UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Ingeniería de Software y Sistemas Informáticos

Comparador de precios colaborativo

Trabajo Fin de Máster

Presentado por: Millán Jaureguizar, Diego

Director/a: Sicilia Montalvo, Juan Antonio

Ciudad:
Fecha:

Resumen

Nos encontramos en un mundo donde Internet es el eje central de las comunicaciones y las relaciones sociales. Para adaptarnos a esta realidad, debemos mantener nuestros conocimientos al día. Con esa finalidad, y tras realizar el presente Master, damos nuestros primeros pasos en ese mundo realizando un proyecto que nos adentra en el desarrollo de aplicaciones actuales, cliente móvil y servidor en la nube, con características sociales. Para ello realizamos un software de búsqueda in-situ de artículos geo-posicionados que nos permita encontrar aquellos productos más baratos, contando con la colaboración de otras personas.

Para su elaboración, contamos de una versión reducida de la metodología Métrica v3 con prácticas de seguridad y consideraciones de usabilidad para el diseño de interfaz de usuario. Este desarrollo nos ha permitido conocer de primera mano las técnicas y tecnologías necesarias y nos ha dado la posibilidad de enfrentarnos a nuevos y desafiantes retos.

Palabras Clave: Internet, Móvil, Cloud, Geo-Posicionamiento, Social

Abstract

We are in a world where Internet is the main communication and social relationship link. We have to stay up to date to adapt ourselves to this reality. In order to accomplish that, and after the completion of the current Master, we proceed to take our very first steps in that world with the developing of a project that introduce us in the current applications state, mobile client and cloud server, with social characteristics. So as to we made an in-situ search application of geo-positioned items that will allow us to find the cheapest ones with the collaboration of other people.

We counted on a stripped-down version of Métrica v3, with security practices and usability considerations for the user interface design included, to prepare it. This development has allowed us to know very first hand methods and technologies required and has given us the possibility to confront new and defiant challenges.

Keywords: Internet, Mobile, Cloud, Geolocation, Social

Índice de contenidos

Índice de tablas

Índice de figuras

1. Introducción

En este documento se recoge el trabajo de preparación, desarrollo y conclusiones relativo al proyecto de realización de una aplicación comparadora de precios colaborativa usando tecnologías y métodos aprendidos durante la realización del master de ingeniería del software y sistemas informáticos.

La finalidad de la realización de esta aplicación no era tanto encontrar una solución novedosa a un problema concreto tanto como demostrar la validez de los conocimientos recibidos durante el año de trabajo en el master. Aun así, tratamos de realizar una aplicación original, aunque en la medida de nuestras limitadas posibilidades. En el mercado, como ya veremos, existen numerosas soluciones para las compras preestablecidas; pero no tanto para artículos no catalogados como son los recuerdos de los viajes (de donde surgió la idea).

La solución se planteó siguiendo los pasos de Métrica v3, adaptados a la realidad del equipo de trabajo y recursos disponibles, así como una aplicación ligera de las metodologías ágiles para el desarrollo. Además, se añadió la consideración y evaluación de los riesgos de seguridad para aplicar buenas prácticas de codificación estandarizadas enfocadas a la creación de aplicaciones seguras, sin olvidar la usabilidad y accesibilidad.

1.1 Justificación

Inicialmente, el problema a resolver era, aparentemente, trivial y puede parecer que una solución tecnológica fuera excesiva: En cualquier viaje llega el momento de comprar recuerdos para la familia, amigos o compañeros de trabajo (los muy trúhanes). Estos recuerdos, o *suvenir,* se suelen encontrar en multitud de tiendas o puestos callejeros diseminados por los lugares que visitamos. Para complicar algo tan trivial en principio, al menos para una persona que no quiere gastar excesivamente o sentirse engañado, aquellos se pueden encontrar en puestos distintos con distinto precio, lo que dificulta recordar y localizar los que nos interesaron al precio más asequible. Vemos que si, como suele ser el caso general, tenemos un presupuesto limitado tanto en tiempo como dinero **¿podemos asegurarnos de comprar aquellos recuerdos que os podemos permitir al mejor precio?** Y **¿Dónde están, y cómo puedo hacer el menor esfuerzo para adquirirlos?** Al final las prisas y la impaciencia nos hacen comprar lo primero o lo último que encontramos, dejando un mal sabor de boca (al menos cuando se hace con un poco de interés).

¿Por qué termina ocurriendo esto? Las personas no dejamos de tener una memoria a corto plazo (sólo tras la noche algunos de los recuerdos pasan a la memoria a largo plazo, quedando fijados en la memoria a base de un mantenimiento más o menos constante [1]) que en el fragor de la tareas que realizamos se ve limitada aún más, sobre todo si estamos en situaciones de cierto estrés, como puede ser estar en un entorno desconocido, con dificultades para la compresión (idioma, cultura…) y falta de tiempo (ese avión que se marcha puntualmente cuando en el viaje de ida acumulaba retraso).

¿Podemos usar la tecnología para facilitarnos tareas ingratas como esta, repetitivas e incluso agotadoras? ¿No es esta su finalidad? Y siendo así, ¿pueden servir los conocimientos aprendidos en el presente máster para idear alguna solución que pueda servir para resolver este pequeño problema y abrir, quizás, las puertas a resolver otros problemas de mayor calado que se resuelvan con fundamentos y tecnologías análogas a las utilizadas? Pensamos que sí y esto hace que un pequeño problema como el presentado en este trabajo, solucionado usando tecnologías que combinan varias plataformas (cliente-servidor, sistemas operativos, bases de datos, lenguajes, movilidad, seguridad, accesibilidad y facilidad de uso, etc.) sea relevante. Porque en cualquier caso se puede considerar una iniciación a la implementación de soluciones de movilidad (en un principio usando s*martphones*, pero podría ampliarse a *wearables* próximamente) en el que prima la correcta implementación del sistema para lograr la mayor satisfacción del usuario. De no ser así, esta como otras soluciones carecen de sentido.

Creemos, además, que la solución que trataremos de explicar con mayor detalle en posteriores capítulos podría usarse para otro tipo de soluciones como bitácoras (de viaje, por ejemplo), auditorías/control/seguimiento de entornos urbanos o rurales, de personas, o de cualquier cosa, lugar o persona que podamos ubicar espacialmente y adjuntar información contextual relevante para nuestro quehacer.

1.2 Planteamiento del trabajo

El problema de ubicar dónde se encuentran los objetos de interés (*suvenir* de nuestro ejemplo) y como llegar a ellos de la manera más directa, se puede plantear desde el aprovechamiento del concepto de “internet de las cosas” tan en boga hoy día. El uso generalizado de teléfonos “inteligentes” o *smartphone* [2] [3], así como el incipiente y prometedor aparición de los dispositivos *wareables* [4], todo ello conectado con internet hace que se puedan implementar soluciones imaginativas que expandan nuestras

capacidades en tareas incluso triviales usando técnicas que mimeticen a las conocidas redes sociales.

Por ello, planteamos la solución al problema original usando, inicialmente, un S*martphone* dotado de cámara de fotos, pantalla táctil, *G.P.S*. y conectividad (internet vía wifi o datos móviles). No creemos que fueran unas exigencias fuera de lo común para lo que hoy en día se encuentra en el mercado, cuando dispositivos de esas características se pueden encontrar en productos considerados de entrada o de las gamas más bajas [2,5] [6].

Bien disponiendo, en principio, de uno de estos teléfonos, un usuario podría utilizar este conjunto de características con una aplicación creada ex-profeso que permita recoger y contextualizar la información para su posterior explotación de una manera automática, que facilite la toma de decisiones al usuario de una manera clara y concisa.

Esta aplicación es nuestro dominio de actuación y lo que hemos pretendido desarrollar a lo largo de este proyecto para solucionar el problema inicial usando las habilidades adquiridas en el presente master. Creamos una aplicación móvil que permita, de manera rápida y sencilla, adquirir la información pertinente. Que además permite compartir dicha información con otros usuarios de la aplicación previamente registrados y aceptados por el usuario, de tal manera que los datos disponibles puedan proceder de otras fuentes aparte de los generados por el mismo. Finalmente, el usuario puede utilizar los datos para su toma de decisiones de forma rápida, fácil e intuitiva. Este fue el objetivo de nuestro trabajo.

En definitiva, la aplicación se apoya en una base de datos centralizada en servidor, alimentada por los dispositivos móviles y, opcionalmente, una aplicación de *backend* web que permite la recolección de información geo-posicionada junto con otra información relativa al contexto (como puede ser fotografía, precio, observaciones, etc.) con la que se construye conceptualmente un elemento (producto). De esta manera, cada información geo-posicionada describe una realización particular de tal elemento. Con la información así agrupada, como si de una categorización por elemento se tratase, el sistema podría escoger qué realizaciones son las más interesantes, según un criterio inicialmente prefijado por el usuario, como puede ser el precio, e indicarle como llegar a ellas; por ejemplo, siguiendo la ruta más corta desde el lugar en el que se encuentra, o solicitando la colaboración al resto de conocidos registrados del usuario.

1.3 Estructura de la memoria

La presente memoria ha sido descompuesta en cinco capítulos principales (aparte de los reservados para las referencias bibliográficas que se realizan a lo largo de la memoria y figuras adicionales que ayuden al lector a profundizar en algunos aspectos mostrados de forma general en los capítulos que a continuación detallaremos).

Los dos primeros capítulos introducirán al lector en el estado actual de las aplicaciones relacionadas con el desarrollo que tratamos de realizar, describiendo las distintas opciones que se pueden encontrar para que introducir al lector al estado actual de las aplicaciones de comparación de precios, los conceptos que subyacen bajo la implementación general de una solución de este tipo y terminaremos detallaremos cuales son los objetivos que buscamos alcanzar con el trabajo.

Los dos siguientes describirán la parte práctica, la metodología usada para conducir el trabajo, indicando los pasos y actividades seguidos, y pormenorizaremos los detalles del trabajo práctico que lleva a la construcción física del sistema final.

Finalizaremos la memoria con las conclusiones extraídas del trabajo durante el periodo que nos ha llegado su realización, en base al devenir del mismo, problemas, incidencias y resultado final obtenido

Más en detalle:

1. Introducción
   El capítulo en el que nos encontramos, se trata de justificar la realización del presente trabajo, situando al lector en el centro de la idea de la que partimos, para a continuación realizar un breve planteamiento del trabajo a realizar y finalmente describiendo las estructura de la memoria para así permitir entender y localizar las partes que interesen revisar en cada momento.

2. Contexto y estado el arte
   En el segundo capítulo de la memoria, presentamos el estado actual de las aplicaciones para móvil o web que realizan una labor parecida a la que pretendemos lograr con nuestro trabajo, así como las diferencias con las mismas.

Además, comentaremos que conceptos teóricos, técnicas y tecnologías se aplican a la construcción de este tipo de aplicaciones, las cuales servirán de ejemplo y referencia para la realización de nuestro trabajo.

3. Objetivos concretos y metodología de trabajo.
   En el tercer capítulo tratamos el objetivo que se trata de lograr con el trabajo de fin de master, así como los objetivos parciales o específicos que se deben ir alcanzando para, al completarlos todos, conseguir dicho objetivo. Estos objetivos específicos se toman como parte importante del proceso de descomposición del problema principal.
   Añadiremos, también, una descripción de la metodología de trabajo, incidiendo en los pasos tomados dentro de la misma, seguidos en del desarrollo práctico del trabajo.

4. Desarrollo específico de la contribución
   En el penúltimo capítulo práctico del presente trabajo presentaremos la realización práctica del desarrollo, paso a paso, siguiendo los procesos de la metodología específica de desarrollo esbozada en el capítulo anterior. Se incide en la identificación de requisitos (funcionales / no funcionales / de seguridad), en el modelado (análisis) y análisis de riesgos de seguridad, para pasar a continuación al diseño concreto de la aplicación (modelado físico, diseño de interfaz de usuario contemplando, en la medida de nuestras posibilidades, usabilidad y accesibilidad).
   Finalmente, se incluye una descripción de los mecanismos que usaremos para evaluar si se consiguen alcanzar los objetivos que se propusieron en el tercer capítulo y con qué grado.

5. Conclusiones y trabajo futuro
   En el último capítulo práctico de la presente memoria indicaremos las conclusiones extraídas de la realización completa del trabajo, de sus problemas, incidencias y soluciones. Además planteamos posibles escenarios futuros o líneas de trabajo a realizar que se presenten interesantes a partir de lo conseguido y de las ideas obtenidas a partir de la realización conseguida.

Por supuesto, se incluye un capítulo (el sexto) de referencias.

2. Contexto y estado del arte

Tuvimos que analizar el panorama actual de aplicaciones desarrolladas para determinar la existencia de una aplicación como la que pretendíamos desarrollar: si hay soluciones parciales que pudieran usarse o si por el contrario la única forma de conseguir el objetivo es desarrollar una nueva. Una visión general del mismo sirve además para recoger ideas, métodos y tecnologías que nos pueden servir para la sacar adelante el proyecto; al fin y al cabo, como decía Bernardo de Chartres "somos como enanos a los hombros de gigantes. Podemos ver más, y más lejos que ellos, no por alguna distinción física nuestra, sino porque somos elevados por su gran altura" [6].

Inicialmente surgen un tipo de aplicaciones online que no dejan de ser una evolución de las antiguas aplicaciones de tareas de las primitivas (a ojos de una persona de hoy día) agendas personales o *P.D.A*. Aquellas han llevado a internet, a la web 2.0, el concepto de lista de comprobación / lista de tareas haciéndolas accesibles desde cualquier lugar y permitiendo compartirlas entre usuarios.

Ya más específicamente, y más relacionado con el fin del presente proyecto, en el mundo de las aplicaciones online se pueden encontrar multitud de aplicaciones y sitios web que tratan de ayudar a sus usuarios con listas de tareas concretas como puede ser la lista de la compra, para ayudarles a ahorrar al realizar la compra con los productos de primera necesidad y, finalmente, también con compras más esporádicas (como bienes de consumo en los que podemos incluir tecnología, regalos, etc.).

En último lugar, trataremos de describir las tecnologías implicadas, que hace que la realización de aplicaciones de este tipo sean posibles y se hayan popularizado en los últimos años, gracias en gran medida a la generalización del uso de los teléfonos inteligentes.

2.1 Estado del arte

Tras una exhaustiva búsqueda de aplicaciones web y móvil para realizar las tareas que pretendemos solventar con nuestro desarrollo, podemos categorizar aquellas en tres tipos de aplicaciones.

- Aplicaciones de listas compartidas (listas de tareas / "*todo*")
- Comparadores de precios
- Aplicaciones mixtas (listas de compra)

Desarrollamos a continuación sus características, desde la óptica del problema que tratamos de solucionar.

2.1.1 Aplicaciones de listas compartidas

Se pueden encontrar aplicaciones para gestionar y compartir listas de tareas (como pueden ser listas de la compra, para el caso que nos compete) con otros usuarios, aparte de aplicaciones de listas personales más básicas que no son de nuestro interés. En su mayoría este tipo de aplicaciones permite su uso y comunicación de cambios casi en tiempo real, como si de una aplicación de mensajería online se tratase; muy útil para los amos de casa olvidadizos. Dejaremos de lado aquellas aplicaciones de tareas, comprobaciones y demás que no sean multiusuario.

Como aplicaciones de listas, que son directamente aplicables a listas de compra, podemos reseñar las siguientes: Google Tasks, Remember de Milk, OutOfMilk (simpáticos nombres que claramente expresa lo que podemos conseguir con ellas) y WunderList entre muchas. Existen aplicaciones que poco a poco han ido ampliando su radio de acción incluyendo utilidades para los negocios como todoist y que quizás no sean tan accesibles para el tipo de usuario al que estamos enfocados, pero que no hay que dejar de lado por su versatilidad. Todas ellas cuentan con cliente web y cliente móvil.

Con uso algo más peculiar de las capacidades de los móviles actuales, encontramos Shopping List Pro, basada en Google Tasks permite introducir las listas usado el reconocimiento de voz de nuestros teléfonos inteligentes, ListOn o FiveFly, aplicaciones en las que se pueden añadir fotos, añadir nuevos elementos y compartir tanto los elementos (con su fotografía) como las listas.

Comparativa (en función de las características comunes que nos son de utilidad).

Aplicación	Web	Artículos	Re-uso	Recursos
ListOn	-	Sí	-	Sí
Google Tasks	Sí	-	-	-
MyShopi	-	Sí	Sí	-
Remember the milk	-	-	-	-
Out of milk	-	-	-	-
WunderList	Sí	-	-	-
TodoIst	Sí	-	-	-
ShoppingListPro	-	-	Sí	Sí

Tabla 1 Comparativa de aplicaciones de listas compartidas

Como se puede extraer de la tabla anterior, ha primado la búsqueda de aplicaciones que permitan compartir listas con otros usuarios. Es interesante apreciar la posibilidad de reusar las listas anteriores, disponer de artículos ya introducidos anteriormente y poder acceder desde la web, aparte de desde nuestro teléfono móvil. Por supuesto, no olvidamos el aprovechamiento de los recursos de los teléfonos móviles.

2.1.2 Comparadores de precios

Como hemos introducido brevemente en el inicio de este capítulo, ya se pueden encontrar multitud de aplicaciones de comparación de precios. Podemos categorizarlas en dos grande grupos, en base al entorno de compras (online / presencial) y al tipo de servicio que prestan, como servicio puntual de compra (en base o no a ofertas publicitadas por el vendedor) o como herramienta de uso más cotidiano y de cercanía:

2.1.2.1 Compra on-line

En esta categoría podríamos incluir sitios web como: www.idealo.es, www.kelkoo.es, www.ciao.es, www.shopydoo.es, www.twenga.es, www.suferpricer.es y otras muchos sitos online de comparación de precios específicos que no entran en el ámbito que queremos abarcar (sitios como www.trivago.es, www.seguros.es, www.rastreator.es, www.acierto.es, por nombrar algunos de los más conocidos gracias a sus campañas de publicidad en televisión). Muchos de estos sitios webs se complementan con aplicaciones móviles, por lo que no deben verse únicamente como aplicaciones web.

La idea general de estos sitios webs es la de proporcionar un buscador de productos, para lo que cuentan con un extenso catálogo categorizados de tal manera que les permiten, una vez realizada dicha búsqueda, presentar al usuario el producto y/o productos similares ordenados por precio. Para ello, por supuesto, se alimentan de los precios más o menos

actuales de los mismos, lo que lleva en algunos casos a pensar en la dependencia de estos lugares con los vendedores finales de los artículos. Al fin y al cabo no dejan de ser un escaparate comercial que atrae visitas a los lugares de los comerciantes [8]. Las ventajas de este tipo de aplicaciones son muchas y están claras. Por una parte, no hace falta moverse de casa. Miles de productos están al alcance de un solo *click*. Por otra está el precio. **Comprar online es más barato que ir a una tienda física.**" [9,10], lo que explica el "crecimiento imparable" [11] de estos sitios.

Como componente social, algunos de estos sitios web incorporan la posibilidad de registrarse para incluir opiniones o consideraciones sobre los distintos artículos hallados, sus condiciones, precios u ofertas concretas.

Podemos tener obtener una visión global de este tipo de aplicaciones a partir de la siguiente figura:

Aplicación	Catálogos	Ofertas	Comparador	Comunidad	Cliente
Idealo.es	Sí	Sí	Sí	Sí	Nativa
Kelkoo.es	Sí	Sí	Sí	Sí	Nativa
Ciao.es	Sí	Sí	Sí	No	Web
Shopydoo.com	Sí	Sí	Sí	Sí	Web
Twenga.es	Sí	Sí	Sí	Sí	Web
SurfPricer.com	Sí	Sí	Sí	No	Nativa

Tabla 2 Comparativa de comparadores compra on-line

Podemos comprobar que las aplicaciones presentan bastantes similitudes, probablemente debido a la feroz competencia entre comparadores (como se puede apreciar además por las continuas apariciones en prensa de algunos tipos de ellos). Lo que apreciamos que les distingue es el tratar de apoyarse en comunidades de usuarios que evalúen los productos que aparecen y en el tipo de aplicaciones que usa, nativas (es decir, con aplicaciones específicas para el móvil en el que se ejecutan) o mediante web *responsive* o de diseño adaptable [12,13].

2.1.2.2 Aplicaciones para compra de proximidad

En este apartado se incluyen aplicaciones como radarprice (android, IOS), soysuper (android, IOS) y dondetuscompras (móvil). Este tipo de aplicaciones (que pueden incluir interfaz web o no, como ocurre con la primera) están más orientadas al comercio de proximidad (entendiendo como tal al que se incluye en una zona arbitraria seleccionada por el usuario y que se presupone alcanzable).

Al igual que en el primer grupo de aplicaciones, estas cuentan con un extenso catálogo de productos diferenciándose quizás, y aquí la parte más interesante en lo que se refiere al nuestro desarrollo, en la forma de seleccionarlos. Mientras que unas aplicaciones se buscan por descripción, a la manera de un buscador web [soysuper, dondetuscompras], otras permiten formas más orientadas a la localización de productos presentes, usando búsqueda por códigos de barra [radarprice], para la que se usa al menos una capacidad de los móviles inteligentes actuales. Al estar más enfocados al día a día de un particular, no suelen incluir capacidades sociales, como compartir opiniones u otras capacidades usuales (mensajería, notificaciones) en las redes sociales.

Mencionaremos, de pasada, aplicaciones que por su ámbito regional de actuación no nos son interesantes inicialmente pero que técnicamente muestran el estado de este tipo de soluciones. Aplicaciones como ShopSavy y Shopular parecida a dondetuscompras, junto con ShopAdvisor que ya incorpora capacidades de red social (como son comentarios y puntuaciones); http://www.scanlife.com/ y Shopping List (IOS) en la órbita de lo que sería radarprice, y con capacidades sociales PurchX más cercano a lo que podríamos llegar a realizar con el presente desarrollo.

Podemos resumir las características de estas aplicaciones con la siguiente tabla:

Aplicación	Catálogo	Tienda	Web	Scan	Envío	Listas
Radarprice	No	Sí	Sí	Sí	No	No
Dondetuscompras	Sí	No	Sí	No	No	No
Shopsavy	Sí	No	No	No	No	No
Shopular	Sí	No	No	No	No	No
Scanlife	Sí	No	No	Sí	No	No

Tabla 3 Comparadores aplicaciones compra proximidad

Estas aplicaciones están pensadas para el uso particular, por lo que la dimensión social no se implementa en dichas soluciones. Así que nos fijamos en la posibilidad de acceder a catálogos de productos, localizar los puntos de venta por cercanía (tienda), cliente web o no (es decir, únicamente mediante aplicación nativa o "*app*"), posibilidad de comprar on-line (envío) y el uso de listas de compras previas. Queda remarcar el uso que hacen radarprice y scanlife de la cámara de fotos para localizar productos mediante códigos de barra.

2.1.3 Aplicaciones mixtas

Más relacionadas con nuestro objetivo final y parecidas al modo en el que pretendemos solucionar nuestro problema original, encontramos este tipo de aplicaciones en el que se combinan listas de compra con búsqueda de artículos, precios y lugares en los que encontrarlos.

