cuando el centro del mapa sale de esos 25 kilómetros se hace una nueva petición al servidor solicitando otra lista de establecimientos que estén dentro de la distancia del nuevo punto.

Esta estrategia tiene el inconveniente de que genera un cuello de botella en la base de datos ya que es esta la que tiene que calcular la distancia a la que se encuentra del usuario. Ésto hace que la base de datos tenga que calcular la distancia de todas los establecimientos y mostrar los que estén a 25 kilómetros de distancia. Para calcular la distancia usaremos la formular de Haversine implementada en SQL

```
SELECT *, ( 6371 * acos( cos( radians($latitude) ) * cos( radians( latitude ) ) * cos(
radians( longitude ) - radians($longitude) ) + sin( radians($latitude) ) * sin( radians( latitude
) ) ) ) AS distance
      FROM store
      HAVING distance < 25
      ORDER BY distance;
```

Para evitar tener que calcular la distancia de todos los establecimientos crearemos un cuadrado que englobe toda la circunferencia de radio 25 kms y filtraremos la latitud y la longitud en base al cuadrado máximo:

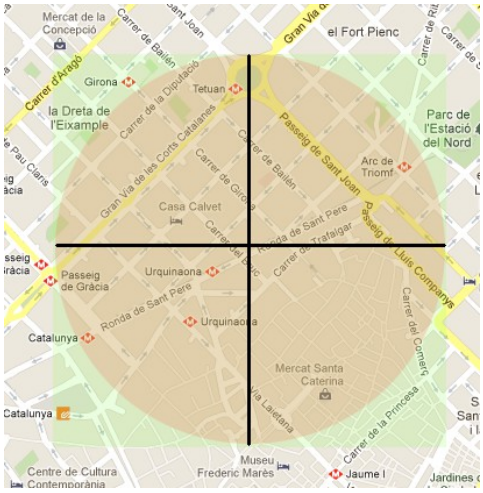


Ilustración 24: Filtrado establecimientos

Con esto ya no tenemos que calcular la distancia de todos los establecimientos sino únicamente los que estén dentro de un rango de latitudes y longitudes, con lo que acotamos mucho el problema.

```
SELECT *, ( 6371 * acos( cos( radians($latitude) ) * cos( radians( latitude ) ) * cos(
radians( longitude ) - radians($longitude) ) + sin( radians($latitude) ) * sin( radians( latitude
) ) ) ) AS distance
      FROM store
      WHERE
        (
          `latitude` BETWEEN ($latitude - ($distance/111.044)) AND ($latitude +
($distance/111.044))
           AND `longitude` BETWEEN ($longitude - ($distance/abs(cos(radians($latitude))*111.044)))
AND ($longitude + ($distance/abs(cos(radians($latitude))*111.044)))
        )

      HAVING distance < $distance
      ORDER BY distance;
```

Peticiones cross-site domain

Por motivos de seguridad los navegadores no permiten realizar llamadas AJAX a dominios diferentes del que está mostrando el site. En el caso de nuestro aplicativo el sistema cliente intenta realizar peticiones AJAX cross-site domain cuando se envían los formularios de los establecimientos o productos, en estos casos los navegadores en vez de realizar una única petición POST para enviar un formulario hacen dos, una de tipo OPTIONS, sin los parámetros del formulario que sirve para comprobar si puede comunicarse con el servidor, y si la respuesta es correcta se hace una segunda llamada con todos los parámetros del formulario.

