Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

GESTIÓN DE PRESUPUESTOS A TRAVÉS DE UNA APLICACIÓN MÓVIL

Alumno: Imanol Bracero Armenteros

Tutor: Arturo Montejo Ráez
Dpto: Departamento de Informática

Junio, 2015

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Arturo Montejo Ráez , tutor del Proyecto Fin de Carrera titulado: GESTIÓN DE PRESUPUESTOS A TRAVÉS DE UNA APP, que presenta Imanol Bracero Armenteros, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2015

El alumno: Los tutores:

Imanol Bracero Armenteros Arturo Montejo Ráez

Contenido

Agradecimientos

Primero me gustaría dar las gracias a mi familia, ya que sin ellos esta etapa de mi vida no habría sido posible. Me han apoyado en los momentos más duros de la carrera, siempre animándome a conseguir mis objetivos.

En segundo lugar agradecer a mis amigos y compañeros de clase, que me han ayudado en todo momento cuando necesitaba un punto de apoyo.

Destacar mi gratitud a la empresa destinataria de este proyecto *Proyectos Comerciales Grupo Rozema S.L.* que ha confiado en mí desde el principio, facilitándome el trabajo tanto como ha podido.

Por último, agradecer a Arturo Montejo Ráez su labor como tutor, ya que me ha aconsejado y resuelto las dudas que me han ido surgiendo durante el desarrollo del trabajo que nos ocupa.

A todos ellos, muchas gracias.

0. PRESENTACIÓN

Mi nombre es Imanol Bracero Armenteros, soy estudiante del Grado en Ingeniería Informática impartido por la Universidad de Jaén. El presente documento es el Trabajo de Fin de Grado de mi titulación (en adelante TFG). Para dicho trabajo, ha sido fundamental la ayuda y asesoría por parte de mi tutor, Arturo Montejo Ráez. Se trata de un proyecto real, dirigido a la empresa "Proyectos Comerciales Grupo Rozema S.L." que se dedica principalmente a la exportación de materiales de construcción, apoyándose en gran medida de los avances tecnológicos.

1. INTRODUCCIÓN

En la actualidad, no concebimos una sociedad en la que no se viva con un uso intensivo de las tecnologías. De hecho, existe una preocupante necesidad de estar siempre conectados y tener la red a nuestra disposición en cualquier momento. Sin duda, el punto más interesante que comparten hoy día estas tecnologías es Internet.

Internet ha conseguido que exista un enlace entre todo el mundo las 24 horas del día a través de numerosos tipos de dispositivos, pasando por ordenadores de todos los tamaños, smartphones, smartwatches, gafas de realidad aumentada e incluso, objetos cotidianos que ya se conectan a la red. Es lo conocido como el Internet de las cosas: acercar la red a nuestra vida cotidiana (lavadoras, frigoríficos, pulseras y accesorios, etc.).

Por supuesto, esto ha supuesto una revolución en el mundo de la empresa. Han sido muchas compañías las que han sabido explotar esta ferviente necesidad de estar siempre on-line. Y no sólo las grandes empresas que son conocidas por todos han aprovechado esta circunstancia. Las empresas pequeñas ya se actualizan y modernizan para llegar al público a través de estos dispositivos.

Las compañías han conseguido obtener cantidades masivas de información de sus clientes directos, lo que ha permitido una adaptación a las necesidades de los usuarios que ha sido beneficiosa de forma bidireccional. Este trato de la información a veces roza el marco de la legalidad, aunque este tema no es el objetivo de este trabajo.

Así me surgió la pregunta que me hizo decidirme por este proyecto: ¿No puedo aprovechar yo esta ventaja de poder llegar a muchos usuarios en cuestión de segundos?

Por supuesto, son muchas las dificultades que existen a la hora de sacar un producto de este tipo al mercado, pero, siendo tan bajos los costes, ¿por qué no intentarlo?

Mi interés se centra en aprovechar esa extensión del brazo que todos tenemos en cualquier momento: el teléfono móvil. Tal vez estas dos palabras terminen perdiéndose al ser sustituidas por “smartphone” y es que, cada vez usamos menos estos dispositivos para la función con la que surgieron y más para aplicaciones diferentes, aunque la esencia no se pierde, la comunicación.

1.1. Motivación

La idea principal de realizar este trabajo es aprovechar este alud tecnológico. Por supuesto esta revolución ofrece grandes beneficios económicos al que consigue hacerse un hueco en el sector, pero no es la motivación principal.

La meta es desarrollar una herramienta que facilite a la empresa y los clientes de la misma tener la información que se necesita en cualquier momento con el simple gesto de desbloquear su móvil.

Esta herramienta actuará como complemento a una aplicación web llamada *TCPline*, que se encarga de gestionar la información de una compañía y ponerla a disposición del cliente. Es una aplicación que está basada en Prestashop, por lo que puede llevar al error de pensar que es una tienda online más en el mercado. No es esa la idea de la aplicación. Se pretende que tanto el comercial como el cliente directo, que están en cualquier parte del mundo puedan realizar sus presupuestos con una serie de productos que la empresa ofrece y después de una serie de pasos coordinados se pueda servir el pedido.

La aplicación realizada es una herramienta para los comerciales de toda la vida, que viajan a muchos sitios intentando vender sus productos.

1.2. Objetivos

Los objetivos de la aplicación se han podido entrever en el apartado anterior y se dividen en 3 grandes bloques, que iremos desglosando a lo largo del trabajo.

- Realizar una APP que sirva de herramienta para que los comerciales puedan trabajar independientemente del lugar en el que se encuentren y la calidad de conexión que tengan.
- Aportar a la herramienta una interfaz clara, sencilla e intuitiva que unifique y simplifique los procesos que suelen resultar tediosos al comercial, facilitando y optimizando así su trabajo.

- Añadir la funcionalidad necesaria para sincronizar la base de datos del terminal con la de la tienda en sí, alojada en un servidor determinado.

1.3. Marco del Proyecto en el panorama actual

La inserción de este producto en el mercado no puede ser más fácil en las fechas en las que nos encontramos, y es que todo el mundo sale a la calle con su teléfono en el bolsillo. Gracias a esto, se puede llegar al público de todas las edades (público, que normalmente serán los propios comerciales, en nuestro caso).

El sector en el que se enmarca este proyecto, está repleto de comerciales, de todas las edades y por consecuente, algunos más adaptados a la tecnología que otros. La misión de esta aplicación es llegar a todos los comerciales, promoviendo una interfaz intuitiva y sencilla, pero a la vez completa. No es tarea sencilla enfocar este programa a un abanico tan amplio de usuarios. Aun así, el diseño pretende abarcar el mayor rango posible.

Antes de desarrollar la temática del proyecto, destacar un titular que se podía ver hace poco tiempo en un periódico digital.

"*España, líder europeo en el uso de teléfonos inteligentes, con un 81 por ciento*" [1]

El desglose de la noticia hace referencia a que el 81% de los móviles usados en España, son *smartphones*, llegando a la cifra de 26,25 millones de españoles conectados regularmente a Internet a finales de 2014.

Además, la noticia desgrana varios datos de estudios y otros informes procedentes del INE (Instituto Nacional de Estadística), la CNMC (Comisión Nacional de los Mercados y la Competencia), agencias y consultoras. En estos datos, algunos puntos que resultan interesantes son los siguientes:

- La mitad de las personas de entre 55 y 64 años dice acceder diariamente a la red.
- En el segundo cuatrimestre de 2014, la facturación de banda ancha superó, por primera vez a la fija.

A continuación, se incluye un gráfico para que se vea el crecimiento que se está produciendo en los últimos años en España (*Gráfico 1.1*).

[1] Anónimo (2015). España, líder europeo en el uso de teléfonos inteligentes, con un 81 por ciento. *El diario*. Recuperado el 2 de febrero de 2015 de: http://www.eldiario.es/turing/Espana-europeo-telefonos-inteligentes-ciento_0_348215609.html

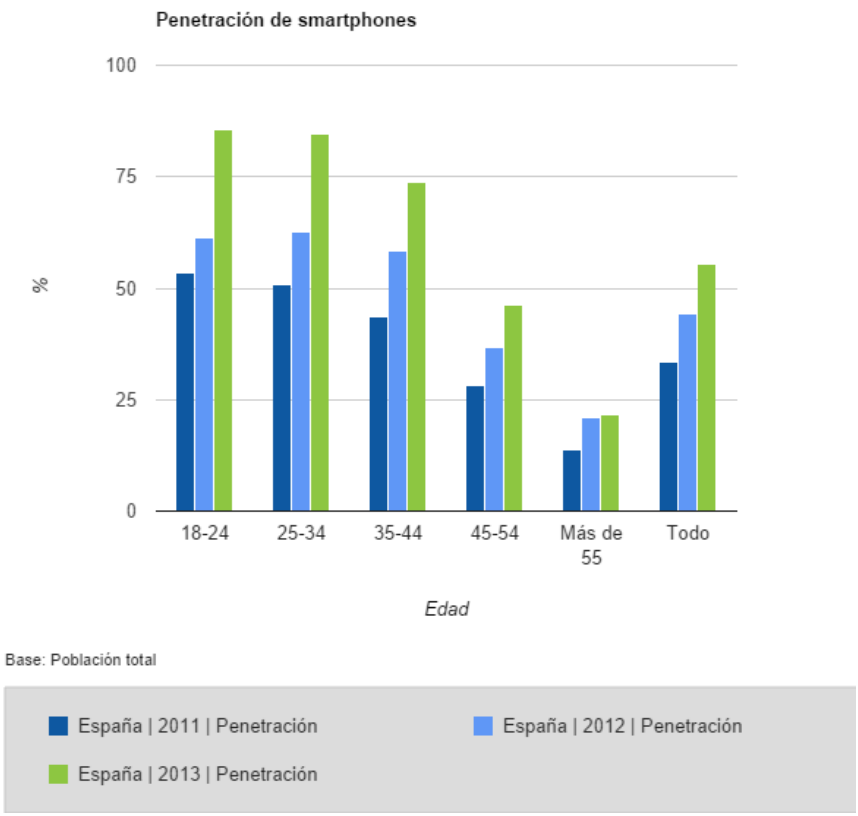


Gráfico 1.1

Se puede ver que la introducción de Smartphones en España ha sido más o menos equilibrada en cuanto a las edades, ya que aumenta considerablemente en todas ellas, lo que puede llegar a suponer beneficios enormes de cara a este proyecto, ya que, como se introduce anteriormente, esta aplicación la usarán comerciales que están en diferentes rangos de edades.

1.4. Metodología y modelo de proceso

En este apartado, se intenta dar una visión global de cómo se presenta el proyecto, las etapas que se han planificado para el mismo y qué modelo de proceso se ha seguido.

En la mayoría de proyectos de características similares al que nos ocupa, se recomienda el uso de un modelo de desarrollo orientado a prototipos, sin embargo, por la facilidad de trabajo y realimentación con el cliente que se tiene en este proyecto se utilizará un *modelo incremental*.

Este modelo combina elementos del MLS (Modelo Lineal Secuencial, basado en el modelo clásico en cascada) con la filosofía interactiva de construcción de prototipos.

En una visión genérica, el proceso se divide en 4 partes: Análisis, Diseño, Código y Prueba. Sin embargo, para la producción del software se usa el principio de trabajo en

cadena o "*pipeline"* (proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior), usado en muchas otras formas de programación. Con esto se mantiene al cliente en contacto con los resultados obtenidos en cada incremento.

En este proceso será el mismo cliente el que incluya o deseche elementos en cada incremento, a fin de que el software se adapte completamente a sus necesidades.

Al final de cada incremento se entrega un producto plenamente operacional.

En este proyecto, se ha utilizado este modelo de desarrollo por la fácil colaboración con la empresa. La realimentación ha sido continua y rápida, facilitando así la creación de estos incrementos y mejorando las funcionalidades que han ido fallando en momentos determinados.

Es una metodología muy práctica para este tipo de proyectos, aunque el inconveniente principal es que conlleva una difícil estimación de costos.

Para el desarrollo de este proyecto con el modelo incremental, ha sido necesario el apoyo en una estrategia de desarrollo ágil, *SCRUM***.**

Es una metodología para gestionar el desarrollo de software. A través de ella se consigue que el cliente se entusiasme y comprometa con el proyecto dado que lo ve crecer interacción a interacción. Además, le permite realinear el software con los objetivos de su empresa, pudiendo introducir cambios funcionales o de prioridad en el inicio de cada iteración.

En esta metodología, cada iteración se conoce como *Sprint* y tiene una duración preestablecida. En cada uno de ellos se va ajustando la funcionalidad y se añaden nuevas prestaciones priorizándose siempre aquellas que aporten mayor valor de negocio.

En este proyecto, los Sprints han sido variables, en función de la complejidad de las funcionalidades que se han ido pidiendo, nunca excediendo las 5 semanas.

Al final de cada Sprint, se presenta el incremento a los encargados de la gerencia, administración y departamento de informática (mencionados en el apartado 3.2.2 de este documento), esperando el feedback preparándose así el siguiente incremento.

A continuación se puede ver el esquema básico del proceso de desarrollo escogido. (*Ilustración 1.1)*

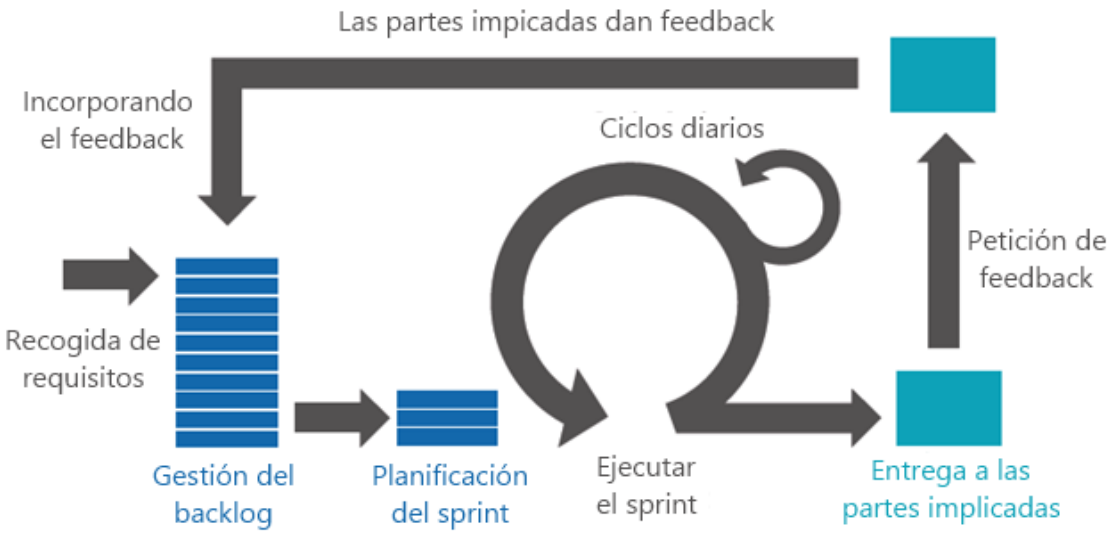


Ilustración 1.1

2. ANÁLISIS DEL SISTEMA ACTUAL

Como se comentó en la presentación, este proyecto es un caso real de desarrollo de software para dispositivos móviles para la empresa *Proyectos Comerciales Grupo Rozema S.L.* En este apartado se realizará una breve descripción de la actividad de la empresa y los recursos y herramientas con los que cuenta en su funcionamiento normal, justificando los puntos donde el desarrollo de esta aplicación móvil puede ser más favorable.

La empresa en cuestión se encarga de la comercialización y distribución de materiales de construcción al por mayor. Forma parte de un sector donde existen grandes fuerzas que ejercen competencia y donde cualquier detalle que les haga distinguirse marca una diferencia importante. Actualmente, es una empresa que apuesta mucho por la tecnología y que se esfuerza por que sus comerciales cuenten con las herramientas más actualizadas.

Esta empresa es una PYME que cuenta con 11 empleados en plantilla, aunque su trabajo habitual se desarrolla en torno a comerciales que trabajan como agentes libres o comisionistas (a los que va destinada esta aplicación).

La forma habitual de trabajo de estos comerciales se basa en el sistema tradicional: los comerciales viajan a distintos puntos nacionales e internacionales intentando realizar las ventas, llevando con ellos los numerosos catálogos y muestras de los productos que van a ofertar. El equipo informático de la empresa desarrolló el sistema *TCPline Web* para evitar este traslado con tanto material, así como la rápida obsolescencia de estos catálogos. Con

esta herramienta, los comerciales pueden preparar presupuestos a los clientes sin miedo a que el producto esté descatalogado, ya que con este sistema siempre está actualizado.

Tras este planteamiento, puede surgir la siguiente pregunta: si ya está hecha la aplicación Web, ¿qué necesidad hay de desarrollar un complemento adicional como puede ser la aplicación móvil?

A continuación se muestra un ejemplo de la versión web de una tienda respaldada por TCPline. (*Ilustración 2.1)*

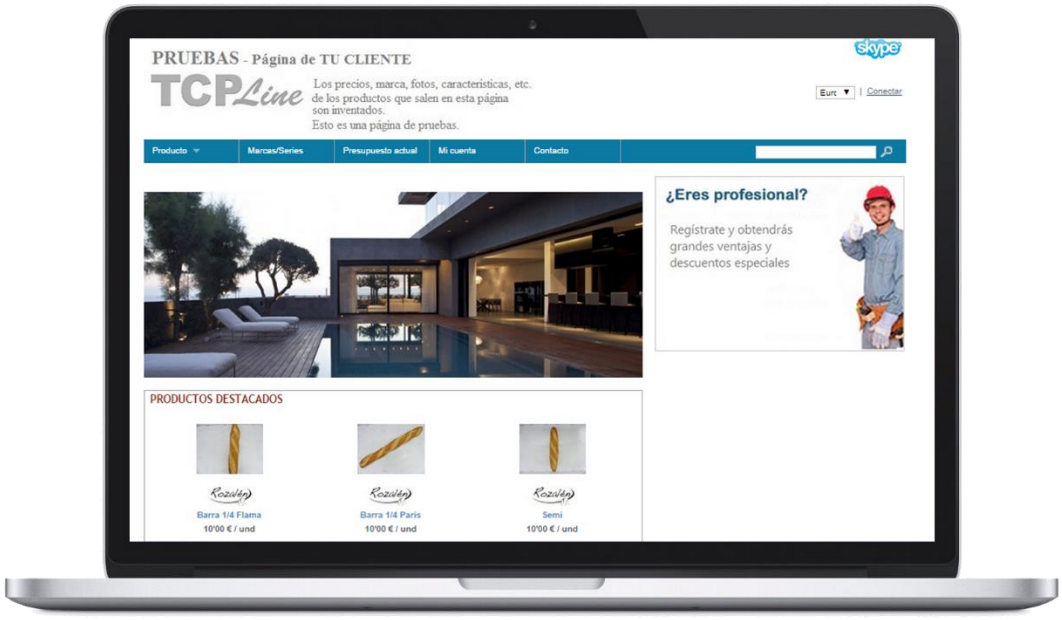


Ilustración 2.1

Se puede contestar a esta pregunta usando dos argumentos principales:

- La versión Web del programa no cuenta con una plantilla responsiva adecuada para un uso intensivo desde un dispositivo móvil. Esto supone una gran incomodidad para el usuario a la hora de trabajar con ella. El sistema con el que se trabaja está basado en Prestashop 1.3 (versión que no incluye *Responsive Design*. Este se incorporará a partir de la versión 1.5) y como se han realizado innumerables aportes a esta base, no se ha podido actualizar.