Ejemplos de este tipo de aplicaciones serían SoySuper y MyShopi, que permiten crear una lista de la compra seleccionando los productos en función del precio y lugar de venta, así como añadir nuestros propios productos.

2.2 Conceptos teóricos

Es muy difícil conocer las interioridades de aplicaciones como las descritas hasta el momento, pero podemos intuir que no serán muy distintas, tecnológicamente hablando del resto de aplicaciones web existentes. Por lo tanto, tomaremos como referencia de tecnologías, técnicas y herramientas las que están en boga hoy día.

Dividiremos el conjunto de técnicas y herramientas necesarias según el entorno o plataforma sobre el que se ejecutan, y siendo nuestro desarrollo una aplicación cliente-servidor en Internet.

2.2.1 Tecnologías y herramientas del lado servidor

Aunque hoy día es asumible, al menos de manera inicial, disponer del **alojamiento** de un sitio o servicio web "*on premise*". No es también menos cierto que con el alojamiento en la nube que ofrecen proveedores [14] como Amazon con su EC3/S3 y Microsoft con su Azure (así como otros proveedores como NTT, HP, IBM, etc.) es más fácil desde un estado inicial ampliar el radio de acción y la disponibilidad de una aplicación o servicio web usando un despliegue del mismo en la nube, como ya hacen multitud de aplicaciones web, buscadores y servicios de almacenamiento. Proveedores tipo "*SaaS"* (*Software as a Service*), "*PaaS"* (*Platform as a Service*) o "IaaS" ("*Infrastructure as a Service"*) copan el mercado y hay que tener en cuenta las diferencias entre estos a la hora de escoger [15]. De todas formas, es un campo en continua expansión [13], siendo la tecnología más popular en este momento, pero sin perder de vista los posibles problemas que plantea, como pueden ser la elevada exposición en el ciberespacio, problemas de interferencias en el rendimiento / ruido derivados por la compartición de recursos con otros sitios, y otros éticos como pueden ser la dudosa denominación de tecnología verde, excesiva dependencia del proveedor (*vendor lock-in*) y la posible monopolización / privatización del ciberespacio [17].

En el caso de la nube habría dos factores a considerar, los problemas de disponibilidad que se hayan producido hasta el momento, es decir si cumplen con el objetivo del 7/24 y el precio. Este último muy importante, sobre todo en fases iniciales del servicio cuando una *start-up* o un grupo de desarrolladores noveles pueden sufrir más por la escasez de fondos. Esto no quita que en condiciones normales no hubieran más consideraciones a tener en cuenta, como pudiera ser la posibilidad de un entorno público o privado, cumplimiento de normas y estándares internacionales reconocidos, soporte y asesoramiento, localización de infraestructuras, etc [10]. Una vez en marcha y disponiendo de una visión más real del cumplimiento de los objetivos, del éxito, se podría tener en cuenta otros parámetros de calidad del hosting (aunque lo antes posible).

Sobre la infraestructura física (virtual o no) debe correr un **sistema operativo** adecuado a las tareas y funciones de un servidor. La pugna por ser el sistema operativo esencial e imprescindible para desplegar un servidor de aplicaciones o un servidor web parece, hasta el momento, cosa de dos. Por un lado Linux, con todas sus distribuciones de las que destacaríamos Redhat, Debian y CentOS [18]. Por otro lado, las versiones para servidores del sistema operativo de Microsoft derivado de la arquitectura NT; nos referimos a los Windows Server. No hay que olvidar un tercer participante en la disputa, el resto de la familia de sistemas operativos UNIX que, aunque mucho menor, tiene también una porción nada desdeñable del mercado [19] [18].

Muchos de los servidores de internet usan Linux (o alguna variante de Unix) [21] y otros Windows. Si nos centramos en esta disyuntiva, el problema para un equipo novel sería el de discernir cual usar en función del conocimiento disponible para su puesta en marcha y mantenimiento y el coste asociado con estas operaciones en el proyecto.

Una vez se dispone de una infraestructura completa (*hardware* + *O.S.*) queda determinar que *software* se encargará de gestionar y proporcionar el soporte necesario a las aplicaciones que ejecutarán los servicios que pondremos a disposición de los clientes. En el campo de los **servidores de aplicación** [13], y posiblemente como efecto colateral del anterior punto, también nos encontramos dividido el mercado en dos grupos principalmente [21]. En uno, aquellos que se basan en productos y soluciones de Microsoft (como puede ser IIS), en otro los que se basan en soluciones de código abierto con licencias libres como puede ser Apache (servidor web), Ngixn o Google, aparte de soluciones basadas en plataforma Java, como Apache Tomcat.

Como hemos comentado, hay soluciones que están íntimamente ligadas al sistema operativo sobre el que se ejecutan. En el caso actual, IIS de Microsoft sólo funciona sobre sistemas operativos Windows de Microsoft al ser, en realidad, un conjunto de servicios que

ya vienen de serie con el mismo y que únicamente hace falta activar para disponer de un completo servidor web o de aplicaciones. Al ser de esta manera, su licenciamiento y coste depende de la versión o gama del sistema operativo.

Por el contrario, otras soluciones como las de Apache, Nginx, JavaEE, etc. en su mayoría son independientes de la plataforma, hay que instalarlos de manera adicional y expresa y disponen de sus propias licencias, soporte externo y coste (en principio suelen ser gratuitas, pero eso no excluye que se produzcan costes por mantenimiento del mismo ya por el contrato de servicios a terceros para su soporte, disponibilidad de administradores u otros).

Finalmente, otra pieza importante es el S.G.B.D. elegido para dar soporte a la aplicación servidora. Hoy en día, sobre todo en lo referente a aplicaciones de internet, el top 3 de los motores de base de datos relacionales lo copan Oracle, MySQL (MariaDB) y MS SQL Server [23]. De entre estos tres, el primero y tercero están sometidos a restricciones de precio y licenciamiento que pueden escaparse de nuestras posibilidades, aunque debemos recordar que según la infraestructura escogida para la nube puede venir incluida (como pude ocurrir en la solución Azure de Microsoft). El segundo motor tuvo un gran auge gracias a ser de acceso libre (*open source*) y gratuito. Además fue adoptado al principio por muchos sitios web, incluyendo Facebook [23] y YouTube [25].

Pero no debemos perder de vista un tipo de almacenamiento alternativo denominado genéricamente noSQL. Este tipo de almacenamiento (generalmente basado en estructuras nombre, valor tipo hash) que suele denominar a cualquier motor que no sigue las pautas relacionales de los S.G.B.D más comunes [26], se ha popularizado enormemente en servidores que requieren una muy rápida respuesta de los servidores de almacenamiento de datos y la gran cantidad de datos [26]. De entre estos cabe destacar Cassandra y HBase [23] [26], adoptadas también por los grandes de internet [28].

2.2.2 Tecnologías y herramientas del lado cliente

En el mundo de las aplicaciones web la pieza principal es, sin lugar a duda, el **navegador.** Tendremos que observar con detenimiento las tecnologías que le rodean y las características que les son propias. Como ocurre con los servidores web y de aplicación, gran parte de los navegadores web se encuentra únicamente en uno o un subconjunto de los sistemas operativos más populares como son Windows, Linux, OSX en escritorio e IOS y Android en tabletas y móviles [19]. Pero al contrario que en aquellos, hay más opciones disponibles con sus propias características y fragmentación. Por ello, hoy en día, pocos desarrollos se abordan centrados en un único navegador o familia de navegadores y tratan

de incluir con el menor esfuerzo posible el mayor número de clientes posibles, ya sean de escritorio o móviles.

Para lograr este fin, está muy de moda [29] usar HTML5 [29], acompañado de *frameworks* de desarrollo (basados en Javascript o Typescript [30]) que proveen de herramientas para el diseño de aplicaciones, tanto desde el lado servidor como para el cliente web que usen patrones de diseño desacoplados del dispositivo de muestra (*render*), como son los populares (de más básico a avanzado) jQuery, Rails, AngularJS (1 y 2). *Frameworks* como estos tratan de conseguir esta abstracción junto con otras características que permiten que la aplicación desarrollada pueda expandirse al máximo número de dispositivos con la mayor reutilización posible.

Otra posibilidad en el campo de las aplicaciones ubicuas es el de crear un **cliente** para terminal **móvil**, en concreto para teléfono. La evolución del mercado de los móviles, especialmente referido a los teléfonos inteligentes que prácticamente copan el mercado, ha dictado la prevalencia de dos plataformas concretas. La primera Android, abierta y propuesta por la *Open Handset Alliance* [31]. La segunda, cerrada y propietaria, es IOS [33] el sistema operativo para plataformas móviles de Apple. Independientemente de la guerra por el mercado en la plataforma móvil, es la de mayor crecimiento en los últimos años dejando muy de lado al tradicional de escritorio, que hasta no hace mucho dominaba en el mercado de las soluciones informáticas.

De las estadísticas disponibles de distribución de mercado [34], podríamos deducir que cualquier desarrollo prácticamente debería tener en cuenta únicamente a Android, por ser la plataforma mayoritaria [34]. Pero como otros estudios de mercado demuestran, son los usuarios de IOS los más propensos a realizar gastos y compras por internet, siendo este con su tienda iTunes, uno de los más lucrativos hasta el momento [5,8] [36]. Así que desde un punto de vista también económico no hay que perderle la pista.

Ambas plataformas tienen en su contra una gran fragmentación en lo referente a las versiones y sus correspondientes características, aunque es más grave en el caso de Android [37] que en el de IOS [38]. Esto suele solventarse realizando el desarrollo para aquella versión inferior desde la que se consigue aglutinar al mayor número de dispositivos compatibles, pero que puede limitarnos a la hora de realizar el desarrollo.

Algo que al escoger la herramienta de desarrollo se debe tener en cuenta es el soporte a tecnologías usadas actualmente para realizar aplicaciones móviles y web. Independientemente del tipo de cliente (web o aplicación móvil), hay una serie de tecnologías que les son comunes:

Push [39], inicialmente desarrollada como forma de ahorrar batería evitando las consultas tipo *pull* contra un servidor, consiste en que es el servidor al que se ha conectado un dispositivo o aplicación que emite los mensajes hacia el cliente [40]. Explicado desde la óptima de un desarrollador, es lo más parecido a una estrategia de notificaciones por medio del patrón *observer* / eventos.

Geolocalización, esta tecnología [41] se ha puesto en manos del público en general gracias a la popularización de los dispositivos móviles dotados de receptores de señal *GPS*. Hasta ese momento, lo más parecido eran los dispositivos de asistencia a la conducción por *GPS* pero estos se trataban de cajas cerradas. En la actualidad, los dispositivos móviles dotados de receptores de señal *GPS* permiten a las aplicaciones que se ejecutan en ellos a acceder a los servicios de geolocalización sin mucho esfuerzo, generalmente a través de *APIs* [42] disponibles en el propio sistema operativo.

Imagen / fotografía integrada. Con la captura de imagen ha ocurrido algo similar. Previamente solo existían las cámaras digitales que como mucho permitían algún modo de conectividad wifi, aunque el auge de la telefonía móvil dotada de cámara la ha arrinconado a nichos especializados. Al igual que ocurre con los servicios de geolocalización, el sistema operativo subyacente ofrece *API* [43] que permiten usar sin mucho esfuerzo la cámara y almacenamiento necesarios.

Reconocimiento de voz. Al igual que ocurre con las capacidades para capturar imágenes, los sistemas operativos actuales para móvil incluyen API para realizar reconocimiento de voz e incluso realizar dictado de texto (*Text-To-Speech*) [44] [45].

Sistema de almacenamiento local. No hay que olvidar que, aunque los dispositivos estén permanentemente conectados no siempre es óptimo que trabajen de esta manera. Por el contrario, suele adoptarse una modalidad híbrida siguiendo el modelo "*briefcase*"; es decir se recuperan datos del servidor, se trabaja en local y se envía posteriormente la información resultante al servidor. Y aunque no se usara, es muy probable que localmente se deba almacenar información de uso habitual que permanece invariante en el tiempo (cache). Para esta información hay que disponer de un almacenamiento local. Para ello podemos fijarnos en las más populares como SQLite y BerkeleyDB, en el lado *open source* [46,23] o SQL Server Compact de Microsoft, Oracle Mobile Embebed o Interbase ToGo en el lado propietario (y menos económico).

2.2.3 Herramientas de desarrollo

Dependiendo de si la aplicación que se construye es nativa o no, es decir se genera exclusivamente para un tipo de dispositivo concreto usando el abanico de instrucciones que pone a la disposición de los desarrolladores o por el contrario se usan herramientas que abstraen completamente de la arquitectura real subyacente proveyendo de unos servicios comunes para una familia de dispositivos a priori incompatibles. A día de hoy los desarrolladores usan las siguientes herramientas de construcción de aplicaciones:

Aplicaciones nativas.

- IOS: XCODE
- ANDROID: Eclipse con ADT *Plugging*, Android Studio
- Multiplataforma: Visual Studio (con Xamarin / Mono), Delphi, HTML5 (http://123freeapps.com/, http://www.appcelerator.com/), herramientas de construcción visual como http://www.appinventor.org/. Solo indicamos estas pero podemos decir que hay casi tantas como lenguajes distintos de programación existen. Algunas de estas generan incluso código nativo para los distintos dispositivos con lo que podrían entrar en los dos grupos anteriores, al modo de "*write once and compile everywhere*" [47], que sería ligeramente distinto del famoso "*write once and execute everywhere*" popularizado por JAVA [48].

Aquí entra en juego los objetivos de quien desarrolla, si pretende una experiencia de usuario más uniforme con el dispositivo en el que se ejecuta, o si por el contrario pretende llegar rápidamente a un mercado más amplio y luego tratar o no de adaptarse o basarse en la experiencia que se tenga con el entorno. O si, finalmente, se pretende un modo u otro de distribución de la aplicación (como puede ser tener que pasar por una tienda autorizada o por el contrario acceder mediante una web).

De todas formas es muy habitual encontrar aplicaciones construidas en HTML5 (CSS + Javascript + HTML) de tal manera que, aunque se tengan que distribuir desarrollo de aplicaciones mixtas, en las que se encuentran tanto clientes nativos adaptados a las características físicas y posibilidades del dispositivo como aplicaciones [49].

Para el desarrollo de **cliente web**, el mercado es mucho más amplio. También es menos exigente, pudiendo realizar en la práctica cualquier desarrollo con un editor de textos sencillo (notepad++, bloq de notas). Además, los navegadores dan soporte adicional al

desarrollador mediante extensiones implementadas (extensiones para el desarrollador) o *plugins* que se pueden instalar adicionalmente (Firebug, por ejemplo).

Por supuesto hay entornos de desarrollos específicos para crear aplicaciones web completas (html, css, javascript) como Aptana Studio, Kompozer, BlueGriffon, Atom, Apache, Visual Studio, Netbeans, Brackets y un largo etcétera.

2.2.4 Riesgos y vulnerabilidades

Una vez establecidas las tecnologías y herramientas más usadas en el desarrollo de este tipo de aplicaciones, no debemos olvidar las vulnerabilidades más comunes que suelen ser usadas por atacantes sin escrúpulos para comprometer los sistemas construidos para su propio beneficio. Teniendo en cuenta que la arquitectura utilizada es la n-capas en internet, aplicable bien para cliente web habitual como cliente móvil, debemos echar una mirada al conocimiento compartido de los expertos en seguridad en este tipo de aplicaciones que ponen a disposición del bien común a través de internet.

Uno de los primeros sitios a consultar es el proyecto OWASP, fundación sin ánimo de lucro que difunde información sobre seguridad de forma colaborativa y de una manera abierta [50]. De entre la multitud de recursos que pone a nuestra disposición de manera libre, podemos encontrar una guía con las 10 vulnerabilidades más comunes [51] [52] con guías, ejemplos y consejos para tratar de evitar su aparición en nuestras aplicaciones.

Aunque a fecha de hoy se encuentran construyendo la que será la lista de las diez vulnerabilidades más importantes de 2016, podemos contar con las más importantes en 2013, que fueron:

A1 – Inyección

> "…ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta." [53]

A2 – Pérdida de autenticación y gestión de sesiones

> "Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente…" [53]

A3 – Secuencia de comandos en sitios cruzados (XSS)

> "…ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada" [53]

A4 – Referencia directa insegura a objetos

> "…un desarrollador expone una referencia a un objeto de implementación interno", "Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados" [53]

A5 – Configuración de seguridad incorrecta

> "…configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el *software* actualizado, incluidas las librerías de código utilizadas por la aplicación." [53]

A6 – Exposición de datos sensibles

> "Muchas aplicaciones web no protegen adecuadamente datos sensibles…" (CITA) "…requieren de métodos de protección adicionales tales como el cifrado de datos…" [53]

A7 – Ausencia de control de acceso a las funciones

> "…las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función…" [53]

A8 – Falsificación de peticiones en sitios cruzados (CSRF)

> "…obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable" [53]

A9 – Uso de componentes con vulnerabilidades conocidas

> "Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos." [53]

A10 – Redirecciones y reenvíos no validados

> "…frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino…" [53]

Otras organizaciones, como MITRE (mitre.org) [54] mantienen extensos catálogos de vulnerabilidades conocidos (*Common Vulnerabilities and Exposures*) [55] en aplicaciones de

todo tipo que pueden ser útiles para estar alerta sobre vulnerabilidades conocidas de herramientas que pretendamos o estemos usando y poner remedio a tiempo.

Por supuesto, hay más guías, como las que pone a nuestra disposición el Instituto Nacional de Ciberseguridad de España [56], donde podemos encontrar guías y herramientas [57,58] que ayudan a la hora de concienciarse, diseñar e implantar sin descuidar la seguridad.

Siguiendo con herramientas gubernamentales, disponemos también del Observatorio Tecnológico del Ministerio de Educación, Cultura y Deporte, en el cual podemos encontrar recursos y formación sobre seguridad [58].

Debemos, por lo tanto, tener presente el uso de herramientas que puedan ayudarnos a detectar vulnerabilidades y a poner remedio antes de que sea demasiado tarde. Estas [59] pueden ser NMAP (rastreo de puertos), Nessus / OpenVAS (evaluación de seguridad), Snort (Sistema de detección de intrusiones), Brutus (autenticador por fuerza bruta), Microsoft Baseline Security Analyzer (MBSA), Nikto (Escáner de vulnerabilidades de servidores web), herramientas de análisis estático de código (https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis), etc.. Muchas son vistas desde un punto de vista más económico (herramientas gratuitas de código abierto). Por supuesto existen muchas más de pago, pero alejadas de nuestro alcance, como pueden ser soluciones de HP como WebInspect o HP Fortify.

No olvidemos que estas herramientas no deben ser motivo para descuidar un desarrollo seguro usando técnicas y buenas prácticas reconocidas durante el desarrollo, como pueden ser apoyarse en los tres pilares fundamentales del desarrollo seguro [60] de Gari McGraw y sus siete "*touchpoints*" [61].

Por lo tanto, y para finalizar, debemos tener en cuenta que soluciones existen a día de hoy para realizar pruebas sobre nuestro desarrollo, antes, durante y tras su finalización. Ya hemos adelantado algunas de ellas, como son el análisis estático de código, pero hay otras técnicas complementarias como son los test unitarios, en un entorno *TDD* (*Test Driven Development*) o no, pruebas de regresión, pruebas de integración en lo referente al código construido, y pruebas de carga para evaluar el rendimiento del producto resultante. Fuera de las pruebas de caja blanca, también hay multitud de herramientas y técnicas de prueba de caja negra como Microsoft Coded UI, Appium (más enfocado en el desarrollo móvil) o el ya mencionado Nikto. En el campo de las pruebas funcionales, podemos incluir a Selenium, SoapUI, Microsoft Test Manager. Para pruebas de carga, FunkLoad y jMeter de la fundación Apache, entre otras muchas.

2.3 Conclusiones

¿Dónde entra nuestro desarrollo en el contexto actual de aplicaciones para listas de compra?

Está claro que debemos tomar como inspiración los usos actuales en el campo del desarrollo de aplicaciones móviles para tratar de minimizar los riesgos y conseguir el éxito de nuestra empresa. Para ello deberemos de abrazar las herramientas y tecnologías más populares que estén a nuestro alcance (medios y conocimientos). Además, no debemos perder de vista uno de los objetivos de este proyecto que es el de demostrar y poner en práctica los conocimientos adquiridos en el presente máster.

Por ello, deberemos seleccionar las siguientes tecnologías y herramientas:

Tecnología nube (*cloud*): Posibilidad de usar de manera gratuita la nube de Amazon o Azure para realizar pruebas (como sería la prueba de concepto de la aplicación que desarrollamos).

Sistema operativo Windows Server o Linux. A corto plazo no sería relevante, al trabajar sobre plataformas a priori gratuitas para desarrollos cortos o pruebas de concepto, pero a la larga sí sería un factor a tener muy en cuenta.

Servidor web/aplicaciones: En el caso de usar un servidor web o servidor de aplicaciones, nos decantaríamos por IIS (a partir de la versión 7) o Apache. Como servidor de aplicaciones Java, Apache Tomcat. Estos servidores han sido los tratados en la asignatura correspondiente del presente Máster y suelen estar disponibles con las versiones de prueba de la infraestructura tipo nube comentada anteriormente.

S.G.B.D: Inicialmente nos decantamos por mySQL, por su popularidad, soporte por parte de las herramientas de programación y gran variedad de utilidades disponibles. De todas formas, durante el desarrollo o posteriormente tras la finalización del proyecto, podemos ir comparando con bases de datos no relacionales como Cassandra.

Navegador: Nos es indistinto. Trataremos de utilizar herramientas lo más genéricas posible (es posible que haya que realizar adaptaciones) que lidien con los problemas que puedan presentar las distintas interpretaciones de los estándares subyacentes a HTML5. En todo caso, se obviarían navegadores antiguos que no cumplen con los estándares propugnados desde el consorcio W3, como pueden ser versiones anteriores de Internet Explorer. Por el contrario, navegadores como Mozilla Firefox, Chrome y Safari estarían entre los candidatos a recibir soporte.

Para lograr este fin, deberemos acompañar el desarrollo del HTML de frameworks de trabajo web y herramientas como jQuery, AngularJs u otras, siendo aquellas dos primeras quizás las más populares [62].

Herramienta de desarrollo: Trataremos de seleccionar una herramienta que nos permita desarrollar para el mayor número de móviles posible. Como en principio no tratamos de monetizar o generar un retorno de inversión, podríamos pasar por alto la plataforma IOS y centrarnos en el desarrollo con JAVA en la plataforma Android. En este caso, muy posiblemente, deberíamos optar por Android Studio, al tratarse de la herramienta de referencia del fabricante.

Otra opción podría ser, ya que se programaría en .NET, usar Visual Studio con Xamarin. Con esto seguiríamos dentro del objetivo de aplicar los conocimientos adquiridos y además podríamos ampliar el número de dispositivos sobre los que la aplicación se ejecutaría ya que esta herramienta permite generar binarios para múltiples dispositivos como Android, IOS o Windows Phone.

Y una última opción podría ser el uso de un entorno de desarrollo basado en html5 que nos permitiera crear tanto la aplicación móvil como el *backend* / aplicación de navegador que tenemos en mente crear.