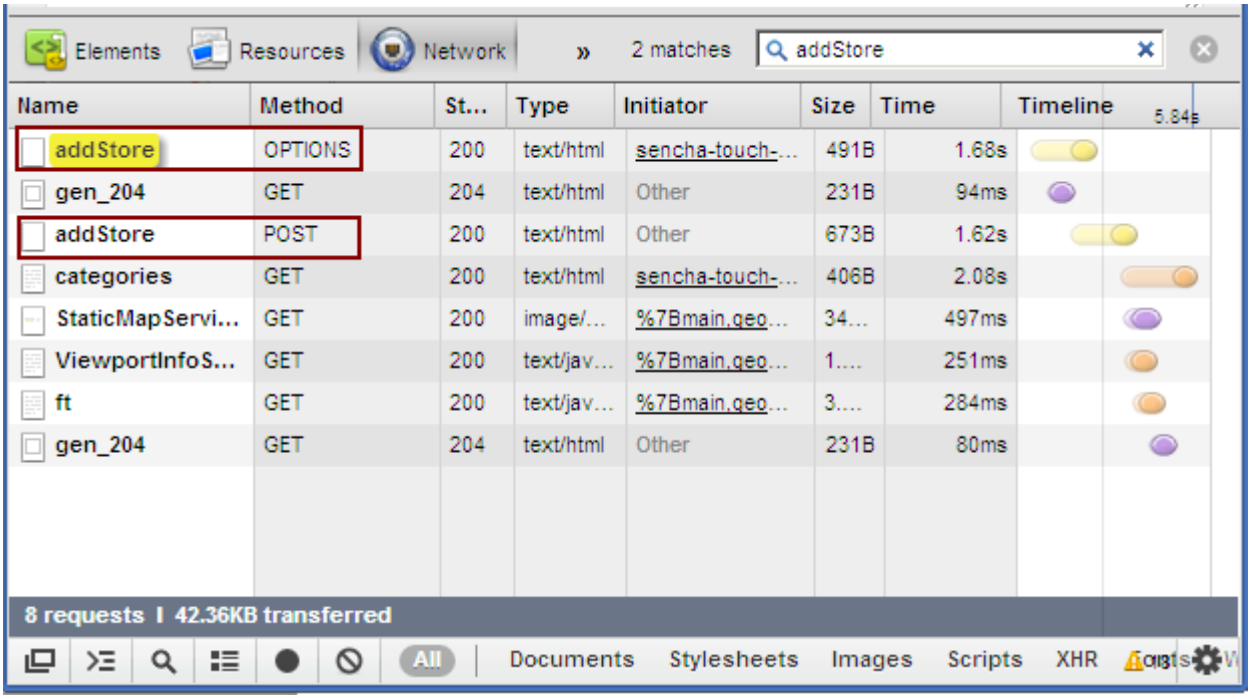


Ilustración 25: Petición AJAX cross-site

Del lado del servidor se tiene que aceptar la petición cross-site enviando unas cabeceras especiales de respuesta y teniendo en cuenta que en vez de una petición tendremos dos peticiones.

```
if ($req->getMethod() == 'POST') {
 $response = new Response();
 $response->headers->set('Access-Control-Allow-Origin', '*');
 $response->headers->set('Access-Control-Allow-Methods', 'POST, GET, OPTIONS');
 $response->headers->set('Access-Control-Allow-Headers', 'x-prototype-version,x-requested-with');

 return $response;
}
```

6.3.- Pruebas validación

El dispositivo sobre el que se han realizado las pruebas para validar el desarrollo es un Samsung Galaxy Ace con Android 2.3.6. Como pruebas para garantizar que la aplicación funciona correctamente se han diseñado una serie de conjuntos de acciones que se deben ejecutar en la aplicación. Los errores detectados en cada serie se anotarán y al acabar todas las series se iniciará el proceso para arreglar los problemas detectados si se considera oportuno.

Se realizarán tantas iteraciones como sean necesarias hasta que todas las series se realicen sin detectar ningún error. Se creará un juego de datos inicial con establecimientos y productos y antes de cada caso de prueba se vaciará la base de datos y se volverá a llenar con los datos de prueba.

Casos de prueba

Como casos de prueba básicos se usarán las historias de usuario que deben funcionar correctamente. A parte se han diseñado unos casos de prueba un poco más complejos para poder asegurar el correcto funcionamiento del aplicativo:

Añadir producto a nuevo establecimiento:

1. Nuevo establecimiento
2. Añadir producto al establecimiento
3. Volver al mapa
4. Acceder al establecimiento y que el producto esté asignado con la foto que hemos añadido
5. Apagar el programa y volverlo a encender
6. Comprobar que el establecimiento se muestra en el mapa y tiene el producto asociado