En la *Ilustración 2.2* se puede observar la apariencia de una tienda en un dispositivo móvil.

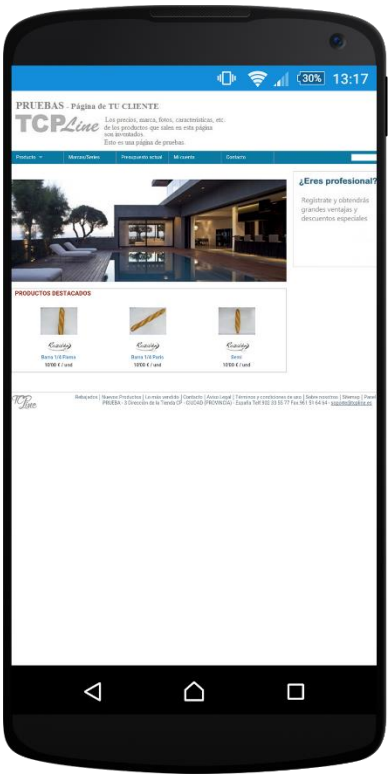


Ilustración 2.2

- Si se pudiera ignorar este primer punto, se presenta el problema de que no se puede trabajar sin conexión a Internet. Teniendo en cuenta la movilidad geográfica de los comerciales y que lamentablemente aún no existe una conexión estable en todo el mundo (apenas se consigue en territorio nacional), lleva a creer en la necesidad de que la forma de trabajar, es llevando toda la información siempre encima, siempre en el dispositivo, esto es: trabajar Offline.

A lo largo del trabajo se detallan estas grandes ventajas y la forma para conseguir "materializarlas".

Los dispositivos con los que cuentan algunos comerciales de la empresa son los siguientes (*Ilustración 2.3*):

- 4 x Samsung Galaxy S4 con Android 4.4.2 Kitkat
- 1 x Dooge Valencia con Android 4.4 Kitkat
- 1 x Samsung Galaxy S3 con Android 4.3 Jelly Bean
- 1 x Samsung Galaxy Core con Android 4.4. Kitkat
- 1 x Sony Xperia Z3 con Android 5.0.1 Lollipop

Todos ellos compatibles con la aplicación desarrollada, que establece el minSDK en la versión 4.0 de Android (Ice Cream Sandwich).

Ilustración 2.3

3. ANÁLISIS DE REQUISITOS

3.1. Entendiendo los requerimientos del usuario

Habiendo realizado esta introducción a las tecnologías, definición de objetivos y motivación del proyecto, a continuación se desglosará el conjunto de técnicas utilizadas en el proceso de desarrollo para lograr la consecución de los objetivos mencionados anteriormente.

En esta fase del proyecto, surge la necesidad de entender una serie de requisitos que serán la base para el diseño y la implementación de la aplicación en cuestión. Es la fase más importante de todo el proyecto, ya que un error cometido en esta etapa, se irá arrastrando (y probablemente incrementando) a medida que avance el proyecto.

Básicamente, se pretende contestar a la siguiente pregunta: ¿qué es lo que debe hacer el sistema?

La respuesta será un conjunto de propiedades y funcionalidades que debe cumplir el desarrollo, así como enmarcarse dentro de las restricciones definidas con la máxima precisión.

Aunque ya se explicó en el capítulo anterior, cabe recalcar la utilidad de la metodología escogida, tanto cara al cliente como para el desarrollador, por permitir un cambio continuo en el producto final y evitando malentendidos entre cliente y desarrollador.

Los requisitos de los que hablamos pueden clasificarse en tres grandes bloques:

- Requisitos funcionales. ¿Qué tiene que hacer el sistema? ¿qué objetivos debe cumplir?
- Requisitos no funcionales. ¿En qué medida se cumplen estos objetivos? ¿qué restricciones debe tener el sistema? (tiempo de respuesta, volumen de datos, seguridad, robustez…)
- Requisitos de facilidad de uso. ¿A qué usuarios va dirigido el producto? ¿Cómo de efectiva será la relación sistema-usuario y qué situaciones pueden surgir?

3.1.1. Requisitos funcionales

1. La aplicación debe ser notablemente usable e intuitiva. Los usuarios deben aprender, entender y manejar la aplicación de forma muy rápida y sencilla.
2. Existirá un control de acceso a la aplicación, que devolverá el conjunto de tiendas a las que el usuario tiene permitido el acceso.
3. Se podrá acceder a las diferentes tiendas a las que un determinado comercial tenga acceso.
4. Se podrá gestionar la información de todos los clientes asignados a un comercial así como añadir nuevos clientes al sistema dentro de una tienda.
5. Se podrá visualizar la información de todos los productos que estén dados de alta en el sistema dentro de una tienda ordenada en categorías.
6. Se proporcionará una forma rápida de acceso a los productos, a través de un buscador.
7. Se mostrarán las diferentes marcas creadas en cada tienda, así como los productos que pertenecen a ellas.
8. Se podrá realizar un presupuesto a partir de los productos introducidos, con la posibilidad de añadir descuentos y diferentes cantidades.
9. Podrá accederse a un histórico de los presupuestos realizados en una tienda determinada y realizar cambios sobre ellos.
10. Podrá cambiarse la moneda en la que se visualizarán los precios.
11. Podrá sincronizarse con el servidor cada vez que el comercial crea oportuno y mantener así la información actualizada en ambos lados de la conexión.

3.1.2. Requisitos no funcionales

Dado que estamos hablando de una aplicación para un dispositivo móvil, se ha de tener en cuenta las restricciones que esto supone. Esta situación hará visibles los siguientes requisitos no funcionales.

1. Al tratarse de un sistema que puede trabajar de forma offline, la descarga de datos inicial conllevará un coste de memoria y tiempo que ha de ser el óptimo.
2. La aplicación será utilizada en pantallas que van de 4” a 10” aproximadamente. Deberá estar optimizada para su usabilidad en estos tamaños.
3. El tiempo de respuesta para sincronizar las tiendas variará con el volumen de datos que éstas tengan desactualizados. Se mostrará al usuario qué datos se están descargando en cada momento.
4. Debe ser una aplicación robusta, capaz de recuperarse rápidamente en caso de fallo.
5. En cuanto al volumen de datos, las cargas estarán optimizadas, usando métodos de compresión para las imágenes que aportan la mayoría del peso. Se calcula que un determinado comercial podrá trabajar con varias tiendas (casi nunca más de 10), cada una contando con una media de 3000 productos, aprovechando eso sí, las imágenes de los productos comunes a varias tiendas, ya que se guardan en un mismo directorio al descargarlas.
6. Los datos que se manejan en la aplicación deberán ser privados, pudiendo usarse sólo desde la aplicación. Una vez que esta se desinstale, los datos serán borrados.
7. Se deberá comprobar que el comercial que use la aplicación conserva los permisos necesarios para seguir usándola en cada una de las actualizaciones.

3.1.3. Requisitos de facilidad de uso

1. La aplicación seguirá los estándares establecidos por la plataforma Android[2], facilitando así el aprendizaje de la misma al usuario.
2. Los iconos utilizados seguirán el estándar de la plataforma mencionada, haciendo más intuitiva la interfaz de la aplicación.
3. El menú principal de la tienda será accesible desde cualquier punto de la aplicación, a excepción de tareas de gran importancia en las que es necesaria la máxima atención por parte del usuario.

[2] Para este proyecto, se han seguido los estándares de Android sin tener en cuenta la novedosa parte de Material Design. Las guías seguidas se pueden encontrar en *http://developer.android.com/design/get-started/principles.html*

3.2. Técnicas de búsqueda de requisitos

Existen numerosas herramientas y técnicas para hallar los requisitos fundamentales del usuario. Se han estudiado las existentes y escogido las que más se ajustan al modelo de este proyecto, siendo desarrolladas a lo largo de este apartado.

3.2.1. Lecturas preparatorias

El objetivo de esta técnica es entender bien la organización a la que va dirigido el producto así como los objetivos del negocio, a través del estudio de documentos existentes en la empresa: informes, manuales de normativas, descripciones de trabajos, documentación de los sistemas existentes, etc.

Este estudio ayuda a conocer la organización y los objetivos que tiene. Una parte fundamental es la documentación del sistema existente para identificar requerimientos y funcionalidades que deberá tener el nuevo sistema. El inconveniente de esta técnica es que no siempre se tiene acceso a esta información y si se consigue, puede no estar actualizada.

Para este proyecto en concreto, la empresa *Proyectos Comerciales Grupo Rozema S.L.* (en adelante Rozema) no ha tenido ningún problema en proporcionar toda la información necesaria, así como la interacción directa con el sistema actual, *TCPline Web*.

3.2.2. Entrevistas con dirección, empleados y usuarios finales

Esta técnica permite obtener un profundo conocimiento de los objetivos de la organización, requerimientos de los usuarios y roles de cada persona en la empresa.

Para ello, se concertó una entrevista a mediados del mes de septiembre de 2014. Asistieron:

- Mariano Rozalén (gerente de Rozema).
- Patricia Rozalén (administradora y usuario directo con el sistema).
- José Rodriguez (encargado del departamento de informática y desarrollador principal de TCPline Web).

De esta reunión se obtuvieron a grandes rasgos los requisitos fundamentales que deberían formar la estructura básica de la aplicación.

La existencia de la versión web del programa a desarrollar facilitó mucho el ejercicio de comprensión de requerimientos del producto. Además, en la reunión se comentaron las metodologías utilizadas para desarrollar esta versión.

3.2.3. Observación

Además de la entrevista realizada con personas de diferentes departamentos de la empresa, el gerente permitió la observación de los trabajadores como usuarios directos del sistema.

Esta técnica permite observar qué sucede realmente y no sólo lo que se dice que sucede en las entrevistas. Incluye:

- Observar como los usuarios interactúan con el sistema actual.
- Obtener datos para mejoras en el nuevo sistema.
- Observar relaciones entre tareas.
- Calcular coste en tiempo y dinero de determinadas tareas.
- Observación de posibles errores en el sistema actual.

3.3. Casos de uso

Los casos de uso describen acciones y reacciones de un sistema desde el punto de vista del usuario. Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno. Son descripciones de la funcionalidad del sistema, independientes de la implementación.

El desarrollo de esta aplicación resalta un caso de uso que representa el objetivo principal de la aplicación: *Realizar un presupuesto.*

Caso de uso	**Realizar un presupuesto**
Actor primario	Comercial
Sistema	TCPline APP
Participantes	Comercial, Servidor de datos
Nivel	Objetivo usuario
Condición previa	El servidor está operativo, hay Internet.
Operaciones básicas	

1	Sincronización de datos inicial
2	Seleccionar cliente
3	Escoger producto
4	Aplicar descuentos
5	Añadir al presupuesto actual
6	Cumplimentar datos necesarios
7	Terminar presupuesto
Alternativas	
1.A.	¿Sincronización realizada correctamente?
1.A.1.	Si sí, continuar
1.A.2.	Si no, volver a 1
2.A.	¿Está el cliente requerido en la lista?
2.A.1.	Si sí, ir a 3
2.A.2.	Si no, crear cliente e ir a 3
3.A.	¿Hay que realizar descuentos para el producto?
3.A.1.	Si sí, ir a 4
3.A.2.	Si no, ir a 5.
5.A.	¿Están todos los productos deseados en el presupuesto actual?
5.A.1.	Si sí, cerrar e ir a 6.
5.A.2.	Si no, ir a 3.

Tabla 3.1

Aunque el caso de uso expuesto es el principal, también se pueden destacar: *Cerrar y guardar presupuesto actual y añadir cliente.*

Caso de uso	**Guardar y cerrar presupuesto actual**
Actor primario	Comercial
Sistema	TCPline APP
Participantes	Comercial, Servidor de datos
Nivel	Objetivo usuario
Condición previa	Hay productos añadidos, hay cliente seleccionado

Operaciones básicas	
1	Seleccionar una dirección de entrega
2	Seleccionar una dirección de facturación
3	Cumplimentar nombre del proyecto y formas de pago
4	Añadir gastos de transporte
5	Añadir descuentos
6	Guardar y salir
Alternativas	
1.A.	¿Está la dirección de entrega en el listado?
1.A.1.	Si sí, seleccionar e ir a 2
1.A.2.	Si no, cumplimentar formulario e ir a 2
2.A.	¿Está la dirección de facturación en el listado?
2.A.1.	Si sí, seleccionar e ir a 3
2.A.2.	Si no, cumplimentar formulario e ir a 3
4.A.	¿Hay gastos de transporte?
4.A.1.	Si sí, rellenar el campo y continuar
4.A.2.	Si no, continuar
5.A.	¿Se pueden aplicar descuentos?
5.A.1.	Si sí, rellenar e ir a 6
5.A.2.	Si no, ir a 6

Tabla 3.2

Caso de uso	Añadir cliente
Actor primario	Comercial
Sistema	TCPline APP
Participantes	Comercial, Servidor de datos
Nivel	Objetivo usuario
Condición previa	
Operaciones básicas	

1	Añadir datos personales
2	Añadir datos de entrega
3	Añadir datos de facturación
4	Guardar y salir
Alternativas	
1.A.	¿Datos personales añadidos correctamente?
1.A.1.	Si sí, ir a 2
1.A.2.	Si no, volver a 1
2.A.	¿Se quiere añadir datos de entrega?
2.A.1.	Si sí, añadir datos y volver a 2 (tantas direcciones como se quiera)
2.A.2.	Si no, ir a 4
3.A.	¿Se quiere añadir datos de facturación?
3.A.1.	Si sí, añadir datos y volver a 3 (tantas direcciones como se quiera)
3.A.2.	Si no, ir a 4

Tabla 3.3

4. PLANIFICACIÓN

La planificación es la etapa del proyecto de la cual se pretende obtener una serie de estimaciones que serán fundamentales a la hora de decidir sobre la viabilidad de un proyecto.

Estas estimaciones giran en torno a tres grandes bloques:

- Estimación de recursos.
  - Humanos
  - Componentes reutilizables
  - Herramientas (software y hardware)
- Estimación de tiempo.
- Estimación de costes y esfuerzos. (basar las estimaciones en proyectos similares ya terminados, técnicas de descomposición en tareas)

4.1. Estimación de recursos

4.1.1. Recursos humanos

Dentro del bloque de planificación de un proyecto, una parte fundamental es fijar con qué recursos humanos contamos a la hora de desarrollarlo.

En el caso de este proyecto, este apartado será breve, ya que desde la fase de análisis hasta la de mantenimiento correrá de la cuenta del autor del proyecto.

Cabe destacar la colaboración del encargado de informática de la empresa en cuestión, que facilitó la API de las tiendas de la que se obtienen los datos de la APP, adaptándola a las necesidades oportunas.

4.1.2. Componentes reutilizables

En cuanto a software, dado que es la primera aplicación Android con un peso importante, que se ha desarrollado por parte del autor, no hay módulos propios que puedan reciclarse. Sin embargo, serán de utilidad algunas librerías externas bastante conocidas

- Picasso[3]. Desarrollada por Square. Facilita el pintado de imágenes como vistas en las diferentes pantallas, accediendo a ellas a través de una conexión HTTP o en las rutas internas del terminal.
- JodaTime[4]. Desarrollada por Joda. Proporciona métodos y procedimientos que ayudan al manejo de las fechas dentro del sistema. Es una librería muy útil para el proyecto, ya que deben tenerse muy en cuenta las fechas de actualización a la hora de realizar las diferentes sincronizaciones de datos.
- Gson[5]. Desarrollada por Google. Proporciona todas las funciones necesarias para manejar la información que fluye en formato JSON. Dado será el lenguaje usado en el protocolo REST para hacer las peticiones a la API, ayudará en gran medida, sobre todo a la hora de generar un JSON a partir de una instancia de un objeto o un array de Instancias.
- Crashlytics. Desarrollada por Fabric IO. Proporciona un sistema de control de errores muy útil para la fase de pruebas. Cualquier dispositivo que instale la aplicación y reciba un error, enviará un informe al desarrollador de la aplicación, que tomará las medidas que considere.

[3] En el proyecto se ha usado la versión 2.1.1. La más reciente (marzo 2015) es la 2.5.2., disponible para descargar en *http://square.github.io/picasso/*

[4] En el proyecto se ha usado la versión 2.3. La más reciente (marzo 2015) es la 2.7., disponible para descargar en *https://github.com/JodaOrg/joda-time/releases*

[5] En el proyecto se ha utilizado la 2.3.1., que continúa siendo la más reciente y se encuentra disponible para descargar en *http://search.maven.org*

4.1.3. Herramientas

En este apartado se describirán las herramientas necesarias para el correcto desarrollo del proyecto. Éstas se pueden dividir en dos bloques:

- Herramientas software**.** Suponen el conjunto de programas utilizados en el desarrollo del proyecto en cuestión.
    - Eclipse**.** Inicialmente fue el entorno de desarrollo escogido al comenzar el proyecto, sin embargo, no es un entorno específico para la plataforma a la que va destinada la app y se hacen ver numerosas carencias.
    - Android Studio**.** El hecho de que este joven software comenzara a lograr versiones estables (aproximadamente la versión 0.6), hizo que se produjera la migración desde Eclipse. Es un IDE específico para la plataforma Android y aporta mayor fluidez y comodidad a la hora de desarrollar.
    - Genymotion**.** Es un emulador externo que se acopla al entorno de desarrollo. Los IDE tratados en los dos puntos anteriores fallan en un mismo pilar, la emulación de la aplicación. Los emuladores que aportan los entornos de desarrollo son lentos e inestables.

      Durante el desarrollo, esto provocó el paso a un emulador externo que sorprendió bastante de forma positiva por su velocidad y robustez.
    - SQLiteManager Mozilla Firefox Extension. Otro pilar fundamental que se puede echar en falta durante el desarrollo es un buen gestor de BBDD. Los problemas surgieron al realizar pruebas de emulación directamente sobre dispositivos reales (simulaciones mucho más rápidas y realistas que en los emuladores software). Gracias a este plug-in de Mozilla se consigue subsanar esta carencia.
- Herramientas hardware**.** Son los componentes físicos que serán necesarios para el desarrollo del proyecto. Como se describe en el apartado anterior, al principio del proyecto se utilizaron varios emuladores, aunque se terminó cediendo a la emulación sobre terminal real, por ser más rápida y testeable.
    - Portátil personal**.** Para el proyecto, se cuenta con el ordenador personal del autor como herramienta principal para el desarrollo. Algunas especificaciones del sistema son:
        - Modelo**:** HP Pavilion g6 (2011)
        - CPU**:** Intel Core-i3 2330 4x2.2Ghz.
        - RAM**:** 6GB (1x4GB + 1x2GB)
        - Discos**:** HDD SATA 500GB 5400rpm + SSD Kingston 120GB
    - Samsung Galaxy S3 I9300

        - CPU**:** Exynos Quad-Core 1.4 Ghz
        - Pantalla**:** 4.8” Super Amoled
        - Versión de Android**:** 4.3. Jelly Bean
        - RAM**:** 1GB
    - Oneplus One
        - CPU**:** Qualcomm Snapdragon 801 Quad-Core 2.5Ghz
        - Pantalla**:** 5.5” LTPS LCD
        - Versión de Android**:** 5.0.2 Lollipop
        - RAM**:** 3GB

4.2. Estimación de tiempo

Al ser implementado este proyecto con la metodología SCRUM de desarrollo incremental, es bastante complicado estimar el tiempo que llevará dicho desarrollo de forma precisa, ya que los requerimientos no siempre han permanecido fijos en el tiempo.

