Facultad
de
Ciencias

Desarrollo de una aplicación sobre los comercios de Santander

Application development about Santander stores

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: África San Román Ortiz

Director: Carlos Blanco Bueno

Junio - 2015

Resumen

Cada vez son más las páginas que proporcionan datos en abierto a los usuarios sobre diversos temas. Un ejemplo lo encontramos en el Ayuntamiento de Santander, quien, a través de la web http://datos.santander.es/ suministra conjuntos de datos en diferentes formatos sobre diversos aspectos de la ciudad, tales como tráfico, transporte, urbanismo o comercios, entre otros.

El problema que presentan muchas veces estos datos es que para los usuarios son difíciles de comprender. Por ello, se desarrollan aplicaciones que facilitan el entendimiento y el acceso.

Centrado en los conjuntos de datos sobre los comercios, el presente trabajo pretende acercar a los usuarios estos conjuntos para que la información esté al alcance de su mano fácilmente.

A través de una aplicación web, el usuario podrá consultar horarios, direcciones, ofertas, así como la localización en el mapa de los comercios deseados. Además, la aplicación incorporará geolocalización para el dispositivo, pudiendo consultar así comercios cercanos, y filtros para encontrar el comercio que mejor se adapte al horario del usuario.

La aplicación también contará con una versión móvil para el sistema operativo Android. Ésta tendrá las mismas funcionalidades que la versión web.

El proyecto, además de lo anterior, tiene como objetivo el aprendizaje de nuevas tecnologías y la adquisición los conocimientos necesarios para el desarrollo de dicha aplicación. Estos servirán para desarrollar nuevas aplicaciones en un futuro en diversas plataformas.

Palabras clave

Aplicación web, aplicación móvil, comercios, geolocalización, smartphone.

Abstract

It is becoming more and more frequent the websites that offer "Open Data" to its users about a variety of topics. An example of this is Santander's Townhall, which uses its website http://datos.santander.es/ to deliver information about the city, such as information about traffic, transport, urbanism and local shops among others.

The amount of data accessible with the use "Open Data" often confuses its users. Applications have been developed in order to solve this problem, which facilitate the comprehension and access to the data.

This paper will focus on the dataset generated about local commerce; in particular it will try to make the dataset within the user's hand reach.

Through a web application, the user would be able to check the schedule, directions, offers and even the location in the map of the desired shops. Furthermore, the application will feature geo-location, which will allow the user to check for the shops near them and even filter to find the shops that best suit their schedules.

The application will also feature a mobile version for the Android operating system. This will have the same functionality as the web version.

In addition to the previous, the project has the aim of being part of a learning process in the use of the new technologies and acquiring new capabilities for the correct and successful development of the application. This will serve as a base to develop new applications in the future for different platforms.

Keywords

Web application, mobile application, stores, geolocation, smartphone.

Índice

Tabla de Ilustraciones

1 Introducción

Según el informe "La sociedad en red 2013" elaborado en 2014, el 53,7% de los españoles posee un smartphone. Se trata de un mercado en alza que prevé un importante desarrollo para los próximos años. Estos dispositivos permiten a sus usuarios estar en todo momento conectados a la red.

El crecimiento de esta tecnología ha derivado en la evolución del software. Se prefiere acceder a los contenidos existentes en la red a través de aplicaciones móviles en vez de usar el navegador. Por su sencillez, su fácil manejo y el acceso directo a la información requerida. Debido a esta evolución, datos e información se hacen muy accesibles.

A su vez, las páginas de datos en abierto en la red han crecido y esto supone un gran avance, pues el usuario tiene a su disposición gran cantidad de información que puede resultarle muy útil.

Estas páginas de datos en abierto presentan una serie de problemas a la hora de ser consultadas por cualquier usuario que no tiene porqué saber conocimientos informáticos:

- Los datos están mal estructurados y son complicados de entender.
- Hay datos que no son relevantes para el usuario.
- Hay relaciones entre los conjuntos de datos que no están detalladas y que el usuario no tiene porqué entender.

Se plantea entonces dar solución a estos problemas seleccionando los conjuntos de datos y atributos que podrían ser más interesantes para el usuario, y sobre ellos, construir aplicaciones que permitan acceder a la información de una forma amigable y útil.

El presente trabajo quiere que ciudadanos, turistas y cualquier usuario que quiera consultar información sobre la ciudad de Santander tengan al alcance de su ordenador o smartphone toda la información relativa a los comercios de esta ciudad que el ayuntamiento proporciona a través de la web http://datos.santander.es/.

Esta web proporciona datos en abierto sobre varias áreas como transporte o cultura en diferentes formatos como JSON, XML o HTML.

Se considera que, desde el punto de vista del usuario, los datos que más pueden interesar son los relativos a los comercios como el nombre, la dirección, el teléfono, la geolocalización, la web, el Facebook, el Twitter, los horarios de apertura y las ofertas relacionadas junto con su descripción.

Acceder a esta información tal y como se presentan los datos en la web podría ser muy tedioso. Por ejemplo, para conocer las ofertas relativas a un comercio, habría que buscar el comercio para conocer su identificador dentro del sistema, y después consultar en el conjunto de datos de las ofertas cuál de ellas es relativa a ese identificador.

El presente trabajo tiene como motivación dar al usuario la posibilidad de encontrar información relativa a los comercios de Santander mediante el desarrollo de una aplicación que lo facilite. Ésta, además de ser web, también será móvil.

1.1 Objetivos

Los principales objetivos del presente proyecto se pueden concretar en:

- El principal objetivo del presente Trabajo Fin de Grado es el de desarrollar una aplicación web que permita acercar a los ciudadanos la información relativa a varios conjuntos de datos. Concretamente, la aplicación se centrará en datos relativos a los comercios de Santander tales como horarios u ofertas. El usuario podrá consultar esta información, aplicar filtros y visualizar los comercios en un mapa.
- Basándose en la aplicación web creada, desarrollar la aplicación móvil que se ejecute en dispositivos móviles que utilicen Android como sistema operativo.

2 Metodología de desarrollo

Para llevar a cabo el proyecto, se ha utilizado una metodología de desarrollo iterativa incremental. Los pasos seguidos han sido los siguientes.

- En primer lugar, se establece cual va a ser la aplicación a desarrollar y se realiza el análisis de requisitos.
- Se divide el desarrollo de la aplicación en cuatro etapas. En cada una de ellas se trabaja sobre la aplicación, incrementando su desarrollo y aprendiendo e investigando sobre las tecnologías utilizadas a la vez.
    - Esqueleto básico de la aplicación.
        - Diseño de interfaz.
        - Selección de ficheros y atributos para mostrar.
        - Presentación de la información en la web.
    - Filtros.
        - Definición de los filtros para los comercios.
        - Implementación de los filtros.
    - Mapa.
        - Añadir la visualización del mapa.
        - Hacer visibles y accesibles los comercios en el mapa.
    - Visualización móvil.
        - Incorporar los plugins necesarios para que la aplicación se pueda visualizar en el móvil.
        - Construir aplicación móvil con PhoneGap.
- Una vez iniciado el desarrollo de la aplicación, se comienza a elaborar la memoria del trabajo.

Se realizó una planificación en el programa Microsoft Project en la que se establecían las tareas que se debían realizar en los plazos de tiempo determinados.

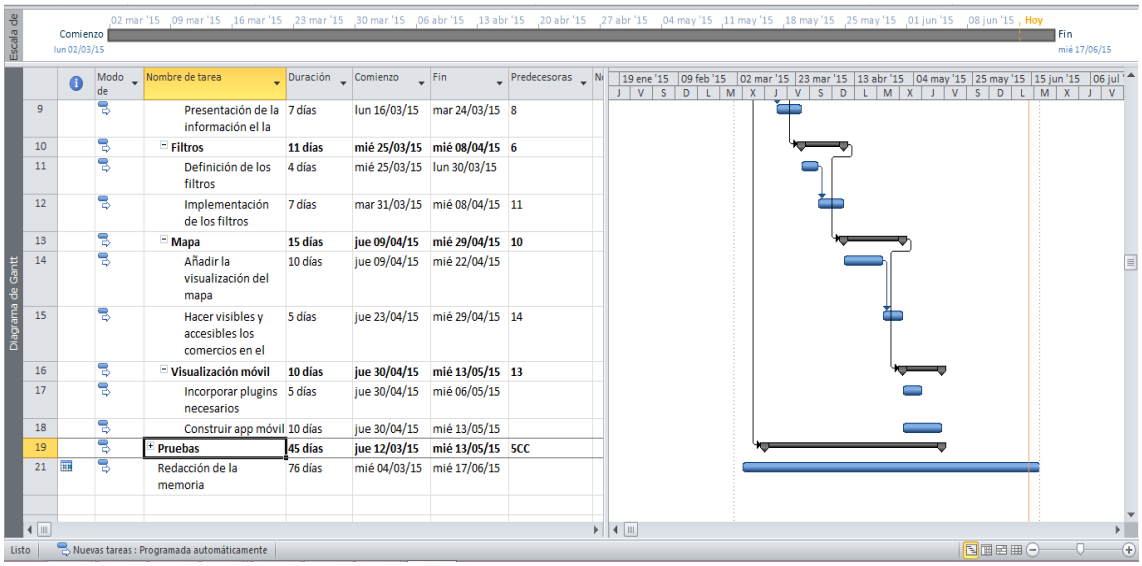

Fig. 1: Flujo de trabajo en Microsoft Project

3 Tecnologías y lenguajes utilizados

En este apartado se expondrán las tecnologías utilizadas así como los lenguajes empleados.

Hasta hace no tanto tiempo, si se quería desarrollar una aplicación específica para un determinado sistema operativo, se debía programar en lenguaje nativo para cada una de sus plataformas. Esto incrementaba el tiempo de desarrollo del proyecto y el coste del mismo.

La tecnología ha avanzado, y a día de hoy, es posible desarrollar una aplicación y exportarla a otras plataformas haciendo simples modificaciones como añadir pluggins específicos para cada sistema operativo en el que se quiere desplegar. Esto supone abarcar más proyectos en una misma cantidad de tiempo y, por lo tanto, mejorar los ingresos.

En el presente trabajo, las tecnologías y lenguajes empleados han sido los siguientes:

- Tecnología web desarrollada en HTML5, CSS3 y JavaScript.
- Tecnología móvil codificada con PhoneGap y jQuerymobile.