Añadir nuevo establecimiento lejano

1. Añadir un establecimiento situado a más de 30 kilómetros de distancia
2. Comprobar que aparece en el mapa
3. Apagar la aplicación y volver a encender
4. Comprobar alejando el zoom que el establecimiento no se muestra
5. Acercar el centro del mapa a donde debería estar el establecimiento y comprobar que se recarga el mapa y se muestra el establecimiento

Filtrar desde el listado de establecimientos

1. Crear un nuevo establecimiento llamado ´Mercado local´
2. Acceder al listado de establecimientos y ver que el establecimiento aparece y tiene una distancia correctamente asignada
3. Escribir ´Mercado´ en el cuadro de búsqueda del listado
4. Volver al mapa y comprobar que solo aparece el establecimiento ´Mercado local´ en el mapa
5. Volver al filtro y limpiar el cuadro de búsqueda
6. Volver al mapa y comprobar que aparecen todos los establecimientos

Filtrar desde formulario de búsqueda

1. Crear un nuevo establecimiento llamada 'Mercado local' con valoración 'Bueno' situado fuera de Barcelona
2. Añadir un producto con categoría Pintura
3. Ir al formulario de búsqueda y deseleccionar la categoría pintura
4. Comprobar que el establecimiento que hemos añadido no aparece
5. Apretar al botón superior para refrescar y comprobar que el establecimiento vuelve a aparecer

6. Ir al formulario de búsqueda y filtrar para que sólo aparezcan los establecimientos que tengan una valoración 'Muy buena'
7. Comprobar que en el mapa no aparece el establecimiento
8. Refrescar, volver al formulario de búsqueda y filtrar los establecimientos que estén a más de un kilómetro de distancia
9. Comprobar que no aparece el establecimiento

6.4.- Problemas conocidos

Compatibilidad multiplataforma

El objetivo de usar PhoneGap es conseguir que el aplicativo funcione sin problemas en diferentes plataformas con diferentes versiones o sistemas operativos. El desarrollo se ha hecho y probado sobre una plataforma con sistema operativo Android 2.3.6, algunas pruebas realizadas sobre Android 4 han confirmado que existe algún tipo de incompatibilidad que hace que no se inicie correctamente.

Reinicio al capturar una fotografía

En ocasiones cuando se intenta capturar una fotografía desde el dispositivo móvil el programa se reinicia no pudiendo completar la tarea. Esto es debido a un problema de PhoneGap y Android. Al iniciar el proceso de captura de fotografías el programa se pone en modo segundo plano y se inicia la aplicación nativa de captura de fotos. El sistema, al tener poca memoria RAM libre, reinicia la aplicación y cuando intenta volver a ponerla en primer plano después de adquirir la fotografía la aplicación se muestra al usuario en la pantalla inicial del mapa.

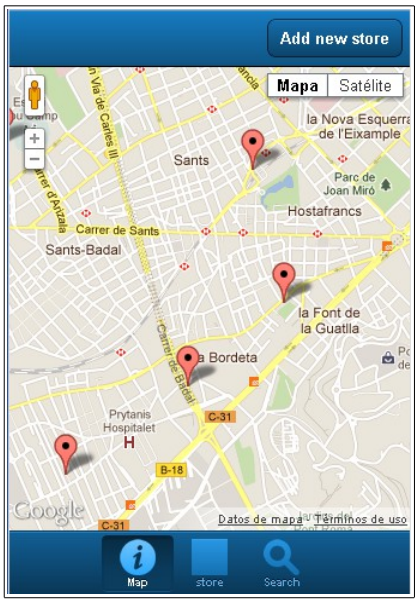
Problema usabilidad al añadir productos

El formulario para introducir nuevos productos tiene muchas opciones que pueden ocasionar dificultades en la usabilidad en dispositivos con pantallas pequeñas. Al hacer clic en las opciones de texto (título, descripción) se despliega el teclado virtual y oculta gran parte del formulario.