Aun así, se consideró necesaria la imposición de dos metas principales:

- Hito 1**.** Conseguir una versión beta, previa a la comercialización, que el equipo de testeo de la empresa pudiera probar.
    - Plazo: hasta el 31 Marzo de 2015.
    - Cumplida
- Hito 2. Conseguir la versión comercializable, pasada la fase de pruebas internas en la empresa, habiendo completado el proceso de subida al Play Store (Google) con los contenidos pertinentes.
    - Plazo: hasta el 30 de Abril de 2015.
    - Cumplida

Estas metas enmarcan el proyecto entre los meses de Septiembre de 2014 y Abril de 2015. Un total de 8 meses de desarrollo que será desglosado en el siguiente apartado.

4.2.1. Diagrama de Gantt

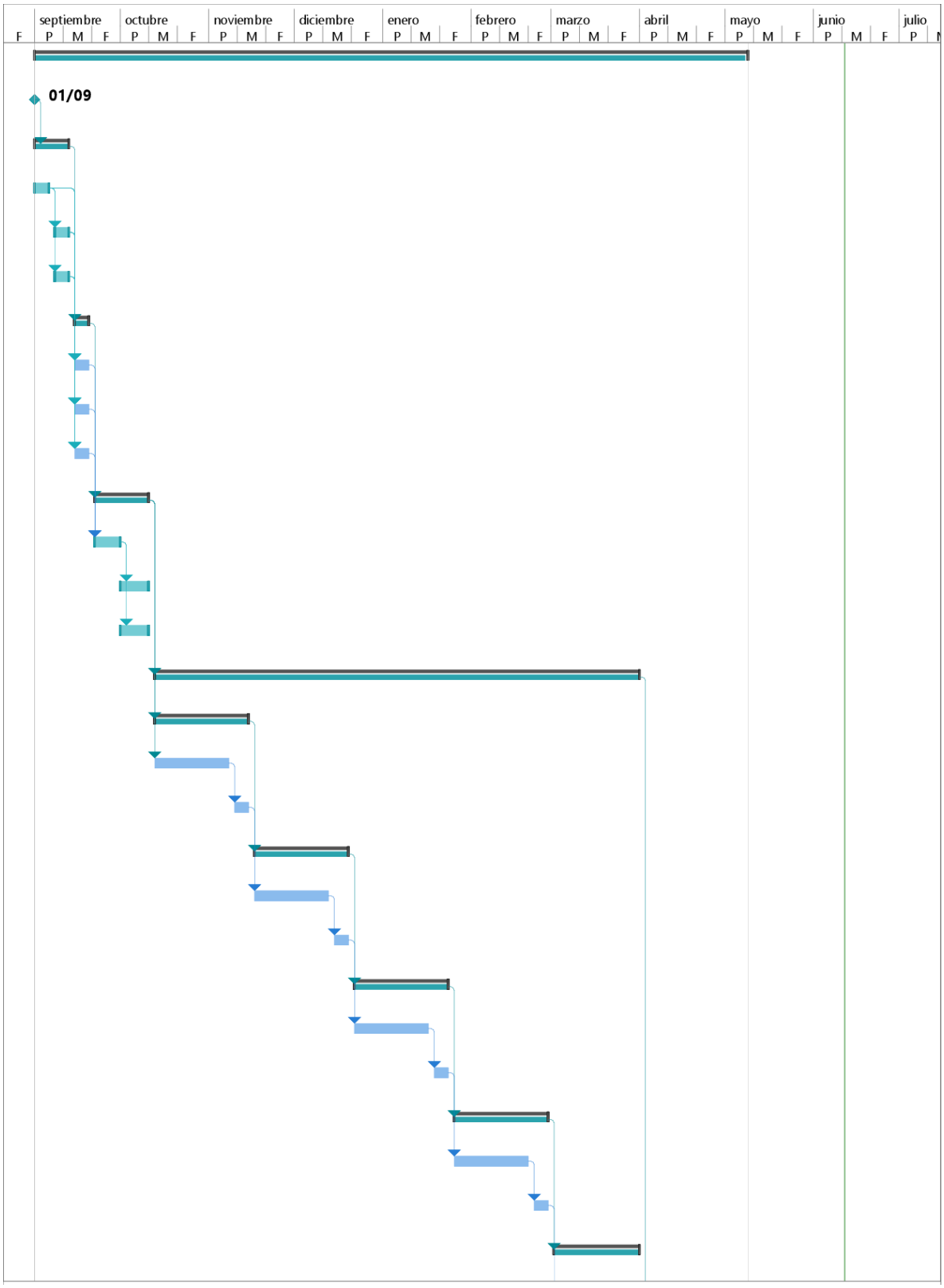
El diagrama de Gantt es una herramienta gráfica cuyo objetivo principal es exponer el tiempo previsto para diferentes tareas o actividades a lo largo del desarrollo del proyecto.

A continuación se muestran las diferentes tareas que han sido necesarias para el desarrollo del proyecto (*Tabla 4.1)* así como el gráfico que las enmarca en el tiempo (*Gráfico 4.1)*.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Proyecto App: TCPline**	**9 mss**	**lun 01/09/14**	**vie 08/05/15**	
2	Inicio	0 días	lun 01/09/14	lun 01/09/14	
3	**Fase 1: Análisis de requerimientos**	**2 sem.**	**lun 01/09/14**	**vie 12/09/14**	**2**
4	Lecturas preparatorias	1 sem	lun 01/09/14	vie 05/09/14	
5	Entrevistas con dirección, empleados y usuarios finales	1 sem	lun 08/09/14	vie 12/09/14	4
6	Observación	1 sem	lun 08/09/14	vie 12/09/14	4
7	**Fase 2: Planificación**	**5 días**	**lun 15/09/14**	**vie 19/09/14**	**3**
8	Estimación de recursos	5 días	lun 15/09/14	vie 19/09/14	4;5;6
9	Estimación de tiempo	5 días	lun 15/09/14	vie 19/09/14	4;5;6
10	Estimación de costes y esfuerzos	5 días	lun 15/09/14	vie 19/09/14	4;5;6
11	**Fase 3: Diseño de la aplicación**	**15 días**	**lun 22/09/14**	**vie 10/10/14**	**7**
12	Selección de tecnologías	7 días	lun 22/09/14	mar 30/09/14	8;9;10
13	Diseño de la BBDD	8 días	mié 01/10/14	vie 10/10/14	12
14	Realización de Storyboards	8 días	mié 01/10/14	vie 10/10/14	12
15	**Fase 4: Implementación (SCRUM)**	**122 días**	**lun 13/10/14**	**mar 31/03/15**	**11**
16	**Iteración 1**	**25 días**	**lun 13/10/14**	**vie 14/11/14**	**11**
17	Implementación de funcionalidades principales	20 días	lun 13/10/14	vie 07/11/14	11
18	Validación de requisitos	5 días	lun 10/11/14	vie 14/11/14	17
19	**Iteración 2**	**25 días**	**lun 17/11/14**	**vie 19/12/14**	**16**
20	Ajuste de requisitos e incorporación de nuevas funcionalidades	20 días	lun 17/11/14	vie 12/12/14	18
21	Validación de requisitos	5 días	lun 15/12/14	vie 19/12/14	20
22	**Iteración 3**	**25 días**	**lun 22/12/14**	**vie 23/01/15**	**19**
23	Ajuste de requisitos e incorporación de nuevas funcionalidades	20 días	lun 22/12/14	vie 16/01/15	21
24	Validación de requisitos	5 días	lun 19/01/15	vie 23/01/15	23
25	**Iteración 4**	**25 días**	**lun 26/01/15**	**vie 27/02/15**	**22**
26	Ajuste de requisitos e incorporación de nuevas funcionalidades	20 días	lun 26/01/15	vie 20/02/15	24
27	Validación de requisitos	5 días	lun 23/02/15	vie 27/02/15	26
28	**Iteración 5**	**22 días**	**lun 02/03/15**	**mar 31/03/15**	**25**

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
29	Ajuste de requisitos e incorporación de nuevas funcionalidades	15 días	lun 02/03/15	vie 20/03/15	27
30	Validación de requisitos finales	7 días	lun 23/03/15	mar 31/03/15	29
31	Hito 1: Conseguir beta para testing	0 días	mar 31/03/15	mar 31/03/15	30
32	**Fase 5: Pruebas**	**1,15 mss**	**mar 31/03/15**	**jue 30/04/15**	**15**
33	Facilitar un ejecutable de la APP a los usuarios finales para su testeo	2 días	mié 01/04/15	jue 02/04/15	30
34	Corrección de bugs	1 ms	vie 03/04/15	jue 30/04/15	33
35	**Fase 6: Publicación**	**5 días**	**vie 01/05/15**	**jue 07/05/15**	
36	Preparación de los documentos necesarios para la subida al Play Store	4 días	vie 01/05/15	mié 06/05/15	32
37	Subida al Play Store	1 día	jue 07/05/15	jue 07/05/15	36
38	Hito 2: Conseguir versión comercializable	0 días	jue 07/05/15	jue 07/05/15	37
39	Fin	0 días	vie 08/05/15	vie 08/05/15	35

Tabla 4.1


septiembre
octubre
noviembre
diciembre
enero
febrero
marzo
abril
mayo
junio
julio
01/09

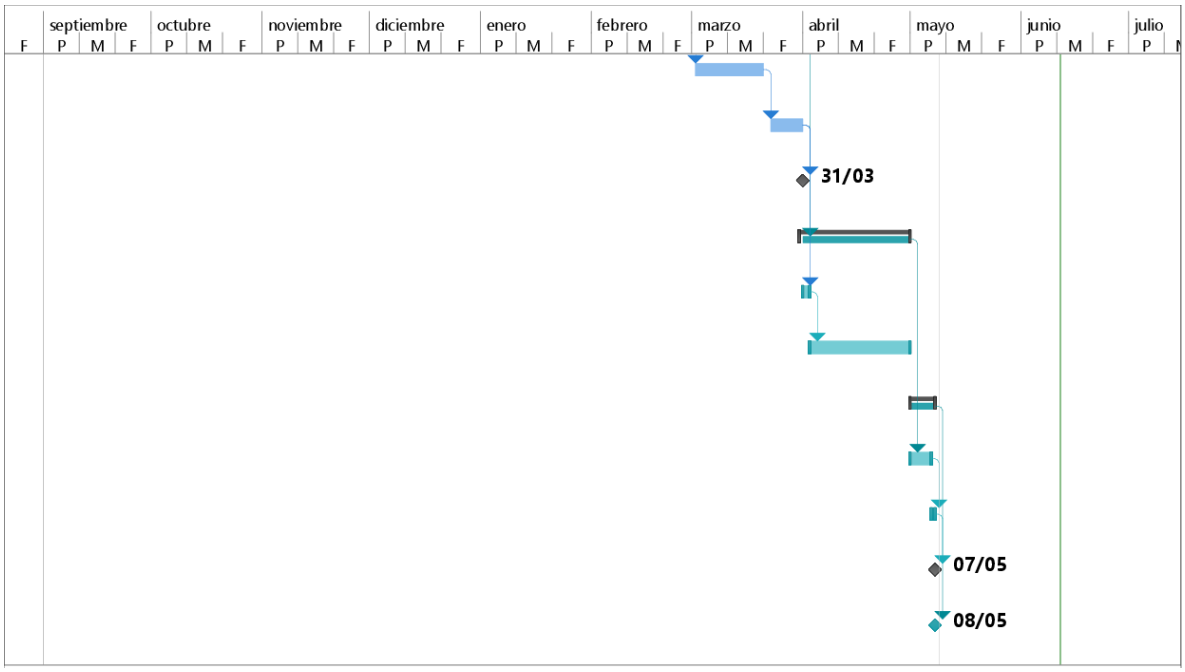


Gráfico 4.1

4.3. Estimación de costes y esfuerzos

En la empresa en cuestión, se trabaja con un sistema de gestión de tareas. Este sistema facilita mucho el desarrollo de esta parte de la documentación del proyecto, ya que en él, se desglosan las distintas tareas necesarias para la realización del proyecto, así como los costes por hora de cada una de las tareas.

Las tareas asignadas por parte de la empresa al desarrollador del proyecto inicialmente fueron las siguientes:

- Desarrollo de aplicaciones – Análisis y diseño.

  En esta tarea se incluye todo el proceso de análisis de requisitos y diseño iniciales del proyecto. Precio de la tarea: 16€ /h.
- Desarrollo de aplicaciones – Bases de datos.

  En esta tarea se incluye todo el diseño de la base de datos de la aplicación, así como la gestión de la misma. Precio de la tarea: 14 €/h.
- Desarrollo de aplicaciones – Programación.

  Es la tarea genérica que engloba las diferentes tareas de programación de las distintas partes de la aplicación. Precio de la tarea: 12 €/h.
- Desarrollo de aplicaciones – Beta tester.

  Esta tarea engloba la fase de pruebas sobre los distintos incrementos de la aplicación. Precio de la tarea: 9€/h.
- Desarrollo de aplicaciones – Corrección de bugs.

En esta tarea se incluye la fase de detección y corrección de errores de la aplicación.

Precio de la tarea: 12 €/h

- Reunión interna – Evaluación del Sprint.

Tarea de presentación del incremento, así como la recepción del feedback necesario para el próximo incremento. Precio de la tarea: 7,5 €/h

A continuación, se muestra una tabla con la estimación realizada para el desglose de cada una de las tareas. Esta estimación ha sido realizada teniendo en cuenta un trabajo semanal aproximado de 25 horas, durante los 8 meses en los que se enmarca el proyecto, por tanto, el esfuerzo estimado estará alrededor de las 850h, contando que cada mes tiene 4 semanas y media de media.

Tarea	Precio (€/h)	Esfuerzo empleado (h) (aprox.)	Total (€)
Análisis y diseño	16	120	1920
Base de datos	14	140	1960
Programación	12	380	3360
Beta tester	9	90	810
Corrección de bugs	12	90	1080
Evaluación del Sprint	7,5	30	225
		850 h	9355 €

Tabla 4.2

La estimación del número de horas para cada una de las tareas se ha realizado con la colaboración del departamento de informática, que ha aportado los datos de proyectos similares anteriores afrontados por la empresa en cuestión.

5. DISEÑO DE LA APLICACIÓN

Una vez realizado el análisis de requisitos de la aplicación y las estimaciones necesarias, comienza la etapa de diseño de la aplicación en cuestión.

A lo largo de este apartado se va a realizar dicho diseño, el cual se puede desglosar en dos grandes bloques:

- Diseño del modelo de la base de datos.
- Realización del *Storyboard* como herramienta gráfica para la comprensión y entendimiento del flujo de la navegación del software por parte del cliente.

La combinación de ellos, puede proporcionar una idea bastante clara de cómo quedará finalmente la aplicación, tras superar los distintos incrementos.

5.1. Diseño de la base de datos

Es una parte fundamental dentro de la etapa de diseño, ya que es cuando se organizará la estructura que contendrá los distintos datos que necesitará nuestra aplicación, así como las relaciones que deben existir entre ellos para proporcionar una base robusta y eficiente.

En el proyecto que se presenta, serán necesarias múltiples bases de datos distintas e independientes, una para cada una de las tiendas y otra genérica que las coordine.

- Base de datos de tienda**.** En ella se encontrarán todos los datos necesarios para representar la estructura de una tienda concreta. Almacenará los datos de sus clientes, productos, presupuestos y configuraciones internas.
- Base de datos general**.** En ella se encontrarán datos de configuración, necesarios para coordinar las diferentes tiendas dentro de la aplicación.

Dadas las características del proyecto se ha considerado el uso del modelo entidad – relación. Este está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre esos objetos.

Para entender esta herramienta de modelado de datos, es importante tener claros los conceptos principales:

- Entidad. Representa un objeto del mundo real con existencia independiente, es decir, se diferencia únicamente de otro objeto incluso siendo del mismo tipo.
- Atributos. Cada una de las características que definen o identifican a una entidad. Pueden ser muchas, y el diseñador sólo utiliza o implementa las que considere relevantes.
- Relación. Describe cierta dependencia entre entidades o permite la asociación de las mismas.

Una vez aclarados estos conceptos básicos, se define cada una de las entidades con sus atributos y para después representarlas gráficamente con ayuda de sus relaciones.

5.1.1. Base de datos general

En ella se organizarán los datos necesarios para coordinar las diferentes tiendas que se encuentran en la aplicación, así como la información de configuración genérica para la misma.

Para esta base de datos, no es necesario realizar el modelo entidad-relación por la simplicidad de la misma, ya que contiene una única entidad, donde cada instancia es independiente de las demás.

5.1.1.1. Modelo relacional

Esta base de datos tendrá básicamente una única entidad principal, la entidad tienda.

Tienda

Esta entidad representa las diferentes tiendas en las que puede estar dado de alta un comercial. La mayoría de los atributos se recibirán del servidor, coincidiendo con la base de

datos interna del mismo. Los atributos que la definen son los siguientes:

Name	Type	Primary Key
id_shop	INTEGER	1
shop_name	VARCHAR	0
url_json	VARCHAR	0
db_name	VARCHAR	0
date_upd	VARCHAR	0
cookie_key	VARCHAR	0
all_images	INTEGER	0
iso_currency	VARCHAR	0
currency_sign	VARCHAR	0
language_name	VARCHAR	0
iso_language	VARCHAR	0
id_employee	INTEGER	0
employee_name	VARCHAR	0
id_discount_access	INTEGER	0
valid_passwd	VARCHAR	0
min1	VARCHAR	0
min2	VARCHAR	0
med1	VARCHAR	0
med2	VARCHAR	0
max1	VARCHAR	0
max2	VARCHAR	0

Tabla 5.1

- **id_shop.** ID único asignado a la tienda. Actúa como clave primaria por no poder existir dos tiendas con un mismo ID.
- **shop_name.** Nombre de la tienda.
- **url_json.** Almacena la dirección http a la que accederá para sincronizar sus productos, clientes y presupuestos.
- **db_name.** Nombre de la base de datos de la tienda. Dado que al crear las diferentes bases de datos específicas para cada tienda hay que asignarle un nombre, se recibe este parámetro desde el servidor para que la estructura interna quede lo más semejante posible.
- **date_upd.** Fecha y hora de la última sincronización realizada por la tienda.
- **cookie_key.** Sirve para la generación del hash con el algoritmo md5 para la codificación de las contraseñas.