Independientemente de las consideraciones que hemos indicado para cada herramienta de desarrollo, deberemos tener en cuenta si ponen a disposición del desarrollador las *API* de acceso a las distintas tecnologías que hemos mencionado en el punto anterior y que serían vitales para lograr el objetivo del proyecto. La primera, el acceso a los servicios de geolocalización. El segundo, el acceso a los dispositivos de adquisición de imágenes (cámara) y de voz (si da tiempo). Y tercero y no menos importante (aunque podría ser suplido por otras tecnologías) el acceso a un sistema de notificación *push*.

3. Objetivos concretos y metodología de trabajo

El desarrollo que se describirá en este trabajo tendrá como objetivo principal demostrar la aplicabilidad de los conocimientos adquiridos en master en el que se inscribe, haciendo especial hincapié en los distintos dominios de conocimiento impartidos durante el curso del mismo y del que el presente tratará de ser fiel reflejo en la medida de lo posible al circunscribirse a unos restricciones muy particulares de medios y tiempo.

Para ello, se realizó la implementación de una solución cliente-servidor en Internet de manera que se aglutine tecnología móvil (Android en principio) con web 2.0 (es decir, uso de html5, JavaScript, Ajax) con buenas prácticas de desarrollo seguro auditables, de accesibilidad y metodológicos. Todo esto, aunque siendo a priori algo ambicioso para una implementación a realizar en el tiempo disponible, se pretendió realizar hasta lograr un primer prototipo funcionalmente completo que sirva para posteriores ampliaciones e incluso otros nuevos desarrollos con las ideas que más adelante se esbozarán en el capítulo correspondiente al trabajo futuro. De aquí que aplicar, de los conocimientos adquiridos, un buen enfoque metodológico fuera importante para conseguir, de manera iterativa e incremental, realizar una construcción en la que no se perdiera tiempo (desperdicio, *Lean*).

3.1. Objetivo general

Nuestro objetivo es desarrollar una aplicación móvil que permita recoger información sobre productos no catalogados previamente, disponibles en múltiples localizaciones y precios, para posteriormente obtener los lugares en los que se encuentran a mejor precio, de manera colaborativa.

Como ya hemos introducido brevemente y aplicado a este proyecto concreto, el objetivo general buscaba la creación de una aplicación que nos sirva para comparar precios de elementos no catalogados ni inventariados previamente de una manera sencilla y rápida, con la menor formalización posible. Es decir, un sistema que permita dar de alta elementos, registrar datos de los mismos y explotarlos con la menor interacción posible.

Para conseguirlo, y centrándonos en el problema de recoger los precios de una serie de productos, realizamos el modelado e implementación de un sistema de información que permita adquirir la información necesaria para nuestro propósito y aprovecharla adecuadamente de una manera automática y cómoda.

Esencialmente la información recogida es una fotografía, una localización y un precio. Esto, n-plicado por cada ocurrencia de una serie de productos, categorizados al momento o posteriormente, permite explotar los datos. Dicho uso es, inicialmente, la localización y establecimiento de rutas para permitir adquirir al usuario o usuarios, desde la posición en la que se encuentra, recorrer y adquirir los productos siguiendo una ruta que le trace el sistema (la más corta inicialmente).

Para conseguir este propósito, se requirió crear una aplicación constituida a partir de servicios en un servidor alojado en la nube, posiblemente Azure de Microsoft, lo que nos brindó una oportunidad muy interesante para estudiar y descubrir esta tecnología.

Para construir una solución lo más segura posible, y siguiendo lo aprendido en el segundo cuatrimestre del Máster, se escogió una implementación en Java (lenguaje fuertemente tipado, orientado a objetos) para construir servlets (servicios) que se alojan en un servidor Apache Tomcat. En la parte cliente, se construyeron dos soluciones, una de ellas enfocada a la recogida de datos y consulta/explotación de los mismos mediante cliente móvil (inicialmente Android/Java) y otra web a modo de *backend* que permite realizar operaciones de mantenimiento y explotación no disponibles en la aplicación cliente móvil.

La información del sistema, en el lado servidor, se recoge en una base de datos escalable de código abierto de la familia mySQL. Aunque la información que a priori se recibe no será sensible, tratamos de seguir las mejores prácticas aprendidas, así como posteriormente auditamos, es decir estudiar, evaluar y analizar, la implementación para comprobar la seguridad del sistema.

Por supuesto, tratamos de que el sistema sea usable y lo más accesible que nos sea posible.

En definitiva, quisimos crear una solución segura, usable que nos permita acercarnos al concepto de internet móvil (ubicuidad en la red), que nos acerca al concepto de internet de las cosas.

3.2. Objetivos específicos

Para alcanzar nuestro objetivo principal, debimos trazar unos pasos que nos permitió descomponer el problema principal (dividir y vencer [63]) esbozado por nuestro objetivo principal para, mediante el paulatino y constante avance, alcanzarlo.

Para ello no debemos olvidar, como ya hemos introducido en puntos anteriores, el objetivo general del trabajo que se trata de realizar con el presente proyecto, aparte de la consecución de la funcionalidad propiamente dicha, era poner en práctica los conocimientos adquiridos en el Máster de ingeniería del software y sistemas informáticos.

Así pues, si básicamente lo que se pretendimos es aplicar en un desarrollo real y aprender del mismo las ventajas y problemas de las soluciones puestas en marcha para la realización del desarrollo usando tecnología móvil y web en un entorno cliente-servidor expuesto a Internet en el que primen la seguridad, usabilidad y accesibilidad, estableceremos los siguientes objetivos secundarios, que nos permitieron alcanzar la meta dictada por el principal:

- Implementar servicios en la nube.
- Desarrollar un cliente móvil (Smartphone).
- Desarrollar un cliente web para las operaciones de mantenimiento.
- Realizar un desarrollo contemplando la seguridad en todos sus procesos (S-SDLC)
- Realizar evaluación de seguridad.
- Implementar una solución de geo-posicionamiento.
- Estudiar nuevas herramientas y metodologías de desarrollo y su aplicabilidad al mundo de aplicaciones Internet (métrica v3).

3.2. Metodología del trabajo

Para realizar el trabajo seguimos los siguientes pasos, siguiendo de manera ligera o resumida, los propuestos por métrica v3 para el desarrollo orientado a procesos.

Para ello dividimos el trabajo en tres procesos, de los que dimos mayor importancia (por ser donde se realiza la mayor parte del trabajo práctico) a los dos primeros. Los procesos fueron:

- Planificación
- Desarrollo
- Construcción
- Mantenimiento

Cada uno de estos procesos los dividimos, a su vez, en las siguientes tareas y actividades que debimos llevar a cabo para su realización:

Planificación

En este primer paso se describió el sistema que pretendíamos desarrollar. Para realizar esta actividad requerimos, como mínimo, realizar las siguientes actividades en las que poder apoyarnos posteriormente en el paso de desarrollo:

1. Definición del catálogo de requisitos De manera resumida registraremos los requisitos necesarios para nuestro sistema.
2. Definición de la arquitectura
    a. Modelo de información.
       Recogeremos el modelo de información en base a los conceptos que se derivan de los requisitos, las reglas que les son impuestas, las relaciones entre ellos y sus operaciones.
    b. Modelo de sistema de información.
       Seleccionaremos el orientado a objetos como modelo del sistema de información para desarrollar los objetivos del proyecto actual.
    c. Arquitectura tecnológica
       Plantearemos la arquitectura escogida para implementar la solución que estamos realizando.

Desarrollo

1. Estudio de viabilidad del sistema.

En nuestro caso, no necesitábamos realizar un estudio de viabilidad del sistema. La solución ya viene determinada desde un principio por lo que no entraremos a valorar otras opciones, que aunque fuesen más económicas (tiempo, recursos, etc.) no cumplirían con el objetivo principal del proyecto: Aplicar el máximo de conocimientos posibles.

a. Valoración de riesgos.

   Una parte importante del proyecto fue determinar, y así poder prevenir, los problemas que puedan surgir durante el proyecto. Para ello necesitábamos realizar un catálogo de riesgos, valorarlos y, en cada caso, determinar las medidas para asumirlos, minimizarlos o anularlos. Así maximizábamos las posibilidades de llevarlo a buen puerto.

   i. Riesgos de seguridad.

      Dentro de los riesgos, y siendo uno de los conocimientos que pretendíamos poner en práctica, los de seguridad en un entorno cliente-servidor en internet eran críticos. Estos riesgos pueden no afectar directamente al desarrollo (aunque al tenerlos en cuenta nos obligará a implantar un ciclo de desarrollo seguro), pero sí a la supervivencia futura de la aplicación una vez puesta en producción.

b. Descripción de la solución propuesta.

   Describe la solución, de manera general, para que los siguientes pasos tengan una referencia que les permita seguir enfocados en el resultado que pretendemos obtener.

2. Análisis del sistema de información.

   Una vez que tenemos una descripción del sistema que pretendemos, entramos en más detalle realizando las siguientes tareas:

   a. Descripción inicial del sistema de información.

      Realizamos una descripción del sistema de información, también de manera general, en base a la descripción de la solución propuesta.

   b. Delimitación del alcance.

      Con las descripciones del sistema y la del sistema de información, pudimos delimitar el alcance de nuestro proyecto, indicando qué se incluye y qué se excluye. De esto último, opcionalmente, pudimos recoger y catalogar aquello que pudiera realizarse en fases posteriores de mantenimiento.

   c. Catálogo de requisitos generales.

      Una vez delimitado el alcance, conocimos a “grosso-modo” que requisitos quedan incluidos en él, por lo que procedimos a catalogarlos, identificándolos unívocamente y realizando una breve descripción general de los mismos.

d. Modelado de alto nivel.

   En base a los resultados de las tareas anteriores, apoyándonos principalmente en los requisitos que hemos determinado en el paso previo, realizamos un modelado estructural de alto nivel: de las clases de análisis (usando UML 1.2 como mínimo), modelado de subsistemas y casos de uso. Esto nos permitió tener una visión general del sistema que pretendemos construir.

3. Diseño del sistema de información. Una vez obtuvimos una visión general y completa del sistema que pretendíamos construir, detallamos el sistema en mayor medida.
   a. Catálogo de requisitos completo.

      Profundizamos en el listado de requisitos para completarlo con la información que hemos podido recopilar hasta el momento, aprovechando para detallarlos lo suficiente para que puedan implementarse sin ambigüedades.

   b. Casos de uso.

      Detallamos los casos de uso completos del sistema que estamos realizando y así poder determinar completamente las funcionalidades que el sistema debía implementar.

   c. Modelo de clases de diseño.

      Recogimos las clases de diseño, partiendo de las clases de análisis que obtuvimos anteriormente, detallándolas desde el sistema y herramientas escogidas para la implementación y así su traslación de modelo a código fuera fluida.

   d. Comportamiento de clases de diseño.

      Otra parte importante de las clases es recoger el comportamiento y servicios que ofrecen al sistema o resto de clases. Este comportamiento requirió nuestro trabajo y aplicación de los conocimientos aprendidos en el presente master al aplicar las técnicas y leguajes impartidos. Además, la realización de estos comportamientos dio como resultado la funcionalidad final.

   e. Diseño de interfaz de usuario.

      Otra parte importante de cualquier sistema orientado a usuarios, más en el mundo móvil y web (sea de escritorio o no), es la interfaz de usuario. En este punto también tratamos de aplicar los conocimientos adquiridos sobre UI y seguridad. No deja de ser lo que un usuario percibe como “el sistema” y es la base para el comportamiento al ser entrada y salida del mismo. La interfaz gráfica, adicionalmente, da una pista importante de si el sistema cumple con los requisitos y objetivos del proyecto.

Construcción

- Implementación (Pruebas unitarias, integración y de sistema).
  Durante la construcción del sistema, usando como "planos" el resultado del punto anterior en sus distintos aspectos (planos parciales o visiones del sistema completo como son los requisitos completos, modelos e UI) implementamos un desarrollo de software seguro, tomando como base el análisis de riesgos, realizando los test que nos permita crear el software y sistema final de manera iterativa e incremental. Teniendo en cuenta las restricciones que se nos aplican (tiempo, recursos), se hizo especial énfasis en seguir unas pautas "lean".

Implantación y aceptación.

- Plan de implantación.
  Conforme a lo que se determine en las fases anteriores, realizamos el plan para llevó a cabo la implantación. Al tratarse de una solución cliente-servidor en el que no se requiere gran despliegue en máquinas cliente, el plan será breve, aunque queda abierto a las modificaciones que puedan producirse debido al resultado y conocimiento adquirido del sistema final en los pasos anteriores.
- Se detallan las pruebas a realizar para formalizar la aceptación del producto resultante en función de los requisitos recogidos en las fases iniciales.

Mantenimiento

De manera breve tratamos de especificar las tareas de mantenimiento que hacen falta para el día a día del producto y que permitan, de continuarse, realizar las mejoras y correcciones futuras.

4. Desarrollo específico de la contribución

En el presente capítulo, a partir de lo aprendido del estado del arte (herramientas y tecnologías así como experiencias de proyectos similares) y con el objetivo en mente a alcanzar (sin olvidar los objetivos secundarios), se detallarán los pasos dados para la realización concreta del proyecto siguiendo nuestra versión libre de Métrica v3.

4.1. Descripción de la planificación del proyecto de desarrollo de software

La planificación del proyecto se llevó a cabo teniendo en cuenta los pasos a seguir dictados por la metodología Métrica v3, intentando tener en cuenta los *touchpoints* de Gary McGraw [61] para llevar a un ciclo de vida de desarrollo seguro.

Recordemos los cinco procesos básicos a seguir en Métrica v3 [64]

1. Estudio de viabilidad del sistema (EVS).
2. Análisis del sistema de información (ASI).
3. Diseño del sistema de información (DSI).
4. Construcción del sistema de información (CSI).
5. Implantación y aceptación del sistema (IAS).

Como comentamos hemos tratado de incorporar las siete buenas prácticas (*touchpoints*) de McGraw para ciclo de desarrollo del software que son:

- Casos de Abuso
- Requerimientos de seguridad
- Análisis de riesgos
- Pruebas de seguridad pasadas en riesgos
- Revisión de código
- Pruebas de penetración
- Seguridad en operación.

Con lo que los pasos a seguir, y procesos de seguridad a incluir quedarían recogidos en la siguiente figura.

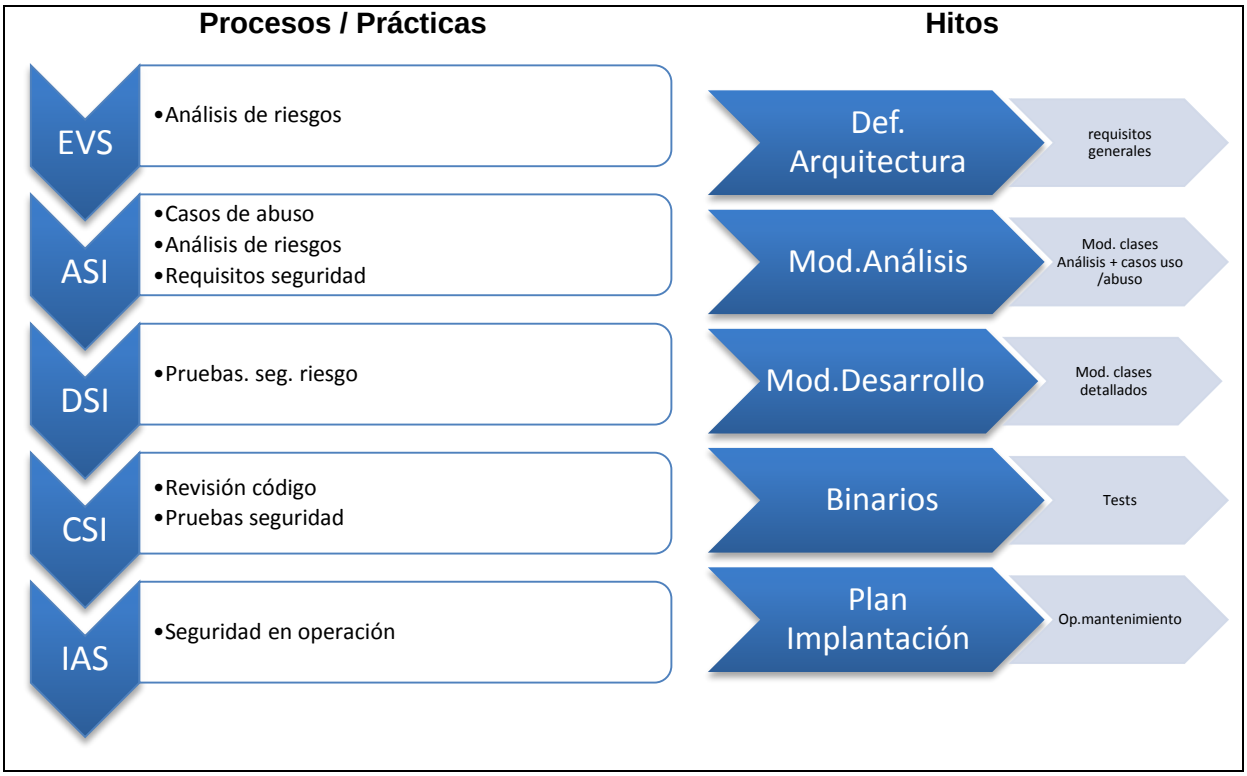


Tabla 4 Procesos desarrollo

4.1.1. Catálogo de requisitos

A partir de los objetivos a alcanzar en la realización del presente proyecto, los requisitos que se derivan fueron los siguientes:

Requisitos funcionales

ID	DESCRIPCIÓN	PRIORIDAD
RF-1	Gestión de usuarios sistema Debe poderse dar de alta/baja usuarios (mail) en el sistema	Media
RF-2	Gestión de usuarios conocidos Un usuario debe poder mantener una lista de usuarios existentes en el sistema que le hayan aceptado.	Baja
RF-3	Gestión de productos El sistema debe poder mantener información de productos	Alta
RF-4	Sincronización datos usuarios conocidos El sistema debe permitir compartir información de productos entre usuarios conocidos	Baja
RF-5	Informes productos	Alta
RF-6	Determinar ruta desde posición actual a productos El sistema debe proporcionar listados por posición de unos productos dados (posición y ruta)	Alta

Tabla 5 Catálogo de requisitos funcionales

Requisitos de diseño

ID	DESCRIPCIÓN	PRIORIDAD
RD-1	Cliente móvil La aplicación de captura de datos se implementará mediante aplicación móvil.	Alta
RF-2	Cliente *backend* web Se desarrollará un cliente web de *backend* de soporte a la aplicación.	Media
RF-3	Nube El sistema implementará el subsistema servidor en infraestructura en la nube.	Alta
RF-4	S-SDLC El desarrollo debe seguir un ciclo de vida seguro.	Alta
RF-5	Evaluación de seguridad Debe realizarse una evaluación de seguridad del sistema implementado.	Media

Tabla 6 Catálogo de requisitos de diseño

4.1.2. Definición de arquitectura

En el presente apartado procederemos a describir los primeros pasos que dimos para establecer una visión global de lo que terminó siendo el sistema desarrollado. Esta visión general nos permitió mantener los objetivos del proyecto a la vez que ir manejando la complejidad de las distintas partes mediante una profundización paulatina en cada uno de los subsistemas a desarrollar.

4.1.2.1 Modelo del sistema de información

A partir de los objetivos y el catálogo de requerimientos que debe satisfacer la solución, se llegó a la conclusión de que necesitaríamos modelar las piezas que compondrían el sistema de información necesario para alcanzar nuestra meta.

Visto desde este punto, a muy alto nivel, el sistema de información quedaría modelado como unos procesos de la siguiente manera:

Ilustración 1 Sistema de información

En la imagen podemos apreciar lo aparentemente sencillo que puede ser un sistema de adquisición de datos y explotación como es, en realidad, nuestra aplicación. Resultan un número relativamente bajo de procesos en complejidad, pero para obtener finalmente una idea más completa del esfuerzo y recursos necesarios necesitamos una descripción de arquitectura tecnológica, como la que presentamos en el siguiente apartado.

4.1.2.2 Arquitectura tecnológica

Con lo aprendido en capítulos anteriores por el estudio del estado del arte, soluciones parecidas, tecnologías y herramientas utilizadas comúnmente en soluciones parecidas, decidimos orientar la decisión arquitectónica en la que basar el desarrollo hacia la más usada en internet, cliente-servidor, de n-capas.

Recordemos que, básicamente, una arquitectura cliente servidor consta de dos piezas fundamentales: el cliente, que se encarga de iniciar el proceso/comunicación, y el servidor, que es la pieza que se encarga de procesar las peticiones de proceso o comunicaciones del cliente. Esto, en internet, no deja de ser el patrón básico de comportamiento de las aplicaciones, en las que un usuario, ya sea con una aplicación partículas o un navegador realiza una solicitud a un servicio alojado en un servidor de cualquier tipo en cualquier lugar de la red.

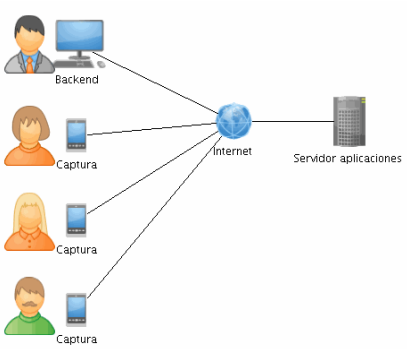


Ilustración 2 Aplicaciones en internet (cliente-servidor)

Indicar n-capas en una solución cliente-servidor desgrana la complejidad subyacente a la arquitectura. Descomponemos la aplicación en las capas que se encargan de tareas específicas lo que nos permite aplicar una solución por descomposición [63] y limitar en la medida de lo posible la complejidad.

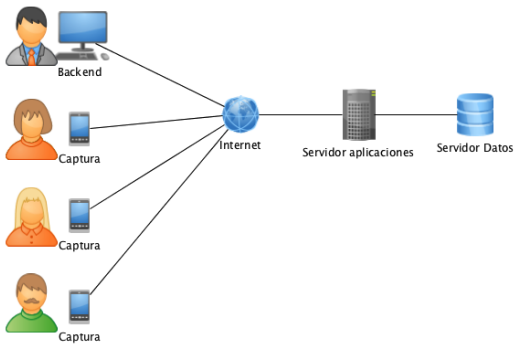


Ilustración 3 Aplicaciones en internet (capas)

Pero la simplificación de las n-capas mostradas en la imagen anterior no deja de ocultar, realmente, la descomposición completa de subsistemas que hacen falta para controlar la complejidad de una aplicación de este tipo y permitir que cada una se encargue de una parte de la tarea a realizar.