Además, para las pruebas unitarias se ha empleado el lenguaje Qunit, para las pruebas de integración la herramienta Selenium, para el almacenamiento de datos el formato JSON y como máquina virtual para probar la aplicación móvil, Genymotion.

En los siguientes apartados se describirán las tecnologías y los lenguajes utilizados con detalle.

3.1 Tecnología web

3.1.1 HTML5

Fig. 2: Logotipo de HTML5

HTML, siglas de HyperText Markup Languaje, es el lenguaje de programación de páginas web basado en etiquetas. Fue creado por Tim Berners en 1990 con el objetivo de facilitar a científicos de diferentes universidades el acceso a los documentos de investigación de cada uno de ellos.

HTML es un lenguaje que hace posible presentar información en internet. Lo que se ve al visualizar una página web es la interpretación que hace el navegador del código HTML.

En concreto, el estándar utilizado en el desarrollo del presente proyecto es el HTML5. Esta versión incluye novedades respecto a las anteriores (HTML4 y XHTML 1.0) que se adaptan a las necesidades de las webs actuales.

- Estructura del cuerpo. Se han incluido nuevas etiquetas que permiten estructurar mejor la página web, estableciéndose cada sección con diferentes etiquetas y reemplazando en muchas ocasiones a las * *. Esta innovación permite desarrollar webs coherentes y fáciles de entender por otras personas. Y lo que es más importante, será trivial de entender para una máquina, que será capaz de dar más importancia a unas secciones que a otras.
- Etiquetas para un contenido específico. Se incorporan nuevas etiquetas específicas para la incorporación de elementos multimedia como *<audio>* y

<video>. Los formularios también añaden nuevos elementos, sobre todo los *<input>*, quienes incorporan nuevos atributos y tipos.

- Canvas. Se trata de una API que controla, con ayuda de JavaScript, la creación de dibujos. Estos permiten movimiento, añadirse fotos y texto e incluso hacer gráficos o animaciones 3D.
- Bases de datos locales. HTML5 permite crear bases de datos en el ordenador del propio usuario y esto sin duda es una ventaja, pues permite trabajar sin conexión.
- Web Workers. Son subprocesos en el hijo que ejecución que evitan que se sature la web cuando procesa un gran volumen de información. Se ejecutan de forma asíncrona, en segundo plano y en procesos diferentes.
- Geolocalización. Se incluye, a través de una API, la posibilidad de localizar geográficamente las páginas web.

Existen diversas características que nos llevan a elegir este lenguaje frente al nativo de cada plataforma. Se enumeran las siguientes:

- Facilidad de desarrollo. Para llevar a cabo las tareas más centradas en el contenido de la aplicación, es más fácil utilizar HTML y CSS para dar forma al mismo que las bibliotecas nativas de Android o IOS.
- Reutilización. Este lenguaje junto a CSS y JavaScript es totalmente reutilizable para las distintas plataformas. El código que escriben se puede reutilizar para webs optimizadas para móviles o entornos táctiles.
- Rapidez de desarrollo. Las herramientas de contenido que existen en la nube permiten a los desarrolladores poder hacer su trabajo mediante el navegador web instalando sobre él herramientas.
- Extensibilidad. HTML es de naturaleza dinámica y resulta fácil mezclar, enlazar y modificar contenidos. El objetivo de esta propiedad es múltiple: por un lado, crear aplicaciones desde una misma plantilla, y por otro, los gestores de contenido pueden actualizar los datos y hacer cambios en cualquier momento en la configuración de las aplicaciones publicadas a través de la nube.

3.1.2 JavaScript

Fig. 3: Logotipo de JavaScript

JavaScript es un lenguaje de programación orientado a las páginas web. Su sintaxis es muy similar a la del lenguaje Java pero no guardan ninguna relación entre ellos. Se define como un lenguaje orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Principalmente se usa del lado del cliente y por ello no requiere compilación ya que son los navegadores los encargados de interpretar estos códigos. Permite mejoras en la interfaz del usuario y páginas web dinámicas. También existe del lado del servidor.

Fue desarrollado por Brendan Eich, trabajador de la empresa Netscape con el nombre de Mocha en 1995. Posteriormente fue renombrado a LiveScript para finalmente quedar como JavaScript.

La característica principal de JavaScript es la de ser un lenguaje de scripting por excelencia.

3.1.3 CSS3

Fig. 4: Logotipo de CSS3

CSS3 es el último estándar de CSS. Se corresponde con las con siglas *Cascading Style Sheets* (hojas de estilo en cascada). Se trata de un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML. El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

El objetivo detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información del estilo puede estar definida en un documento separado o en el mismo HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo *<style>*.

3.2 Tecnología móvil

3.2.1 PhoneGap

Fig. 5: Logotipo de PhoneGap

Desarrollar aplicaciones multiplataforma es posible gracias a frameworks como PhoneGap, también conocido como Apache Cordova.

PhoneGap es un framework libre de código abierto que permite crear aplicaciones móviles para diferentes plataformas utilizando APIs web estandarizadas. Creado por Nitobi y comprada en octubre de 2011 por Adobe Systems, quien ha entregado el código a la fundación Apache, dónde se encuentra bajo el nombre de PhoneGap, manteniéndose así como Open Source.

El reto al crear PhoneGap consistía en desarrollar un marco que permitiera a los desarrolladores web usar HTML, CSS y JavaScript para aplicaciones que pudieran abarcar la funcionalidad nativa del dispositivo móvil, como la cámara, almacenamiento y características de geolocalización.

Aunque fue pensado para funcionar únicamente con iPhone, PhoneGap ha ido creciendo y a día de hoy permite desarrollar aplicaciones en las siguientes plataformas:

- IOS
- Android

- Windows phone
- Blackberry OS
- Symbian
- Bada
- webOS

PhoneGap almacena librerías JavaScript que han sido desarrolladas en el lenguaje específico de cada plataforma y nos garantizan el acceso a las distintas características del dispositivo móvil.

Podemos percibir PhoneGap como un conjunto de páginas web almacenadas y empaquetadas dentro de una aplicación móvil. A través de estas webs tenemos acceso al DOM por lo que es posible usar otros frameworks como jQuery Mobile, como se hace en este trabajo.

Según el sistema operativo en el que se quiera desarrollar la aplicación, se han de tener en cuenta una serie de exigencias. Por ejemplo, en el caso de iOS, es necesario tener un Mac e instalado en él, su software de desarrollo llamado Xcode. En el caso de Android, se debe utilizar Eclipse y para BlackBerry, aunque no hay un entorno específico, debe usarse Java SDK, BlackBerry SDK y Apache Ant.

El presente trabajo se centrará en el desarrollo de una aplicación para Android.

3.2.2 JQuery Mobile

Fig. 6: Logotipo de jQuery Mobile

jQuery Mobile es un Framework optimizado para dispositivos táctiles que está siendo desarrollado actualmente por el equipo de proyectos de jQuery. El desarrollo se centra en la creación de un Framework compatible con la gran variedad de smartphones y tablets.

Este Framework sigue el eslogan "*write less, do more*" (en español "escribe menos, haz más"). En vez de escribir una aplicación para cada Smartphone o Tablet, este Framework permite diseñar un único sitio web o aplicación que funcione en todas las plataformas.

3.3 QUnit

Fig. 7: Logotipo de QUnit

Es un Framework para pruebas unitarias fácil de usar para JavaScript. Actualmente es usado por los proyectos JQuery, JQuery UI y JQuery Mobile. Es un Framework robusto y tiene un gran alcance.

Esta herramienta fue desarrollada por John Resing como parte del proyecto llamado JQuery y ahora es usada para desarrollos independientes. En el año 2008 QUnit tuvo su sitio web, nombre y documentación del

API, permitiendo a otras personas usar sus propias herramientas de pruebas unitarias. En aquel entonces, este framework tenía una gran dependencia de JQuery, pero después, en el año 2009, se hicieron cambios sustanciales y QUnit pudo pasar a ejecutarse solo.

3.4 Selenium

Fig. 8: Logotipo de Selenium

Selenium es un entorno de pruebas software para aplicaciones basadas en web. Provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de scripting para pruebas. Software de código abierto y licencia apache 2.0 que puede ser descargado y usado sin cargo. Selenium IDE es la herramienta utilizada en este caso, es un entorno de desarrollo integrado para pruebas con Selenium. Está implementado como una extensión de Firefox y permite grabar, editar y depurar pruebas.

3.5 JSON

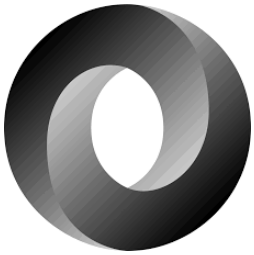

Fig. 9: Logotipo de JSON

JSON (JavaScript Object Notation – Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para los humanos, mientras que para la máquina es simple interpretarlo y generarlo.

JSON es un formato de texto completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidas por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termine con } (llave de cierre). Cada nombre es seguido por: (dos puntos) y los pares nombre/valor están separados por , (coma).

Ejemplo:

```
{
   "Frutas": [
         { "NombreFruta":"Manzana" , "cantidad":10 },
         { "NombreFruta":"Pera" , "cantidad":20 },
         { "NombreFruta":"Naranja" , "cantidad":30 }
         ]
}
```

3.6 Genymotion

Fig. 10: Logotipo de Genymotion

Se trata de un emulador de Android. Permite ejecutar y probar aplicaciones en él. Sus principales características son:

- Rapidez. Permite crear en tan solo tres clicks un dispositivo Android en el que poner a prueba cualquier aplicación.
- Simpleza. Fácil de instalar y ejecutar. Su interfaz es muy intuitiva.
- Potencia. Genymotion permite controlar sensores de gran alcance para probar las características de la aplicación que se desee. Usa una arquitectura de virtualización x86 muy eficiente.

4 Presentación de la solución y análisis de requisitos

El presente proyecto busca dar uso de los datos que el Ayuntamiento de Santander proporciona a través de la web *datosSantander.es*. Pretende desarrollar una aplicación web, y, a partir de ella una móvil, basada en los conjuntos de datos que proporciona la web en abierto. Para su desarrollo, se emplearán tres conjuntos de datos con información sobre:

- Comercios. De este conjunto de datos, se seleccionan como atributos
  - Nombre del comercio
  - Teléfono
  - Comentarios
  - Email
  - Descripción
  - Facebook
  - Twitter
  - Web

- Horarios. Sus atributos escogidos son
  - Hora de apertura
  - Hora de cierre
  - Días de apertura
- Ofertas. Se seleccionan los siguientes datos
  - Nombre de la oferta
  - Nombre del comercio que publica la oferta
  - Fecha de inicio de la oferta
  - Fecha de fin de la oferta
  - Descripción de la oferta

Los datos pertenecen a diferentes conjuntos y esto añade una problemática adicional, ya que la aplicación deberá relacionar entre sí toda la información.