- **all_images.** Proporciona la información para saber qué imágenes han de descargarse en cada sincronización de la tienda (0 sólo imagen de portada, 1 para todas las imágenes).
- **iso_currency.** Codigo ISO de la moneda por defecto de la tienda.
- **currency_sign.** Símbolo de la moneda.
- **iso_language.** Código ISO del idioma por defecto de la tienda (previsto para versiones posteriores con multi-idioma).
- **id_employee.** ID único del comercial para una tienda. Se utilizará cuando se realicen los GET. Así se controlan los accesos a la información que tiene cada comercial.
- **employee_name.** Nombre del comercial.
- **id_discount_access.** Código de acceso a descuentos. Dependiendo del rango del comercial dentro de una tienda, tendrá acceso a realizar descuentos más o menos agresivos a los productos.
- **valid_password.** Conjunto de contraseñas codificadas para abrir el candado. Si el comercial obtiene una de estas contraseñas, podrá realizar cualquier tipo de descuento dentro de la tienda.
- **min1, min2, med1, med2, max1, max2.** Conjunto de pares descuento-incremento. Contendrán uno o varios caracteres que identificaran los diferentes descuentos dentro de la tienda. A simple golpe de vista, el comercial podrá saber qué descuentos tiene permitidos para cada uno de los productos gracias a estos caracteres.

5.1.2. Base de datos de tienda

Esta base de datos contendrá las diferentes entidades y sus relaciones, encargadas de gestionar el correcto funcionamiento de cada una de las tiendas.

5.1.2.1. Modelo relacional

En este apartado se van a definir cada una de las entidades colaboradoras en la base de datos de cada tienda.

Cliente

Esta entidad define los atributos y características de los clientes que están o serán dados de alta en cada una de las tiendas. Sus atributos son los siguientes:

Name	Type	Primary Key
id_own_customer	INTEGER	1
id_web_customer	INTEGER	0
id_gender	INTEGER	0
email	VARCHAR	0
passwd	VARCHAR	0
date_add	VARCHAR	0
date_upd	VARCHAR	0
dni	VARCHAR	0
firstname	VARCHAR	0
lastname	VARCHAR	0
contact_person	VARCHAR	0
contact_phone	VARCHAR	0
special_conditions	VARCHAR	0
editable	INTEGER	0
comercial_asignado	VARCHAR	0
sincronizado	INTEGER	0

Tabla 5.2

- **id_own_customer.** ID único asignado a un cliente en la APP. Es el ID con el que se trabajará internamente en la aplicación. Funcionará como clave primaria de la tabla.
- **id_web_customer.** ID único asignado a un cliente en la versión web. Es necesario para poder sincronizar los datos desde la web.
- **id_gender.** ID asociado al género del cliente. (0 para masculino, 1 para femenino, 9 para empresa)
- **email.** Dirección de correo electrónico del cliente.
- **password.** Contraseña del cliente. Se guarda codificada con MD5 y la cookie key de la tienda.
- **date_add.** Fecha y hora en la que se dio de alta el cliente en el sistema.
- **date_upd.** Fecha y hora de la última actualización de los datos del cliente.
- **dni.** Documento de identidad del cliente.
- **firstname.** Nombre completo del cliente.
- **lastname.** Alias del cliente.
- **contact_person.** Nombre de la persona de contacto del cliente.
- **contact_phone.** Teléfono de contacto del cliente.
- **special_conditions.** Condiciones generales de cliente.
- **editable.** Campo que controlará si un comercial tiene permiso para editar un cliente o solamente puede realizarle presupuestos.

- **comercial_asignado.** Nombre del comercial por defecto que tiene asignado a este cliente.
- **sincronizado.** Controla la sincronización de los clientes. Si es 0, significa que aún no está sincronizado con el servidor, si es 1, quiere decir que los datos están actualizados en el servidor.

Dirección

Es la entidad que contendrá los diferentes datos de la dirección de entrega o facturación de un cliente. Sus atributos son los siguientes:

Name	Type	Primary Key
id_own_address	INTEGER	1
id_web_address	INTEGER	0
id_customer	INTEGER	0
id_gender	INTEGER	0
destination_name	VARCHAR	0
apodo	VARCHAR	0
dni	VARCHAR	0
address	VARCHAR	0
address_alias	VARCHAR	0
postcode	VARCHAR	0
city	VARCHAR	0
province	VARCHAR	0
country	INTEGER	0
contact_person	VARCHAR	0
contact_phone	VARCHAR	0
contact_email	VARCHAR	0
observ	VARCHAR	0
date_add	VARCHAR	0
date_upd	VARCHAR	0
tipo	CHAR	0
sincronizado	INTEGER	0

Tabla 5.3

- **id_own_address.** ID único asignado a una dirección en la APP. Se trabajará internamente con este ID. Funcionará como clave primaria de la tabla.
- **id_web_address.** ID único asignado a una dirección en la versión web. Es necesario para poder sincronizar los datos desde la web.
- **id_customer.** ID del cliente al que pertenece la dirección.

- **id_gender.** ID asociado al género del destinatario. (0 para masculino, 1 para femenino, 9 para empresa)
- **destination_name.** Nombre completo del destinatario.
- **apodo.** Alias del destinatario.
- **dni.** Documento de identidad del destinatario (necesario sólo para las direcciones de facturación).
- **address.** Dirección completa del destinatario.
- **address_alias.** Alias para identificar rápidamente la dirección. (ej. Mi casa)
- **postcode.** Código postal.
- **city.** Municipio del destinatario.
- **province.** Región / provincia del destinatario.
- **country.** ID único de país (perteneciente a la entidad País).
- **contact_person.** Nombre completo de una persona de contacto.
- **contact_phone.** Teléfono de la persona de contacto.
- **contact_email.** Correo electrónico de la persona de contacto.
- **Observ.** Campo de observaciones de la dirección.
- **date_add.** Fecha y hora de alta en el sistema.
- **date_upd.** Fecha y hora de la última actualización de la dirección.
- **tipo.** 'F' para facturación y 'E' para entrega.
- **sincronizado.** Controla la sincronización de las direcciones. Si es 0, significa que aún no está sincronizado con el servidor, si es 1, quiere decir que los datos están actualizados en el servidor.

País

Esta entidad contendrá los datos necesarios para acceder a las distintas nacionalidades de los clientes. Podría pensarse que esta entidad debe pertenecer a la base de datos general, sin embargo, debido a que en versiones posteriores podrá cambiarse el idioma de una tienda, y cada tienda podrá tener uno diferente, esta entidad debe pertenecer a esta base de datos, para mostrar los países en el idioma por defecto de la tienda, por ejemplo. Sus atributos son:

Name	Type	Primary Key
id_country	INTEGER	1
name	VARCHAR	0
iso_code	VARCHAR	0

Tabla 5.4

- **id_country.** ID único para identificar un país. Funcionará como clave primaria de la tabla.
- **name.** Nombre del país.
- **Iso_code.** Código internacional del país.

Moneda

Contendrá los datos necesarios para acceder a las distintas monedas en las que pueden estar expresados los precios de una tienda. Sus atributos son:

Name	Type	Primary Key
name	VARCHAR	0
iso_code	VARCHAR	1
sign	VARCHAR	0
conversion_rate	DECIMAL	0
selected	INTEGER	0

Tabla 5.5

- **name.** Nombre de la moneda.
- **iso_code.** Código internacional de la moneda. Funcionará como clave primaria de la tabla.
- **sign.** Símbolo de la moneda.
- **conversión_rate.** Tasa de conversión entre la moneda por defecto de la tienda y las demás a las que pueden cambiarse los precios.
- **selected.** Valor booleano que sirve para distinguir la moneda que está seleccionada en una tienda en un momento determinado (cuando selected = 1). Por tanto, sólo habrá una fila que tenga este campo a 1 simultáneamente.

Categoría

Entidad que servirá para organizar los productos del sistema de forma jerárquica. Pueden contener subcategorías (entidades del mismo tipo) y éstas a su vez tener también subcategorías. Sus atributos son:

Name	Type	Primary Key
id_category	INTEGER	1
id_parent	INTEGER	0
level_depth	INTEGER	0
date_upd	VARCHAR	0
name	VARCHAR	0

Tabla 5.6

- **id_category.** ID único que servirá para identificar cada una de las categorías dentro del sistema. Funcionará como clave primaria en la tabla.
- **id_parent.** ID de la categoría padre (en el nivel inmediatamente superior).
- **level_depth.** Profundidad de la categoría dentro del árbol de categorías. Este valor pertenece al intervalo [0-2].
- **date_upd.** Fecha de la última modificación de los datos de una categoría.
- **name.** Nombre de la categoría.

Producto

Es la entidad que identificará cada uno de los productos que serán almacenados en la base de datos. Sus atributos son los siguientes:

Name	Type	Primary Key
id_product	INTEGER	1
date_upd	VARCHAR	0
id_tienda	INTEGER	0
id_marca	INTEGER	0
ref	VARCHAR	0
name	VARCHAR	0
id_categorias	VARCHAR	0
id_category_default	INTEGER	0
peso	DECIMAL (6,3)	0
volumen	DECIMAL(6,3)	0
price	DECIMAL(14,3)	0
reduction_price	DECIMAL(14,3)	0
reduction_percent	DECIMAL(4,3)	0
reduction_from	VARCHAR	0
reduction_to	VARCHAR	0
formato_venta	VARCHAR	0
cantidad_minima_venta	DECIMAL	0
unidades_x_envase	DECIMAL	0
tipo_envase	VARCHAR	0
dtomin1	DECIMAL	0
dtomin2	DECIMAL	0
dtomed1	DECIMAL	0
dtomed2	DECIMAL	0
dtomax1	DECIMAL	0
dtomax2	DECIMAL	0
disp_tienda	VARCHAR	0
disp_proveedor	VARCHAR	0
short_description	VARCHAR	0
long_description	VARCHAR	0
extra_description	VARCHAR	0

Tabla 5.7

- **id_product.** Identificador único de producto. Actuará como clave primaria de la tabla.
- **date_upd.** Fecha de la última modificación de este producto.
- **id_tienda.** Identificador del proveedor del que procede.
- **id_marca.** Identificador de su marca (si la tiene). Irá vinculado con la tabla de marcas.
- **ref.** Referencia del artículo en la tienda.
- **name.** Nombre del artículo en la tienda.
- **id_categorías.** Cadena compuesta por los ID's de las categorías a las que pertenece este producto.
- **id_category_default.** Id de la categoría por defecto.
- **peso.** Peso físico del artículo.
- **volumen.** Volumen físico que ocupa el producto.
- **price.** Precio de venta al público del objeto (sin ofertas).

- **reduction_price.** Descuento por precio neto de un producto en oferta.
- **reduction_percent.** Descuento en porcentaje de un producto en oferta. Este campo y el anterior son excluyentes, es decir, si un campo está relleno, el otro deberá estar vacío. Sólo podrá tener un tipo de descuento simultáneamente.
- **reduction_from, reduction_to.** Fecha de comienzo y fin de la oferta respectivamente.
- **formato_venta.** Formato de venta del artículo (m2, litro, cm3, etc.).
- **cantidad_mínima_venta.** Número de unidades mínimo para poder vender el artículo.
- **unidades_x_envase.** Cantidad del artículo por envase.
- **tipo_envase.** Tipo de envase del producto (caja, blíster, botella, etc.).
- **dtomin1, dtomin2, dtomed1, dtomed2, dtomax1, dtomax2.** Descuentos del artículo destinados al comercial. El comercial podrá ver estos descuentos en la ficha de producto, sabiendo qué descuentos puede aplicar para cada uno de ellos.
- **disp_tienda, disp_proveedor.** Disponibilidad del artículo en la tienda y para el proveedor respectivamente (Verde = Disponible, Amarillo = Pocas unidades, Rojo = Sin Stock, Negro = Artículo descatalogado, Blanco = Stock desconocido).
- **short_description.** Descripción corta del artículo. Proporcionada por el proveedor.
- **long_description.** Descripción detallada del artículo. Proporcionada por el proveedor.
- **extra_description.** Descripción adicional del artículo aportada por el responsable de la tienda.

Marca

Entidad que contendrá los atributos necesarios para definir las marcas que agruparán los distintos productos del sistema. Sus atributos son:

Name	Type	Primary Key
id_marcas	INTEGER	1
name	VARCHAR	0
date_upd	VARCHAR	0
img	VARCHAR	0

Tabla 5.8

- **id_marcas.** Identificador único de marca. Actuará como clave primaria de la tabla.
- **name.** Nombre de la marca. Es el que se mostrará en el listado de marcas.
- **date_upd.** Última modificación de la marca.

- **img.** Url interna del logo de la marca.

Presupuesto

Es la entidad que contendrá los datos generales de un presupuesto. Los atributos que la forman son:

Name	Type	Primary Key
id_own_order	INTEGER	1
id_web_order	INTEGER	0
writing	INTEGER	0
id_customer	INTEGER	0
nombre_cliente	VARCHAR	0
email_cliente	VARCHAR	0
iso_code_moneda	VARCHAR	0
sign_moneda	VARCHAR	0
iso_idioma	VARCHAR	0
estado	INTEGER	0
payment	VARCHAR	0
total_products	DECIMAL(5,3)	0
total_shipping	DECIMAL(5,3)	0
total_paid	DECIMAL(5,3)	0
date_add	VARCHAR	0
date_upd	VARCHAR	0
nombre_asignado	VARCHAR	0
nombre_proyecto	VARCHAR	0
observ	VARCHAR	0
servir	VARCHAR	0
persona_contacto	VARCHAR	0
telf_contacto	VARCHAR	0

Tabla 5.9

id_delivery_address	INTEGER	0
de_id_gender	INTEGER	0
de_nombre	VARCHAR	0
de_apodo	VARCHAR	0
de_direccion	VARCHAR	0
de_cp	VARCHAR	0
de_ciudad	VARCHAR	0
de_provincia	VARCHAR	0
de_pais	INTEGER	0
de_persona_contacto	VARCHAR	0
de_telefono	VARCHAR	0
de_email	VARCHAR	0
de_observ	VARCHAR	0
id_invoice_address	INTEGER	0
df_id_gender	INTEGER	0
df_nombre	VARCHAR	0
df_apodo	VARCHAR	0
df_direccion	VARCHAR	0
df_cp	VARCHAR	0
df_ciudad	VARCHAR	0
df_provincia	VARCHAR	0
df_pais	INTEGER	0
df_persona_contacto	VARCHAR	0
df_telefono	VARCHAR	0
df_dni	VARCHAR	0
df_email	VARCHAR	0
df_observ	VARCHAR	0
dto1_porcentaje	DECIMAL(3,5)	0
dto1_cantidad	DECIMAL(5,3)	0
dto2_porcentaje	DECIMAL(3,5)	0
dto2_cantidad	DECIMAL(3,5)	0

Tabla 5.10

- **id_own_order.** Identificador único propio de un presupuesto. Actuará como clave primaria.
- **Id_web_order.** Identificador único de la web. Es necesario mantener los dos, ya que no sabemos qué ID será el siguiente en cada momento en la aplicación web.
- **writing.** Valor booleano que será 1 cuando el presupuesto en cuestión esté incompleto (presupuesto actual) y 0 para el resto de presupuestos.
- **id_customer.** ID de cliente al que está destinado el presupuesto.

- **nombre_cliente.** Nombre del cliente en el momento que cerró el presupuesto. Es necesario guardarlo por si se modificara la entidad Cliente.
- **email_cliente.** Email de contacto del cliente en el momento que cerró el presupuesto.
- **iso_code_moneda.** Identificador internacional de la moneda en la que está realizado el presupuesto.
- **sign_moneda.** Símbolo de la moneda de los precios del presupuesto.
- **iso_idioma.** Idioma por defecto en el que estaba la tienda en el momento que se cerró el presupuesto.
- **estado.** Código propio que nos dirá en qué estado se encuentra el presupuesto en cada momento, desde que se comienza el presupuesto hasta que realiza o cancela el pedido.
- **payment.** Forma de pago para el pedido.
- **total_products.** Precio de la suma total de los precios de todos los artículos del presupuesto.
- **total_shipping.** Precio de los gastos de envío del presupuesto.
- **total_paid.** Total a pagar.
- **date_add.** Fecha en que se creó el presupuesto.
- **date_upd.** Fecha de la última actualización del presupuesto.
- **nombre_asignado.** Nombre del comercial asignado al presupuesto.
- **nombre_proyecto.** Nombre o breve descripción para identificar el presupuesto.
- **observ.** Observaciones generales del presupuesto.
- **persona_contacto.** Nombre de la persona de contacto del presupuesto.
- **telf_contacto.** Teléfono de la persona de contacto del presupuesto.
- **id_delivery_address, id_invoice_address.** Identificadores de las direcciones seleccionadas del cliente para el presupuesto.
- **Campos con prefijos “de” y “df”.** Irán rellenados en el caso de que se escoja introducir una dirección personalizada para este presupuesto. Dicha dirección no será almacenada en la tabla de direcciones del cliente.
- **dto1_cantidad.** Primer descuento por precio neto 1 general del presupuesto.
- **dto1_porcentaje.** Primer descuento por porcentaje 1 general del presupuesto. Este campo y el anterior son excluyentes.
- **dto2_cantidad.** Segundo descuento por precio neto general del presupuesto.
- **dto2_porcentaje.** Segundo descuento por porcentaje general del presupuesto. Este campo y el anterior son excluyentes.

- **sincronizado.** Controla la sincronización de las direcciones. Si es 0, significa que aún no está sincronizado con el servidor, si es 1, quiere decir que los datos están actualizados en el servidor.

Línea de presupuesto

Esta entidad hace referencia a cada una de las líneas de un presupuesto, es decir, de cada producto, qué cantidad, con qué descuentos y en qué condiciones se han añadido al presupuesto. Sus atributos son los siguientes:

Name	Type	Primary Key
id_own_order_detail	INTEGER	1
id_web_order_detail	INTEGER	0
id_order	INTEGER	0
product_id	INTEGER	0
product_name	INTEGER	0
product_quantity	INTEGER	0
product_price	DECIMAL (4,3)	0
product_reference	VARCHAR	0
descuento_cantidad	DECIMAL(4,3)	0
descuento_porcentaje	DECIMAL(3,3)	0
incremento_porcentaje	DECIMAL(3,3)	0
destino	VARCHAR	0
observ	VARCHAR	0
formato_venta	VARCHAR	0
medidas_especiales	INTEGER	0
unidades_x_envase	DECIMAL(4,3)	0
tipo_envase	VARCHAR	0
cantidad_minima_venta	DECIMAL (3,2)	0
condiciones	VARCHAR	0
dtomin1	DECIMAL	0
dtomin2	DECIMAL	0
dtomed1	DECIMAL	0
dtomed2	DECIMAL	0
dtomax1	DECIMAL	0
dtomax2	DECIMAL	0
tipo	VARCHAR	0
sincronizado	INTEGER	0

Tabla 5.11

- **id_own_order_detail.** Identificador único propio para la línea de presupuesto. Actuará como clave primaria de la tabla.
- **id_web_order_detail.** Identificador único de la aplicación web. Servirá a la hora de realizar las sincronizaciones con el servidor.
- **id_order.** Identificador del presupuesto al que pertenece la línea.
- **product_id.** Identificador del producto añadido al presupuesto.
- **product_name.** Nombre del producto. Este campo será necesario por si el producto es borrado de la base de datos.
- **product_quantity.** Cantidad del producto añadida al presupuesto.
- **product_price.** Precio del producto cuando se añadió al presupuesto. Es necesario guardarlo por si se alterara el producto.
- **product_reference.** Referencia del producto.
- **descuento_cantidad.** Descuento por precio neto realizado por el comercial en el momento de añadir el producto al presupuesto.
- **descuento_porcentaje.** Descuento por porcentaje realizado por el comercial en el momento de añadir el producto al presupuesto.
- **incremento_porcentaje.** Incremento por porcentaje realizado por el comercial en el momento de añadir el producto al presupuesto. El sentido del incremento es que habrá determinados productos que tendrán un precio mayor si se compran en pocas cantidades y el precio puede estar expresado para venta al por mayor.
- **destino.** Forma de agrupar productos dentro de un presupuesto. Por ejemplo, si se está realizando un presupuesto para una habitación de hotel, habrá algunas líneas de presupuesto que estén agrupadas en “baño” y otras en “dormitorio”.
- **observ.** Observaciones al agregar el producto al presupuesto.
- **formato_venta.** Formato de venta del artículo (m2, litro, cm3, etc.).
- **cantidad_mínima_venta.** Número de unidades mínimo para poder vender el artículo.
- **unidades_x_envase.** Cantidad del artículo por envase.
- **tipo_envase.** Tipo de envase del producto (caja, blíster, botella, etc.).
- **dtomin1, dtomin2, dtomed1, dtomed2, dtomax1, dtomax2.** Descuentos del artículo destinados al comercial. El comercial podrá ver estos descuentos en la ficha de producto, sabiendo qué descuentos puede aplicar para cada uno de ellos.
- **condiciones.** Campo de condiciones especiales. Por ejemplo, para pedir una variación en el color si fuera posible.
- **tipo.** Especifica con una “p” si la línea añadida es un producto o con “m” si la línea es una muestra.