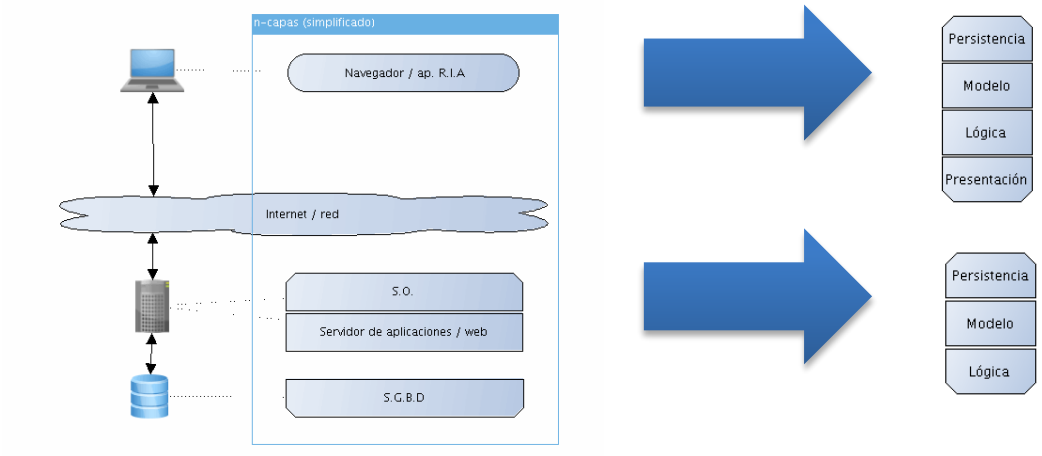


Ilustración 4 Aplicación n-capas

4.2 Desarrollo

4.2.1 Estudio de viabilidad

Como ya se ha establecido en pasados apartados, no hizo falta realizar un estudio de viabilidad en profundidad. Quedó a la consideración del equipo la valoración de medios, esfuerzo y oportunidad que diera partida a la realización del presente proyecto.

Aun así, no hay que dejar de controlar los riesgos con los que nos encontramos, por lo que se tuvo a bien realizar las siguientes valoraciones que tuvimos presente posteriormente en los siguientes pasos del ciclo de vida del desarrollo llevado a cabo.

4.2.1.1 Delimitación de alcance

Por las razones ya esgrimidas anteriormente, el alcance del estudio de viabilidad se centró en la valoración de riesgos, con especial énfasis en los de seguridad.

4.2.1.2 Valoración de riesgos

Principalmente los riesgos a los que nos enfrentamos son los siguientes, cuantificados como Leve, Medio o Crítico:

- Conocimiento / Desconocimiento de las herramientas: Leve
- Falta de tiempo: Crítico
- Falta de recursos: Medio

Para afrontar estos tres riesgos principales, disponemos de las siguientes estrategias.

Conocimiento: Uso de herramientas impartidas en el master:

Servidor Web/Aplicaciones: IIS | Tomcat | Apache

Codificación: Eclipse | Visual Studio

Falta de tiempo: en este caso solo podemos asumir este riesgo, teniendo en cuenta que se dispone de una segunda oportunidad de entrega en septiembre. Aun así, el módulo web sería prescindible en caso de necesidad.

Falta de recursos: Los recursos necesarios para el desarrollo no son exigentes. Con los ordenadores personales bastarían. Desgraciadamente disponer de un servidor en internet es problemático, por lo que delegaremos este problema en un servicio gratuito de Microsoft (Azure) o Amazon.

4.2.1.2.1 Riesgos de seguridad

Disponer de un servidor abierto a Internet conlleva una serie de riesgos que debemos tener muy en cuenta a la hora de realizar el desarrollo. La posibilidad de intrusiones externas en el sistema es algo que puede poner en peligro el proyecto y su viabilidad futura.

En nuestro caso, al tratarse de una arquitectura cliente-servidor en la que la lógica de negocio se concentrará en el servidor este será el elemento más crítico y más atractivo a ojos de los posibles atacantes.

El alcance del análisis de riesgos, por lo tanto, se centrará en el servidor (Servidor de aplicaciones, Sistema operativo y SGBD) así como en los distintos clientes a implementar.

Los activos son muy limitados, debido a la inexistencia de una organización física ni de servidores propios.

ID	Nombre	Descripción	Tipo	Ubicación	Crítico
1	Servidor	Servidor del sistema	Nube	Externo	Sí

Las amenazas a las que nos encontraremos serán las habituales en el tipo de aplicaciones on-line de hoy día. Como hemos comentado anteriormente, la inexistencia de una organización física limita la tipología de riesgos a estos que acabamos de comentar. No contar con instalaciones propias hace que podamos descartar el tipo de riesgos asociados a las mismas (desastres naturales, intrusión física, etc.).

Id	Descripción	Control	Nivel amenaza
Fallo en servidor	Caída del servidor debido a incidencias en proveedor	no corresponde	Medio
Intrusión	Acceso no autorizado por un tercero	Implantación de medidas de autentificación y autorización	Alto
Suplantación	Acceso de un tercero en lugar de un usuario registrado	Implantación de medidas de autentificación y autorización	Medio
Robo de información	Extracción no autorizada de información del sistema	Implantación de política de autorización basada en roles	Bajo
Intercepción	Escucha y/o modificación de las comunicaciones de la aplicación	Implantación comunicaciones seguras punto a punto (SSL/TSL)	Alto
Errores	Problemas en el código que puedan alterar o impedir el correcto funcionamiento de la aplicación	Implantación de pruebas unitarias, integración y sistema	Medio

Tabla 7 Amenazas

4.2.2. Análisis del sistema de información

4.2.2.1 Descripción inicial

Para iniciar la descripción del sistema de información, comenzaremos proporcionando unos DFD que describan los flujos de información empezando por describir el contexto y llegando hasta un mayor nivel.

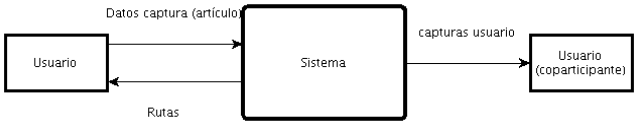


Ilustración 5 DFD contexto

Con un poco más de detalle:

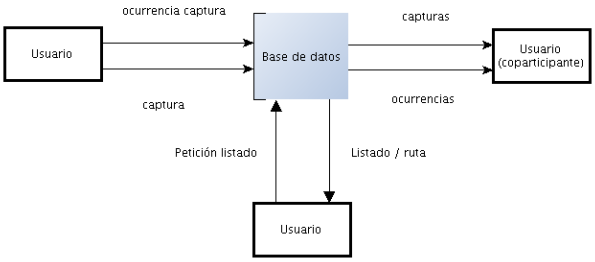


Ilustración 6 DFD nivel 1

Ya podemos profundizar en los flujos de datos para cada uno de los distintos procesos esbozados, tanto de entrada como salida del sistema:

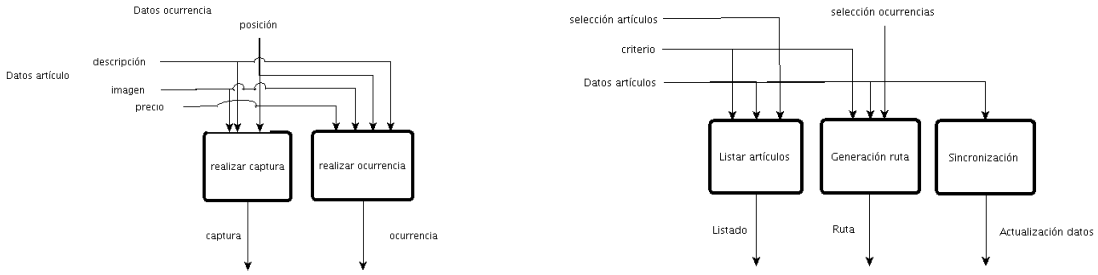


Ilustración 7 DFD Nivel 2

4.2.2.2 Catálogo de requisitos generales

Hemos categorizado los casos de uso en cuatro grupos principales, los de captura de datos, los de explotación de dichos datos, operaciones adicionales de mantenimiento y los de comunicación entre participantes de las búsquedas.

Sesión

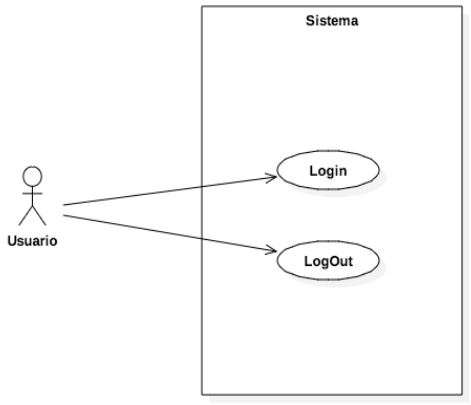


Ilustración 8 Casos de uso: Sesión

CU-1	**Conexión**
Descripción	El usuario se autentifica
Dependencias	El usuario no debe tener una sesión iniciada. En caso contrario se ignorará
CU-2	**Desconexión**
Descripción	El usuario cierra la sesión
Dependencias	El usuario debe tener sesión iniciada. En caso contrario se ignorará.

Tabla 8 Casos de uso: Sesión

Captura

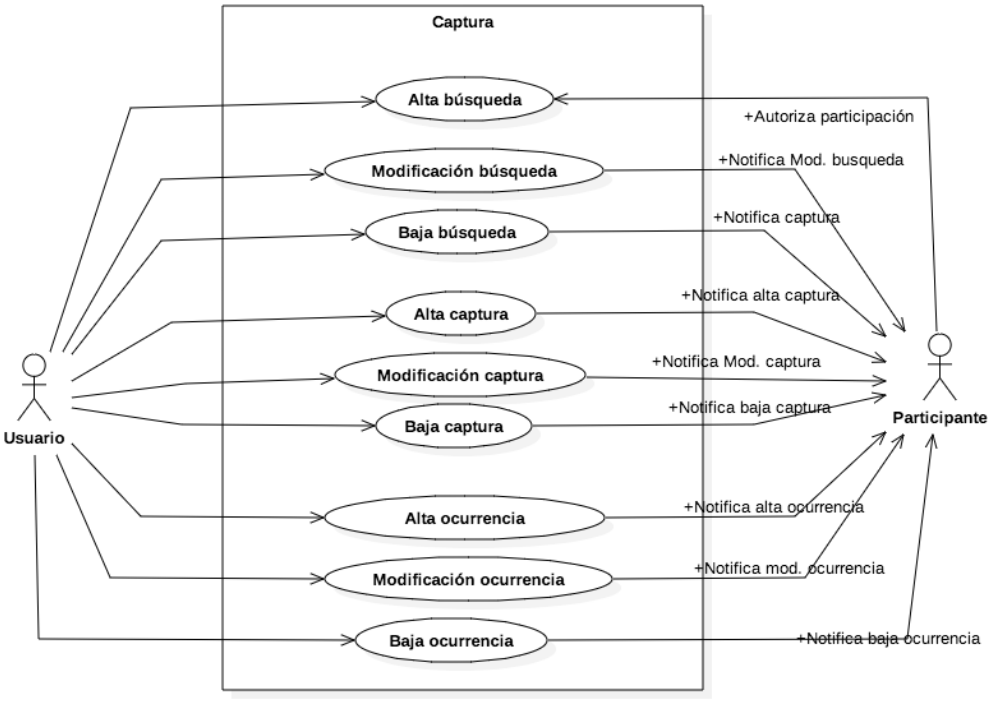


Ilustración 9 Casos de uso: Captura

CU-3	**Alta búsqueda**
Descripción	El usuario da de alta una nueva búsqueda en el sistema
Dependencias	-
CU-4	**Modificación búsqueda**
Descripción	El usuario cambia las propiedades de una búsqueda
Dependencias	Se requiere una búsqueda existente
CU-5	**Baja búsqueda**
Descripción	El usuario da de baja una búsqueda existente
Dependencias	Baja existente
CU-6	**Activación búsqueda**
Descripción	El usuario marca una búsqueda como la activa (en curso)
Dependencias	Búsqueda existente
CU-7	**Alta captura**
Descripción	El usuario captura un elemento nuevo (captura) y una ocurrencia del mismo
Dependencias	Búsqueda existente
CU-8	**Modificación captura**
Descripción	El usuario modifica manualmente alguna de las propiedades

	de una captura
Dependencias	Captura existente
CU-9	**Baja captura**
Descripción	El usuario da de baja una captura existente
Dependencias	Captura existente
CU-10	**Alta ocurrencia**
Descripción	El usuario da de alta una ocurrencia de una captura
Dependencias	Captura existente
CU-11	**Modificación ocurrencia**
Descripción	El usuario modifica alguna de las propiedades de una ocurrencia de una captura
Dependencias	Ocurrencia existente
CU-12	**Baja ocurrencia**
Descripción	El usuario da de baja una ocurrencia existente
Dependencias	Ocurrencia existente

Tabla 9 Casos de uso: Captura

Explotación

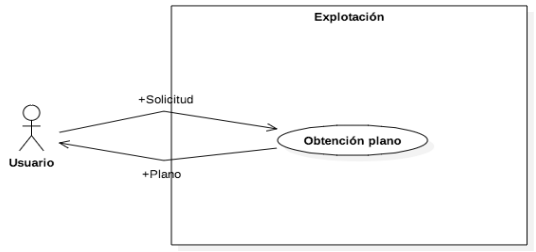


Ilustración 10 Casos de uso: Explotación

CU-13	**Obtención plano (ocurrencias por criterio)**
Descripción	Obtener la posición en plano de unas ocurrencias
Dependencias	Capturas y sus ocurrencia, criterio de selección

Tabla 10 Casos de uso: Explotación

Términos usados:

Captura: Artículo o ítem.

Ocurrencia: Datos de una captura particulares para una ubicación concreta.

Se completó el diagrama de casos de uso con el diagrama de casos de abuso siguiente:

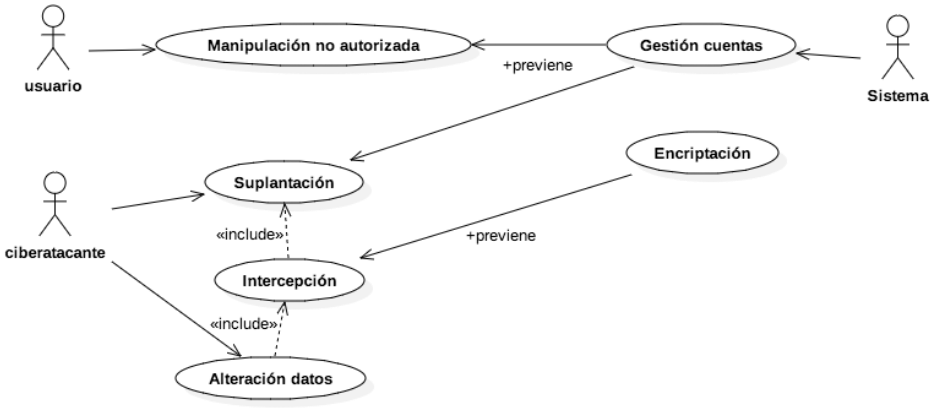


Ilustración 11 Casos de abuso

Requisitos funcionales

Id	**1 Alta búsqueda**
Resumen	Crear búsqueda que aglutinará las capturas realizadas y permitirá añadir otros participantes en la búsqueda
Entradas	Fecha/Hora, Descripción, Lista de participantes
Resultado	Se añade una búsqueda
Id	**2 Nueva captura**
Resumen	Realiza la captura de un elemento nuevo
Entradas	Fotografía, precio (manual), posición, descripción (opcional), observaciones (opcional)
Resultado	Se añade a una búsqueda el nuevo elemento junto con sus atributos de la entrada introducidos.
Id	**3 Nueva ocurrencia de captura**
Resumen	Para un elemento existente, captura nueva ocurrencia de datos
Entradas	Fotografía (opcional), precio (manual), posición, observaciones (opcional)
Resultado	Se añade una nueva entrada en los atributos de un elemento existente.
Id	**4 Modificación captura**
Resumen	Para un elemento existente, poder modificar sus datos
Entradas	Datos principales elemento existente (Fotografía por defecto, descripción y observaciones)
Resultado	Se modifica la fotografía (nueva o existente en sus ocurrencias), descripción u observaciones del elemento.

Id	**5**	**Modificación ocurrencia**
Resumen	Poder modificar los atributos de una ocurrencia dada de una captura.	
Entradas	Datos ocurrencia.	
Resultado	Se modifica cualquiera de los atributos de la ocurrencia.	
Id	**6**	**Baja ocurrencia**
Resumen	Poder eliminar una ocurrencia de una captura.	
Entradas	Ocurrencia	
Resultado	Se elimina la ocurrencia de la captura.	
Id	**7**	**Baja captura**
Resumen	Poder eliminar una captura con sus ocurrencias.	
Entradas	Captura	
Resultado	Se elimina la captura, localmente, junto con sus ocurrencias	
Id	**8**	**Representación capturas sobre plano**
Resumen	Dado una selección de capturas y un criterio, representar en el plano las ocurrencias de las capturas que cumplan con el criterio.	
Entradas	Selección de capturas, Criterio, Ocurrencias de capturas	
Resultado	Plano con las ocurrencias marcadas (y descritas), junto con la posición (opcional) del terminal (usuario).	
Id	**9**	**Modificar datos búsqueda**
Resumen	Modificar los datos (fecha/hora, descripción, lista de participantes) de una búsqueda	
Entradas	Búsqueda	
Resultado	Búsqueda modificada	
Id	**10**	**Baja búsqueda**
Resumen	Dar de baja una búsqueda	
Entradas	Búsqueda	
Resultado	Búsqueda dada de baja.	
Id	**11**	**Compartir capturas**
Resumen	Los datos de una captura (o sus ocurrencias) deben estar disponibles para todos los participantes de una búsqueda.	
Entradas	Captura, Ocurrencias captura, Participantes	
Resultado	Los datos de la captura son accesibles por todos los participantes.	
Id	**12**	**Compactación capturas**
Resumen	Poder unificar en una única captura las ocurrencias de varias. El	

	resto de capturas quedarán dadas de baja.
Entradas	Captura principal, lista de capturas que unificar
Resultado	La captura principal contendrá todas las ocurrencias de la lista de capturas, mientras que el resto quedarán dadas de baja.
Id	**13 Segregar ocurrencias de captura**
Resumen	Dadas las ocurrencias de una captura, poder extraer un subconjunto de ellas en una nueva captura.
Entradas	Selección de ocurrencias captura, datos nueva captura
Resultado	Las ocurrencias seleccionadas se traspasarán a una captura, quedando la captura que las albergaba con el resto.

Tabla 11 Requisitos funcionales

Requisitos no funcionales

IdNF	**1 Móvil (Android/Multiplataforma)**
Resumen	La aplicación requerirá terminal móvil de captura de datos. Se recomendará que sea multiplataforma o al menos Android (JAVA o .NET)
IdNF	**2 Cliente web HTML5**
Resumen	La aplicación requerirá una aplicación de backend que permita realizar las funcionalidades que se determinen incómodas para realizar en el terminal móvil (jQUERY / Angular 2)
IdNF	**3 Cliente-Servidor JAVA**
Resumen	La aplicación debe construirse siguiendo una arquitectura cliente servidor con servicios JAVA que permita compartir datos fácilmente.
IdNF	**4 Seguridad**
Resumen	La aplicación debe construirse siguiendo un S-SDLC que permita asegurar un buen nivel de seguridad.
IdNF	**5 Accesible / Usable**
Resumen	La UI de la aplicación debe construirse siguiendo normas de accesibilidad y usabilidad.
IdNF	**6 Comunicar posición**
Resumen	La aplicación debe permitir indicar la posición al resto de participantes de una búsqueda (podría desactivarse temporalmente).
IdNF	**7 Comunicación texto**

Resumen	La aplicación debe permitir enviar mensajes de texto (al menos) al resto de participantes de una búsqueda (al modo de un chat grupal).
IdNF	**8 Uso encriptación**
Resumen	Se requerirá el uso de encriptación para las comunicaciones entre cliente y servidor.
IdNF	**9 Gestión de cuentas**
Resumen	Se deben poder gestionar las cuentas de usuario y gestionar las políticas de seguridad asociadas

Tabla 12 Requisitos no funcionales

4.2.2.3 Modelado de alto nivel

Con la información de flujos de datos realizamos los siguientes modelos (E/R y clases):

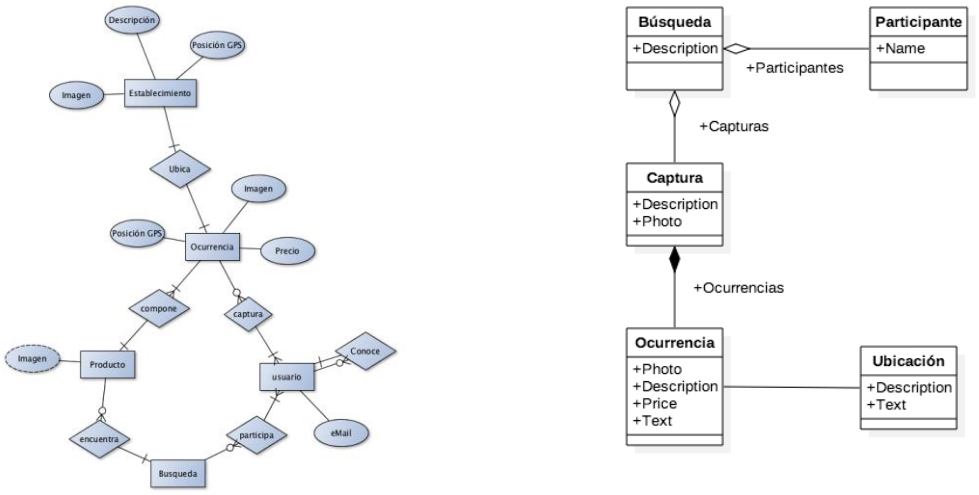


Ilustración 12 E/R y Modelo de clases de análisis

Del enunciado del proyecto podemos obtener las siguientes clases: Búsqueda, Captura, Ocurrencia, Ubicación y Participante. Adicionalmente añadimos las clases referentes a la mensajería.