4.1 Requisitos funcionales

Los requisitos funcionales describen la funcionalidad o servicios del software. A continuación se detallan los requisitos funcionales de la aplicación.

ID	Descripción del requisito
RF01	El usuario podrá consultar el listado de comercios de Santander.
RF02	El usuario podrá consultar el teléfono, los comentarios, el email, la descripción, el facebook, la dirección, el twitter, la página web y el horario del comercio seleccionado siempre que esta información esté disponible.
RF03	El usuario podrá consultar la localización del comercio seleccionado.
RF04	El usuario podrá consultar las ofertas del comercio seleccionado.
RF05	El usuario podrá geolocalizarse a la vez que se muestran las geolocalizaciones de los comercios

RF06	El usuario podrá filtrar los comercios mostrados en mapa y en lista por días de apertura.
RF07	El usuario podrá filtrar los comercios mostrados en mapa y en lista por horarios de apertura.
RF08	El usuario podrá filtrar los comercios mostrados en mapa y en lista por horarios y días de apertura.
RF09	El usuario podrá filtrar los comercios para que se muestren en mapa y en lista los abiertos en el instante de la consulta.
RF10	El usuario podrá acceder a la información del comercio requerido en el mapa
RF10	El usuario podrá listar todas las ofertas.
RF11	El usuario podrá listar las ofertas activas.
RF12	El usuario podrá consultar las fechas de inicio y fin y la descripción de la oferta seleccionada.
RF13	El usuario podrá acceder a la información del comercio relativa a la oferta seleccionada.

4.2 Requisitos no funcionales

Los requisitos no funcionales toman en consideración otras características importantes que afectan al sistema desde diversos ámbitos (fiabilidad, usabilidad, rendimiento, etc.). Los requisitos no funcionales que deberá seguir el sistema desarrollado se recogen en la siguiente tabla.

ID	Descripción del requisito	Importancia
RNF01	El sistema ha de ser compatible con los navegadores Firefox, Chrome y Safari.	Media
RNF02	El sistema ha de ser multiplataforma y multidispositivo.	Media
RNF03	La interfaz del sistema deberá diseñarse utilizado iconos fácilmente identificables	Alta

4.3 Diagrama de casos de uso

A continuación se recoge el diagrama de casos de uso que describe los requisitos funcionales descritos en el apartado anterior.

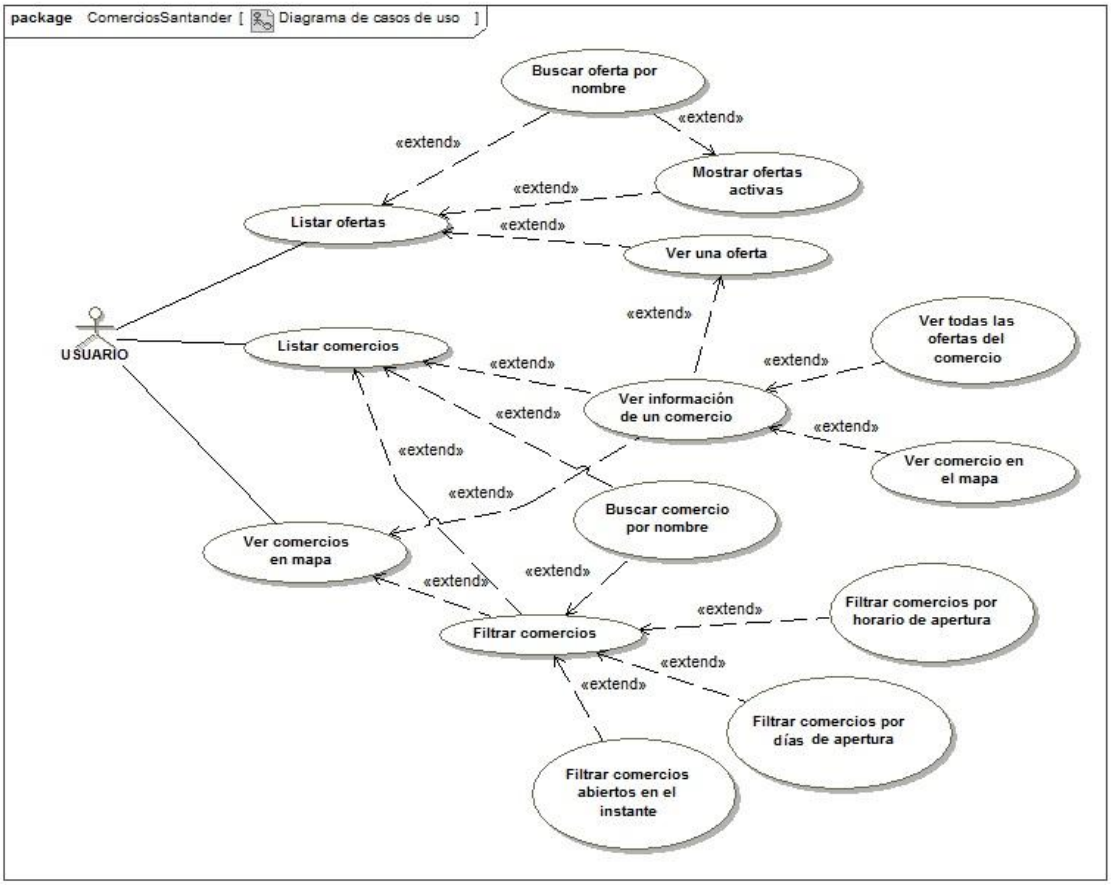

Fig. 11: Diagrama de casos de uso

4.3.1 Plantillas

En las siguientes plantillas se explican detalladamente cada uno de los casos de uso.

Plantilla	
ID	P01
Nombre	Listar ofertas
Descripción	El usuario selecciona la opción de listar ofertas y el sistema le muestra la lista de ofertas disponibles.
Actores primarios y secundarios	Usuario
Precondiciones	
Flujo principal	1. El usuario pulsa sobre el botón de ofertas. 2. El sistema muestra las ofertas disponibles.
Flujos alternativos	2.1 Las ofertas no están disponibles y no se muestra ninguna.

Plantilla	
ID	P02
Nombre	Buscar oferta por nombre
Descripción	El usuario selecciona la opción de listar ofertas y el sistema le muestra la lista de comercios disponibles que puede filtrar por nombre.
Actores primarios y secundarios	Usuario
Precondiciones	EL usuario ha seleccionado la opción de ofertas.
Flujo principal	1. El usuario teclea el nombre de la oferta o del comercio del cual busca ofertas. 2. El sistema muestra las ofertas disponibles filtradas por el texto introducido.
Flujos alternativos	2.1 No se muestra ninguna oferta 2.1.1 El texto introducido no coincide con ningún nombre de comercio 2.1.2 El texto introducido no coincide con ningún nombre de oferta

Plantilla	
ID	P03
Nombre	Mostrar ofertas activas
Descripción	El usuario selecciona la opción de listar ofertas y el sistema le muestra la lista de ofertas disponibles que puede filtrar mostrando solo las activas.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario ha seleccionado la opción de ofertas.
Flujo principal	1. El usuario activa el filtro de ofertas activas. 2. El sistema muestra las ofertas activas disponibles.
Flujos alternativos	2.1 No hay ninguna oferta activa y no se muestra ningún resultado.

Plantilla	
ID	P05
Nombre	Ver una oferta
Descripción	El usuario selecciona la opción de listar ofertas y el sistema le muestra la lista de ofertas disponibles.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario ha seleccionado la opción de ofertas.
Flujo principal	1. El usuario aplica los filtros oportunos a las ofertas. 2. El sistema muestra las ofertas disponibles para esos filtros. 3. El usuario selecciona la oferta que quiere ver. 4. El sistema muestra la información relativa a la oferta seleccionada.
Flujos alternativos	

Plantilla	
ID	P05
Nombre	Listar comercios
Descripción	El usuario selecciona la opción de listar comercios y el sistema le muestra la lista de comercios disponibles.
Actores primarios y secundarios	Usuario
Precondiciones	
Flujo principal	1. El usuario accede a la sección de comercios. 2. El sistema muestra los comercios disponibles.
Flujos alternativos	2.1 Los comercios no están disponibles y no se muestra ninguno.

Plantilla	
ID	P06
Nombre	Ver información de un comercio
Descripción	El usuario selecciona la opción de listar comercios y el sistema le muestra la lista de comercios disponibles filtrados por los parámetros especificados.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario ha seleccionado la opción de comercios.
Flujo principal	1. El sistema muestra los comercios disponibles. 2. El usuario selecciona el comercio del que quiere ver información 3. El sistema muestra toda la información relativa a ese comercio.
Flujos alternativos	

Plantilla	
ID	P07
Nombre	Buscar comercio por nombre
Descripción	El usuario selecciona la opción de listar comercios y el sistema le muestra la lista de comercios disponibles filtrados por los parámetros especificados.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario ha seleccionado la opción de comercios.
Flujo principal	1. El usuario introduce el nombre del comercio. 2. El sistema muestra los comercios disponibles filtrados por el texto introducido.
Flujos alternativos	2.1 No se muestra ninguna oferta. 2.1.1 El texto introducido no coincide con ningún nombre de comercio.

Plantilla	
ID	P08
Nombre	Ver todas las ofertas del comercio
Descripción	El usuario selecciona ver todas las ofertas del comercio del que está viendo información.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario está visualizado la información de un comercio.
Flujo principal	1. El usuario selecciona ver todas las ofertas del comercio del que está visualizando información. 2. El sistema muestra todas las ofertas disponibles de ese comercio.
Flujos alternativos	2.1 No hay ofertas disponibles para ese comercio.

Plantilla	
ID	P09
Nombre	Ver comercio en mapa
Descripción	El usuario selecciona ver la geolocalización del comercio.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario está visualizado la información de un comercio.
Flujo principal	1. El usuario selecciona ver la geolocalización del comercio del que está visualizando información. 2. El sistema muestra en el mapa la geolocalización del comercio.
Flujos alternativos	2.1 Error en la visualización del mapa. 2.1.1 No hay conexión a internet.