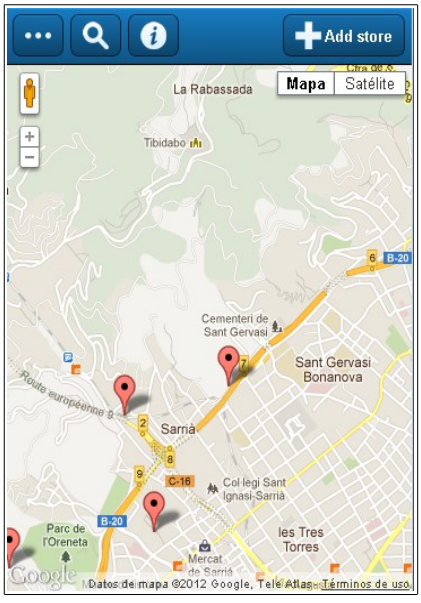
6.5.- Diseño final

A pesar de haber hecho una fase para desarrollar el prototipo y probarlo durante la fase de desarrollo el aplicativo ha continuado sufriendo cambios de diseño. El más claro es el cambio del color general de la interficie:

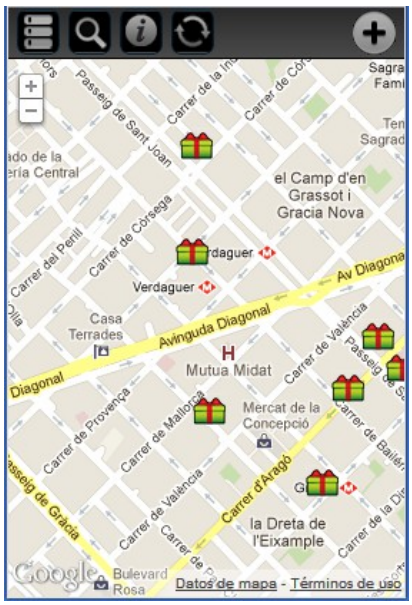
Versión inicial



Versión revisada



Versión final



Los iconos del menú superior se han hecho ligeramente más pequeños y se han modificado, añadiendo un nuevo botón para refrescar la lista de establecimientos que se están mostrando. También se ha configurado el mapa de Google para que muestre menos opciones de configuración y así tener más área visible.

6.6.- Desarrollo multiplataforma

El objetivo del proyecto es desarrollar una aplicación multiplataforma pero hasta ahora todas las pruebas se han realizado sobre Android. Para desarrollar para Iphone/Ipad es necesario utilizar el sistema operativo MacOS y disponer de una licencia (de pago) de desarrollador los por lo que no se ha podido probar en este sistema operativo.

Para realizar las pruebas sobre BlackBerry únicamente se necesita un ordenador con sistema operativo Windows y el SDK gratuito de Blackberry por lo que se han podido realizar pruebas de compatibilidad.

Después de realizar la compilación para generar un fichero instalable en Blackberry se han encontrado diferentes problemas.

Generar fichero instalable

Al parecer el compilador de Blackberry no admite ficheros con espacios en blanco en su nombre. SenchaTouch tiene un único fichero que no cumple esta regla y genera problemas al intentar compilar el fichero. El fichero es una imagen que al parecer no está referenciada por ningún fichero y ha podido ser borrando no afectando al resto de la aplicación, solucionando el problema al compilar los ficheros.

Ejecutar programa en el simulador

Al iniciar el programa se mostraban varios mensajes de error impidiendo la inicialización del aplicativo, esto era debido a un problema con algunos ficheros css para Android de SenchaTouch. Al solucionar este problema y conseguir iniciar el aplicativo surge otro problema, una vez la aplicación está cargada se muestra una capa negra en la pantalla que oculta toda la vista. Este problema parece que es que es de SenchaTouch, que no es capaz de determinar el alto en píxeles de la pantalla. Es necesario sobreescribir la clase que representa la pantalla en SenchaTouch, Ext.viewport.Viewport, para hacer un pequeño arreglo y que funcione correctamente.

Tras este último problema la aplicación se inicia con normalidad en el simulador aunque la velocidad del simulador hace impracticable la aplicación.

Ilustración 26: Simulación Blackberry

De forma general la aplicación no parece funcionar correctamente, aunque se ha conseguido añadir establecimientos y visualizarlos en el mapa general, ni siquiera el navegador de Google Maps reacciona correctamente cuando se aumente o disminuye el zoom, tarda varios minutos en refrescar la vista. Quedaría probar la aplicación en un dispositivo real para comprobar si el problema es la velocidad del simulador o la aplicación.