Entidad	**Búsqueda**
Descripción	Búsqueda por parte de participantes de capturas.
Atributos	**Descripción**: Texto
	Participantes: Lista de Participante
	Capturas: Lista de Capturas
Entidad	**Participante**
Descripción	Usuario del sistema que participa en una búsqueda
Atributos	**Nombre**: Texto corto
Entidad	**Captura**
Descripción	Representación de un elemento del mundo real del cual se realiza una búsqueda.
Atributos	**Descripción**: Texto corto
	Photo: Fotografía
Entidad	**Ocurrencia**
Descripción	Realización de un elemento a capturar en una ubicación dada
Atributos	**Photo**: Imagen
	Description: Texto corto
	Price: Valor monetario
	Text: Texto
Entidad	**Ubicación**
Descripción	Reflejo de un lugar o establecimiento
Atributos	**Description**: Texto corto
	Location: Coordenadas
	Text: Texto

Tabla 13 Modelado de alto nivel

4.2.3 Diseño del sistema de información

4.2.3.1 Casos de uso reales

A parte de los casos de uso ya aparecidos en el apartado de análisis, se ampliaron con unos nuevos aparecidos por necesidades de diseño, como son los siguientes:

Mantenimiento

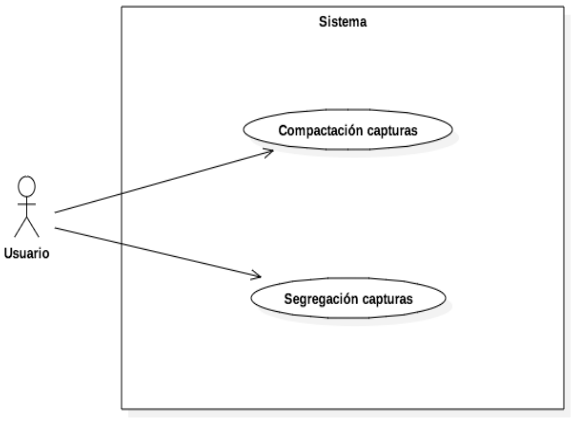


Ilustración 13 Casos de uso diseño: Mantenimiento

CU-14	**Compactación de capturas**
Descripción	El usuario agrupa varias capturas en una
Dependencias	Conjunto de capturas, una de ellas seleccionada como receptora.
CU-15	**Segregación de capturas**
Descripción	Creación de una captura a partir de ocurrencias de otra
Dependencias	Captura con ocurrencias

Tabla 14 Casos de uso diseño: Mantenimiento

Comunicación

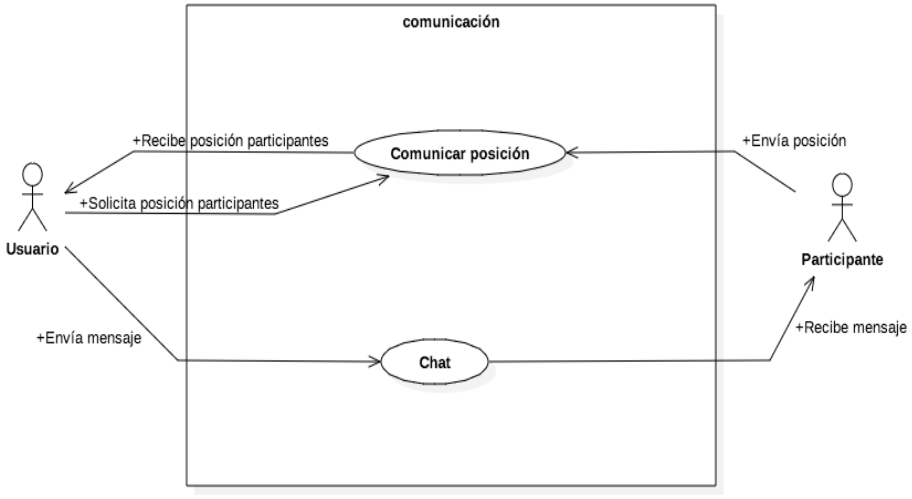


Ilustración 14 Casos de uso diseño: Comunicación

CU-16	**Comunicar posición**
Descripción	El usuario solicita la posición en plano del resto de participantes
Dependencias	Búsqueda activa
CU-17	**Chat**
Descripción	Usuario manda un mensaje al resto de participantes
Dependencias	Búsqueda activa

Tabla 15 Casos de uso diseño: Comunicación

4.2.3.2 Modelo de clases de diseño

Tras adaptar las clases resultantes del análisis (que se pueden ver en el apartado 4.2.2.4), y reflejando las distintas características aplicables a cada uno de los módulos que componen el sistema, el modelo de clases de diseño quedó conformado, mostrado de menor a mayor detalle, como se muestran en las siguientes figuras:

Cliente

Las clases de diseño se tuvieron que adaptar al patrón arquitectónico elegido para la realización física de la aplicación, en nuestro caso MVC. Para cada elemento de este patrón mostraremos como quedó afectado el modelo:

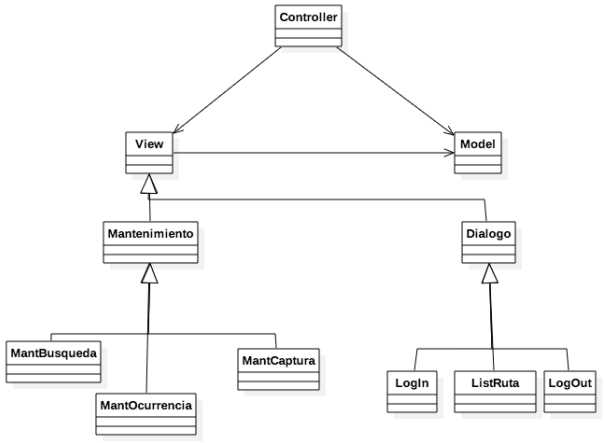


Ilustración 15 Modelo de clases de diseño: Cliente 1

Pasamos a describir brevemente las clases más importantes del modelo mostrado. Dejaremos de lado las realizaciones particulares para cada elemento concreto ya que no aportan información relevante, aparte de una implementación concreta.

Entidad	**Controller**
Descripción	Control de la interactividad usuario-aplicación
Atributos	**Vista**: View
	Modelo: Model
Entidad	**View**
Descripción	Se encarga de representar la entidad (en nuestro caso, visualmente)
Atributos	**Modelo**: Model
Entidad	**Model**
Descripción	Recoger y almacenar la información de las distintas entidades
Atributos	<depende de la entidad>
Entidad	**Mantenimiento**
Descripción	Representa la edición de un elemento
Entidad	**Diálogo**
Descripción	Representación sólo lectura de un elemento

Tabla 16 Modelo de clases de diseño: Cliente 1

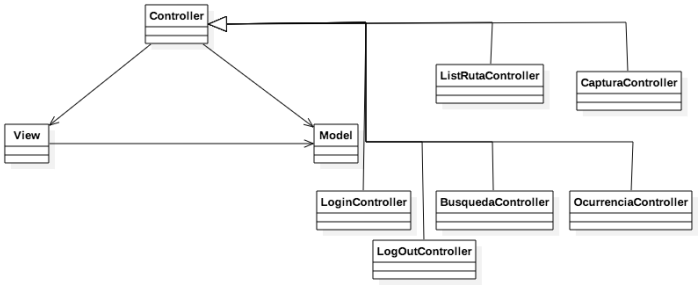


Ilustración 16 Modelo de clases de diseño: Cliente 2

En este caso concreto no introducimos más clases que no sean derivadas de una dada, como es *controller*. Cada de los indicados en la figura tienen como objetivo servir a la comunicación particular entre los módulos de vista y modelo (así como la interacción con el usuario / eventos externos) que terminarán implementando de manera visual lo que se espera de cada caso de uso de análisis.

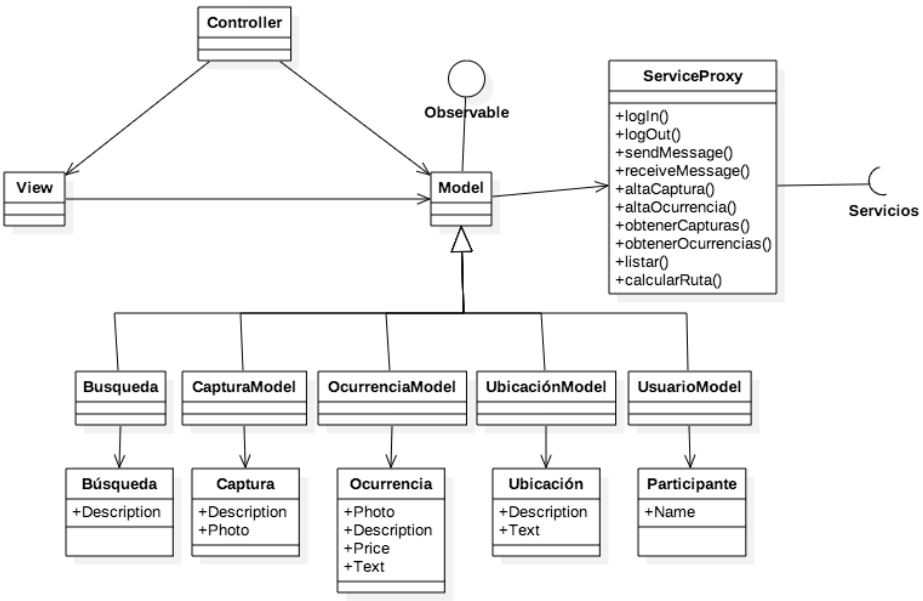


Ilustración 17 Modelo de clases de diseño: Cliente 3

Aquí aparecen nuevas clases, los modelos particulares (que envuelven los objetos de negocio aparecidos en la fase de análisis), como el objeto ServiceProxy que se encargará de mediar con el servidor, facilitando la comunicación con aquel al ocultar la complejidad inherente a la arquitectura cliente/servidor al resto de la aplicación cliente.

Entidad	**Model<T>**
Descripción	Implementa la recuperación y mantenimiento de los datos de una cierta entidad T
Atributos	**Elemento**: T
	Observers: Lista de observadores
Entidad	**ServiceProxy**
Descripción	Se encarga de implementar localmente las llamadas remotas a los servicios de la capa servidora.

Tabla 17 Modelo de clases de diseño: Cliente 3

Servidor

Lo mismo ocurre en el lado servidor de la aplicación. Las clases que han aparecido debido a la arquitectura elegida, así como por la el diseño concreto son las siguientes:

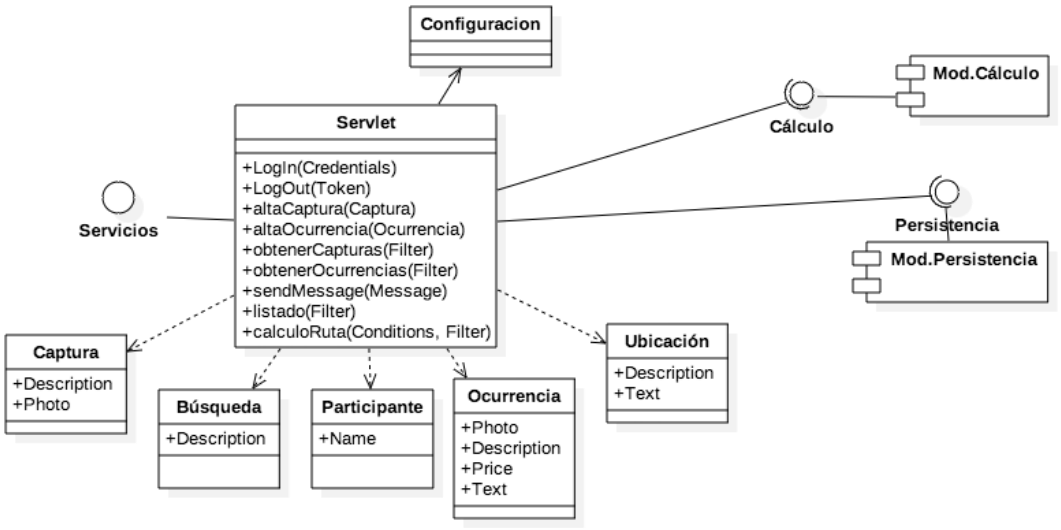


Ilustración 18 Modelo de clases de diseño: Servidor 1

En este caso concreto, podemos destacar la aparición de una clase de configuración, que permitirá instanciar las clases y factorías que se usan en el servidor con la configuración adecuada para la realización de sus tareas concretas.

Vuelven a aparecer las clases de análisis (Captura, Búsqueda, etc.) ahora usadas en el contexto del Servlet que las enviará o recibirá en los servicios que implementa. Estas clases ya encontradas en análisis se usarán para que tanto el módulo de cálculo realice su misión (listados y cálculos de rutas) de explotación de datos, como para conseguir la persistencia de los datos que encierran.

Entidad	**Servlet**
Descripción	Implementa la recuperación y mantenimiento de los datos de una cierta entidad T
Atributos	**Elemento**: T
	Observers: Lista de observadores
Entidad	**Configuracion**
Descripción	Se encarga de implementar localmente las llamadas remotas a los servicios de la capa servidora.

Tabla 18 Modelo de clases de diseño: Servidor 1

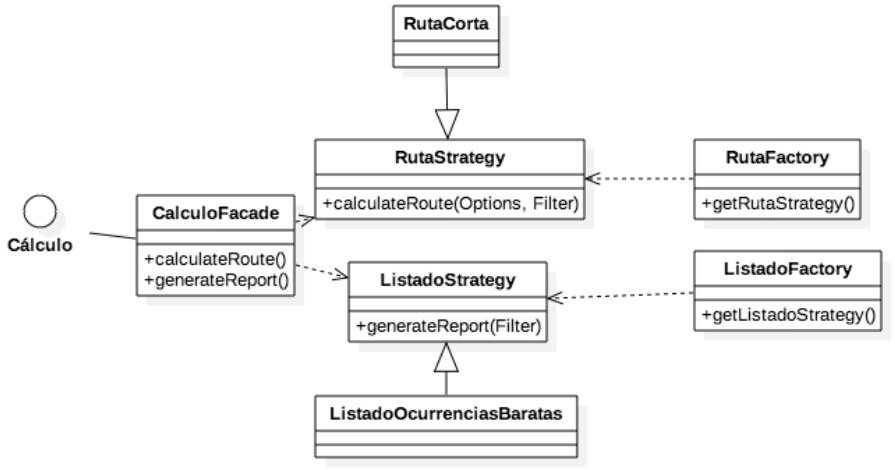


Ilustración 19 Modelo de clases de diseño: Servidor 2

En el módulo de cálculo reflejamos el uso de varios patrones de diseño, como son el *Strategy*, *Facade* y *Factory*. Estos patrones nos permitirán ocultar la complejidad de este módulo hacia el exterior por medio de una interfaz definida, cambiar los algoritmos a usar para cada una de las tareas, seleccionándolos de una colección de piezas intercambiables (aunque en la actualidad han quedado reducidos a uno para cada caso, esperamos que en un futuro a corto plazo se implementen otros), que nos devolverán las factorías.

Entidad	**CalculoFacade**
Descripción	Expone los métodos que permiten invocar los servicios del módulo de cálculos (listados y rutas).
Entidad	**RutaStrategy**
Descripción	Implementa un algoritmo de cálculo de ruta a partir de las ocurrencias de unas capturas dadas.
Entidad	**ListadoStrategy**
Descripción	Implementa un algoritmo de generación de listado.
Entidad	**RutaFactory**
Descripción	Construye los objetos necesarios para el cálculo de rutas
Entidad	**ListadoFactory**
Descripción	Construye los objetos necesarios para la generación de un listado a partir de una configuración dada.

Tabla 19 Modelo de clases de diseño: Servidor 2

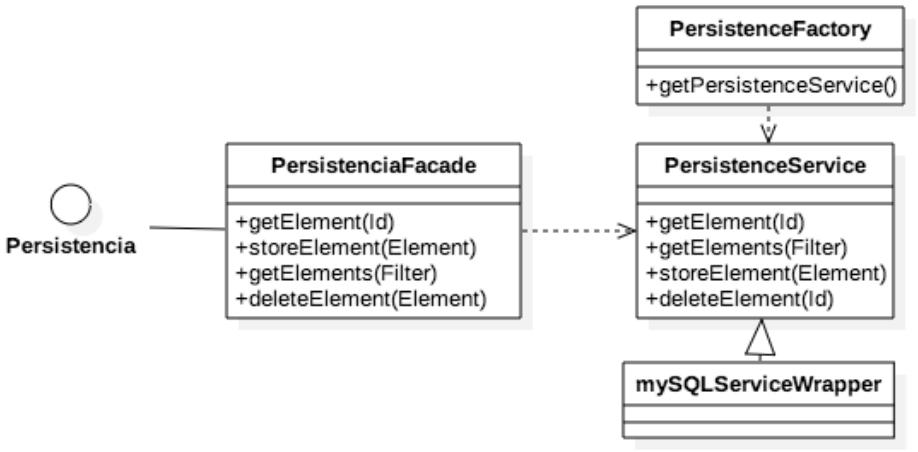


Ilustración 20 Modelo de clases de diseño: Servidor 3

En este caso concreto, el módulo encargado de la persistencia de los datos, reflejamos las clases que interactuarán con la capa física de datos, abstrayendo al resto del servicio de este trabajo. Reflejamos con estas clases que hemos desacoplado al resto de la aplicación de la capa física concreta, lo que nos permitirá en un futuro abordar otros motores de bases de datos relacionales (en principio mySQL o su derivado MariaDB), u otros tipos de persistencia noSQL.

Entidad	**PersistenciaFacade**
Descripción	Expone los métodos que permiten al servicio interactuar con la capa de persistencia sin conocer sus detalles.
Entidad	**PersistenceService**
Descripción	Implementan los métodos necesarios para recuperar y almacenar datos en el almacén físico de datos concreto.
Entidad	**PersistenceFactory**
Descripción	Devuelve una implementación concreta de PersistenceService
Entidad	**mySQLServiceWrapper**
Descripción	Implementación de PersistenceService para mySQL.

Tabla 20 Modelo de clases de diseño: Servidor 3

Aparecieron, adicionalmente, nuevas clases para la mensajería entre cliente y servidor que no se detectaron en la fase de análisis:

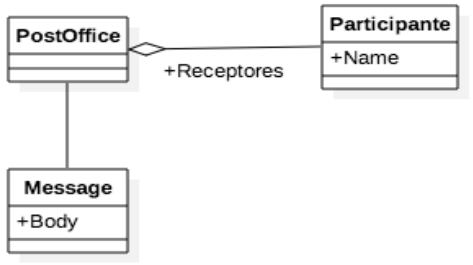


Ilustración 21 Modelo clases Mensajería

Estas clases se encargarán de mantener la información que se quiere enviar, así como los destinatarios de manera que el módulo de comunicaciones tenga lo suficiente para su envío, así como el receptor la posibilidad de aceptar o rechazarlo.

Entidad	**PostOffice**
Descripción	Entidad encargada del reparto de mensajes
Atributos	Ninguno
Entidad	**Mensaje**
Descripción	Recoge la información que se transmite entre participantes de una búsqueda
Atributos	**Origen**: Participante
	Body: Text

Ilustración 22 Modelo clases mensajería

4.2.3.3 Comportamiento de clases de diseño

Para mostrar el comportamiento de las clases de diseño usaremos los siguientes diagramas de interacción. En el primero de ellos mostraremos de manera resumida como son las interacciones entre las clases de la capa cliente. Hemos evitado que aparezcan las clases concretas (por ejemplo: los modelos, vistas y controladores particulares) ya que no aportan información adicional al modelo de interacción expuesto.

Debemos notar que, como se muestra en el modelo, los mensajes dentro del cliente son síncronos mientras que los mensajes hacia el servidor (y por lo tanto sus respuestas) son asíncronos. Para reforzar esto, hemos añadido en el modelo el sufijo *async.*

El mensaje ReceiveMessageAsync hace referencia al servicio *push* implementado por la infraestructura en la nube que usaremos y que nos remitirá a los clientes los mensajes que el servidor deba enviarnos (generalmente enviados por otros clientes, lo que no excluye que se puedan usar para mensajes del sistema u otro tipo de notificaciones futuras.).

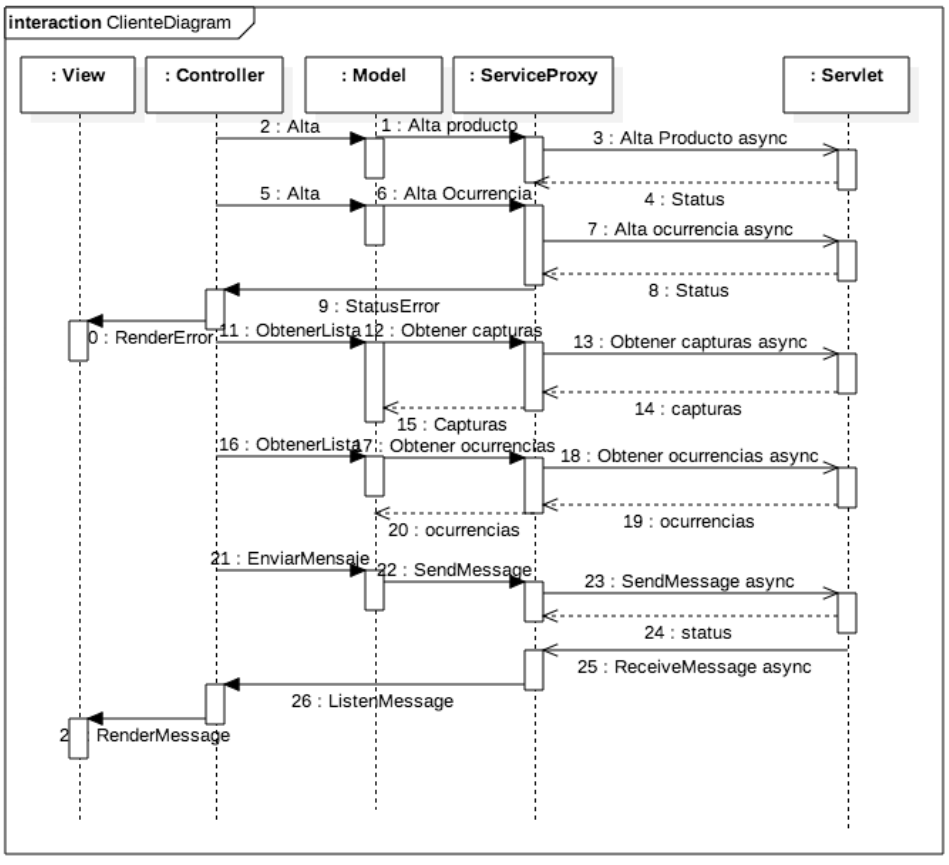


Ilustración 23 Diagrama interacción clases cliente

En el diagrama que acabamos de ver, faltan algunos mensajes que exponemos en el siguiente diagrama, el que muestra las interacciones en la capar servidora (concretamente en el *servlet* y los módulos que usa). Los mensajes en cuestión, GenerarReport y CalcularInforme son como el resto de mensajes ya descritos y poco más aportan al diagrama anterior. Aun así, los hemos incluido en el siguiente ya que afectan a un módulo distinto al del resto de mensajes que recibe el *servlet*.

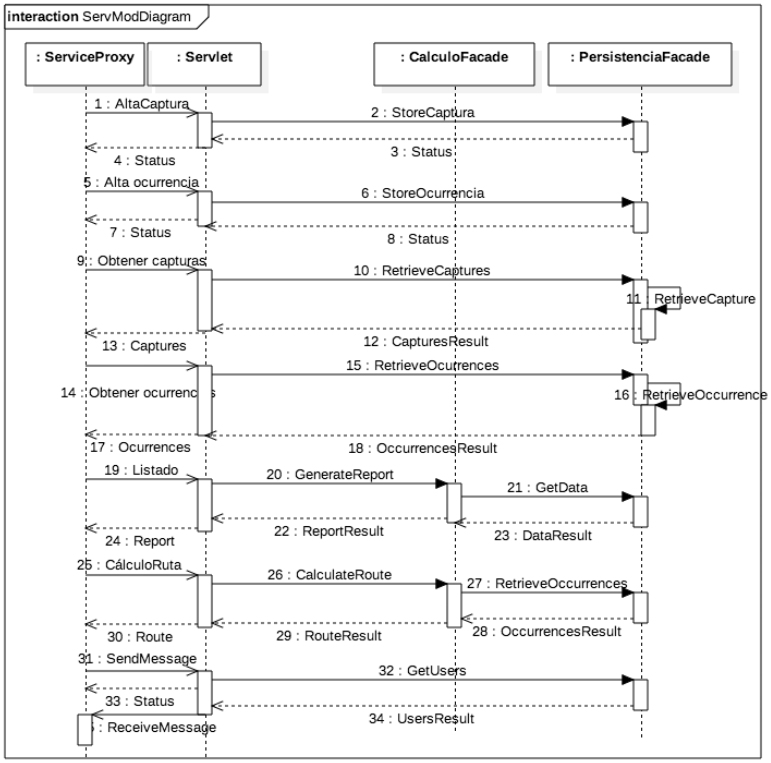


Ilustración 24 Diagrama interacción clases servidor

En el siguiente diagrama mostramos las interacciones del módulo de persistencia, en el que se creará un módulo concreto de acceso a datos y persistencia (en nuestro caso para mySQL) que se esconderá tras la clase / interfaz de PersistenceService y que nos habrá creado (según configuración) la factoría de persistencia (PersistenceFactory).