Plantilla	
ID	P10
Nombre	Ver comercios en mapa
Descripción	El usuario selecciona la opción de ver comercios en mapa y el sistema le muestra un mapa con la geolocalización de los comercios disponibles con los filtros aplicados
Actores primarios y secundarios	Usuario
Precondiciones	
Flujo principal	1. El usuario selecciona la opción de ver comercios en mapa 2. El sistema le muestra la geolocalización de todos los comercios disponibles con los filtros aplicados y la geolocalización del usuario como centro del mapa.
Flujos alternativos	2.1 No se muestra el mapa 2.1.1 No hay conexión a internet. 2.2 El dispositivo no puede acceder a la geolocalización del usuario. 2.2.1 El sistema notifica al usuario que la búsqueda de la localización del usuario ha fallado. 2.2.1 El sistema muestra el mapa con centro el Ayuntamiento de Santander.

Plantilla	
ID	P12
Nombre	Filtrar comercios por días de apertura
Descripción	El usuario selecciona la opción de listar comercios o mapa y podrá filtrar los comercios que mejor se adapten a su horario de apertura.
Actores primarios y secundarios	Usuario
Precondiciones	EL usuario ha seleccionado la opción de comercios o mapa.
Flujo principal	1. El usuario selecciona la opción de filtros. 2. El usuario marca los días que quiere aplicar como filtro. 3. El sistema muestra los comercios que abren los días que el usuario ha seleccionado.
Flujos alternativos	2.1 El usuario también ha marcado un horario 2.1.1 Se aplican ambos filtros a los comercios mostrados.

Plantilla	
ID	P13
Nombre	Filtrar comercios por horario de apertura
Descripción	El usuario selecciona la opción de listar comercios o mapa y podrá filtrar los comercios que mejor se adapten a su horario de apertura.
Actores primarios y secundarios	Usuario
Precondiciones	EL usuario ha seleccionado la opción de comercios o mapa.
Flujo principal	1. El usuario selecciona la opción de filtros. 2. El usuario selecciona las horas que quiere aplicar como filtro. 3. El sistema muestra los comercios que abren las horas que el usuario ha seleccionado.
Flujos alternativos	2.1 El usuario también ha marcado días de apertura 2.2 Se aplican ambos filtros 3.1 Las horas no están bien introducidas 3.2 El sistema se lo notifica al usuario y le solicita que vuelva a introducirlas

Plantilla	
ID	P14
Nombre	Filtrar comercios abiertos ahora mismo
Descripción	El usuario selecciona la opción de listar comercios o mapa y podrá filtrar los comercios que mejor se adapten a su horario de apertura.
Actores primarios y secundarios	Usuario
Precondiciones	EL usuario ha seleccionado la opción de comercios o mapa.
Flujo principal	1. El usuario selecciona la opción de filtros. 2. El usuario selecciona el filtro para mostrar los comercios abiertos en ese instante.

	3. El sistema notifica al usuario que solo se mostrarán los comercios abiertos en ese instante 3.1 Los demás filtros aunque estén marcados no se aplicarán 3.2 No será posible marcar o desmarcar ninguno de los demás filtros. 4. El sistema muestra los comercios que abren en ese instante.
Flujos alternativos	3.1 El sistema no puede acceder al reloj del dispositivo 3.1.1 No se muestra ningún comercio.

5 Diseño

5.1 Diseño arquitectónico

Para desarrollar completamente el sistema son necesarios tres pilares fundamentales

- Una interfaz gráfica sencilla e intuitiva que permita un fácil manejo de la aplicación
- Una capa dónde estará todo el código que haga posible la lógica del sistema.
- Otra capa que proporcione las operaciones para la gestión de los datos.

Ante este problema, se determina que una arquitectura basada en tres capas encaja perfectamente como solución: presentación, negocio y datos. La principal ventaja que nos ofrece esta separación en niveles es que es soportada por cualquier navegador web o sistema operativo de cualquier dispositivo móvil.

Esto ahorra mucho tiempo a la hora de desarrollar un proyecto de envergadura y aún más si el cliente solicita que el sistema se adapte a distintas plataformas.

Otro de los puntos clave que se deben tener en cuenta en el presente proyecto es la dificultad de programar una interfaz gráfica que se adapte a cualquier dispositivo Android en cualquiera de sus versiones o tamaños de pantalla.

Para el desarrollo del presente trabajo se ha utilizado una arquitectura de tres capas: capa de presentación, capa de negocio y capa de datos. Se trata de una división lógica que busca la independencia de cada capa y su posible adaptación ante cualquier cambio.

La funcionalidad de cada capa será:

- Capa de presentación. Es la capa que ve el usuario y que se encarga de mostrar y comunicarle la información requerida. Esta capa se comunica únicamente con la capa de negocio. Debe ser atractiva a la vista del usuario.
- Capa de negocio. Es la capa encargada de recibir las peticiones del usuario y enviar las respuestas tras el proceso. Es aquí donde se establecen los procesos que podrá llevar a cabo la aplicación. Se comunica con la capa de presentación para recibir las peticiones del usuario y devolverla los resultados, y con la capa de datos para solicitar o almacenar los datos requeridos.
- Capa de datos. Es la capa en la que se almacenan todos los datos y la única que accede a ellos. Esta capa recibe peticiones de almacenamiento o recuperación de datos desde la capa de negocio.

La división en capas proporciona, como ya se ha comentado, independencia entre capas. Esto facilita la reutilización de las capas en aplicaciones similares y que se puedan realizar cambios en cada una de ellas sin que afecten al conjunto de la aplicación.

En la figura 12 se muestra el diagrama de componentes del sistema en el que se representa la interconexión de las capas de la arquitectura.

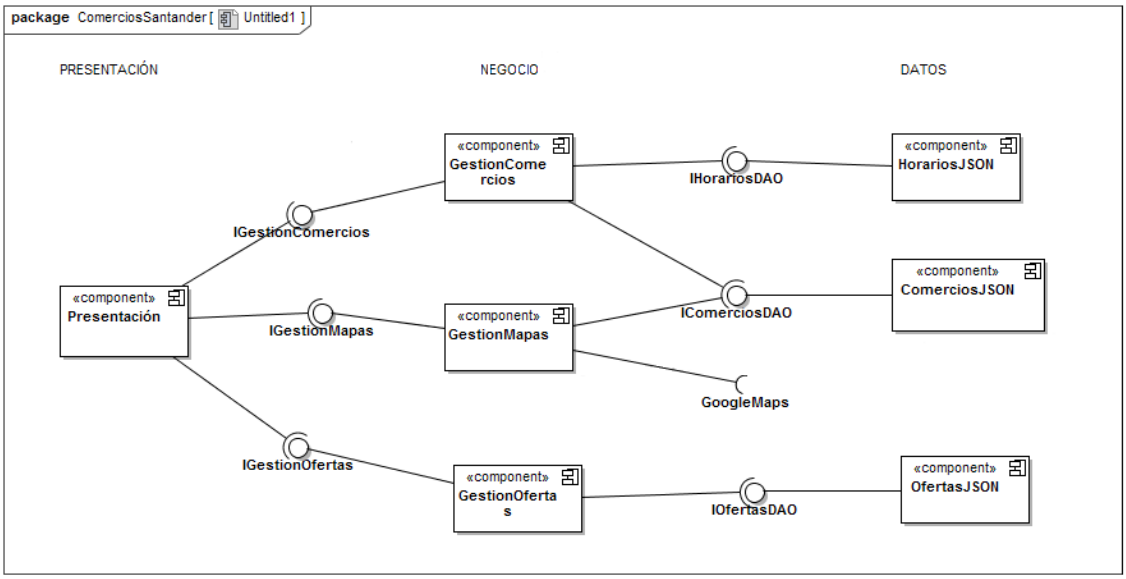

Fig. 12: Diagrama de componentes

5.1.1 Operaciones de la interfaz presentación - negocio

Detalle de las operaciones realizadas por las interfaces que conectan la capa de presentación con la capa de negocio.

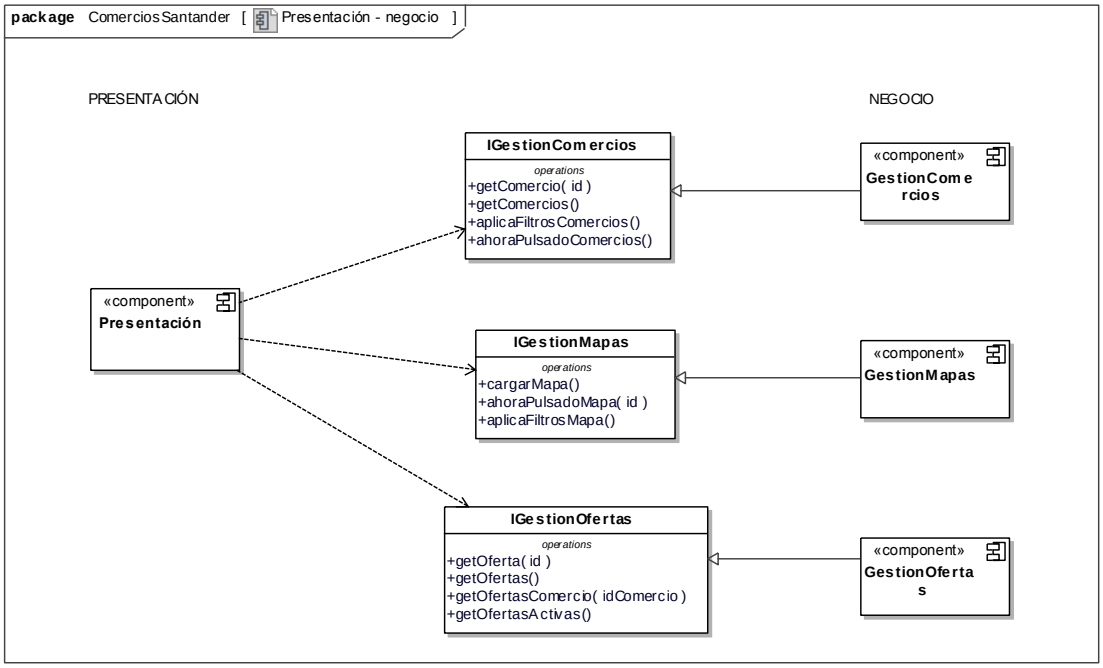

Fig. 13: Especificación de operaciones presentación - negocio

5.1.2 Operaciones de la interfaz negocio - datos

Detalle de las operaciones realizadas por las interfaces que conectan la capa de negocio con la capa de datos.