- **sincronizado.** Controla la sincronización de los clientes. Si es 0, significa que aún no está sincronizado con el servidor, si es 1, quiere decir que los datos están actualizados en el servidor.

Imagen

Es la entidad que contiene los atributos necesarios para localizar las diferentes imágenes de los productos dentro del sistema. Sus atributos son:

Name	Type	Primary Key
id_image	INTEGER	1
id_product	INTEGER	0
url	VARCHAR	0
position	INTEGER	0
cover	INTEGER	0

Tabla 5.12

- **id_image.** Identificador único de la imagen. Se utilizará como clave primaria para la tabla.
- **id_product.** Identificador del producto al que pertenece la imagen.
- **url.** Ruta local donde se sitúa la imagen.
- **position.** En caso de que un producto tenga varias imágenes, ésta especificará el orden en que se mostrarán las mismas.
- **cover.** Este valor será 1 si la imagen es portada y 0 si no lo es.

Adjunto

Es la entidad que contendrá los valores necesarios para asignar adjuntos a los productos. Sus atributos son los siguientes:

Name	Type	Primary Key
id_attachment	INTEGER	1
id_product	INTEGER	0
name	VARCHAR	0
url	VARCHAR	0

Tabla 5.13

- **id_attachment.** Identificador del adjunto. Actuará como clave primaria en la tabla.
- **id_product.** Identificador del producto al que pertenece el adjunto.
- **name.** Nombre del adjunto. Será el texto del enlace de este adjunto.

- **url.** Dirección externa del enlace.

Estado de presupuesto

Esta entidad contiene el posible estado en el que se encuentra un presupuesto en un momento determinado. Sus atributos son:

Name	Type	Primary Key
id_state	INTEGER	0
state	VARCHAR	0
ord	INTEGER	1

Tabla 5.14

- **id_state.** Identificador de la aplicación web para localizar el estado de un presupuesto. Sirve para la sincronización.
- **state.** Nombre del estado del presupuesto.
- **ord.** Será un entero que marcará el orden en el que se mostrarán los estados, además funcionará como clave primaria, ya que no podrá haber dos estados en la misma posición.

Condiciones específicas

Entidad que contendrá una condición específica cliente-producto, es decir, un cliente podrá tener condiciones especiales sobre unos determinados productos. Sus atributos son:

Name	Type	Primary Key
id_product	INTEGER	1
id_customer	INTEGER	2
conditions	VARCHAR	0

Tabla 5.15

- **id_product.** Identificador del producto al que se asocia la condición.
- **id_customer.** Identificador del cliente al que se le asocia la condición. Este campo y el anterior forman clave primaria, por no poder ser repetidos conjuntamente.
- **conditions.** Condiciones especiales cliente-producto.

5.2. Storyboard

Un storyboard es un guión gráfico que muestra la secuencia de imágenes que se mostrarán en el software teniendo en cuenta las decisiones tomadas por el usuario. Su

objetivo es servir de guía para entender el funcionamiento de la aplicación antes de realizar la implementación.

En el proyecto en cuestión, se ha dividido el storyboard en partes, debido a su gran tamaño, para facilitar la maquetación y visualización de las diferentes partes que lo componen.

Para comprender esta secuencia, se han numerado cada una de las pantallas de la aplicación. Para cada una de las acciones posibles, se ha situado el número correspondiente de pantalla al que se accederá a través de esa acción justo al lado.

Ilustración 5.1

Breve explicación:

- Pantalla 1. Se compone de un breve formulario de log-in con un botón de confirmación. Una vez rellenados los campos, se pulsará sobre dicho botón y se accederá a la pantalla número 2.
- Pantalla 2. Muestra un listado de tiendas. Se podrá pulsar en cada una de las tiendas, accediendo así a la pantalla número 3.
- Pantalla 3. Hace referencia a la pantalla de configuración. Aquí se escogerán las diferentes opciones configurables de la tienda. Una vez seleccionados estos parámetros, accederemos a la pantalla 4 a través del botón “Aceptar”.
- Pantalla 4. Es el menú principal de la tienda. A través de él se podrá navegar por toda la aplicación. Dependiendo de qué opción se pulse sobre el menú, se mostrarán las diferentes pantallas que serán explicadas a continuación.

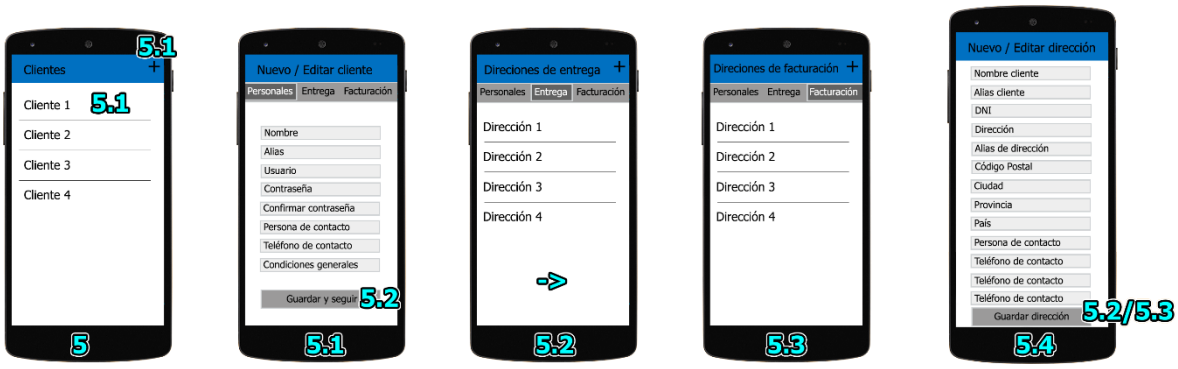


Ilustración 5.2

- Pantalla 5. Muestra el listado de clientes dados de alta en la tienda. Las opciones posibles sobre esta pantalla son:
  - Mantener pulsado sobre un cliente para editarlo.
  - Pulsar sobre “+” para añadir uno desde cero.
- Pantalla 5.1. Se muestra un formulario para rellenar los datos personales del cliente. Una vez cumplimentados, accederemos a los datos de entrega pulsando sobre el botón “Guardar y seguir”, pulsando sobre “Entrega” o “Facturación” en la barra superior o bien deslizando el dedo hacia la izquierda.
- Pantalla 5.2. y 5.3. Esta pantalla hace referencia al listado de direcciones de entrega y facturación de un cliente. Se podrán añadir nuevas direcciones pulsando sobre el botón “+”.
- Pantalla 5.4. Formulario de alta de direcciones. Una vez cumplimentado se pulsará sobre “Guardar dirección” y el programa volverá al listado de direcciones del que proviene.

Ilustración 5.3

- Pantalla 6, 6.1 y 6.2. Se mostrará un listado de categorías / subcategorías / familias sobre las que podremos realizar búsquedas y pulsar, navegando así entre las diferentes categorías de la tienda.

- Pantalla 6.3. Está compuesta por un listado de productos donde se podrá observar rápidamente el precio, la referencia, el nombre y las disponibilidades de venta. Si se pulsa sobre uno de ellos, se abrirá la ficha de producto, reflejada en la pantalla 6.4.
- Pantalla 6.4. Se corresponde con la ficha de producto. En ella se encontrará toda la información que pueda querer el cliente en el momento de la venta (descripciones, referencias, descuentos, precios, imágenes, etc.). Una vez rellenados los campos de cantidad, descuento y observaciones, se podrá continuar accediendo así a la pantalla número 9, que hace referencia al presupuesto que se está realizando en ese momento.

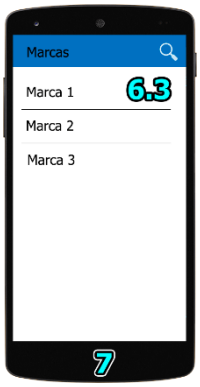


Ilustración 5.4

- Pantalla 7. Muestra la clasificación de los productos según sus marcas. A través de una pulsación sobre cualquiera de ellas, podremos acceder al listado de productos de esta marca, como en la pantalla 6.3.

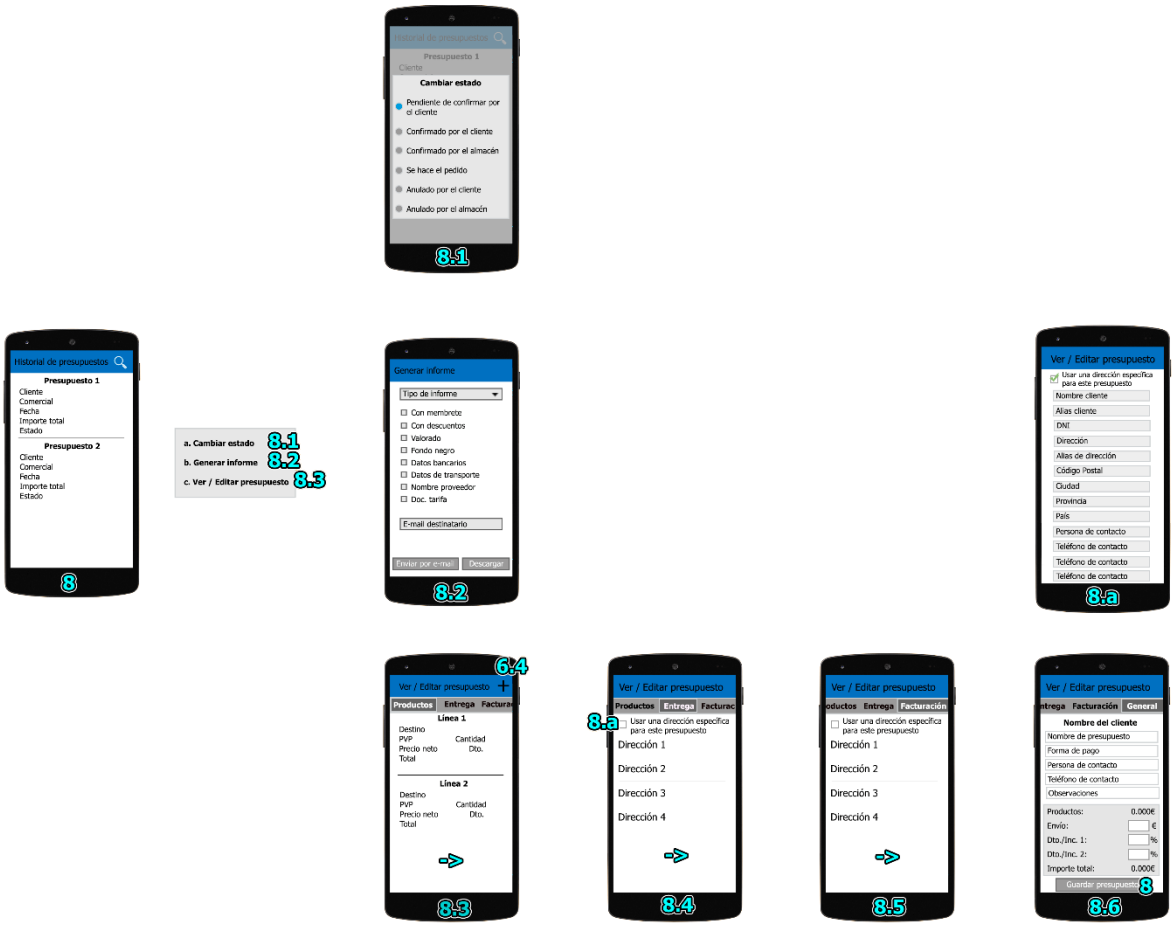


Ilustración 5.5

- Pantalla 8. Mostrará un listado con los distintos presupuestos realizados en la tienda. Mediante una pulsación larga, se mostrará una ventana emergente que nos dará opción a:
    - Cambiar un estado (Pantalla 8.1)
    - Generar un informe (Pantalla 8.2)
    - Ver / Editar presupuesto (Pantalla 8.3)
- Pantalla 8.1. A través de esta pantalla se podrá cambiar el estado de un presupuesto gracias a la selección mediante RadioButtons.
- Pantalla 8.2. Consta de varios parámetros que habrá que seleccionar para generar un informe ajustado a las necesidades del cliente. Se podrá descargar en el dispositivo o mandar por e-mail pulsando en los respectivos botones.
- Pantalla 8.3, 8.4, 8.5 y 8.6. Nos aparecerá el detalle del presupuesto. Cada una de las líneas se componen de los datos de los productos añadidos con sus descuentos y observaciones. Se puede editar cada línea mediante una pulsación larga y añadir una nueva pulsando sobre “+”. Esto nos llevará a la ficha del producto y el listado de categorías respectivamente. Podremos revisar también los datos de entrega (8.4),

facturación (8.5) y generales del presupuesto (8.6) mediante deslizamientos laterales.

- Pantalla 8.a. Es una pantalla alternativa a las de direcciones de entrega y facturación. Al clicar sobre el checkbox de estas pantallas, desaparecerá el listado de direcciones y aparecerá este formulario, preparado para rellenar una dirección específica para el presupuesto que no se guardará en la ficha de cliente.

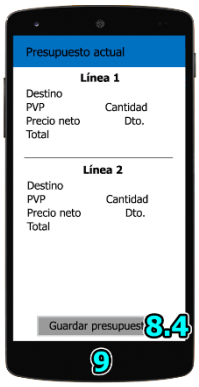


Ilustración 5.6

- Pantalla 9. Se corresponde con la pantalla de presupuesto actual. En ella veremos el estado del presupuesto en un momento determinado. Podemos cerrar el presupuesto pulsando sobre “Guardar presupuesto”. Esto nos llevará a la pantalla de selección de dirección (8.4.).

Como se dijo antes de comenzar esta breve descripción del storyboard, se ha desglosado en varias partes para mejorar su comprensión. Aun así, el storyboard completo se encontrará como anexo del proyecto.

6. SELECCIÓN DE TECNOLOGÍAS

A lo largo de este apartado se explicará el porqué de escoger las tecnologías usadas en el desarrollo, acompañado de una breve comparativa con las tecnologías más competitivas en cada uno de los casos.

Las tecnologías necesarias para un proyecto de este tipo se pueden englobar en dos grandes bloques:

- Plataforma para la que se desarrollará. Es muy importante tener claro el público al que irá destinada la aplicación en sí, por tanto tenemos que tener claro cuál es la plataforma más accesible a este público objetivo.
- Servicios Web necesarios**.** Este bloque engloba el conjunto de protocolos que serán necesarios para interconectar Cliente y Servidor.

Teniendo en cuenta esto, se explicará a continuación qué tecnología se ha escogido para cada caso, porqué y qué otras opciones había disponibles.

6.1. Android

Android es una plataforma de Software basada en el núcleo de Linux. Diseñada para dispositivos móviles, Android permite controlar dispositivos por medio de bibliotecas desarrolladas o adaptadas por Google mediante el lenguaje Java.

Es una plataforma de código abierto. Esto quiere decir que cualquiera puede crear y desarrollar aplicaciones en lenguaje C y compilarlas a código nativo de ARM (API de Android).

Inicialmente desarrollada por Android Inc., pronto fue adquirida por Google y unida al Open Handset Alliance, un consorcio de 78 grandes compañías del mundo del hardware, software y las telecomunicaciones.

El código fuente de Android está publicado bajo la licencia de software Apache, una licencia de software libre y código abierto a cualquier desarrollador.

En el siguiente apartado se verán las características principales de este gran sistema operativo.

6.1.1. Características principales

Entre sus características, destacan:

- Framework de aplicaciones. Permite el remplazo y la reutilización de componentes.
- Navegador integrado. Basado en el motor Open Source Webkit.
- SQlite. Base de datos relacional que se integra directamente en las aplicaciones.
- Multimedia. Soporte para un gran listado de formatos comunes de audio, vídeo e imágenes planas.
- Máquina virtual Dalvik. Base de llamadas de instancias similar a Java.

6.1.2. Arquitectura

La arquitectura interna de Android se distribuye en un sistema de capas formada básicamente por los siguientes componentes:

- Aplicaciones. Este nivel contiene, tanto las incluidas por defecto de Android como aquellas que el usuario ha ido añadiendo posteriormente, ya sean de terceros o de

propio desarrollo. Todas estas aplicaciones utilizan los servicios, API y librerías de los niveles inferiores.

- Framework de Aplicaciones. Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación desarrollada para Android, utilizan el mismo conjunto de API y el mismo framework, representado por este nivel. Entre las API más utilizadas se encuentran:
    - Activity Manager. Controla el ciclo de vida de las aplicaciones.
    - Window Manager. Gestiona las ventanas de las aplicaciones.
    - Content Provider. Permite a cualquier aplicación compartir los datos con las demás aplicaciones.
    - Location Manager. Posibilita la obtención de información de localización y posicionamiento.
- Librerías. La siguiente capa se corresponde con las librerías utilizadas por Android. Éstas proporcionan a Android la mayor parte de sus capacidades más características. Junto con el núcleo Linux, estas librerías componen el corazón de Android. Se podrán ver algunas de ellas en la ilustración que sigue a continuación.
- Runtime Android. Se sitúa en el mismo nivel que las librerías Android. Lo constituyen las Core Libraries, que son librerías con multitud de clases Java y la máquina virtual Dalvik.
- Núcleo Linux. Se usa como capa de abstracción para el hardware de los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo hardware, lo primero que debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel.