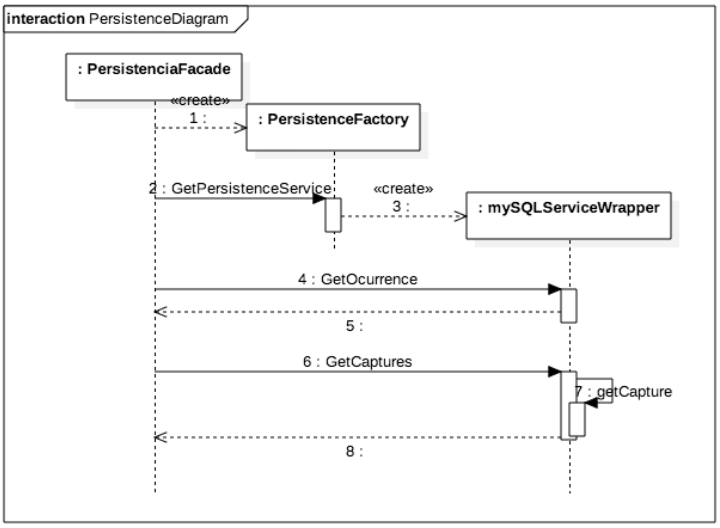


Ilustración 25 Diagrama interacción clases servidor: Persistencia

Terminamos con el diagrama que muestra la interacción, muy parecida a la que acabamos de ver en la capa de persistencia, de las clases de cálculos. Lo mostramos con dos mensajes, generación de listado (GenerateReport) y cálculo de ruta (CalculateRoute) que, como ya hemos visto anteriormente, usan piezas intercambiables (en este caso estrategias) para permitir el intercambio de algoritmos de cálculo de listado y rutas generadas por sendas factorías.

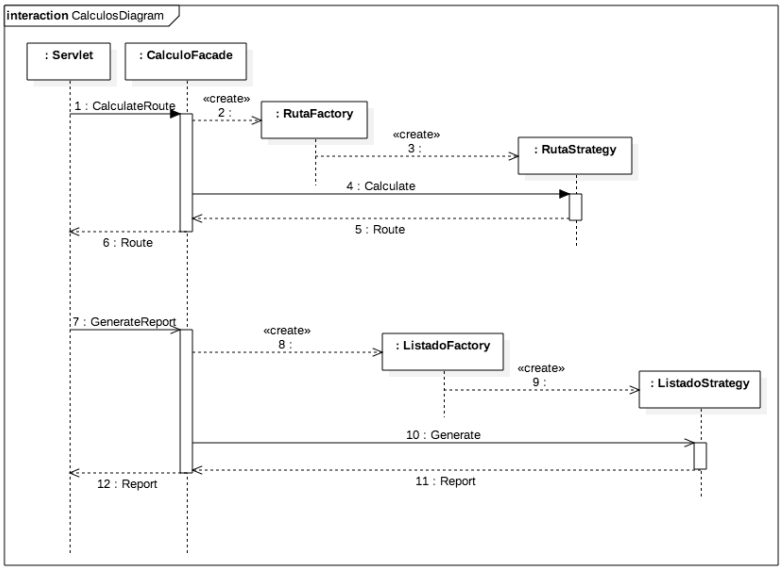


Ilustración 26 Diagrama interacción clases servidor: Cálculos

4.2.3.4 Diseño módulos del sistema

Los distintos módulos del sistema quedaron configurados tal y como se puede ver en las siguientes imágenes. Al proporcionar una visión general hemos unificado los módulos correspondientes a las distintas aplicaciones cliente en uno sólo, al ser equivalentes (aunque construidos con las particularidades de cada entorno concreto).

De manera general, los módulos del sistema quedaron:

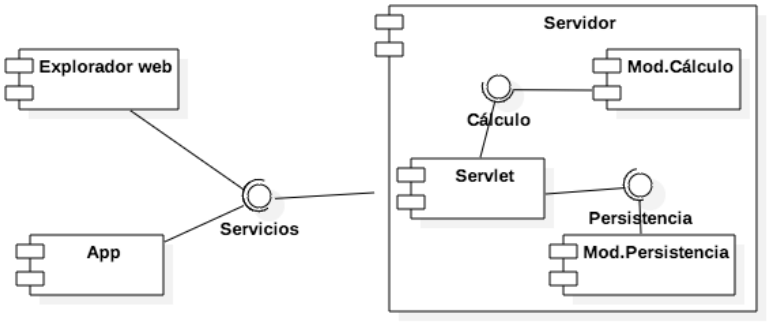


Ilustración 27 Módulos del sistema: Vista Global – Servidor

Hemos detallado en el módulo que refiere al servidor los componentes que lo forman y descrito sus relaciones por medio de interfaces, que permiten desacoplarlos entre sí. También se puede observar cómo hacemos notar que tanto la aplicación web como la aplicación móvil (*app*) usan, realmente, los mismos servicios. Esto es así porque tanto la aplicación nativa para móvil como la aplicación web se encargan de la presentación y lógica de control relativa a la interacción con el usuario. La lógica de negocio se concentra en el servidor.

En relativo a los clientes de la aplicación, la descomposición en módulos se hizo siguiendo el patrón de diseño de arquitectura MVC (*Model/View/Controller*) practicados durante el máster en la asignatura de programación en el servidor web. Aunque se usó PHP concretamente, este patrón es independiente del lenguaje y se usa comúnmente tanto en aplicaciones de escritorio, cliente servidor y, concretamente, web de manera profusa [58].

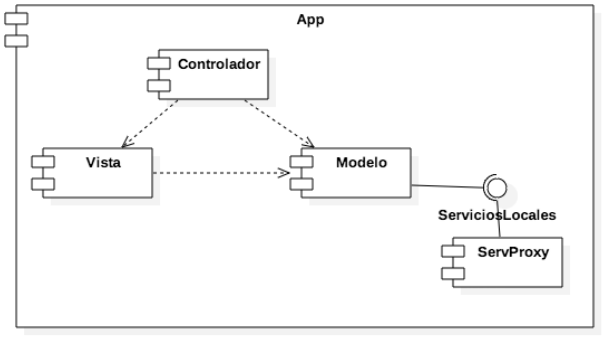


Ilustración 28 Módulos del sistema: Cliente

4.2.3.5 Modelo físico de datos

A partir del modelo lógico introducido en los pasos anteriores y de las características físicas del S.G.B.D. inicialmente escogido para la realización, se definieron las siguientes estructuras de datos para la implementación.

Nota definiremos el metadato ***Guid*** para referir a **binary(16)** de mySQL (por lo que no indicaremos tamaño en los campos en los que aplique). Las columnas con tipo *Guid* se generarán automáticamente en base de datos usando disparadores (*triggers*).

User	**Usuarios / Amigos / Participantes**					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
eMail	Varchar	320			Sí	Sí
Alias	Varchar	64				
Passhash	Binary	64				Sí

Tabla 21 Modelo físico de datos: Usuarios

En esta tabla se almacenan los usuarios del sistema, que internamente tiene asociados un GUID y el hash de hasta 512 bits correspondiente a la contraseña.

Los usuarios, debido a sus roles, se relacionan tanto con otros usuarios (amigos) como con búsquedas (participantes). Para mantener esta relaciones decidimos crear dos tablas (en vez de una única tabla de relaciones entre entidades y así evitar un crecimiento desmedido que pudiera penalizar el acceso), que exponemos a continuación. Por optimización, y aunque se tratase de una relación 1-n, decidimos añadir un id a cada tabla para evitar el uso de claves compuestas (aunque nos permita crear relaciones n-n, añadiendo algo de complejidad lógica que requiera mayor documentación).

UserContacts	Amigos usuario					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
UserId	Guid		Sí	Users.Id		Sí
ContactId	Guid			Users.Id		Sí

Tabla 22 Modelo físico de datos: Amigos (de usuario)

SearchContacts	Participantes					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
SearchId	Guid			Users.Id		Sí
ContactId	Guid			Users.Id		Sí

Tabla 23 Modelo físico de datos: Participantes (de búsqueda)

La siguiente entidad importante es la búsqueda, agrupadora de los artículos que se tratan de localizar.

Searchs	Búsqueda					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
Description	Varchar	64				Sí
DateCreation	Datetime					Sí
Owner	Guid			Users.Id		Sí
PhotoId	Guid			Resources.Id		

Tabla 24 Modelo físico de datos: Búsquedas

Como se puede apreciar, usamos esta tabla para mantener los datos básicos de una búsqueda, como son su descripción, usuario poseedor (creador) y otros como la fecha de creación y recurso (foto) que se usa visualmente para su identificación (podríamos haber evitado este campo pero decidimos usarlo para evitar, en caso de no tener una foto seleccionada, hacer productos cartesianos [*join*] o búsquedas inversas desde una tabla de recursos).

La siguiente entidad son los artículos (capturas) objetivo de las búsquedas.

Items	Búsqueda					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
Description	Varchar	64				Sí
DateCreation	Datetime					Sí
Owner	Guid			Users.Id		Sí
Photo	Guid			Resources.Id		

Tabla 25 Modelo físico de datos: Capturas (productos)

Al igual que ocurre con las tablas de relaciones 1-n, hemos preferido mantener separadas las tablas *Items* de *Searchs* (aunque son físicamente iguales) para usar la diferencia semántica para separar los datos.

La asociación entre un artículo y una búsqueda se materializa en la siguiente tabla de relación (con la misma consideración que con las anteriores tablas de relaciones).

SearchItems	Participantes					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
SearchId	Guid			Search.Id		Sí
ItemId	Guid			Items.Id		Sí

Tabla 26 Modelo físico de datos: Capturas de búsqueda

Para concluir con las capturas (artículos) la tabla que refleja sus ocurrencias.

ItemOcurrences	Participantes					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
ItemId	Guid			Items.Id		Sí
PhotoId	Guid			Resources.Id		No
Price	Numeric	10,2				Sí
Location	Spacial:Point					Sí

Tabla 27 Modelo físico de datos: Ocurrencias de captura

A parte de referencias a la captura de la que forma parte, su foto y precio, hemos usado las extensiones mySQL para datos espaciales para almacenar la ubicación GPS de la ocurrencia. Con esto, hemos obviado en primera versión el crear una tabla propia para almacenar las ubicaciones.

Las tablas adicionales, para almacenar los recursos

Resources	Recursos (fotos)					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			Sí
Data	LONGBLOB					Sí

Tabla 28 Modelo físico de datos: Recursos

Inicialmente decidimos almacenar los recursos de gran tamaño (fotos) en base de datos. Según el rendimiento que obtengamos, es posible que en posteriores versiones busquemos otro método de almacenamiento (como puede ser usar el sistema de ficheros del S.O. subyacente, por ejemplo).

Por último, requeriremos una tabla para almacenar los mensajes a enviar entre los distintos usuarios de la plataforma, ya sean mensajes de usuario (conversación / *chat*) como de sistema (notificaciones internas). Aunque rompamos un poco con la política seguida hasta el momento de particionar los datos para facilitar el trabajo del s.g.b.d, en este caso hemos decidido mantener los mensajes en una única tabla, con visos a realizar pruebas futuras con un motor noSQL.

Messages	Recursos (fotos)					
Nombre	**Tipo**	**Tamaño**	**PK**	**FK**	**IDX**	**Obligatorio**
Id	Guid		Sí			
SourceId	Guid					Sí
TargetId	Guid					Sí
Status	Integer					Sí
Data	TEXT					

Tabla 29 Modelo físico de datos: Mensajería

4.2.3.6 Diseño de interfaz de usuario

Siguiendo las directrices de usabilidad dictadas por las plataformas sobre las que se ejecutará la aplicación móvil, realizamos el siguiente prototipo visual sobre el que se fundamentará la interfaz visual de la aplicación. Hemos de tener en cuenta, también, que debido a la herramienta seleccionada para la implementación no dispondremos de un acceso total a las posibilidades de la plataforma subyacente. Debimos asumir este compromiso por la naturaleza multiplataforma de la solución y a las diferencias entre las guías de usabilidad establecidas por los propietarios.

En la mayoría de los casos la herramienta seleccionada ya realiza parte del trabajo de adaptarse a las guías de usabilidad y experiencia de usuario, lo que nos facilitó el trabajo ulterior. Aun así no debemos dejar de lado que se siguió, adaptados al desarrollo móvil, una

guía de principios heurísticos de usabilidad recibida durante las clases de “Usabilidad, Accesibilidad y Métricas para sitios Web” [66].

A continuación, mostraremos el primero prototipo de navegación por la aplicación realizada con la herramienta de desarrollo multiplataforma Xamarin [67]. Las capturas se realizaron sobre el cliente Android, aunque se mostrarán algunas capturas de la versión iOS para ilustrar la adaptabilidad de la herramienta a las diferencias en la experiencia de usuario exigida por las plataformas reales.

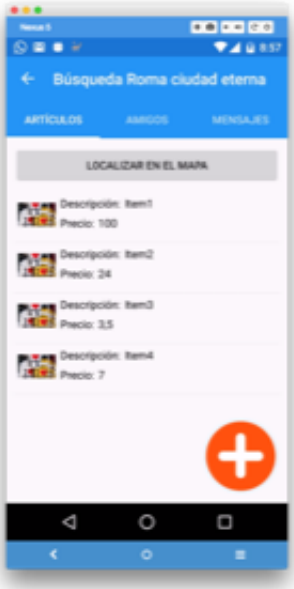
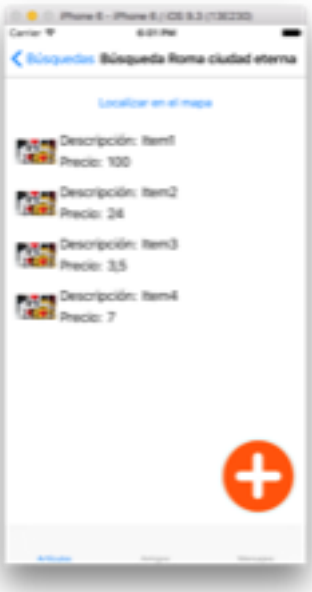
El flujo diseñado para la aplicación móvil se refleja en la siguiente figura:

Ilustración 29 UI: Flujo aplicación móvil

En rojo hemos marcado la ruta para llegar al objetivo (ubicación espacial de los artículos encontrados con menor precio). En verde, parte del trabajo a realizar para conseguir el objetivo. Hemos notado con una doble flecha los caminos que permiten regreso y con una flecha aquellos que no permiten retorno (hemos obviado en el flujo la opción de salir, que nos devolvería al inicio del flujo. Esta opción se encuentra en el desplegable de opciones de sistema).

4.3.2.6.1 Multiplataforma

Para mostrar la capacidad de la aplicación para adaptarse a las distintas plataformas, mostraremos a continuación un par de ejemplos de pantallas de la misma en los dos entornos sin requerir código especial o condicionado.

Android	iOS
	
	

Como podemos apreciar en las imágenes que mostramos por parejas (en la primera fila, el ejemplo de autenticación en el sistema y en la segunda la pantalla central de búsquedas), constatamos las diferencias sutiles entre las dos plataformas generadas por el mismo código.

Aunque muy similares, vemos como los botones cambian de apariencia, notablemente en iOS con su diseño minimalista. Además, las pestañas cambian de apariencia y posición siguiendo los dictados de la plataforma en la que se ejecutan (Android en la parte superior, iOS en la inferior).

Ilustración 30 UI: Diferencias plataforma

4.3.2.6.2 Autenticación y Acceso

Ilustración 31 UI: Autenticación y Acceso

En esta figura nos centramos en las primeras dos columnas del flujo mostrado al principio del capítulo.

En la primera pantalla podemos observar las opciones que hemos presentado al usuario para autenticarse o darse de alta en el sistema. Como se puede apreciar, hemos optado por usar el correo electrónico como identificador de usuario, lo que nos permite enviarle instrucciones en caso de que olvide la contraseña o, en el futuro, pasos a seguir para registrase completamente (evitar ataques de denegación de servicio o suplantación).

Una vez que el usuario introduzca las credenciales o se de alta en el sistema, accederá a la pantalla principal de la aplicación.

4.3.2.6.3 Opciones de la aplicación

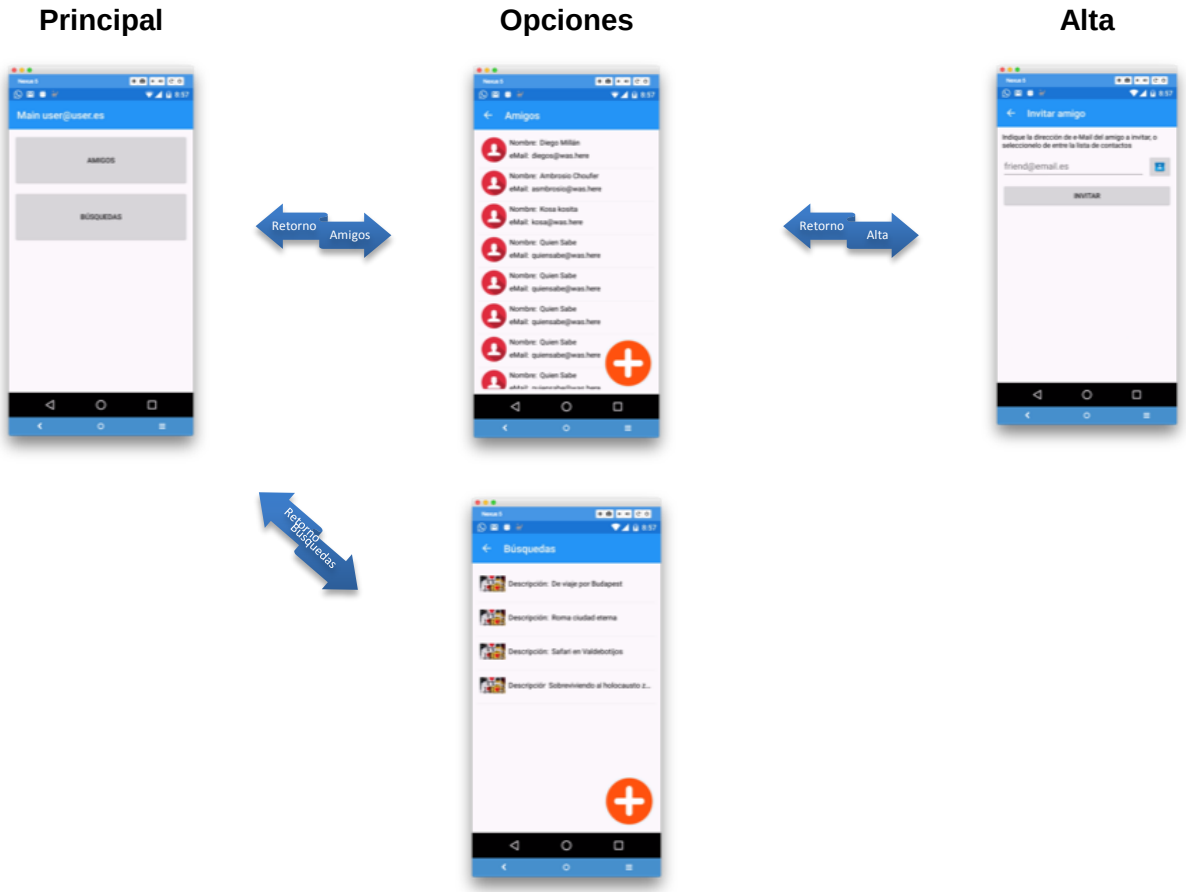


Ilustración 32 UI: Opciones principales

En esta figura mostramos las opciones principales de esta primera versión de la aplicación. Como ya hemos indicado anteriormente, no reflejamos la opción de salir del sistema, ya que la encontramos en las opciones de sistema que se encuentra en todas las pantallas y oscurecería el flujo.

En esta primera versión, nos hemos centrado en lo fundamental para poder realizar una funcionalidad completa aunque limitada. Para ello, hemos diseñado un mantenimiento de “amigos” o contactos, a los cuales se les mandará una invitación (que deberán aceptar). Se dispondrá de un botón para seleccionarlos de entre los contactos del móvil.

La otra opción es la que nos permite disponer de un mantenimiento de distintas búsquedas realizadas en el tiempo, ayudados de un subconjunto de los contactos o amigos matriculados.

4.3.2.6.4 Búsquedas

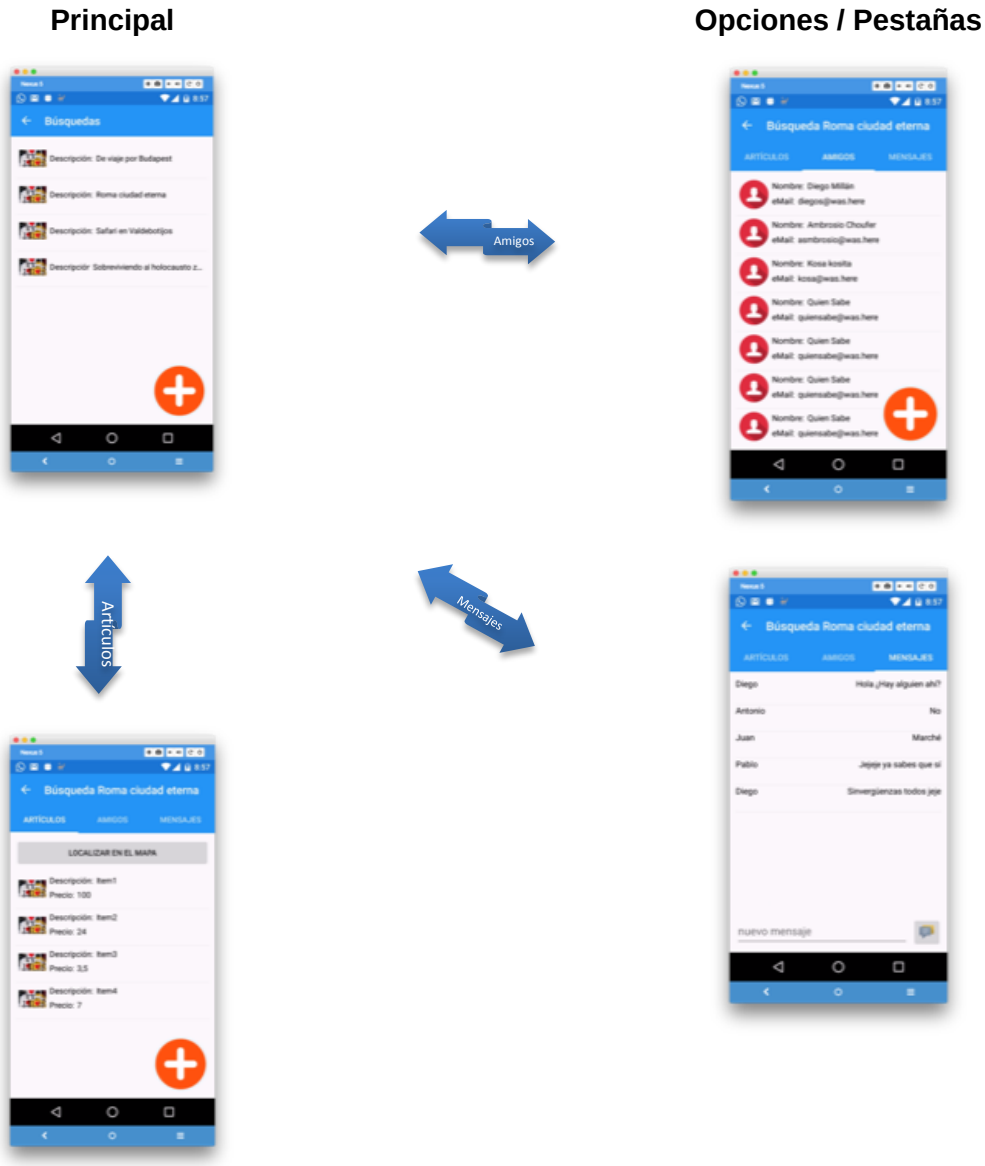


Ilustración 33 UI: Opciones Búsqueda

En esta ilustración nos centramos en las opciones disponibles para una búsqueda existente. El alta de artículos será detallada en la siguiente figura.