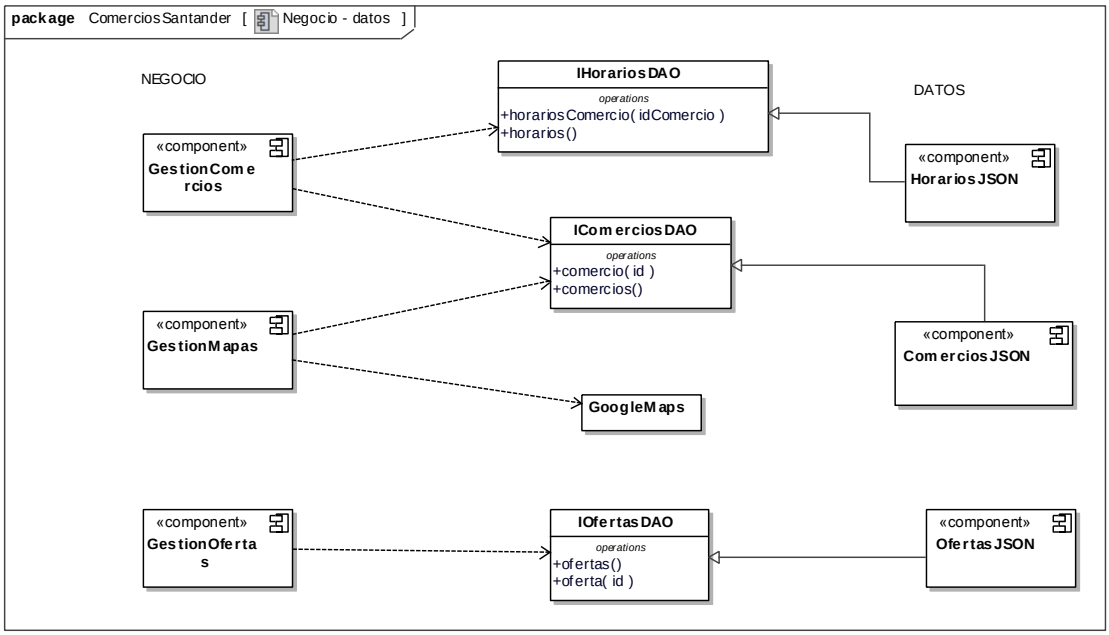

Fig. 14: Especificación de operaciones negocio - datos

5.2 Diseño detallado

En el siguiente apartado se muestra el diseño detallado de la aplicación, el cual ha sido realizado mediante un diagrama de clases.

Se trata de un modelo de dominio muy sencillo, en el que la complejidad está en lograr adaptar el modelo a los datos de los archivos JSON.

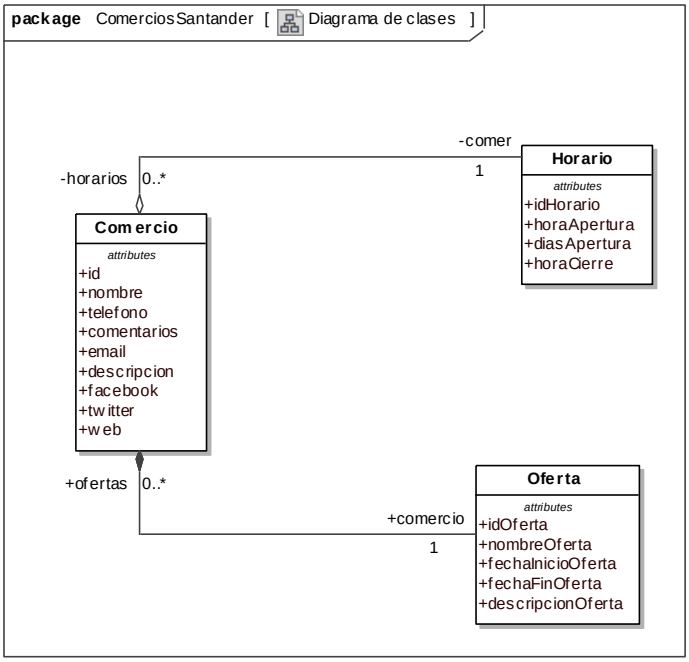

Fig. 15: Modelo de dominio

6 Implementación

El orden en que se explicará la implementación de la aplicación no es el mismo que el seguido en la realidad. Esto se debe a que constantemente se van haciendo modificaciones entre capas para desarrollar el sistema según las necesidades entre capas.

6.1 Implementación de la capa de presentación

En esta capa se recoge el código HTML5, CSS3, JQueryMobile y, ocasionalmente, JavaScript como mero creador de código mediante alguna función. También se incluyen las imágenes que el sistema requiere.

El esqueleto de la capa de presentación queda de la siguiente manera

Fig. 16: Captura de pantalla con el esqueleto de la aplicación

En la aplicación se ha empleado un solo archivo HTML sobre el que se van realizando los cambios pertinentes para mostrar la información requerida.

```
<form id="buscador">
        <input style="font-family:'Dosis'"  id="divOfPs-input" data-type="search" placeholder="Nombre del comercio">
</form>
<div id="filtros"></div>
<div id="filtrosMapa"></div>
<div id="ofertasActivas"></div>
<div class="ui-grid-a">
        <div id="atras"  style="width:auto"class="ui-block-a"></div>
        <div id="botonOfertas" style="width:auto"class="ui-block-b"></div>
        <div id="verEnMapa"  style="width:auto"class="ui-block-c"></div>
</div>
<div class="ui-grid-a">
    <div id="botonComercio"style="width:auto" class="ui-block-a"></div>
    <div id="atrasOfer"style="width:auto" class="ui-block-b"></div>
</div>
<div id="volver"></div>
<div id="map"></div>
<div id="resultadoComercios" data-filter="true" data-input="#divOfPs-input"></div>
<div id="resultadoOfertas" data-filter="true" data-input="#divOfPs-input"></div>
```

Fig. 17: Captura de pantalla de código HTML

Una vez implementado el esqueleto de la capa de presentación, resulta interesante comprobar cómo se le ha proporcionado el estilo, que al final es lo que el usuario ve y le va a resultar más o menos atractivo. Para ello, se ha usado la herramienta ThemeRoller. Ésta permite fácilmente dar el estilo deseado arrastrando los colores y las formas que se quieren poner a cada elemento. Una vez acabado el diseño, proporciona los archivos css para importarlos desde el HTML.

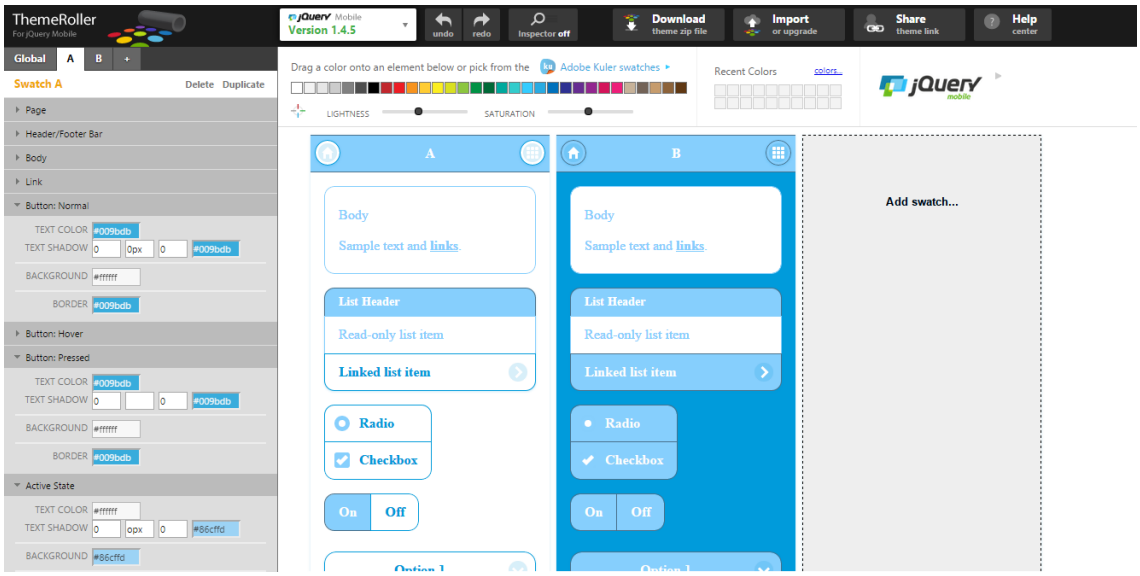

Fig. 18: Captura de pantalla de la herramienta ThemeRoller

Los archivos css que genera son fácilmente editables si algo no queda como se quiere.

```
html {
    font-size: 100%;
}
body,
input,
select,
textarea,
button,
.ui-btn {
    font-size: 1em;
    line-height: 1.3;
       font-family: 'Dosis' /*{global-font-family}*/;
}
span{
    color: #86cffd;
    font-weight: bold;
}
h2{
    color: #86cffd;
}
#map{
    border-radius: 10px;
}
```

Fig. 19: Captura de pantalla de código CSS

A continuación se muestran una serie de capturas de pantalla del sistema desarrollado.