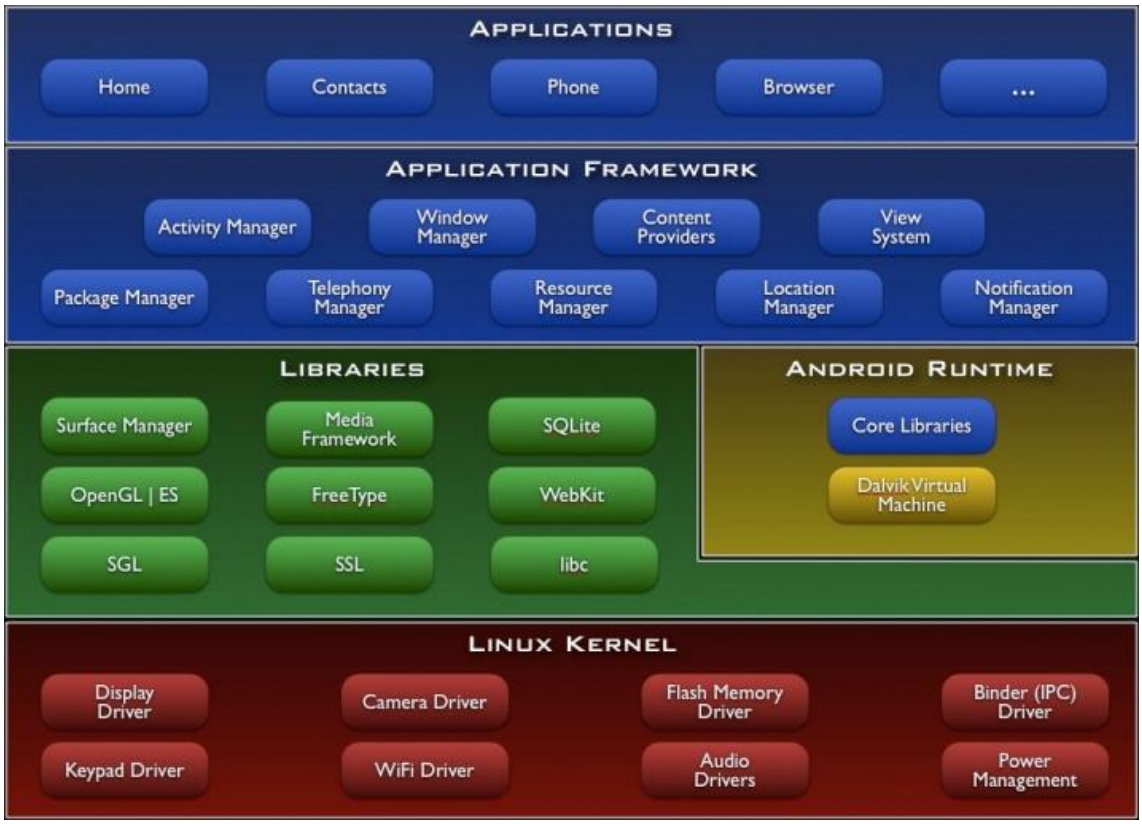


Ilustración 6.1

6.1.3. Android Studio

Existen varios entornos de programación (en adelante IDE) en los que se puede programar para la plataforma Android. Los más conocidos son Android Studio y Eclipse.

Para este proyecto se ha optado por Android Studio y más adelante se explicarán los motivos.

Android Studio es el IDE oficial para el desarrollo de aplicaciones Android. Está basado en IntelliJ IDEA. Sus características principales son:

- Sistema de construcción flexible basado en Gradle.
- Múltiples variantes de construcción y generación de archivos .apk.
- Plantillas de código para proporcionar construcciones de apps comunes de forma rápida y sencilla.
- Editor de layouts muy enriquecido, posibilidad de drag&drop con los elementos de la interfaz.
- Soporte de funcionamiento y compatibilidad de versiones.
- ProGuard como sistema de seguridad anti-decompilación y firma de aplicaciones.
- Integración con la nube de Google.
- Y mucho más.

La alternativa a Android Studio que encontramos es Eclipse. Es un IDE más maduro que Android Studio (aunque este está basado en IntelliJ, destacado entorno de programación Java), sin embargo, abarca muchos más lenguajes de programación y plataformas. Esto presenta una ventaja a favor de Android Studio, que enfoca todo su desempeño en una tarea específica.

Para este proyecto se ha usado Android Studio por algunas razones:

- Es el recomendado por Android, optimizado para esta plataforma.
- Utilización de Gradle**.**
    - Facilita el reciclaje de código y recursos.
    - Gestiona las dependencias de forma cómoda y potente (basado en Maven).
    - Permite compilar desde línea de comandos.
    - Hace realmente fácil la tarea de crear distintas versiones de aplicación.
- Recibe numerosas actualizaciones apoyadas en la gran comunidad Android.

6.1.4. Alternativas

Existen gran cantidad de alternativas en cuanto a sistemas operativos optimizados para dispositivos móviles se refiere, como pueden ser iOS (Apple), Windows Phone (Microsoft), BlackBerry OS o Symbian.

Entonces, ¿por qué Android?

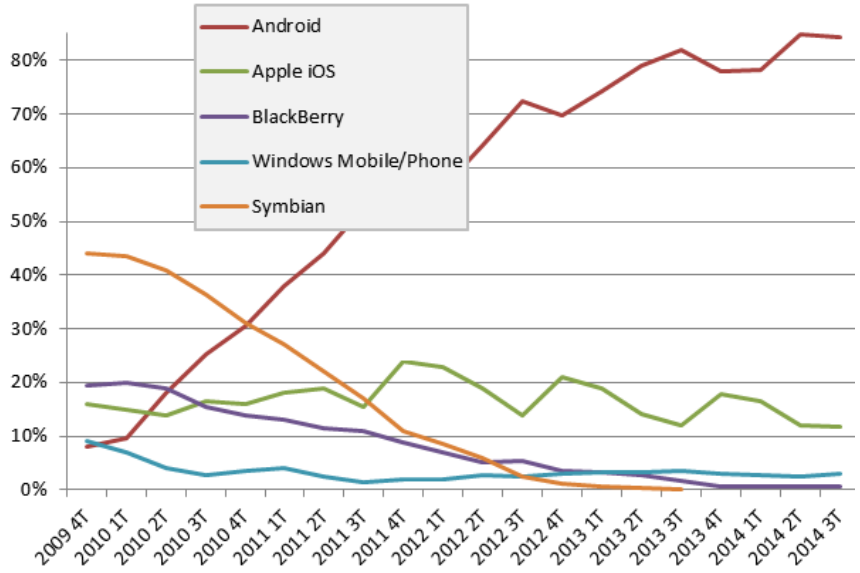


Gráfico 6.1

El gráfico 6.1 muestra la cuota de mercado de los principales sistemas operativos para dispositivos móviles. Como vemos, Android supera el 80% de la cuota, siendo su principal competidor el software de Apple, que se encuentra más de un 60% por debajo.

Esto puede deberse tal vez a la gran variedad de dispositivos ofertados que cuentan con Android como sistema operativo, que van desde las gamas más bajas a los denominados flagship (buques insignia) de las compañías.

	Apple iOS 8	**Android 5.0**	**Windows Phone 8**	**BlackBerry 10**	**Firefox OS 2.2**
Compañía	Apple	Open Handset Alliance	Microsoft	BlackBerry	Mozilla Foundation
Núcleo del SO	Mac OS X	Linux	Windows NT	QNX	Linux
Licencia de *software*	Propietaria	Libre y abierto	Propietaria	Propietaria	Libre y abierto
Año de lanzamiento	2007	2008	2010	1999	2013
Fabricante único	Sí	No	No	Sí	No
Variedad de dispositivos	Modelo único	Muy alta	Media	Baja	Muy baja
Soporte memoria externa	No	Sí	Sí	Sí	Sí
Motor del navegador web	WebKit	WebKit/ Chromium (Blink)	Trident	WebKit	WebKit
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry World	Firefox Marketplace
Número de aplicaciones	800.000 (marzo 2013)	800.000 (marzo 2013)	130.000 (enero 2013)	100.000 (enero 2013)	¿?
Coste publicar	$99 / año	$25 una vez	$99 / año	Sin coste	Sin coste
Otras tiendas sin supervisión	No	Si	No	Si	Si
Familia CPU soportada	ARM	ARM, MIPS, x86	ARM	ARM	ARM, x86
Soporte 64 bits	Si	Si	No	No	No
Máquina virtual	No	Dalvik / ART	.net	No	Navegador Web
Lenguaje de programación	Objective-C, C++	Java, C++	C#, Visual Basic, C++	C, C++, Java	HTML5, CSS, JavaScript
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows	Windows, Mac	Windows, Mac, Linux
Multiusuario	No	Si	No	No	No
Modo invitado	Si	Si	No	No	No

Tabla 6.1

Por si con la aplastante cuota de mercado que posee Android no bastara, la decisión de realizar este proyecto sobre Android viene de la oportunidad de trabajar con software libre y poder contar con una amplia comunidad de desarrolladores.

6.2. Servicios Web

Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones desarrolladas sobre diferentes plataformas pueden utilizar estos servicios para intercambiar datos a través de Internet. Los proveedores de estos servicios los ofrecen como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la web.

Proporcionan interoperabilidad y extensibilidad entre las aplicaciones y realizan operaciones complejas, liberando así de carga de trabajo a estas aplicaciones.

6.2.1. Rest / RESTful

Las famosas APIs que publican muchos de los sitios web actualmente no son más que servicios web de este tipo, aunque en la mayoría de los casos cuentan con medidas de seguridad adicionales como OAuth o similares.

REST se asienta sobre el protocolo HTTP como mecanismo de transporte entre cliente y servidor. En cuanto al formato de los datos transmitidos, a diferencia de otros servicios como SOAP, no se impone ninguno en concreto, aunque lo más habitual es intercambiar la información en formato XML o JSON.

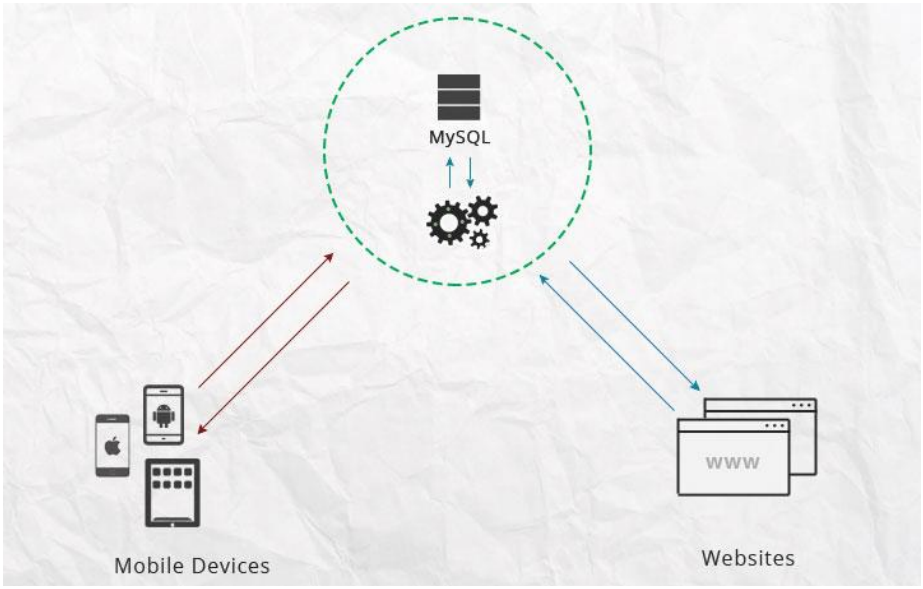


Ilustración 6.2

Los principios de RESTful son:

- Protocolo cliente-servidor sin estado. Cada mensaje HTTP contiene toda la información necesaria para comprender la petición.
- Conjunto de operaciones definidas. Las más importantes son POST, GET, PUT y DELETE.
- Sintaxis universal para identificar recursos. En este protocolo, adquiere gran importancia el concepto de recurso como elemento de información accesible mediante un identificador global (URI).
- Uso de hipermedios. La representación de la información en un sistema REST suele estar en formato XML o JSON. Como resultado de esto, es posible navegar de un recurso REST a otros, simplemente siguiendo enlaces.

Sólo se considerará servicio RESTful cuando cumpla estas condiciones sin excepción posible.

En este proyecto se ha seleccionado esta tecnología por la sencillez que implica. Además, el autor del trabajo ya conocía el lenguaje JSON, por lo que los posibles retrasos en el desarrollo de este apartado han sido mínimos.

6.2.2. Ventajas

Entre las ventajas principales que aporta la utilización de servicios web se encuentran:

- Interoperabilidad entre aplicaciones software, independientemente de sus propiedades o plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto. Esto facilita el acceso a su contenido y el entendimiento de su funcionamiento.
- Proveer servicios integrados fácilmente desde diferentes lugares geográficos combinando servicios y software de diferentes compañías.

6.2.3. Inconvenientes

También tiene algunos puntos débiles, entre los que se encuentran:

- No puede compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA.
- Su rendimiento es bajo comparado con otros modelos de computación distribuida. Es uno de los inconvenientes asociados al formato de texto, ya que entre los objetivos de XML no se encuentra la eficacia del procesamiento.
- Puede esquivar barreras de seguridad basadas en firewall ya que se apoya en HTTP.

7. IMPLEMENTACIÓN DE LA APLICACIÓN

La implementación constituye la etapa en la cual se pretende convertir el trabajo realizado en las etapas anteriores en algo tangible. Para ello, se utiliza un lenguaje de programación (en este caso Java) y algunas herramientas como pueden ser:

- Entorno de programación o IDE. En este proyecto se ha optado por el Software recomendado por Google, Android Studio.

- Máquina virtual de simulación o emulador. A lo largo del proyecto se han usado varios métodos de simulación para realizar las pruebas con la aplicación. Se detallará en los siguientes apartados.
- Control de versiones Git. Herramienta de gestión de cambios en el proyecto.

A lo largo de este apartado, veremos las diferentes partes que conforman el sistema y la forma que tienen para coordinarse. Para entender estas partes por separado, conviene dar una visión global de la actuación normal del sistema.

Básicamente se pueden encontrar dos grandes estructuras: el cliente (en este caso será la aplicación Android) y el servidor (al cual se accederá a través de una API adaptada). El funcionamiento es simple y se fundamenta en dos operaciones REST:

- **GET.** El cliente realiza una petición reclamando unos datos determinados. Se realizarán varias peticiones independientes (reclamando datos de clientes, productos, presupuestos, eliminados, etc.) para facilitar la modularidad del sistema.
- **PUT.** El cliente realiza una petición enviando datos al servidor. El servidor actuará actualizando la información en su base de datos. Como desde la aplicación solo se pueden crear / editar clientes y presupuestos, son las dos únicas operaciones posibles de este tipo.

El sistema da prioridad siempre a los datos obtenidos de la aplicación móvil. Por ejemplo, si un presupuesto ha sido editado en las dos plataformas, prevalecerán los campos que la aplicación móvil envía mediante su petición PUT en el momento de la sincronización, aunque es un caso que no se da con demasiada frecuencia. En las versiones venideras se mejorará el sistema de tratamiento de este tipo de conflictos, pero de momento y a petición de la empresa, el objetivo principal era salir cuanto antes al mercado con una primera versión funcional.

Una vez realizada esta aclaración de funcionamiento, se explicarán las diferentes partes que intervienen.

7.1. Java (lenguaje de programación)

Para programar una aplicación en Android, es fundamental conocer el lenguaje que se utiliza para ello. En este apartado se va a hablar de Java como lenguaje, lo que lo diferencia de otros lenguajes de programación y las ventajas que ofrece al desarrollador.

Java es un lenguaje de programación de propósito general, concurrente y orientado a objetos, diseñado para evitar las famosas dependencias de implementación en la medida

posible. Su propósito principal fue permitir que los desarrolladores escriban el programa una única vez y lo puedan ejecutar en cualquier dispositivo ("*write once, run anywhere"*).

Fue desarrollado inicialmente por *Sun Microsystems*. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos.

Se creó con cinco objetivos principales:

- Usar el paradigma de la programación orientada a objetos.
- Permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Incluir por defecto soporte para trabajo en red.
- Ejecutar código en sistemas remotos de forma segura.
- Fácil de usar, tomando lo mejor de otros lenguajes orientados a objetos como C++.

Entre sus ventajas ofrecidas al programador podemos destacar:

- Java realiza comprobación estricta de tipos durante la compilación, evitando problemas como el desbordamiento de pila.
- Todas las instancias de clase se crean con el operador *new()*, de manera que un *recolector de basura* se encarga de liberar la memoria ocupada por los objetos que ya no están referenciados. La *máquina virtual de Java* gestiona la memoria de forma dinámica.
- Una fuente común de errores en otros lenguajes proviene del uso de punteros. En Java se eliminan, el acceso a las instancias de clase se hace a través de *referencias.*
- El programador está obligado a tratar las posibles excepciones que se produzcan en tiempo de ejecución.
- Preparado para la programación concurrente, sin necesidad de utilizar ninguna librería.
- Java posee un *gestor de seguridad* con el que poder restringir el acceso a los recursos del sistema.

7.2. Aplicación android (cliente)

7.2.1. Conceptos

Ya se ha explicado la estructura de Android como sistema operativo, sin embargo, falta detallar cuáles son las entidades principales que intervienen en el desarrollo de una aplicación para dicho sistema. Es fundamental tener claros los siguientes conceptos:

- **Actividad.**
- **Fragmento.**

- **Adaptador.**
- **Vista / Layout.**

7.2.1.1. Actividad

Las aplicaciones de Android funcionan principalmente por actividades. Una actividad es una clase java que suele corresponderse con la lógica y la interfaz que un usuario puede encontrar en una pantalla determinada. Hay casos especiales, mediante los cuales se pretende abstraer a las actividades alejándolas de las vistas de la aplicación, aunque este no es el caso que nos ocupa.

Una parte de las actividades que cabe mencionar es su ciclo de vida. Es el conjunto de etapas por las que pasa una actividad desde el momento que es lanzada hasta que finaliza su ejecución. La forma más fácil de comprenderlo es mediante el siguiente gráfico.

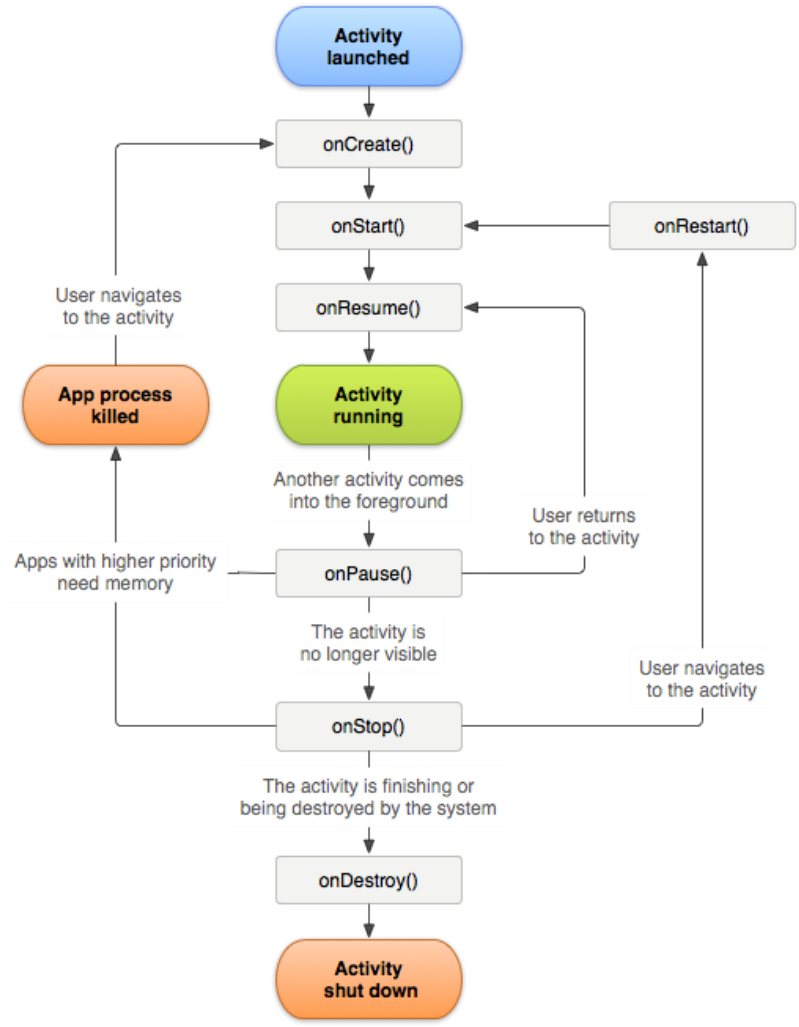


Ilustración 7.1

7.2.1.2. Fragmento

Un fragmento representa una tarea o una porción de la interfaz de usuario dentro de una actividad. Se pueden combinar múltiples fragmentos dentro de una actividad sencilla o reutilizar un mismo fragmento desde varias actividades. Se puede entender como una sección modular de una actividad, con su ciclo de vida propio, recibiendo sus eventos de entrada y actuando en consecuencia. Se puede añadir o quitar de una actividad mientras ésta está corriendo.