Como podemos apreciar, se ha diseñado una búsqueda con tres opciones principales. Mantenimiento de artículos, que podemos ver debajo de la pantalla principal, mantenimiento de amigos que participan en la búsqueda y ventana de mensajes entre los mismos.

En el mantenimiento de artículos se ha incluido el botón que nos permitirá acceder al mapa o listado que nos devolverá el resultado de la búsqueda.

El mantenimiento de amigos nos permitirá seleccionar a uno de los disponibles en el mantenimiento global de amigos (previa invitación y aceptación). La ventana de mensajes

nos permitirá enviar a cada uno de los participantes un mensaje, al modo de las aplicaciones de mensajería instantánea como *whatsapp*, *hangouts* y demás, salvando las distancias.

4.3.2.6.5 Búsqueda: Artículos

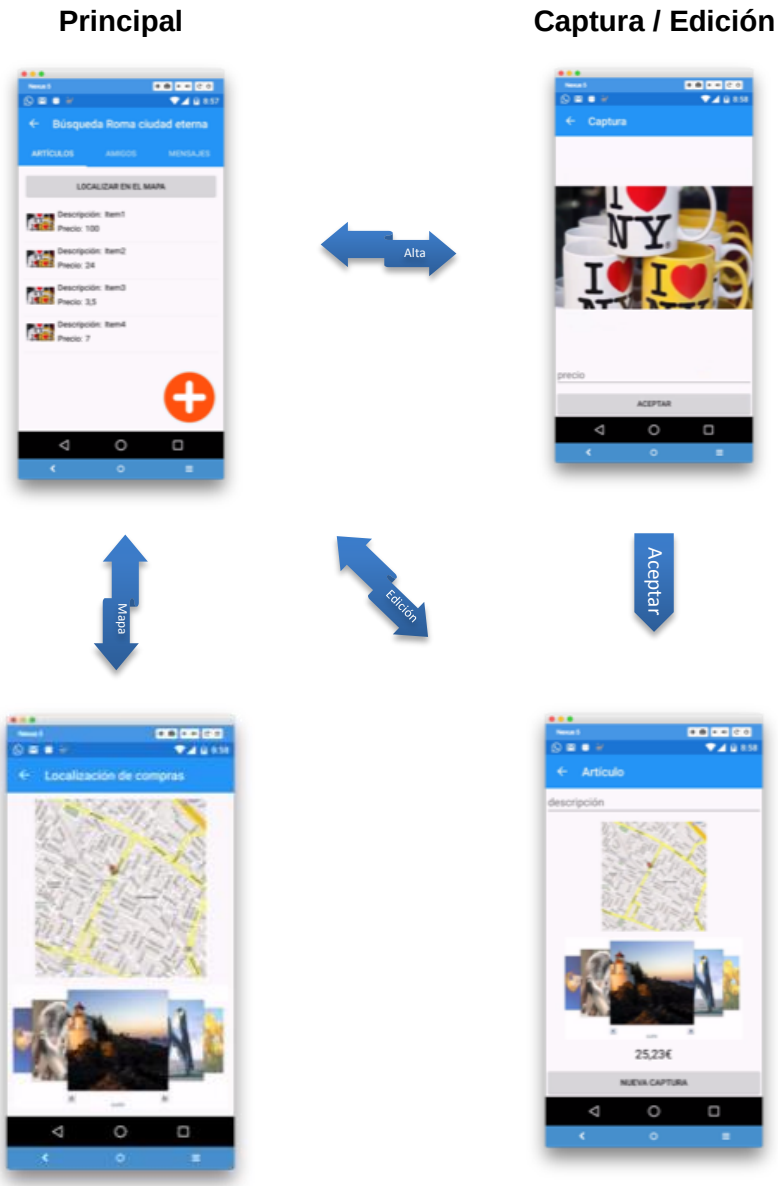


Ilustración 34 UI: Búsqueda - Artículos y Localización

Finalmente, quizás el grupo de pantallas más importante en la aplicación. Aquí mostramos el flujo de alta y trabajo con los artículos que componen una búsqueda. Como la aplicación está pensada para el trabajo de campo (*in situ*), diseñamos el alta como manera de recoger los datos de la primera ocurrencia del artículo encontrado. Para ello tomamos una foto e indicamos el precio de la ocurrencia. La aplicación acompañará estos datos con la localización (*gps*). Posteriormente, ya dado de alta, se accede a la pantalla de edición del

mismo artículo, en el que se muestran mediante un carrusel, las fotos de las distintas instancias (si se proporcionaron) y los precios, así como su ubicación en el mapa. Aquí se podrá indicar la descripción del artículo. A esta pantalla será la obtenida si editamos desde la selección de artículos. Dispondremos aquí, del botón para repetir el proceso y agregar una nueva ocurrencia al artículo.

En la selección de artículos disponemos del botón que nos mostrará sobre el plano la ubicación de las ocurrencias más económicas de los distintos artículos seleccionados.

4.3 Construcción

Inicialmente se planteó la construcción final de la solución siguiendo un plan dividido en cuatro fases consecutivas, como se muestra en el siguiente cronograma:

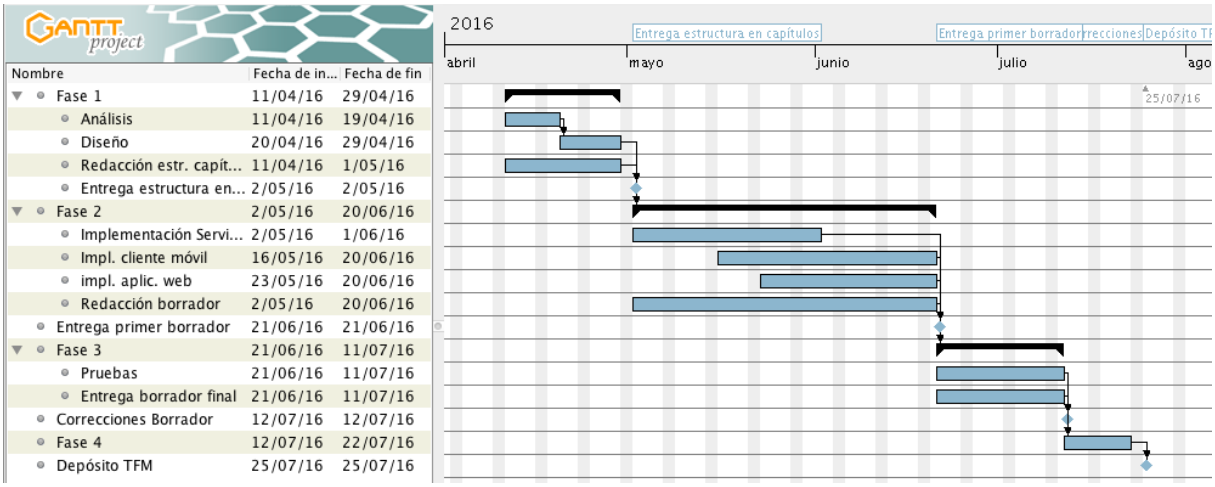


Ilustración 35 Construcción: Cronograma inicial

Debido a la falta de recursos (principalmente tiempo), debimos desplazar las fases dos y tres al verano, para realizar la entrega en septiembre en vez de en julio, con lo que la planificación para la construcción final quedó:

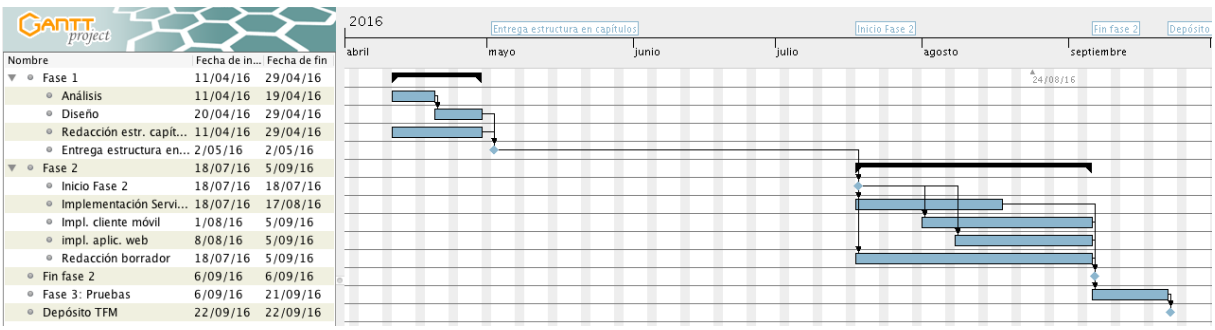


Ilustración 36 Construcción: Planificación final

La cuarta fase, que inicialmente era un colchón en la primera planificación, se pierde en la final, como también se eliminó la necesidad de la entrega del borrador.

Pero lo importante quedó prácticamente intacto. La fase 2, la fase de construcción propiamente dicha, siguió el camino dictado por la planificación inicial, con la salvedad del desplazamiento temporal. Pero durante la primera semana de trabajo, debido a nuestra poca experiencia con las herramientas, vimos que no era del todo posible una primera versión de la solución con los dos clientes previstos. Por esto, decidimos descartar momentáneamente la creación del cliente web. No es esencial para el funcionamiento de la aplicación, aunque si será útil para un futuro mantenimiento de la misma.

Por lo tanto, se decidió construir tanto la capa servidora como la cliente móvil. Para ello se dividió en varias actividades. Cada una estaba relacionada con el módulo a realizar, y cada una en una serie de tareas.

4.3.1 Servicios (capa servidora)

Configuración servidor.

Debido a que contemplamos la seguridad desde el principio, comenzamos nuestros primeros pasos en la nube trabajando en el bastionado del servidor y su repercusión en la visibilidad externa del mismo. Para ello nos servimos de herramientas como NMAP. Hicimos nuestras primeras pruebas con certificados digitales para securizar la comunicación usando TLS para dar soporte a https.

Construcción base de datos mySQL.

La base de datos era un aspecto importante de la aplicación, no sólo por los aspectos de rendimiento e integridad de los datos como por ser otra pieza del rompecabezas que conforma la seguridad. Aparte de diseñar las tablas, debimos restringir los permisos sobre las mismas para que sólo unos procedimientos almacenados tuvieran los privilegios necesarios para realizar las operaciones *CRUD*, así como servir de base en la posterior autenticación en la aplicación.

Construcción servicios JAVA

Una vez contamos con una infraestructura mínima, pudimos ir creando (en parte solapadamente) los esqueletos de los servicios java (*Servlets*). Los implementamos usando el protocolo SOAP debido a que están específicamente diseñados para dar soporte a una mensajería sin estado. Aunque añade una carga adicional por el procesamiento de mensajes XML, también permite en un futuro usar *gateway XML y*

firewall XML que podrían ayudar a dotar de mayor seguridad a la comunicación y abrir los servicios a otras aplicaciones y clientes.

4.3.2 Cliente móvil

Como habíamos planificado, el desarrollo del cliente móvil se solapó en el tiempo con la del servidor. Así, pudimos resolver los cambios que surgieron por la realización práctica y el conocimiento que fuimos adquiriendo durante el mismo, así como pulir aspectos del diseño. Esto no quiere decir no hubiera problemas, como la necesidad constante de buscar información para realizar cualquier tarea con la herramienta elegida (Xamarin). Pero pudimos disfrutar de una curva de aprendizaje muy suave gracias a los conocimientos de la plataforma .net así como de Android adquiridos durante el curso.

En el cliente móvil también se ve reflejado, aunque de manera parcial, el trabajo de diseño de la base de datos al requerirse cierta persistencia local de datos para un funcionamiento más óptimo y minimizar la demanda de una comunicación continúa con el servidor. Esto lo conseguimos con un almacén local de datos, a modo de *cache*. Podemos decir que tuvimos que implementar una pequeña versión de la base de datos contenida en el servidor, con la que dispondremos de los datos más recientes referidos únicamente al usuario particular del cliente móvil.

4.4 Implantación y aceptación

Debido a la ausencia de un equipo completo de personas dedicadas a la puesta en marcha, implantación y mantenimiento de la solución creada, es el propio equipo de desarrollo el que llevó a cabo la implantación aceptación. Sabemos que no es la situación ideal, pero permite simplificar las tareas de implantación al no requerir un esfuerzo de comunicación entre partes. Aun así, es recomendable crear una serie de documentos que guíen la implantación y provean de unos criterios claros de aceptación, para que tanto el equipo de desarrollo actual como el de implantación futuro (y mantenimiento) tengan unas guías claras y no se pierda conocimiento.

4.4.1 Plan de implantación

Debemos destacar que la naturaleza cliente servidor de la solución, así como la implementación móvil del cliente que fuerza a un modelo de instalación desatendido en el que el equipo de implantación y mantenimiento tienen poco o ningún control (modelo tienda de aplicaciones), hace que los esfuerzos se vuelquen en el lado servidor: el más crítico.

Así, llevamos a cabo los siguientes pasos para la implantación del servidor y su puesta en marcha con garantías de éxito, lo que nos sirve como documentación para futuras puestas en marcha o cambios.

1) Servidor. Para el acceso externo a la aplicación sita en la nube (Azure) debimos hacer visible el servidor, así como los puertos necesarios para la comunicación necesaria de manera segura. Para ello se permitió el acceso al puerto 8443 y se configuró el almacén de certificados necesario para la comunicación usando TLS. Debido a la naturaleza de Azure, no se requirió desactivar servicios no usados ya que se deben activar de manera explícita.

2) Servidor de aplicaciones. La instalación de TOMCAT y su configuración fue el siguiente paso. Para ello se necesitó consultar las guías del fabricante (Apache Foundation) [68].

3) S.G.B.D. El siguiente paso fue instalar y configurar mySQL. Para ello aprovechamos las capacidades de Azure creando una nueva máquina virtual en la que se instaló únicamente mySQL para disponer de un servidor dedicado. Se configuró para ser accedido únicamente desde la red privada virtual. Con esto conseguimos que no estuviera expuesto al exterior, simulando un entorno real. Para concluir, ya que en seguridad hay que pensar que la red está comprometida seguimos la guía del fabricante [69] para su configuración.

4) Servicios. Finalmente, instalamos los paquetes WAR en el servidor TOMCAT con los servicios implementados (parte servidora de la aplicación). Ubicamos los ficheros de configuración y modificamos los permisos para que sólo el servidor de aplicaciones y el usuario administrador tuviera permisos sobre los mismos.

4.4.2 Pruebas

Debido a las distintas capas introducidas por la arquitectura usada, debimos realizar las siguientes pruebas, tanto funcionales, de usabilidad, de rendimiento y de seguridad.

Para empezar, las pruebas funcionales, sin las que no podemos considerar alcanzados los objetivos. Para ello, de la lista de requisitos funcionales (capítulo 4.1.1) derivamos una lista de comprobación que repartimos a los usuarios voluntarios de la aplicación. En primera fase, siguiendo los pasos de dicha lista, los usuarios debían validar el grado de cumplimiento de cada funcionalidad. En segunda fase, libre, los usuarios usaron la aplicación a su aire para detectar posibles problemas derivados de un uso real.

Para las pruebas de usabilidad, y debido a la escasez de tiempo para la realización del proyecto, optamos por derivar una lista de comprobación (*checklist*), a partir de los principios heurísticos recibidos en el trascurso de la asignatura de Usabilidad ya mencionados con anterioridad [66,6]. Para ello dispusimos de familiares “voluntarios” que realizaron las pruebas y enriquecieron con sus aportaciones. El equipo de desarrollo realizó las adaptaciones necesarias para pasar con éxito las pruebas.

Para medir el rendimiento, y debido a la plataforma usada en la nube en la que el tráfico dicta en parte el precio, nos contentamos con realizar una aplicación que permitía realizar pruebas concurrentes como si tres usuarios estuviesen trabajando al unísono. No pudimos realizar pruebas masivas ya que sería prohibitivo, pero nos puede servir lo creado para realizar tales pruebas en un entorno real futuro.

Para concluir las pruebas, las de seguridad, el equipo de desarrollo realizó pruebas usando las herramientas aprendidas durante el master, como NMAP, OWASP ZAP, OpenVAS (intrusión), Fortify SCA (análisis estático de código JAVA) y FxCOP (análisis estático código .NET). Estas pruebas se pasaban en paralelo y cada vez que se generaba una versión nueva debido a cambios (debido a resultados de las pruebas o mejoras introducidas).

5. Conclusiones y trabajo futuro

5.1. Conclusiones

A estas alturas no está de más recordar cuál era el problema original que tratábamos de resolver con la aplicación realizada. Demos un paso atrás y veamos en perspectiva el conjunto. Para ello releamos las preguntas originales "**¿podemos asegurarnos de comprar aquellos recuerdos que os podemos permitir al mejor precio?** Y **¿Dónde están, y cómo puedo hacer el menor esfuerzo para adquirirlos?".**

Bien, para dar respuesta a estas, decidimos realizar una aplicación cliente servidora que hiciera uso de las tecnologías móviles, la nube con una pizca de experiencia colaborativa, realizando todo el uso, que nos fuera posible, de los conocimientos adquiridos a lo largo del Master del cual este proyecto es su último hito. Es decir, planteamos el siguiente objetivo "desarrollar una aplicación móvil que permita recoger información sobre productos no catalogados previamente, disponibles en múltiples localizaciones y precios, para posteriormente obtener los lugares en los que se encuentran a mejor precio, de manera colaborativa." a solucionar con los conocimientos y práctica adquiridos.

Para determinar si la solución resultante del trabajo de desarrollo seguido es lo que esperábamos, también debemos desgranar el objetivo principal recuperando los objetivos secundarios que determinamos para el proyecto y comprobar su cumplimiento.

Implementar servicios en la nube.	De una manera limitada, inicial, hemos encontrado muy interesante la primera experiencia con una infraestructura de este tipo, aunque se haya tratado de un servicio gratuito, a prueba, para la toma de contacto. Hemos podido desmitificarla y tratar de manera cercana sus pros y contras.
Desarrollar un cliente móvil (Smartphone).	Nos ha alegrado en gran medida comprobar que la primera toma de contacto con este mundo realizado durante el master nos ha servido como guía a la hora de realizar una aplicación de cierta envergadura que, seguramente, de otra manera no habríamos abordado.

Desarrollar un cliente web para las operaciones de mantenimiento.	Durante la realización debimos descartar este punto por la falta de tiempo. De todas formas se deberá abordar en segunda fase.
Realizar un desarrollo contemplando la seguridad en todos sus procesos (S-SDLC)	Por fin, en un desarrollo real, hemos tenido una toma de contacto con un proceso de desarrollo en el que se ha tratado de seguir desde el principio sus actividades incluyendo la seguridad. Aunque quizás de manera superficial, se ha comprobado que aparentemente cualquier SDLC puede ampliarse para convertirse en un desarrollo seguro.
Realizar evaluación de seguridad.	Durante todo el desarrollo, especialmente en la fase de análisis contemplamos la evaluación de la seguridad lo que nos ha
Implementar una solución de geo-posicionamiento.	Pudimos explorar lo que es realizar una aplicación de geo-posicionamiento, utilizando las capacidades de los móviles actuales para algo más que para conversar y mandar mensajes. Las posibilidades que nos abren son casi ilimitadas.
Estudiar nuevas herramientas y metodologías de desarrollo y su aplicabilidad al mundo de aplicaciones Internet (métrica v3).	Por último, aplicar una metodología de desarrollo usada en entornos académicos y gubernamentales adaptables a multitud de proyectos es una experiencia enriquecedora que permite valorar su seguimiento como pieza clave para alcanzar los objetivos.

Podemos concluir que, ciertamente, se han alcanzado los objetivos propuestos, a satisfacción del equipo que ha participado en su elaboración. Ciertamente no ha sido al ciento por ciento, pero ha dejado abierto el camino a continuar la aplicación con desafíos atractivos e interesantes, aunque no se pudiera sacar provecho económico de la misma.

5.2. Líneas de trabajo futuro

Debido a las limitaciones de recursos para la realización del proyecto, debimos dejar de lado muchas interesantes opciones que, en un futuro a corto o medio plazo, podrían incorporarse y dotar de mayor interés a la aplicación.

Pero centrándonos exclusivamente en el objetivo, crear una aplicación cooperativa para localizar precios bajos, esta tomar varias direcciones de desarrollo futuro. Una podría ser incorporarse, como módulo o por adquisición, a una aplicación existente de comparación de precios y otra crecer (rendirse no es una opción, dicen). Descartaremos la primera ya que, primero es remota la posibilidad y segundo limitaría las posibilidades de aprendizaje y emprendimiento que podríamos tomar. Así que esperamos y deseamos que la aplicación crezca. Y aquí es donde se abren multitud de posibilidades, de las que destacaremos dos:

La primera es combinar y añadir nuevas capacidades a la aplicación actual para que deje de ser una herramienta casual, para búsquedas puntuales y pase a ser una herramienta que permita la compra diaria, usada por una comunidad de usuarios que compartan información, apoyándose en la confianza, guiada con un sistema de recompensas o valoraciones de usuarios tal vez (como puede ser el karma [70]), se conseguiría una buena reputación siempre y cuando hubiese un número suficiente de usuarios. Esto, junto con una estrategia de *gamificación* [71] a propósito del sistema de valoración de usuarios, podría incrementar la participación de los usuarios y por lo tanto dotar de la validez de lo inmediato a los datos recogidos.

Para que fuera útil en este propósito, la aplicación debería incorporar una gestión de listas de compra, con un catálogo de artículos inicialmente proporcionado por el sistema pero ampliable por los usuarios. Un registro, incluyendo información de localización, de comercios y puntos de venta. Cruzando toda la información anterior, el sistema podría calcular distintas estrategias de compra que pudieran ser útiles a los usuarios, como dónde poder ir a comprarlo todo, a que sitios ir para conseguir la compra más barata, las compras con el menor desplazamiento, etc.