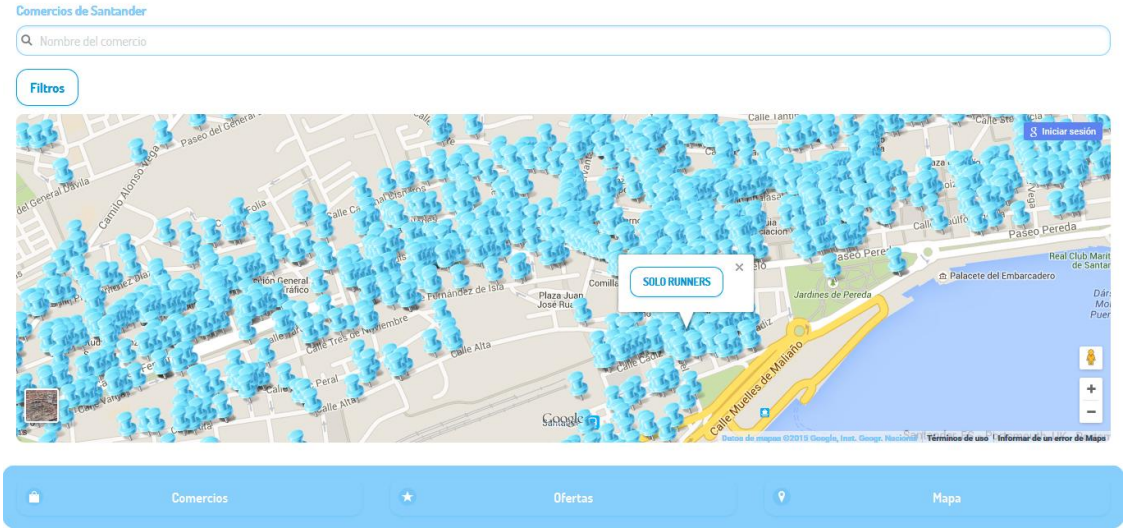

Fig. 20: Captura de pantalla del mapa

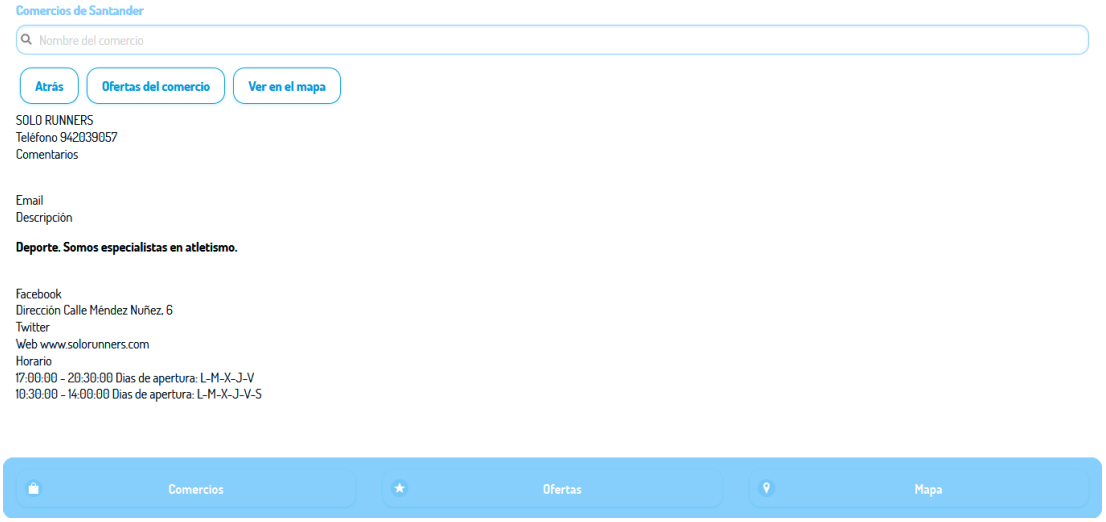

Fig. 22: Captura de pantalla de la información de un comercio

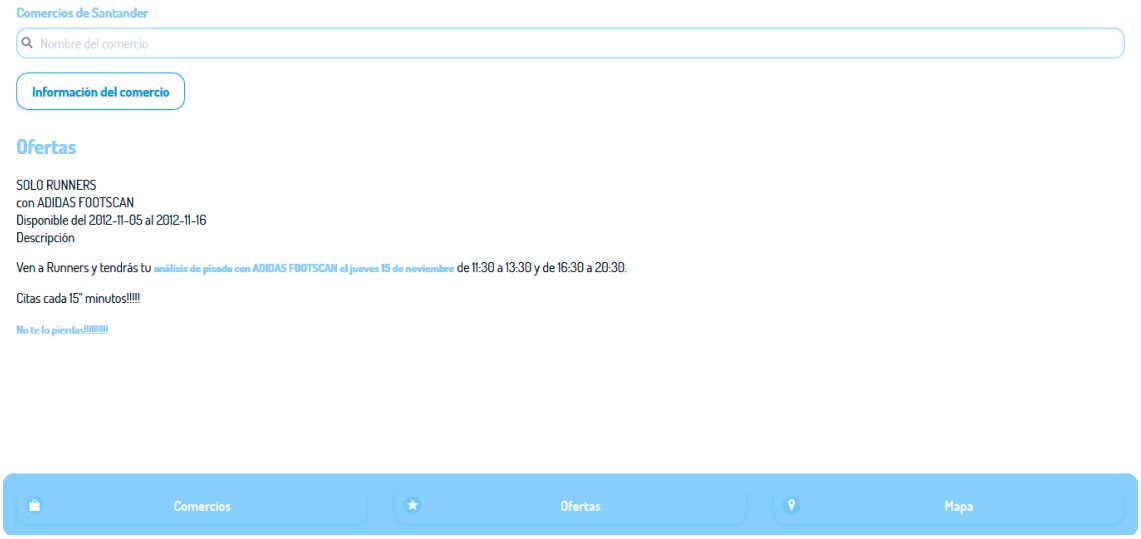

Fig. 23: Captura de pantalla de las ofertas de un comercio

Fig. 24: Captura de pantalla de los comercios

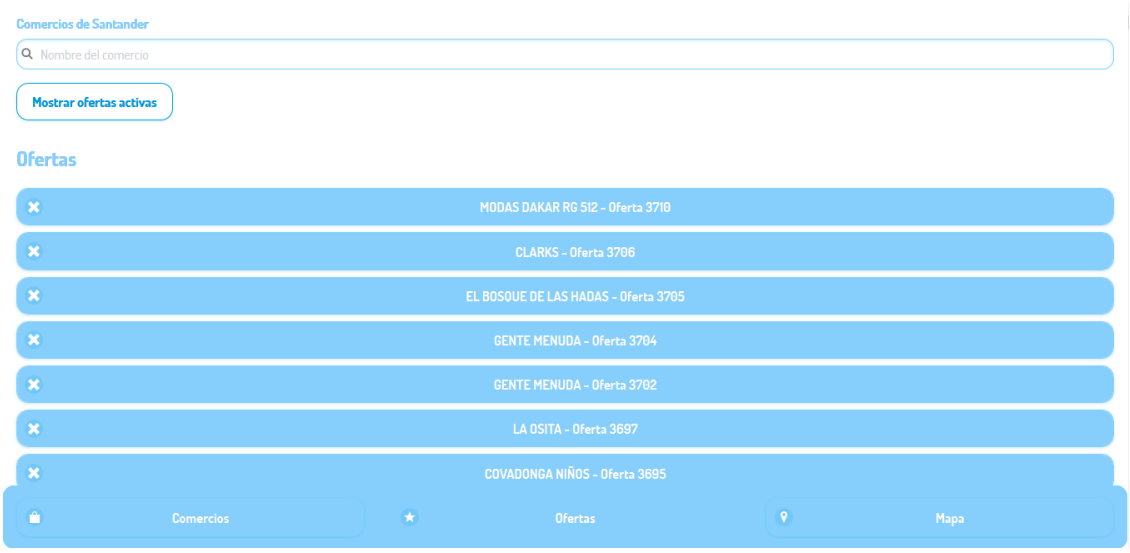

Fig. 25: Captura de pantalla de las ofertas

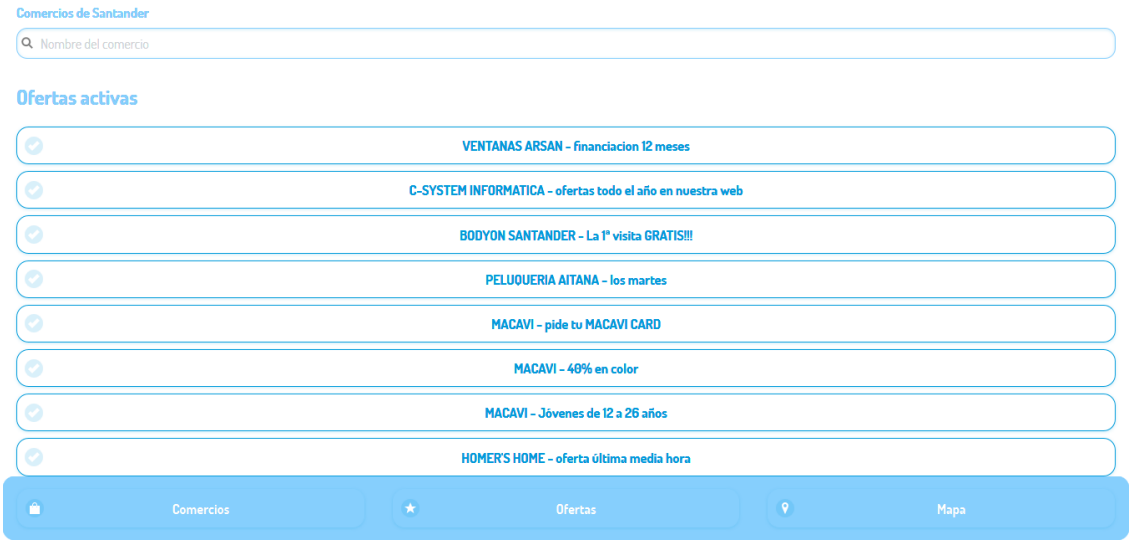

Fig. 26: Captura de pantalla de las ofertas activas

Fig. 27: Captura de la pantalla de filtros

6.2 Implementación de la capa de negocio

En la capa de negocio se sitúa toda la lógica de la aplicación. En esta aplicación, esta capa se ha desarrollado en JavaScript.

Fig. 28: Esqueleto de la capa de negocio

La figura 29 muestra parte del código JavaScript de un método de la capa de negocio. Este método filtra los comercios en función de los parámetros que se le pasan en cuanto a horas y días de apertura.

```
$.buscameDiasHoras = function(datos){
    //Primer separo del array las horas
    var horaInicio =  datos[datos.length-2];
    var horaFinal = datos[datos.length-1];
    var hInicio = Date.parse("01/01/2016 "+horaInicio);
    var hFinal = Date.parse("01/01/2016 "+horaFinal);
    var arrayId = new Array();
    datos.splice(datos.length-2,2);
    if(hFinal<=hInicio){
        alert("Las horas no están bien introducidas");
    }else{
        //recorro todos los horarios
        for (index = 0; index < horarios.length; index++){
                //Recorro los días que abre a ver si tiene los días que me pide
                var diasAbre = horarios[index].dias;
                var result;
                var continua = true;
                    for(i=0;i<datos.length;i++){
                        if(continua){
                            //Miro si tiene el día de mi lista
                            result = diasAbre.indexOf(datos[i]);
                            //si no lo tiene dejo de buscar en este comercio
                            if(result == -1){
                                continua = false;
                            }
                        }
                    }
                    var sigue = true;
```

Fig. 29: Parte del código de un método de la capa de negocio

6.3 Implementación de la capa de datos

La capa de acceso a datos está implementada con JavaScript. Mediante este lenguaje se accede a los archivos JSON y se extrae la información requerida. Los archivos JSON con los datos sobre los comercios son proporcionados, como ya se ha dicho, por la web de datos en abierto que promueve el Ayuntamiento de Santander http://datos.santander.es/.