Su utilidad potencial reside en la realización de aplicaciones multi-plataforma, destinadas a distintos tipos de pantalla, donde a partir de un determinado tamaño se combinan fragmentos distintos en una misma vista. El ejemplo más común es el conocido Lista-Detalle.

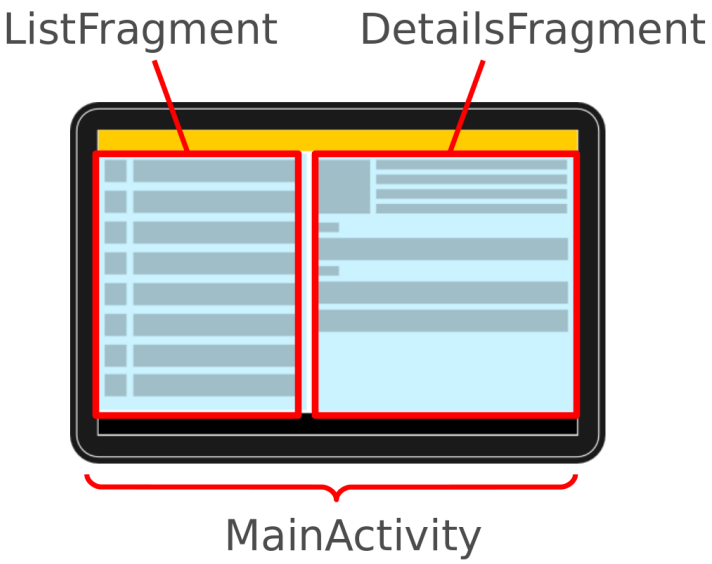


Ilustración 7.2

En este proyecto, los fragmentos han sido trabajados para aprovechar su fluidez y versatilidad. Por ejemplo, se han realizado pantallas usando un ViewPagerAdapter (adaptador de vistas completas o páginas), aprovechando una única actividad con varios fragmentos, optimizando así los accesos a la base de datos. El ejemplo mencionado se puede ver en el alta de un nuevo cliente o al formalizar el guardado de un presupuesto (ya sea nuevo o editado).

7.2.1.3. Adaptador

Los adaptadores son clases java que actúan como puentes entre las vistas y las actividades o fragmentos. Proporcionan una capa de abstracción entre ellas, traduciéndose esto en un desarrollo más modular. Además, aportan ciertas ventajas a la hora de

representar determinadas vistas, por ejemplo, las listas (listviews). Gracias a la correcta implementación de estos adaptadores, se permite filtrar por determinados campos, consiguiendo resultados muy potentes.

7.2.1.4. Vista / Layout

Representan el bloque básico para la construcción de componentes de interfaz de usuario. Una vista ocupa un área rectangular de a pantalla y responde a las acciones del usuario. Las vistas pueden ser agrupadas y organizadas de diferentes formas. La estructura se decide mediante la programación de un fichero XML que representa a este conjunto de vistas. Esta estructura podrá ser cargada como una pantalla completa de la interfaz o como parte de ella.

7.2.2. Estructura del proyecto Android

Para detallar algo más profundamente la estructura que tiene el proyecto en Android Studio, a continuación se realizan una serie de indicaciones, acompañado de la jerarquía de directorios que sigue el proyecto.

La estructura principal del proyecto tiene una apariencia similar a esta:

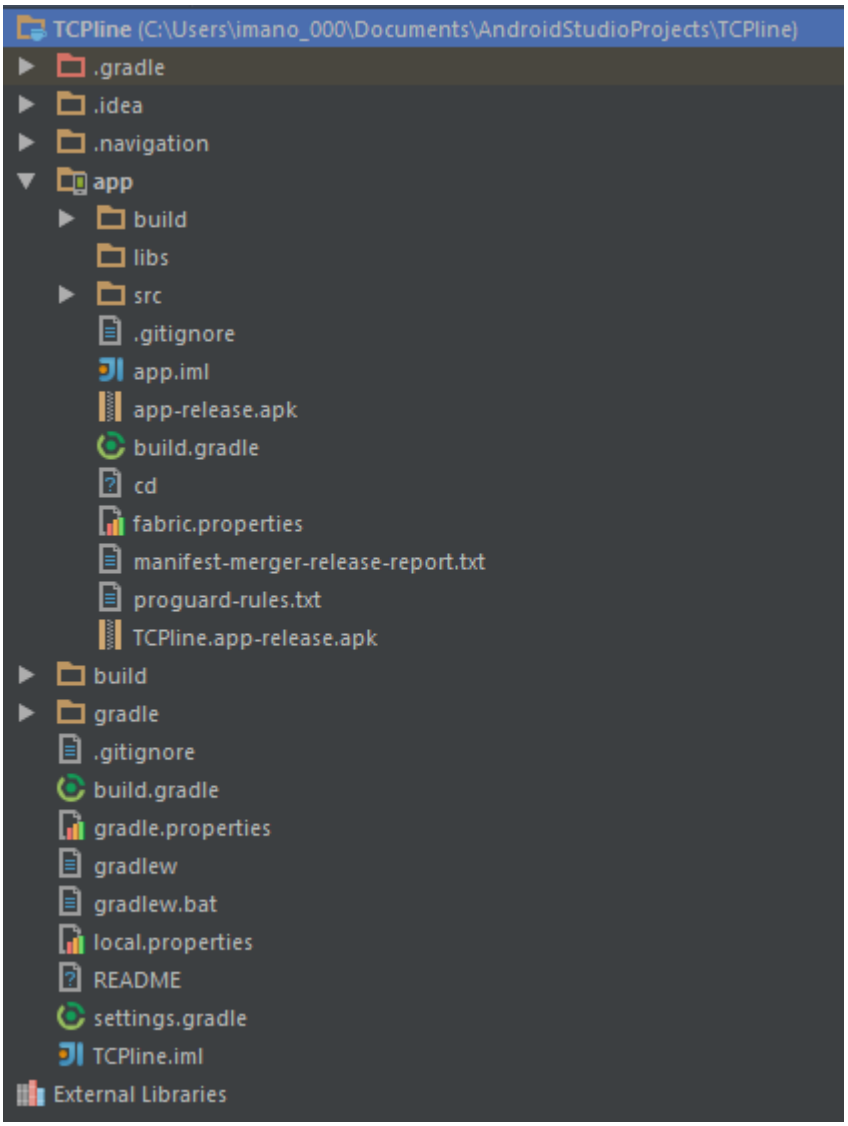


Ilustración 7.3

Estos directorios son los que se crean por defecto en cualquier proyecto nuevo Android. Hay variables muy importantes como la carpeta *build* y *gradle*, que se encargan de las funciones principales del versionado y construcción del archivo apk, sin embargo, la atención va a recaer sobre los ficheros que se encuentran en "*app -> src"*.

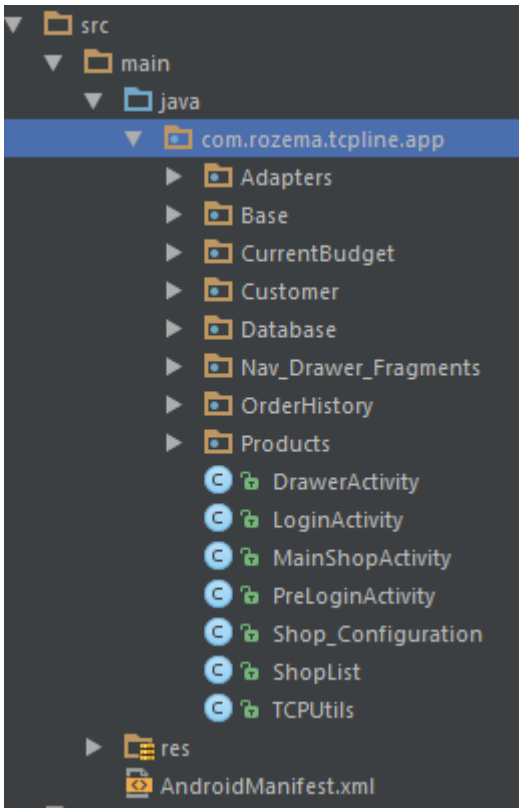


Ilustración 7.4

Una vez desplegado el directorio mencionado, podemos ver una serie de carpetas y ficheros cuya explicación puede resultar más que interesante para el desarrollo de este proyecto. Hay tantas formas de organización del código como programadores. Aunque hay determinados patrones como el MVC (Model-View-Controller) o el MVVM (Model-View-View-Model) que optimizan la abstracción y modularización de una aplicación, éstos requieren conocimientos muy amplios.

Debido al gran número de clases con las que cuenta el proyecto, se explicarán los diferentes directorios en los que se han organizado y qué tipo de clases contienen:

- Directorio Adapters. Contiene todos los adaptadores necesarios para mostrar las vistas en cada una de las pantallas.
- Directorio Base. Contiene las clases más básicas que se utilizarán a la hora de sacar la información del modelo de datos y trabajarla de forma cómoda, usándola en arrays y listas. Un posible ejemplo es la clase “Producto” o “Presupuesto”.
- Directorio Customer. Contiene las actividades y fragmentos que se utilizan en el alta y edición de los clientes.
- Directorio Database. Contiene las clases java que se encargan de construir y acceder a las bases de datos con las que cuenta el sistema: la general y las de las diferentes tiendas.

- Directorio Nav_Drawer_Fragments. Contiene los diferentes fragmentos que son llamados al pulsar sobre cualquiera de las opciones del menú desplegable. Todos ellos se adjuntan al mismo activity, que dependerá de la pantalla desde la que se realice este despliegue.
- Directorio OrderHistory. Contiene las clases y fragmentos necesarios para la edición sobre el historial de presupuestos.
- Directorio Products. Compuesto por las clases y fragmentos que participan en la navegación entre productos, a través de las categorías hasta la ficha de producto.
- Fichero DrawerActivity. Es una actividad de la cuál heredarán cada una de las actividades desde las que se puede desplegar el menú principal. Aporta las funcionalidades necesarias para realizar este despliegue y controla los eventos una vez desplegado.
- Fichero LoginActivity. Es una actividad que controla el inicio de sesión a la aplicación a través de peticiones al servidor.
- MainShopActivity. Actividad principal de la tienda. Sobre ella se cargarán todos los fragmentos del menú principal.
- PreLoginActivity. Actividad que se encarga de comprobar si la sesión está abierta para mostrar el formulario de inicio de sesión o el listado de tiendas. No posee ninguna vista asociada.
- Shop_Configuration. Actividad que controla la vista y la lógica asociada a la configuración de una tienda.
- TCPUtils. Librería propia que contiene métodos genéricos usados frecuentemente desde diversos puntos de la aplicación. Un ejemplo de método de esta librería puede ser “capitalize”, que se encarga de mostrar un String formateándolo de forma que la primera letra será mayúscula y las demás minúsculas, entre otros.

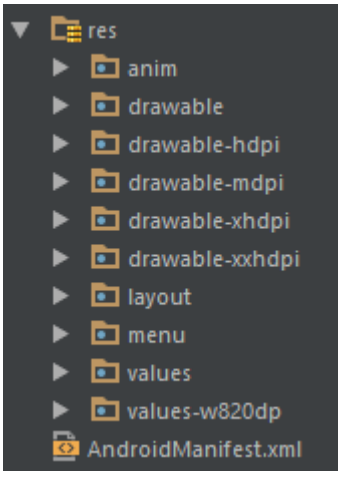


Ilustración 7.5

- Directorio res. Esta carpeta contiene todos los recursos necesarios a los que accederán las clases java (actividades, fragmentos y adaptadores).
- Directorio anim. Contiene los ficheros XML que indican las posibles animaciones de la aplicación. Por ejemplo, al finalizar una actividad o la transición de una a otra.
- Directorios drawable. Contiene los ficheros “drawable” que contendrán los recursos que tienen que pintar las vistas, iconos, etc. Se utilizan diferentes directorios para introducir las imágenes en diferentes calidades de modo que los soporten la mayor cantidad posible de dispositivos, tengan la densidad de píxeles por pulgada que tengan.
- Directorio layout. Contiene los ficheros XML con los que se construyen las vistas.
- Directorios values. Contiene los ficheros de estilo, strings y dimensiones para homogeneizar lo máximo posible las diferentes pantallas de la aplicación.
- AndroidManifest. Es un fichero de configuración donde podemos aplicar las preferencias básicas de la aplicación. Todas las actividades que forman parte de la aplicación deben ser declaradas aquí para el correcto funcionamiento.

7.3. Aplicación web (servidor)

Una vez explicada la estructura de la aplicación Android y los conceptos más importantes de este sistema operativo, el siguiente paso es describir a grandes rasgos cómo está implementada la parte del servidor.

En la parte del servidor, se cuenta con una serie de “tiendas” cuya infraestructura está basada en Prestashop que ha sufrido modificaciones para adaptarse a las funcionalidades requeridas por el sistema. Más que una tienda, se busca una plataforma donde se pueda mostrar un catálogo y realizar presupuestos de forma rápida, sencilla y compacta.

Existe una plantilla sobre la que se van gestionando todos los cambios funcionales de la aplicación web. Cada cierto periodo de tiempo, todas las tiendas actualizan su estructura basándose en esta plantilla, dejando intacta la base de datos que las forman.

Cada una de las tiendas tiene implementada una API, que se actualiza de la misma forma mencionada anteriormente, pudiéndose acceder así a los datos de cada tienda a través de ella. Básicamente la API está preparada para recibir los dos tipos de peticiones que se han descrito en la introducción del apartado 6, GET o PUT.

El método **GET** siempre se hará mediante parametrización de la URL, es decir, la url de la API llevará ciertos atributos que determinarán que datos ha de devolver. En las

próximas versiones, se espera mejorar este sistema añadiendo un token de seguridad. Un ejemplo de petición a la API actualmente podría ser:

http://tcpline.sytes.net/tutienda/api/index.php?tabla=clientes&id_empleado=4&date_from=2015-06-01+11:04:24

Con esta petición devolvería todos los clientes que han sufrido modificaciones desde el 1 de junio de 2015 a las 11:04:24 hasta el momento actual. Dependiendo del empleado (o comercial) tendrá asignados una serie de clientes, por lo que la respuesta variará también en función de su id_empleado. Una posible respuesta obtenida sería:

```
{
    "status": "success",
    "error_msg": "",
    "data": [
        {
            "id_customer": 57,
            "date_upd": "2015-06-03 19:35:24",
            "date_add": "2015-05-11 18:06:56",
            "tipo": "MAN",
            "nombre_completo": "Francisco Gimeno Hernandez",
            "apodo": "Fran",
            "contacto": {
                "nombre": "Fran",
                "tfn": "616657604"
            },
            "condiciones": "Aplicar Maximos Dtos.",
            "notes": "",
            "usuario": "contabilidad11@rozema.es",
            "passwd": "1c8cc71d60ad28671337888c8edabae5",
            "editable": 1,
            "direccion_facturacion": [
                {
                    "id_address": 103,
                    "date_add": "0000-00-00 00:00:00",
                    "date_upd": "2015-05-11 18:13:02",
                    "direccion": "Urb. Pla de Marco S/N",
                    "alias": "Mi direccion",
                    "cp": "46169",
                    "localidad": "Olocau",
                    "provincia": "Valencia",
                    "id_pais": 6,
                    "contacto": {
                        "nombre": "Fran 11",
                        "email": "contabilidad11@rozema.es",
                        "tfn": "616657604"
                    },
                    "observ": "Enviar una copia de la Ftra. Por correo Ordinario",
                    "tipo": "MAN",
                    "nombre_completo": "Francisco Gimeno Hernandez 11",
                    "apodo": "Fran 11",
                    "dni": "48382307J"
                }
            ],
            "direccion_entrega": [
                {
                    "id_address": 104,
                    "date_add": "0000-00-00 00:00:00",
                    "date_upd": "2015-05-11 18:12:21",
                    "direccion": "Urb. Pla de Marco S/N",
                    "alias": "Mi casa",
                    "cp": "46169",
                    "localidad": "Olocau",
                    "provincia": "Valencia",
                    "id_pais": 6,
                    "contacto": {
                        "nombre": "Fran 11",
                        "email": "contabilidad11@rozema.es",
                        "tfn": "616657604"
                    },
                    "observ": "Entregar por las maÃ±anas",
                    "tipo": "MAN",
                    "nombre_completo": "Francisco Gimeno Hernandez 11",
                    "apodo": "Fran 11"
                }
            ]
        }
    ]
}
```

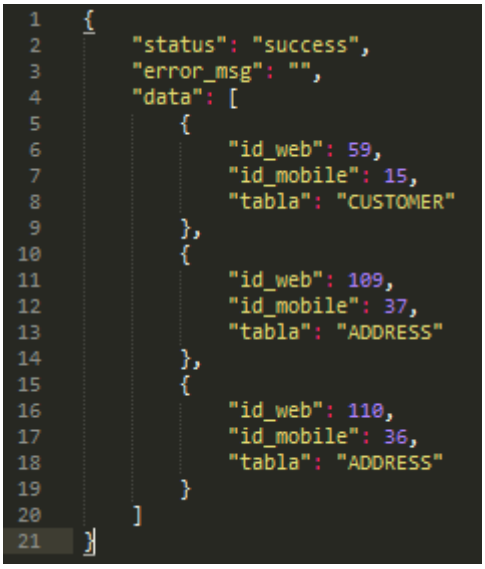

Ilustración 7.6

El método **PUT** sin embargo, maneja datos formateados en JSON. Por tanto, la API estará preparada para recibir grandes volúmenes de información estructurada, de donde

obtendrá todos los campos cuyo nombre se ha acordado previamente. Un ejemplo de petición PUT:

Ilustración 7.7

Este JSON solicita el alta de un cliente nuevo, con una dirección de entrega y una dirección de facturación. Ante una petición de este tipo y si todo ha salido correctamente, el sistema respondería con algo similar a esto:



```
{
    "status": "success",
    "error_msg": "",
    "data": [
        {
            "id_web": 59,
            "id_mobile": 15,
            "tabla": "CUSTOMER"
        },
        {
            "id_web": 109,
            "id_mobile": 37,
            "tabla": "ADDRESS"
        },
        {
            "id_web": 110,
            "id_mobile": 36,
            "tabla": "ADDRESS"
        }
    ]
}
```


Ilustración 7.8

Básicamente es una tabla de correspondencias entre los IDs que se han registrado en la web y los que se tienen en la base de datos del dispositivo móvil. En este caso, nos encontramos una correspondencia de tipo *cliente* y dos de tipo *dirección.*

La forma de generar esta información desde el cliente se ha convertido en una tarea realmente sencilla gracias a la librería *Gson* de Google. Esta librería implementa un método que convierte la información contenida en una instancia de una clase cualquiera a formato JSON.