Finalmente, como estrategia de mantenimiento económico, podría incluirse información patrocinada (debidamente identificada) como pueden ser ofertas de negocios y empresas u optar por una estrategia de *crowdfunding* o mecenazgo.

La segunda opción es generalizar el mecanismo, ya sea en la aplicación actual o en una aplicación derivada (*spin off*), que permita cierta personalización de la base de datos por parte de los usuarios. Los usuarios podrían compartir (o no) bitácoras o registros de lugares y lo que se encuentra en ellos, a modo de guía de viajes, registro de lugares interesantes etc. También podría usarse como herramienta de localización e identificación de objetos en la vía pública, desperfectos, etc. [4] para organismos públicos (de manera abierta, pública, o cerrada a los trabajadores).

Incluso, con un sistema de alertas de proximidad, podría aparecer al usuario y notificar de la presencia de alguno de estos registros.

6. Bibliografía

[1] Sleep promotes branch-specific formation of dendritic spines after learning. (2014, Junio) Sciencemag. [Online]. http://science.sciencemag.org/content/344/6188/1173

[2] Consumo Móvil en España 2014 Revolución y evolución. (2014) Deloitte. [Online]. http://www2.deloitte.com/content/dam/Deloitte/es/Documents/tecnologia-media-telecomunicaciones/Deloitte_ES_TMT_Consumo-movil-espana-2014-def.pdf

[3] Informe Mobile en España y en el Mundo 2015. (2015, Julio) Ditrendia. [Online]. http://www.ditrendia.es/wp-content/uploads/2015/07/Ditrendia-Informe-Mobile-en-Espa%C3%B1a-y-en-el-Mundo-2015.pdf

[4] La evolución de los wearables: Pasado - Presente y futuro. (2015) MediaTrends. [Online]. http://www.mediatrends.es/a/34117/evolucion-de-los-wearables-pasado-presente-y-futuro/

[5] iOS App Store brings in 75% more revenue than Play Store despite difference in downloads. (2016, Enero) http://9to5mac.com/. [Online]. http://9to5mac.com/2016/01/20/app-store-ios-downloads-vs-android-revenue/

[6] U.S. Smartphone Use in 2015. (2015, Abril) Pew Research Center Internet, Science & Tech. [Online]. http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/

[7] Sobre la frase de Newton “A HOMBROS DE GIGANTES” y el mal genio del genio. (2013, Marzo) InstintoLogico.com. [Online]. http://instintologico.com/sobre-la-frase-de-newton-a-hombros-de-gigantes-y-el-mal-genio-del-genio/

[8] Michael Leinster (ShoppyDoo): "El tráfico que aporta un comparador de precios es un tráfico cualificado". (2014, Marzo) ecommaster.es. [Online]. http://ecommaster.es/comparador-precios-shoppydoo

[9] Las ventajas de utilizar un comparador de precios al hacer tus compras. (2015, Octubre) Estrella Digital. [Online]. http://www.estrelladigital.es/articulo/empresas/ventajas-utilizar-comparador-precios-hacer-tus-compras/20150930202315255453.html

[10] 10 Consejos para elegir un proveedor cloud. (2013, Marzo) BT España. [Online].

https://www.bt.es/noticias/10-consejos-para-elegir-un-proveedor-cloud

[11] Los comparadores en Internet - un negocio en alza en España. (2014, Enero) 20 Minutos. [Online]. http://www.20minutos.es/noticia/2036788/0/comparadores/ofertas/internet/

[12] Qué es responsive. Lo básico. (2013, Febrero) [Online]. http://lostiumproject.com/2013/02/que-es-responsive-lo-basico/

[13] Servidor de aplicaciones. (2015, Septiembre) Wikipedia. [Online]. https://es.wikipedia.org/wiki/Servidor_de_aplicaciones

[14] Cloud Computing Providers. (2016, Junio) Wikipedia. [Online]. https://en.wikipedia.org/wiki/Category:Cloud_computing_providers

[15] How to choose a cloud hosting service. (2012, Abril) PCWorld. [Online]. http://www.pcworld.com/article/253486/how_to_choose_a_cloud_hosting_service.html

[16] Cloud Computing Trends: 2015 State of the Cloud Survey. (2015, Febrero) Right Scale. [Online]. http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey

[17] Cloud Computing Issues. (2016, Julio) Wikipedia. [Online]. https://en.wikipedia.org/wiki/Cloud_computing_issues

[18] Las mejores distribuciones de linux para el 2016. (2016, Enero) VicktorHck In the Free World. [Online]. https://victorhckinthefreeworld.wordpress.com/2016/01/21/las-mejores-distribuciones-de-linux-para-el-2016/

[19] Usage share of operating systems. (2016, Julio) Wikipedia. [Online]. https://en.wikipedia.org/wiki/Usage_share_of_operating_systems

[20] OS Platform Statistics. (2016, Junio) w3schools.com. [Online]. http://www.w3schools.com/browsers/browsers_os.asp

[21] 7 Razones por las que Linux es más usado que Windows en servidores. (2014, Mayo) Hypertextual. [Online]. https://hipertextual.com/archivo/2014/05/linux-servidores/

[22] Web Server Survey. (2016, Abril) Netcraft. [Online].

http://news.netcraft.com/archives/2016/04/21/april-2016-web-server-survey.html

[23] DB-Engines Ranking. (2016, Agosto) DB-Engines. [Online]. http://db-engines.com/en/ranking

[24] La base de datos 'MySQL' utilizada por Facebook se encuentra al límite. Nerdilandia. [Online]. http://www.nerdilandia.com/la-base-de-datos-mysql-utilizada-por-facebook-se-encuentra-al-limite/

[25] MySQL Customers. (2016) MySQL. [Online]. https://www.mysql.com/customers/

[26] Utilidad y funcionamiento de las bases de datos NoSQL. (2015, Noviembre) Universidad de la Rioja. [Online]. https://dialnet.unirioja.es/descarga/articulo/5029469.pdf

[27] Top Five NoSQL Databases and When to Use Them. ITBusinessEdge. [Online]. http://www.itbusinessedge.com/slideshows/top-five-nosql-databases-and-when-to-use-them.html

[28] Cassandra: la base de datos de facebook - twitter y digg. (2010, Julio) [Online]. http://danielmartinez.info/2010/07/cassandra-la-base-de-datos-de-facebook-twitter-y-digg/

[29] Top 10 Reasons to Use HTML5 Right Now. (2011, Noviembre) http://tympanus.net/. [Online]. http://tympanus.net/codrops/2011/11/24/top-10-reasons-to-use-html5-right-now/

[30] HTML5. (2016, Julio) Wikipedia. [Online]. https://es.wikipedia.org/wiki/HTML5

[31] Typescript - Javascript that scalates. (2016) Typescrypt. [Online]. https://www.typescriptlang.org/

[32] Android. (2016) Android. [Online]. https://www.android.com/intl/es_es/

[33] IOS9 - ¿Qué es IOS? (2016) Apple.com. [Online]. http://www.apple.com/es/ios/what-is/

[34] Hacia 2016: 20 estadísticas de marketing en 2015. (2016, Enero) Antevenio. [Online]. http://www.antevenio.com/blog/2016/01/20-estadisticas-de-marketing-en-2015/

[35] 2015 Q2 Smartphone OS Market Share. (2015) IDC. [Online].

http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[36] Google Play Store vs the Apple App Store: by the numbers (2015). (2015, Abril) Android Authority. [Online]. http://www.androidauthority.com/google-play-store-vs-the-apple-app-store-601836/

[37] Dashboards. (2016) developer.android.com. [Online]. https://developer.android.com/about/dashboards/index.html?hl=es

[38] App Store. (2016) apple.com. [Online]. https://developer.apple.com/support/app-store/

[39] Tecnología Push. (2016) Wikipedia. [Online]. https://es.wikipedia.org/wiki/Tecnolog%C3%ADa_Push

[40] Notificaciones Push. (2013, Noviembre) Wikipedia. [Online]. https://es.wikipedia.org/wiki/Notificaciones_push

[41] Definición de Geolocalización. (2016) definicionabc. [Online]. http://www.definicionabc.com/geografia/geolocalizacion.php

[42] Aprendiendo Android V: Inicialización a la API del GPS. (2010, Agosto) El Android Libre. [Online]. http://www.elandroidelibre.com/2010/08/aprendiendo-android-v-inicializacion-a-la-api-del-gps.html

[43] Taking Pictures and Movies. (2016) developer.apple.com. [Online]. https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/CameraAndPhotoLib_TopicsForIOS/Articles/TakingPicturesAndMovies.html

[44] TextToSpeech. (2016) developer.android.com. [Online]. https://developer.android.com/reference/android/speech/tts/TextToSpeech.html

[45] AVSpeechSynthesizer. (2016) developer.apple.com. [Online]. https://developer.apple.com/library/ios/documentation/AVFoundation/Reference/AVSpeechSynthesizer_Ref/

[46] Five popular databases for mobile. (2014, Septiembre) developereconomics. [Online]. https://www.developereconomics.com/five-popular-databases-for-mobile/

[47] Write once - compile everywhere. (2016, Junio) Wikipedia. [Online].

https://en.wikipedia.org/wiki/Write_once,_compile_anywhere

[48] Write once run anywhere. (2016, Junio) Wikipedia. [Online]. https://en.wikipedia.org/wiki/Write_once,_run_anywhere

[49] Tipos de aplicaciones móviles ventajas e inconvenientes. (2016, Febrero) LanceTalent. [Online]. https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/

[50] The OWASP Foundation. (2016) OWASP. [Online]. https://www.owasp.org/index.php/About_OWASP#The_OWASP_Foundation

[51] Top Ten Mobile Risks. (2016) OWASP. [Online]. https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#Top_Ten_Mobile_Risks

[52] OWASP Top Ten Project. (2016) OWASP. [Online]. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[53] OWASP Top 10 2013. (2013) OWASP. [Online]. https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Español.pdf

[54] MITRE. (2016) The MITRE Corporation. [Online]. https://www.mitre.org/

[55] MITRE. (2016) Common Vulnerabilities and Exposures The Standard for Information Security Vulnerability Names. [Online]. https://cve.mitre.org/cve/cve.html

[56] INCIBE. (2016) Página web del Instituto Nacional de Ciberseguridad. [Online]. https://www.incibe.es/

[57] Catálogo de Ciberseguridad. (2016) Instituto Nacional De Ciberseguridad. [Online]. https://www.incibe.es/sites/default/files/contenidos/guias/doc/catalogo_ciberseguridad.pdf

[58] Understanding Model View Controller. (2008, Mayo) Coding horror. [Online]. https://blog.codinghorror.com/understanding-model-view-controller/

[59] Introducción a la seguridad informática. (2016) recursos EducaLAB. [Online]. http://recursostic.educacion.es/observatorio/web/gl/software/software-general/1040-

introduccion-a-la-seguridad-informatica

[60] Las 75 Herramientas de Seguridad Más Usadas. (2016) http://insecure.org/. [Online]. http://insecure.org/tools/tools-es.html

[61] Three Pillars of Software Security. (2016) Software Security Building Security In. [Online]. http://www.swsec.com/resources/pillars/

[62] Seven Touchpoints for Software Security. (2016) Software Security Building Security In. [Online]. http://www.swsec.com/resources/touchpoints/

[63] Usage statistics and market share of AngularJS for websites. (2016, Agosto) W3Techs. [Online]. https://w3techs.com/technologies/details/js-angularjs/all/all

[64] Algoritmo divide y vencerás. (2016, Abril) Wikipedia. [Online]. https://es.wikipedia.org/wiki/Algoritmo_divide_y_vencer%C3%A1s

[65] Introducción a Métrica v3. Sede de Administracción Electrónica. [Online]. http://administracionelectronica.gob.es/pae_Home/dms/pae_Home/documentos/Documentacion/Metodologias-y-guias/Metricav3/METRICA_V3_Introduccion.pdf

[66] Principios heurísticos. (2015, Noviembre) colimbo.net. [Online]. http://www.colimbo.net/documentos/documentacion/fipo/Principios_heurisiticos_20151120.pdf

[67] Microsoft. Xamarin. [Online]. https://www.xamarin.com/

[68] Tomcat 7.0 Setup. Apache.org. [Online]. https://tomcat.apache.org/tomcat-7.0-doc/setup.html

[69] Installing mySQL. (2016) mySQL. [Online]. http://dev.mysql.com/doc/refman/5.7/en/installing.html

[70] Reddit FAQ. (2016) Reddit.com. [Online]. www.reddit.com

[71] Gamificacion: 7 claves para entender que es y como funciona. (2014, Junio) [Online]. http://justificaturespuesta.com/gamificacion-7-claves-para-entender-que-es-y-como-funciona/

7. Anexos

7.1. Artículo

Comparador de precios colaborativo

Diego Millán Jaureguizar

Escuela de Ingeniería, Universidad Internacional de la Rioja. Logroño, Spain

Abstract - We are in a world where Internet is the main communication and social relationship link. We have to stay up to date to adapt ourselves to this reality. In order to accomplish that, and after the completion of the current Master, we proceed to take our very first steps in that world with the developing of a project that introduce us in the current applications state, mobile client and cloud server, with social characteristics. So as to we made an in-situ search application of geo-positioned items that will allow us to find the cheapest ones with the collaboration of other people.

We counted on a stripped-down version of Métrica v3, with security practices and usability considerations for the user interface design included, to prepare it. This development has allowed us to know very first hand methods and technologies required and has given us the possibility to confront new and defiant challenges.

Keywords: Internet, Mobile, Cloud, Geolocation, Social

I. Introducción

Con la finalidad de demostrar la validez de los conocimientos recibidos durante el año de trabajo en el master, surgió la presente aplicación que trata de solventar el pesado problema del viajero a la hora de comprar recuerdos: ¿Podemos asegurarnos de comprar aquellos recuerdos que os podemos permitir al mejor precio? Y ¿Dónde están, y cómo puedo hacer el menor esfuerzo para adquirirlos?

Para ello, planteamos una solución mediante el aprovechamiento del concepto de “internet de las cosas” tan en boga hoy día y el uso de teléfonos “inteligentes” [3] o *smartphone*, interconectados.

II. Contexto y estado del arte

Es fundamental conocer el tipo de aplicaciones, métodos y tecnologías existentes para no caer en el error de reinventar la rueda y explorar otros puntos de vista. Existen a día de hoy multitud de aplicaciones como la que buscamos; Las categorizamos como aplicaciones de listas compartidas entre usuarios y comparadores de precios en sí. Este último tipo, a su vez se divide en aplicaciones de compra online, comparadores de precios y mixtas.
Junto con las aplicaciones existentes, las tecnologías que lo hacen posible requieren un repaso: La nube (*hosting*), el sistema operativo (donde dominan Windows y Linux), servidor de aplicaciones (destacan IIS, Apache, Tomcat), motor de base de datos para la persistencia (el top 3: Oracle, MySQL y MS SQL Server).

En el lado cliente, el navegador (mayoritariamente usando entornos multi-navegador como jQuery, etc.), el dispositivo móvil y su S.O. (Android e IOS principalmente) y tecnologías como *Push* (que permite al servidor enviar mensajes al cliente), Geolocalización (mediante el uso de GPS), fotografía digital, reconocimiento de voz y dictado y finalmente, almacenamiento local. Todo esto, generalmente, soportado a nivel de API de S.O.

Aparte hay que valorar la herramienta de desarrollo: De generación de código nativo o web. En el primer grupo podemos encontrar a XCODE (IOS), Android Studio (Android) o multiplataforma como Xamarin, Delphi y otros. En el lado web, al abanico de herramientas se abre mucho por la gran popularidad de este: Aptana Studio, Kompozer, BlueGriffon, Atom, Apache, Visual Studio, Netbeans, Brackets y un largo etcétera.

No hay que olvidar los riesgos inherentes a este tipo de desarrollos. Para ello no tenemos que perder de vista el catálogo que nos proporciona OWASP [51], así como asegurar el uso de buenas prácticas reconocidas durante el desarrollo, como pueden ser apoyarse en los tres pilares fundamentales del desarrollo seguro de Gari McGraw y sus siete "*touchpoints*" [62].

III. Objetivos concretos y metodología de trabajo

Nuestro objetivo es desarrollar una aplicación móvil que permita recoger información sobre productos no catalogados previamente, disponibles en múltiples localizaciones y precios, para posteriormente obtener los lugares en los que se encuentran a mejor precio, de manera colaborativa. De aquí derivamos los siguientes objetivos específicos:

Implementar servicios en la nube, Desarrollar un cliente móvil (Smartphone), desarrollar un cliente web para las operaciones de mantenimiento, realizar un desarrollo contemplando la seguridad en todos sus procesos (S-SDLC), realizar evaluación de seguridad, implementar una solución de geo-posicionamiento y estudiar nuevas herramientas y metodologías de

desarrollo y su aplicabilidad al mundo de aplicaciones Internet (Métrica v3 [65]).

Metodología de Trabajo - Para realizar el trabajo seguimos los siguientes pasos, siguiendo de manera ligera o resumida, los propuestos por Métrica v3 para el desarrollo orientado a procesos. Para ello dividimos el trabajo en tres procesos, de los que dimos mayor importancia (por ser donde se realiza la mayor parte del trabajo práctico) a los dos primeros. Los procesos fueron los de Planificación (Definición del catálogo de requisitos y definición de arquitectura), Desarrollo (Viabilidad, Riesgos, Análisis y Diseño del sistema de información), Construcción (Codificación y pruebas) y Mantenimiento (plan de implantación y aceptación).

IV. Desarrollo específico de la contribución

Para realizar el desarrollo nos basamos en una versión libre y más ligera de la metodología Métrica V3, de la que seguimos los cinco procesos básicos (incluyendo los *touchpoints* para convertirlo en un S-SDLC): Estudio de viabilidad del sistema (EVS), Análisis del sistema de información (ASI), Diseño del sistema de información (DSI), Construcción del sistema de información (CSI) e Implantación y aceptación del sistema (IAS).

Siguiendo los pasos de la metodología, y tras extraer el catálogo de requisitos, pudimos modelar a alto nivel el sistema de información de la siguiente manera:

Determinamos que el sistema se implementaría usando una arquitectura n-capas, habitual en aplicaciones de Internet. Una vez hecho esto, estudiamos la viabilidad y se realizó un análisis de riesgos, de los que destacamos los conocimientos obtenidos, la falta de tiempo y recursos, y especialmente los de seguridad. Para cada uno decidimos las estrategias que los mitigaran o anularan.

Pudimos continuar con el modelado de la aplicación: casos de uso, requisitos funcionales y no funcionales, y modelado de alto nivel. Llegado a este punto, debimos descartar realizar un cliente web para la aplicación, al no contar con tiempo ni recursos necesarios. Al haberlo valorado durante el análisis de riesgos, pudimos alterar el plan. Con el diseño ya listo, pudimos realizar un diseño físico, centrándonos en el servidor y cliente móvil.

Para la construcción decidimos usar Xamarin [67] para la parte cliente, con lo que obtuvimos dos clientes (IOS y Android) con el mismo código y pocas o ninguna modificación particular. En la parte servidora implementamos *servlets* para un contendor de aplicaciones Tomcat (Java), apoyándonos en un S.G.B.D. mySQL.

Una vez concluido, debemos destacar que la implantación tiene dos partes, la parte cliente y la servidora. La primera, en producción, se hará a través de las tiendas de las plataformas móviles. Aquí no tenemos control. Por el contrario, en el servidor sí. Para ello, seguimos las recomendaciones y guías de los distintos fabricantes de las herramientas usadas, empezando por la base de datos y terminando por el S.O.

Para la aceptación montamos una lista de comprobación (*checklist*) siguiendo los requisitos que definimos en fases anteriores y usamos un equipo de voluntarios para realizar las comprobaciones. A la vez, realizamos pruebas de penetración para comprobar la robustez de a seguridad implementada. Tras los ajustes pertinentes, dimos por concluido el desarrollo.

V. Conclusiones y trabajo futuro

Conclusiones – Debemos preguntarnos si hemos logrado los objetivos que nos planteamos inicialmente:

Hemos guiamos el desarrollo, a la vez que nos enfrentamos por primera vez, usando una versión ligera de Métrica v3, lo que nos sirvió como aprendizaje adicional.

Realizamos una implementación de servicios en la nube (usando *servlets* Tomcat), con lo que nos ha servido como toma de contacto tanto con una infraestructura en la nube como practicado parte de lo aprendido en el Master de tecnologías en el lado servidor (S.O, Servidor de aplicaciones, codificación).

Se implementó en el cliente móvil una aplicación conectada con los servicios anteriores, usando una solución multiplataforma, que nos permite recoger los datos y realizar una explotación inicial de los mismos para lograr nuestro objetivo de encontrar los artículos más baratos de manera colaborativa. Todo ello apoyándonos en geo-posicionamiento.

Durante el desarrollo se tuvo presente la seguridad en todas sus tareas y así prevenir problemas futuros. Sin olvidar, por supuesto, evaluación de seguridad.

Entonces podemos concluir que se han alcanzado los objetivos propuestos, y por lo tanto contestado afirmativamente a las preguntas que pusieron en marcha el proyecto. No ha sido al ciento por ciento, pero ha dejado abierto el camino a continuar la aplicación con desafíos muy interesantes.

Líneas de trabajo futuro – Hemos determinado que se podrían tomar varias direcciones de desarrollo a continuación:

Una implicaría que la aplicación creciera hasta incorporar características de comparadores de precios actuales, así como características de compra en proximidad. Para que fuera útil en este propósito, la aplicación debería incorporar una gestión de listas de compra, con un catálogo de artículos inicialmente proporcionado por el sistema, y ampliable por los usuarios, junto con un registro de comercios y puntos de venta. El sistema podría calcular distintas estrategias de compra que pudieran ser útiles a los usuarios, como dónde poder ir a comprarlo todo, a que sitios ir para conseguir la compra más barata, las compras con el menor desplazamiento, etc.

La segunda opción es generalizar el mecanismo, ya sea en la aplicación actual o en una aplicación derivada (*spin off*), que permita cierta personalización de la base de datos por parte de los usuarios. Los usuarios podrían compartir (o no) bitácoras o registros de lugares y lo que se encuentra en ellos, a modo de guía de viajes, registro de lugares interesantes etc. También podría usarse como herramienta de localización e identificación de objetos en la vía pública, desperfectos, etc. para organismos públicos (de manera abierta, pública, o cerrada a los trabajadores).

VI. Bibliografía

[1] Informe Mobile en España y en el Mundo 2015. (2015, Julio) Ditrendia. [Online]. http://www.ditrendia.es/wp-content/uploads/2015/07/Ditrendia-Informe-Mobile-en-Espa%C3%B1a-y-en-el-Mundo-2015.pdf

[2] Top Ten Mobile Risks. (2016) OWASP. [Online]. https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#Top_Ten_Mobile_Risks

[3] Seven Touchpoints for Software Security. (2016) Software Security Building Security In. [Online]. http://www.swsec.com/resources/touchpoints/

[4] Introducción a Métrica v3. Sede de Administracción Electrónica. [Online]. http://administracionelectronica.gob.es/pae_Home/dms/pae_Home/documentos/Documentacion/Metodologias-y-guias/Metricav3/METRICA_V3_Introduccion.pdf

[5] Microsoft. Xamarin. [Online]. https://www.xamarin.com/