Para el desarrollo de la aplicación se usan tres archivos JSON:

- Comercios. Contiene la información general sobre los comercios.
- Horarios. En este archivo encontramos la información sobre los horarios de apertura de los comercios.
- Ofertas. Incluye información sobre las ofertas de los comercios.

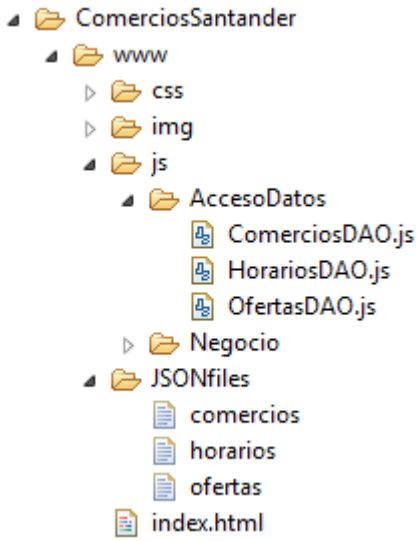

Fig. 30: Esqueleto de la capa de datos

En la siguiente imagen se muestra un ejemplo de cómo se accede a estos ficheros. Este método guarda en un array todos los comercios que hay en el fichero JSON.

```
$.getComerciosdao = function(){
    var comerDAO;
    $.getJSON("comercios", function(result){
        $.each(result.resources, function(i,comer){
            comerDAO = new Comercio(comer["ayto:codigo"], comer["dc:name"],
            comer["ayto:latitud"],comer["ayto:longitud"],comer["ayto:telefono"],
            comer["ayto:otrosComentarios"], comer["ayto:email"],
            comer["dc:description"],comer["ayto:facebook"],comer["ayto:direccion"],
            comer["ayto:twitter"], comer["ayto:web"]);
            comercios[comercios.length] = comerDAO;
        });
    });
}
```

Fig. 31: Código JS de un método de la capa de datos

6.4 Aplicación móvil

Una vez desarrollada la aplicación y comprobado su correcto funcionamiento mediante las pruebas necesarias, se procede a realizar los cambios pertinentes para poder migrar esta aplicación de web a móvil. Para ello se una el framework Phonegap.

Para que la aplicación funcione a la perfección en el sistema operativo Andorid, se le añade el pluging de geolocalización. Una vez hecho esto, se despliega en Phonegap y éste nos proporciona la apk de la aplicación, instalable en cualquier dispositivo Android.

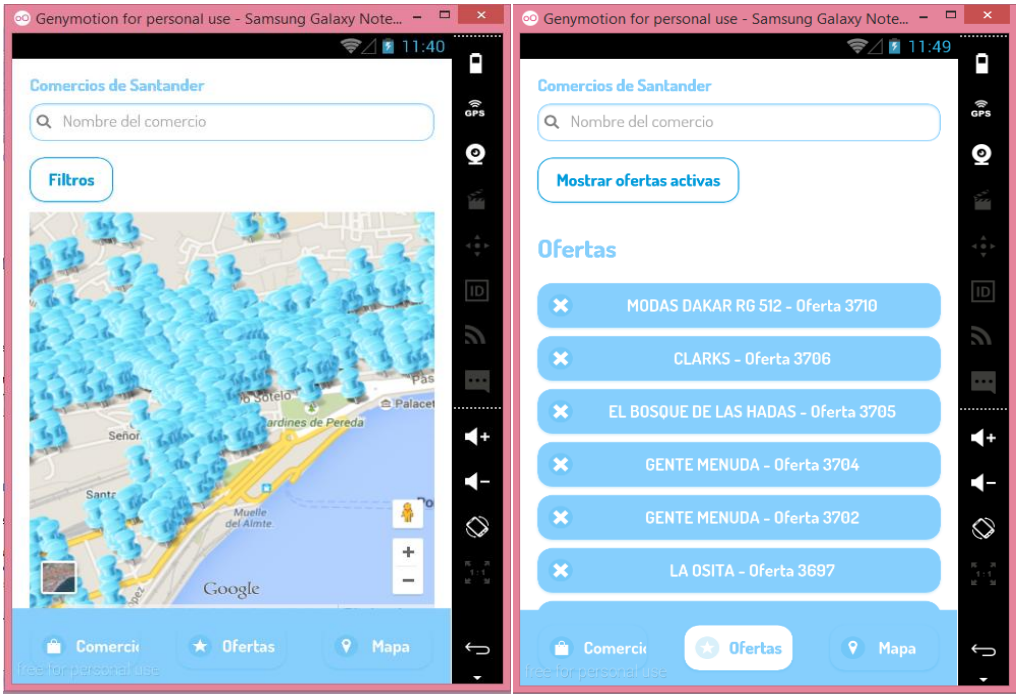

Fig. 32: Capturas de pantalla en dispositivo Android

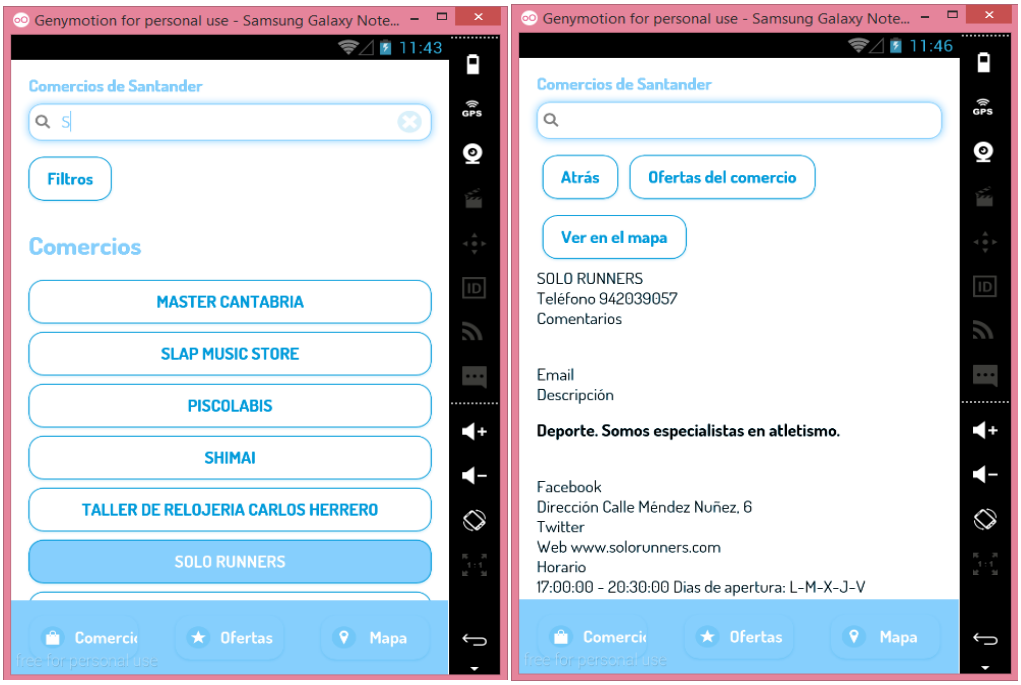

Fig. 33: Capturas de pantalla en dispositivo Android

7 Evaluación y pruebas

Es importante verificar y validar el software desarrollado puesto que cualquier sistema es propenso a contener defectos, errores o fallos que impiden su correcto funcionamiento.

La realización de pruebas para llevar a cabo la verificación y validación del software se han definido dos bloques de pruebas:

- Pruebas de desarrollo. Bloque compuesto por pruebas unitarias, de integración y de sistema.
- Pruebas de usuario. Bloque compuesto por las pruebas de aceptación.

7.1 Pruebas de desarrollo

7.1.1 Pruebas unitarias

Para desarrollar las pruebas unitarias se ha utilizado el framework para pruebas de JavaScript QUnit. Este framework permite probar los métodos desarrollados individualmente y así comprobar su correcto funcionamiento tal y como lo hace JUnit con las clases Java.

A medida que se ha ido implementando código, se han ido desarrollando dichas pruebas para comprobar que el código funciona tal y como se espera.

Sería muy extenso recoger en la memoria todas las pruebas realizadas al detalle, únicamente se muestra un ejemplo del framework utilizado con los métodos getComerciosdao() (método que guarda todos los comercios que haya en el archivo JSON en un array) y getComercio(id) (método que devuelve el comercio cuyo id se pasa como parámetro).

```
$.getComerciosdao();
test("getComercio", function(assert){
    var c = new Comercio("2852","EL GATO EN EL BALCON","43.46293211378524",
    "-3.8056039810180664","942361429","<p><strong>Tenemos los diseños de: POÈTE . MOLLY BRACKEN. EL ARMARIO DE LULÚ."
    +"TRINIDAD CASTILLO.</strong></p>\n<p><strong>Y un espacio preferencial para NUEVOS CREADORES en complementos.</strong></p>",
    "","<p><strong>Moda y complementos seleccionados para la mujer.</strong></p>",
    "www.facebook.com/elgatoenelbalcon", "Calle Rualasal, 23","","");
    assert.strictEqual($.getComercio("2852").id,
    c.id,"Se ha encontrado el comercio");
    assert.strictEqual($.getComercio("2852").nombre,
    c.nombre,"Se ha encontrado el comercio");
    assert.strictEqual($.getComercio(2852).nombre,
    "EL GATO EN EL BALCON", "Se ha encontrado el comercio");
    assert.strictEqual($.getComercio("4000"),
    undefined,"No hay comercio con ese id");
});
test("getComercios",function(assert){
    assert.strictEqual(comercios.length, 2000,"Los comercios se han cargado");
});
```

Fig. 34: Código de pruebas unitarias

Los resultados de la ejecución de estas pruebas son los siguientes

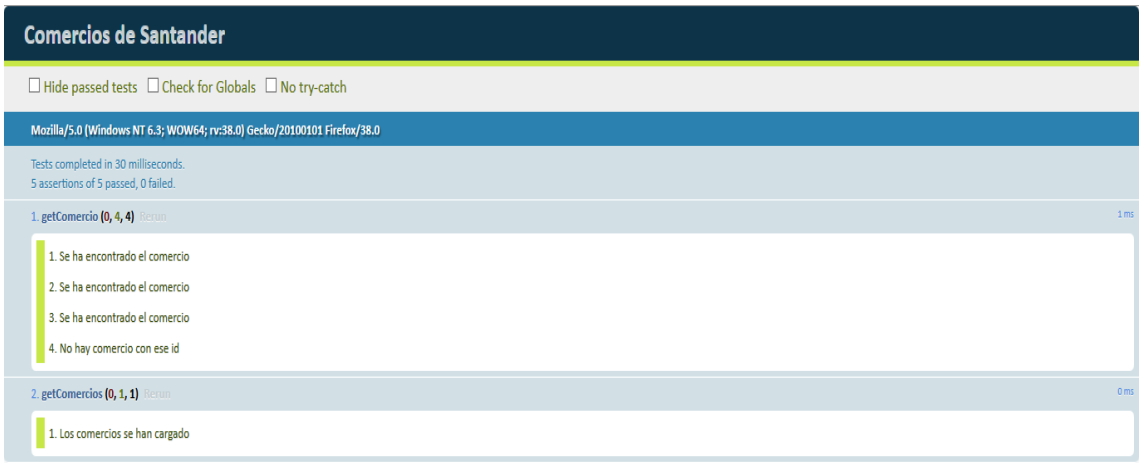

Fig. 35: Resultados de las pruebas unitarias

7.1.2 Pruebas de integración

Tras las pruebas unitarias de cada módulo de código, hay que comprobar si la integración entre estos se realiza de forma correcta. En este sentido, han de realizarse las pruebas de integración.