Seguramente para conseguir hacer funcionar correctamente la aplicación se tengan que realizar pequeños ajustes como los efectuados para que la aplicación se iniciara pero la mayor parte del código funcionaría sin problemas.

7.- Líneas de futuro

Como línea de futuro la principal prioridad debería ser la de intentar crear una comunidad alrededor de la aplicación, entre otras posibles mejoras:

- Asegurar la compatibilidad con Ios y Blackberry
- Permitir a los usuarios registrarse para crear comunidad
- Ver listados de los usuarios más activos
- Mensajes entre usuarios para que puedan consultar sus dudas, etc...
- Valorar las mejores propuestas
- Integración con otras redes sociales, Facebook, Pinterest, Twitter...
- Versión web/tablet con interficie adaptada a las características del dispositivo. La actual versión está adaptada a un dispositivo con pantalla de tamaño reducido.
- Versión offline con almacenamiento de los datos en el dispositivo cliente y posibilidad de sincronizar posteriormente ya que no siempre se dispone de conexión a Internet.
- Test unitarios y funcionales automatizados en los sistemas cliente y servidor para asegurar la calidad en el desarrollo.
- Algunos parámetros de configuración se inicializan en la aplicación cliente, como el radio de búsqueda de establecimientos. Si en un futuro se decidiese cambiar este parámetro se tendría que reinstalar la aplicación cliente. Se debería modificar la aplicación para que toda esta configuración se leyera desde la aplicación servidor.

8.- Conclusiones

EL objetivo principal de este proyecto era aplicar los conocimientos obtenidos durante las diferentes asignaturas de la carrera y en ese aspecto el proyecto ha cumplido perfectamente su objetivo. Se ha hecho la recogida inicial de requerimientos, planificación, diseño, desarrollo y validación del proyecto.

Como objetivo secundario la finalidad era desarrollar un aplicativo multiplataforma usando plataformas de desarrollo ágiles. Es en este punto donde se han encontrado las mayores dificultades:

- Aunque PhoneGap y SenchaTouch se definan como alternativa perfectamente válida a un desarrollo en lenguaje nativo para las diferentes plataformas, durante el desarrollo del proyecto se ha demostrado que a los productos les queda mucho para llegar a ese punto:
    - **Rendimiento**: el rendimiento de la aplicación es muy inferior al código nativo, eso se nota principalmente a la hora de navegar por los mapas de Google y comparar el rendimiento con la aplicación oficial de Google Maps.
    - **Multiplataforma:** la aplicación únicamente se ha probado bajo algunas versiones de Android y sobre el simulador Blackberry, no funcionando correctamente con la versión 4 de Android y teniendo que hacer algunas adaptaciones para que funcione correctamente en otras. Para que funcionara en Blackberry se ha tenido que modificar el código de la aplicación y algunos ficheros de SenchaTouch, consiguiendo que la aplicación se inicie pero no funcionando correctamente.
    - **Documentación**: este punto a sido el más problemático. Después de analizar las diferentes alternativas, SenchaTouch parecía ofrecer una buena documentación. Después del desarrollo se ha comprobado que la documentación es incompleta y errónea. En muchos ejemplos del sitio web oficial no eran completos o funcionaban correctamente.

- Programación orientada a dispositivos móviles, a pesar de haber realizado un prototipo y haber hecho varias pruebas con usuarios, durante la fase de desarrollo el diseño de la interficie ha sufrido cambios para mejorar la usabilidad y la sensación de estar usando un producto profesional. En general y más aun en el mundo de los programas orientados a dispositivos móviles, es imprescindible invertir un gran esfuerzo en trabajar la interficie gráfica ya que de esta depende en gran medida que el aplicativo sea un éxito o una aplicación más dentro de los millones de aplicaciones que el usuario tiene a su alcance.