Como ocurre en las pruebas unitarias, el método de trabajo empleado implica que estas pruebas se lleven a cabo de modo incremental. Esto es, cada vez que se incorpore nuevo código y se realicen las pruebas unitarias correspondientes, se deberá comprobar la interacción entre todas las partes presentes hasta ese momento en el sistema.

El hecho de realizar este tipo de pruebas de forma incremental supone que, finalmente, se acabe comprobando el funcionamiento del sistema completo. La última prueba une todos los componentes para evaluar el flujo de trabajo global del sistema.

Para realizar estas pruebas se ha usado la herramienta "SelenimHQ". Ésta captura cada uno de los pasos que se van realizando en la página web manualmente, de modo que consigue registrar los elementos que interactúan en cada uno de los procesos ejecutados durante el uso de la misma. Además, permite repetir el flujo de acciones capturado, posibilitando así la automatización de las pruebas realizadas.

Nuestra aplicación antes de ser una aplicación para Android, es una aplicación web, por lo que estas pruebas pueden realizarse antes de la migración a Android.

En la figura 36 se muestra la secuencia grabada en la que se inicia la aplicación, se aplica el filtro sobre el mapa para ver los comercios abiertos en ese instante y se muestra el mapa con el resultado.

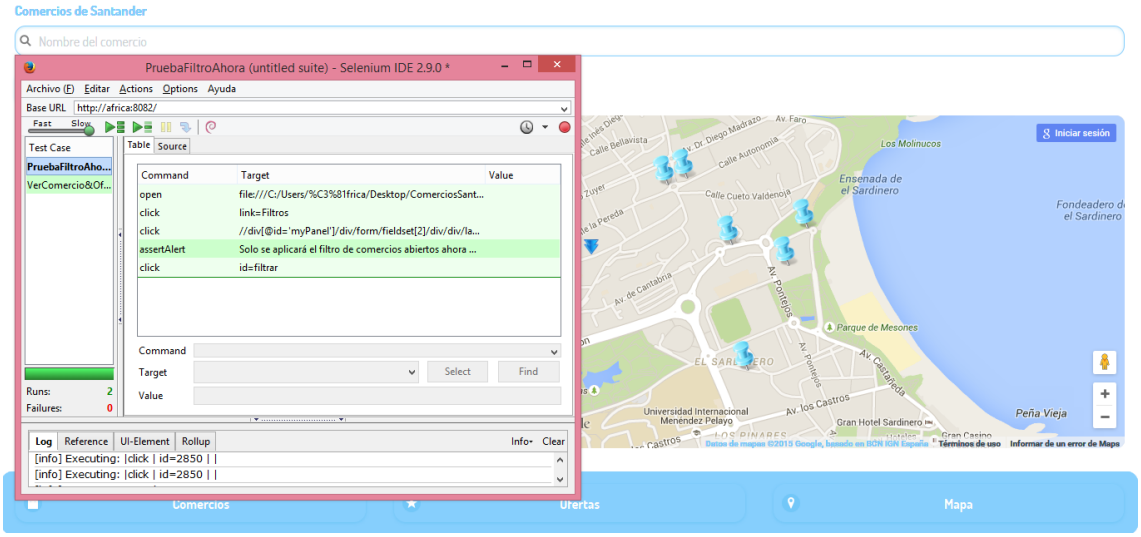

Fig. 36: Puebas con Selenium

7.1.3 Pruebas de sistema

Las pruebas de sistema verifican el comportamiento del sistema en su conjunto. Normalmente los fallos funcionales se suelen detectar en los niveles anteriores (pruebas unitarias y de integración). Este nivel es más adecuado para comprobar los requisitos no funcionales como rendimiento o usabilidad.

7.1.3.1 Pruebas de rendimiento

Estas pruebas se han realizado en varios dispositivos. La versión web de la aplicación se ha probado en los navegadores Chrome, Firefox e Internet Explorer tanto en un ordenador portátil como en un dispositivo Android (modelo BQ E5). Además, la aplicación móvil se ha probado en el mismo dispositivo Android y en la máquina virtual de Genymotion sobre los dispositivos Samsumg Galaxy Note 4.1.1, Google Nexus 10 y HTC One 4.4.4.

La aplicación se ejecuta de manera fluida en cualquier dispositivo aunque requiere un tiempo mínimo de carga de datos y acceso a las funcionalidades del dispositivo. En general se han obtenido buenos registros.

7.1.3.2 Pruebas de usabilidad

Se dio a probar la aplicación a diferentes usuarios pertenecientes a grupos de edades distintos para comprobar que la aplicación presentaba una interfaz amigable y su funcionamiento es fácil de intuir.

Los resultados de dichas pruebas fueron los esperados.

7.2 Pruebas de usuario

7.2.1 Pruebas de aceptación

El objetivo de las pruebas de aceptación es validar que un determinado sistema cumpla con el funcionamiento esperado y permitir también que el cliente o usuario final determine su aceptación desde un punto de vista relativo a la funcionalidad, rendimiento y sencillez.

En el presente proyecto estas pruebas se han ido realizando a medida que se terminaba un nuevo módulo. Se recopilaban opiniones y los puntos en que se debía mejorar o cambiar, aunque esto es muy ambiguo, debido al número de usuarios que esta aplicación puede tener.

8 Conclusiones y trabajos futuros

Al finalizar un proyecto, es importante hacer una reflexión sobre el trabajo realizado. Es conveniente analizar los éxitos, las dificultades y las partes que se podrían mejorar.

8.1 Conclusiones generales

Se ha implementado una aplicación web y, a partir de esa implementación, una aplicación móvil para dispositivos Android. Se ha llevado a cabo sobre una arquitectura en tres capas. Esta arquitectura nos facilita la modificación de alguna de las capas debido a la independencia entre ellas.

El sistema permite que el usuario final pueda obtener información geolocalizada de los comercios de la ciudad de Santander, así como de las ofertas que estos ofrecen. Además, incorpora filtros que permiten consultar al cliente los comercios abiertos a cierta hora cierto día o en el propio instante de la consulta.

Se ha conseguido que la aplicación sea compatible con dispositivos móviles de diferentes tamaños de pantalla, además de funcionar en cualquier navegador web, objetivo principal del trabajo. Para ello, se han utilizado los lenguajes HTML5, CSS3 y JavaScript, además de otros frameworks y plugins.

8.2 Conclusiones personales

El hecho de desarrollar una aplicación desde cero, partiendo del desconocimiento de los lenguajes utilizados para tal fin me ha aportado lo que expongo a continuación:

- Capacidad de desarrollo de un proyecto útil para los ciudadanos o visitantes de la ciudad de Santander.
- Afán de superación. El reto era desarrollar una aplicación desde cero implementada con lenguajes y herramientas desconocidas y se ha conseguido.
- Adquisición de nuevos conocimientos informáticos hasta ahora desconocidos que permiten ampliar mis estudios.
- Puesta en práctica de conocimientos adquiridos durante la carrera.

8.3 Líneas de trabajo futuras

El planteamiento de líneas futuras de trabajo sobre el sistema desarrollado se pueden valorar diversos aspectos.

Por un lado, la ampliación del sistema, añadiendo datos de otros conjuntos disponibles también en la web *http://datos.santander.es/*. Este servicio de datos en abierto también proporciona otros datasets sobre comercios como noticias o ferias de stock que podrían resultar interesantes para los usuarios de la aplicación.

Por otro lado, la mejora del sistema desarrollado. Este perfeccionamiento incluiría:

- Desarrollo de una base de datos en la que se alojarían los datos consultados y actualizados al abrir la aplicación. De ese modo, las consultas se harían sobre la base de datos y no sobre el fichero JSON, lo que haría la aplicación más eficiente.
- Implementar un trazado de ruta que permita conocer el camino desde la posición del usuario hasta que comercio que desee.

Bibliografía

[1] Sommerville, 2012. Ingeniería del Software. 9a Edición, Addison-Wesley. 2012.

[2] J.Arlow e I.Neustadt. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Addison-Wesley. 2005.

[3] Alistair Cockburn, “Writting Effective Use Cases”. Addison-Wesley, 2000.

[4] Richard N. Taylor, Nenad Medvidovic y Eric M. Dashofy. “Software Architecture: Foundations, Theory, and Practice”. 2009.

[2] Estudio 'Sociedad en Red 2013' elaborado por la ONTSI
http://goo.gl/mI6Edt

[3]. SeleniumHQ
http://goo.gl/FnS6Wh

[4] PhoneGap
http://goo.gl/b3DkoG

[5] PhoneGap
http://goo.gl/eWN2Ke

[6] HTML
http://goo.gl/dg1dvU

[7] HTML
http://goo.gl/w2P2e5

[8] HTML
http://goo.gl/pmbmCn

[8] JavaScript
http://goo.gl/hVLuQs

[9] JavaScript
http://goo.gl/Mr3vk7

[10] CSS
http://goo.gl/fj8AN4

[11] Qunit
http://goo.gl/z5hNAA

[12] JSON
http://goo.gl/ExSK14

[13] Genymotion
https://goo.gl/3YIWPF