Glosario

Glosario de términos por orden alfabético:

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML). Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. http://es.wikipedia.org/wiki/AJAX

API, Interfaz de programación de aplicaciones o API (del inglés Application Programming Interface). Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. http://es.wikipedia.org/wiki/API

CDN, red de entrega de contenidos o red de distribución de contenidos (CDN, content delivery network en inglés). Es una red superpuesta de computadoras que contienen copias de datos, colocados en varios puntos de una red con el fin de maximizar el ancho de banda para el acceso a los datos de clientes a través de la red. http://es.wikipedia.org/wiki/Red_de_entrega_de_contenidos

Codificación Base64, es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII. http://es.wikipedia.org/wiki/Base64

Garbage collector, un recolector de basura es un mecanismo implícito de gestión de memoria implementado en algunos lenguajes de programación de tipo interpretado o semi interpretado. http://es.wikipedia.org/wiki/Recolector_de_basura

GPS, acrónimo de Global Positioning System (sistema de posicionamiento global) o NAVSTAR-GPS1. Es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto. http://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

JSON, acrónimo de JavaScript Object Notation. Es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. http://es.wikipedia.org/wiki/JSON

Peticiones CROSS-SITE DOMAIN, peticiones AJAX realizadas entre sitios web con diferente dominio o subdominio.

Reserve proxy, es un servidor proxy instalado en el domicilio de uno o más servidores web. Todo el tráfico entrante de Internet y con el destino de uno de esos servidores web pasa a través del servidor proxy. http://es.wikipedia.org/wiki/Proxy#Reverse_Proxy_.2F_Proxy_inverso

SQL, lenguaje de consulta estructurado (por sus siglas en inglés Structured Query Language). Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas. http://es.wikipedia.org/wiki/SQL

URL, URI, acrónimo de Uniform Resource Identifier (en español «identificador uniforme de recurso»). Es una cadena de caracteres corta que identifica inequívocamente un recurso. http://es.wikipedia.org/wiki/Uniform_Resource_Identifier

Webservice, servicio web es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. http://es.wikipedia.org/wiki/Servicio_web

Bibliografía

Para el desarrollo de este proyecto se han consultado las siguientes fuentes de información:

Otros: Apache Cordoba API Documentation [en línea]:

<http://docs.phonegap.com/en/1.4.0/index.html> [Consulta: marzo-abril de 2012]

Otros: Sencha Docs – Touch 2 [en línea]:

<http://docs.sencha.com/touch/2-0/> [Consulta: marzo-abril de 2012]

Otros: Learn Symfony - Symfony [en línea]:

<http://symfony.com/doc/current/index.html> [Consulta: abril-mayo de 2012]

Otros: Multiple phone web-based application framework [en línea]:

<http://en.wikipedia.org/wiki/Multiple_phone_web_based_application_framework> [Consulta: febrero de 2012]

Otros: BEST MOBILE WEB HTML5 FRAMEWORK FOR MOBILE APP DEVELOPMENT [en línea]:

<http://dumaslab.com/2011/05/best-mobile-web-html5-framework-for-mobile-app-development/> [Consulta: febrero de 2012]

Rubin, Alexander: Geo/Spatial Search with MySQL [en línea]:

<http://es.scribd.com/doc/2569355/Geo-Distance-Search-with-MySQL > [Consulta: mayo de 2012]

Otros: Haversine formula [en línea]:

<http://en.wikipedia.org/wiki/Haversine_formula> [Consulta: mayo de 2012]

Otros: Calculate distance between two points on a globe [en línea]:

<http://www.codecodex.com/wiki/Calculate_Distance_Between_Two_Points_on_a_Globe#MySQL> [Consulta: mayo de 2012]

Otros: Phonegap app quits after Camera captures image. [Android] [en línea]:

<http://stackoverflow.com/questions/9303701/phonegap-app-quits-after-camera-captures-image-android> [Consulta: mayo de 2012]

Otros: PhoneGap camera restarts the application [en línea]:

<http://stackoverflow.com/questions/8368091/phonegap-camera-restarts-the-application/9524643#> [Consulta: mayo de 2012]

Otros: Google Maps Javascript API V3 Reference [en línea]:

<https://developers.google.com/maps/documentation/javascript/reference> [Consulta: marzo-junio de 2012]