Antes de realizar cada una de las peticiones, se cotejaran los credenciales de usuario con el servidor. En caso de que se le hayan revocado los permisos para entrar a una determinada tienda o haya sido bloqueado por algún motivo, la aplicación echará al usuario a la pantalla de log-in. Al intentar loguearse, el usuario recibirá un mensaje con el motivo de su bloqueo.

7.3.1. Modo OFFline

Ya se ha descrito la forma de intercambiar información entre el cliente y el servidor, sin embargo, ¿qué se hace con esta información?

Uno de los requisitos principales que se exponían para este proyecto era la necesidad de poder trabajar siempre con los datos, con o sin conexión a Internet. La explicación es la siguiente: imagina un comercial que va a países de todo tipo a vender materiales, posiblemente en muchos de estos países no exista una buena conexión a la red que sea estable, por lo que la forma de tener siempre disponibles los datos es descargando esta información previamente, almacenándola en el dispositivo portátil que el comercial lleve (Smartphone o táblet).

Por tanto, la pregunta queda resuelta. El tratamiento que recibirán los datos al ser recibidos por el dispositivo será almacenarlos en la base de datos de la tienda para así poder acceder a ellos cuando sea necesario.

Cuando el usuario recupere una buena conectividad, tendrá la oportunidad de sincronizar los datos con el servidor, haciendo así que los datos estén actualizados en ambas partes.

8. VALIDACIÓN DE LA APLICACIÓN

Una vez terminada la aplicación, sólo faltaría la etapa de pruebas de la misma. En este capítulo se revisará el cumplimiento de los requisitos funcionales y no funcionales que se veían en el apartado 3.1 del capítulo de Análisis de requisitos.

Requisitos funcionales

- Existe un control de acceso a la aplicación, que devuelve el conjunto de tiendas a las que el usuario tiene permitido el acceso. ✓ Conseguido. El acceso será controlado a través de un panel de control externo a la APP, controlado por la empresa.

Ilustración 8.1

- Se puede acceder a las diferentes tiendas a las que un determinado comercial tenga acceso. ✓ Conseguido

Ilustración 8.2

- Se puede gestionar la información de todos los clientes asignados a un comercial así como añadir nuevos clientes al sistema dentro de una tienda. ✓ Conseguido

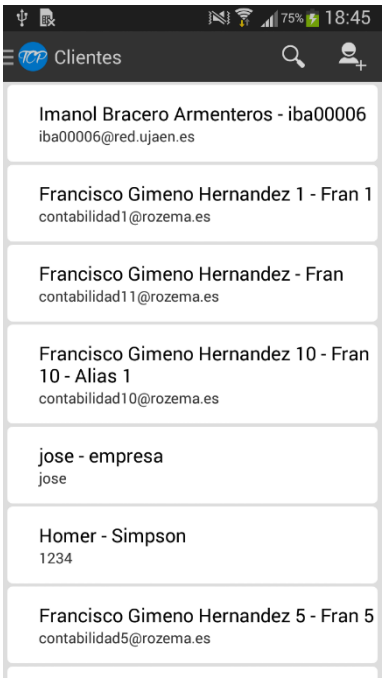


Ilustración 8.3

- Se puede visualizar la información de todos los productos y que estén dados de alta en el sistema dentro de una tienda ordenada en categorías. ✓ Conseguido
- Se proporciona una forma rápida de acceso a los productos, a través de un buscador. ✓ Conseguido. Este buscador encontrará los productos filtrando por una palabra que esté contenida en cualquiera de sus campos. Además, se proporcionarán filtros por disponibilidad y precios.

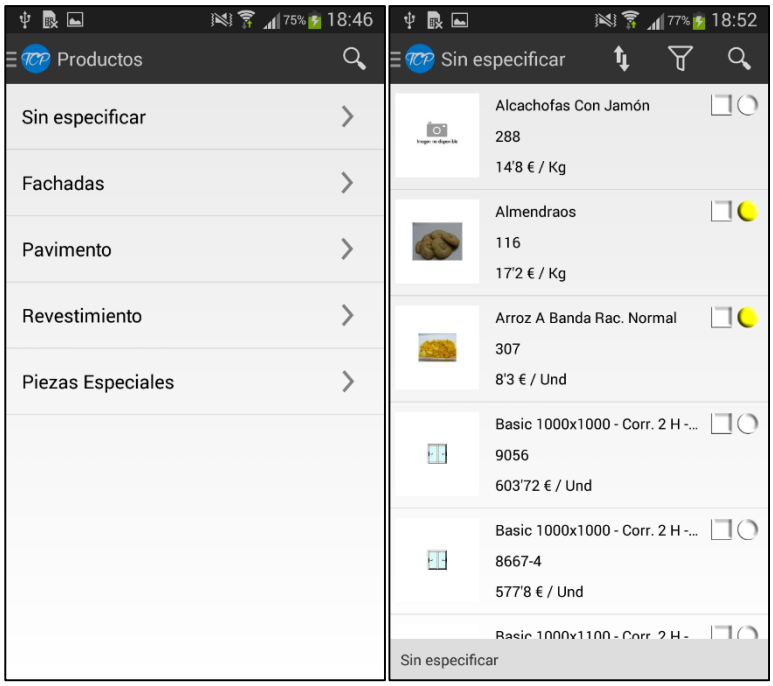


Ilustración 8.4

- Se muestran las diferentes marcas creadas en cada tienda, así como los productos que pertenecen a ellas. ✓ Conseguido

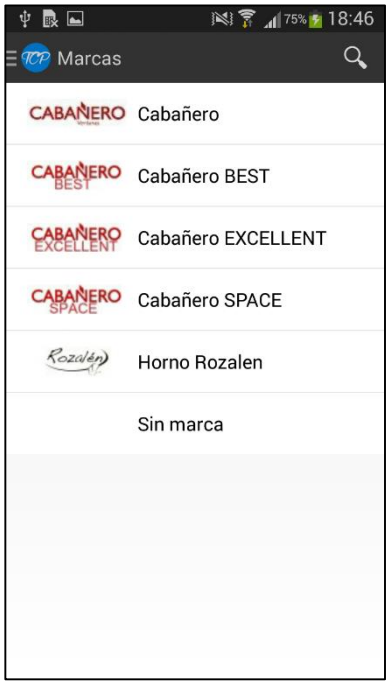


Ilustración 8.5

- Se puede realizar un presupuesto a partir de los productos introducidos, con la posibilidad de añadir descuentos y diferentes cantidades. ✓ Conseguido

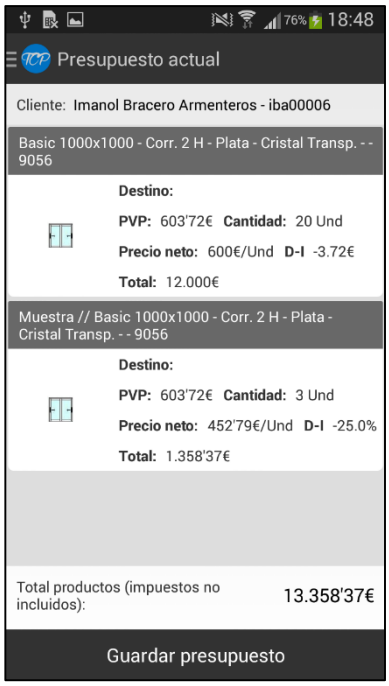


Ilustración 8.6

- Puece accederse a un histórico de los presupuestos realizados en una tienda determinada y realizar cambios sobre ellos. ✓ Conseguido

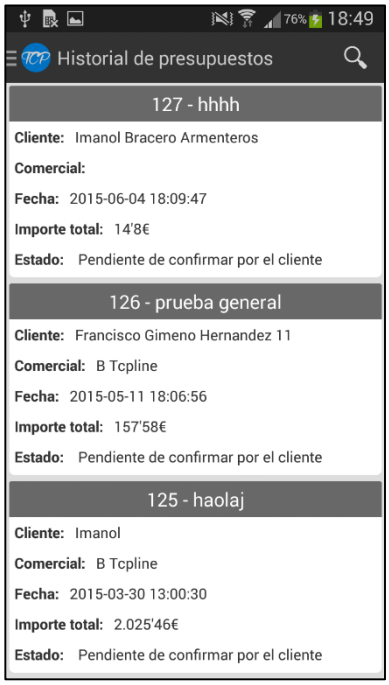


Ilustración 8.7

- Se puede cambiar la moneda en la que se visualizan los precios. ✓ Conseguido

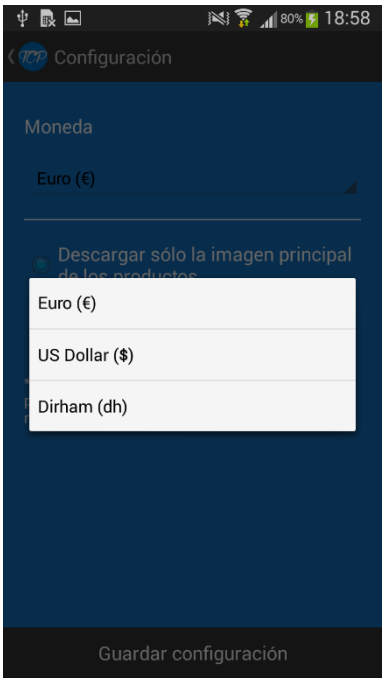


Ilustración 8.8

- La APP puede sincronizarse con el servidor cada vez que el comercial crea oportuno y mantener así la información actualizada en ambos lados de la conexión. ✓ Conseguido. Podrá hacerlo pulsando sobre el botón “Sincronizar datos”.

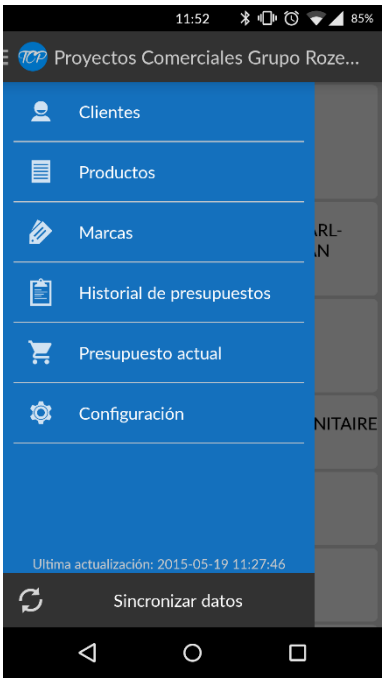


Ilustración 8.9

Requisitos no funcionales

- Al tratarse de un sistema que puede trabajar de forma offline, la descarga de datos inicial conlleva un coste de memoria y tiempo que ha de ser óptimo.

  El tiempo de respuesta para sincronizar las tiendas varía con el volumen de datos que éstas tengan desactualizados. Se muestran al usuario qué datos se están descargando en cada momento. ✓ Conseguido

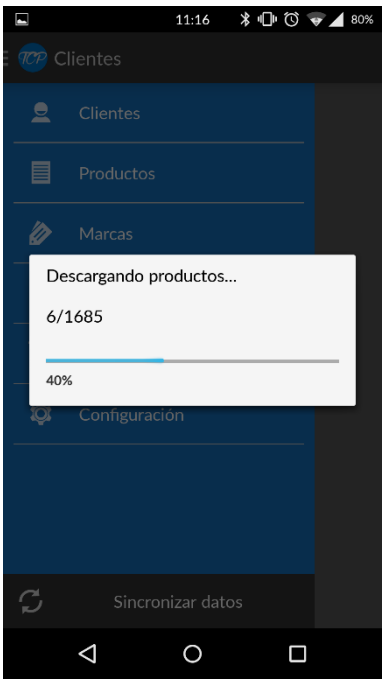


Ilustración 8.10

- La aplicación será utilizada en pantallas que van de 4” a 10” aproximadamente. Está optimizada para su usabilidad en estos tamaños. ✓ Conseguido. Se han realizado pruebas con dispositivos de diferente tamaño: el más pequeño ha sido un Google Nexus S (4”) y el más grande una tablet Samsung Galaxy Note 10.1 (10”).
- Es una aplicación robusta, capaz de recuperarse rápidamente en caso de fallo. ✓ Conseguido. Mediante manejo de excepciones, se ha conseguido una aplicación robusta en la mayoría de los casos. Para posibles fallos no controlados, la aplicación se apoya en una herramienta de control de errores (Crashlytics) que notifica directamente al equipo de desarrollo dónde ha sido el error.
- Los datos que se manejan en la aplicación son privados, pudiendo usarse sólo desde la aplicación. Una vez que esta se desinstale, los datos serán borrados. ✓ Conseguido. Esto se ha conseguido mediante el conocido MODE_PRIVATE de Android, guardando todos los datos e imágenes de forma segura y privada.
- Se comprueba que el comercial que use la aplicación conserva los permisos necesarios para seguir usándola en cada una de las actualizaciones.

✓ Conseguido. Cada vez que se realiza una conexión al servidor, se envían los datos de log-in del comercial, guardados en preferencias de la APP. En caso de haber retirado los permisos, se expulsará al usuario a la pantalla de log-in, notificando las razones del caso.

9. CONCLUSIONES

Cuando se habla de tecnología, no se puede negar la cantidad de ventajas y herramientas de todo tipo que ésta aporta, siempre que se use con moderación. En la actualidad, sobra mencionar la importancia que tiene la tecnología en la vida de las personas y como facilita el día a día. Con este proyecto, se pretendían aprovechar algunas de estas herramientas para optimizar el trabajo de determinados trabajadores dentro de sus empresas. Aunque más enfocado a la labor comercial, la aplicación que se ha desarrollado puede usarse por todo tipo de profesionales.

En este punto, me encantaría enfatizar en la suerte que he tenido de poder trabajar en un proyecto real para la empresa Proyectos Comerciales Grupo Rozema S.L., dándome la oportunidad de realizar esta primera inmersión en un proyecto importante sin haber terminado aún la carrera.

Además, me gustaría señalar las facilidades aportadas por esta empresa a la hora de realizar los distintos Sprints dentro de la metodología SCRUM, ayudando enormemente en las distintas iteraciones que se han llevado a cabo durante el desarrollo.

El desarrollo del proyecto ha sido algo extenso, debido a las diferentes modificaciones que han ido surgiendo, así como la aportación de nuevas funcionalidades. Sin embargo, el desarrollo no ha sido un camino demasiado difícil, aunque sí reconozco que he tenido algunos puntos débiles que he conseguido reforzar gracias a este trabajo.

Mi experiencia con la plataforma Android no había sido muy profunda, sin embargo, conocía los fundamentos para lograr una rápida incorporación al ritmo normal de trabajo dentro de la empresa. Cabe destacar la gran cantidad de información aprendida que ha terminado convirtiéndose en experiencia propia, así como el procedimiento de subida al Play Store de Google.

En cuanto al lenguaje de programación, JAVA, no he tenido grandes dificultades, debido al amplio conocimiento aportado en algunas asignaturas de la titulación universitaria. Sí que es cierto que he fortalecido los conocimientos en herencia, polimorfismo e implementación de interfaces.

Tras finalizar el proyecto, han quedado pendientes algunas mejoras o funcionalidades extra que se irán añadiendo en posteriores versiones. Entre estas mejoras podemos encontrar:

- Mejorar apariencia en la ficha de Producto.
- Añadir datos bancarios en la ficha de cliente.
- Añadir datos de transporte en la ficha de presupuesto.
- Añadir pestaña para productos relacionados.
- Añadir lector de QR para direccionar a productos.
- Añadir bloc de notas genérico para un cliente.
- Posibilidad de generar varios tipos de informes en PDF y enviarlos por correo a los clientes.
- Adaptar el diseño de la interfaz a Material Design.
- Optimizar la seguridad de las peticiones con protocolo OAuth2 o similar.
- Optimizar el sistema de conflictos a la hora de la sincronización.

Para finalizar este bloque de conclusiones, me gustaría señalar la importancia de desarrollar un proyecto de este tipo de principio a fin, pasando por todas las etapas de forma íntegra.

BIBLIOGRAFÍA

[1] "*España, líder europeo en el uso de teléfonos inteligentes, con un 81 por ciento*". Recuperado el 2 de febrero de 2015.

http://www.eldiario.es/turing/Espana-europeo-telefonos-inteligentes-ciento_0_348215609.html

[2] "*Metodologías de desarrollo*". Recuperado el 12 de febrero de 2015.

Apuntes propios asignatura *Fundamentos de Ingeniería de Software (2º curso Grado en Ingeniería Informática, EPS Jaén).*

[3] "*SCRUM*". Recuperado el 24 de febrero de 2015.

http://es.wikipedia.org/wiki/Scrum

[4] "*El gran libro de Android Avanzado*", Jesús Tomás Gironés, 2013. Última consulta 5 de mayo de 2015.

[5] "*Android Studio Overview*". Recuperado el 7 de abril de 2015.

http://developer.android.com/tools/studio/index.html

[6] "*Android vs otras plataformas*". Recuperado 29 de febrero de 2015.

http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/98-comparativa-con-otras-plataformas

[7] REST / RESTful. Última consulta 21 de abril de 2015.

http://es.wikipedia.org/wiki/Representational_State_Transfer

[8] Conceptos Android. Última consulta 14 de abril de 2015.

http://developer.android.com/guide/components/index.html

[9] Realización de casos de uso. Última consulta 29 de marzo de 2015.

Apuntes propios asignatura *Fundamentos de Ingeniería de Software (2º curso Grado en Ingeniería Informática, EPS Jaén).*

[10] Planificación. Última consulta 13 de marzo de 2015.

Apuntes propios asignatura *Gestión y Control de Proyectos Informáticos (3º curso Grado en Ingeniería Informática, EPS Jaén).*

[11] "*Experto en Java*" – Antonio Jesús Galiano Sánchez, 2013

ANEXO I: MANUAL DE USUARIO DE TCPLINE APP

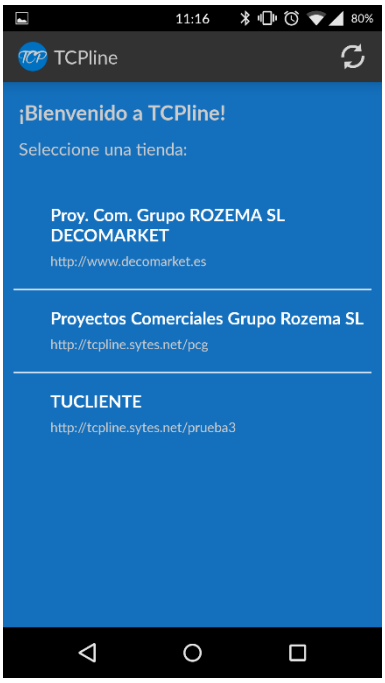
1. Inicio de sesión

Para utilizar la aplicación de TCPline usted debe poseer una licencia de uso, la cual se podrá adquirir poniéndose en contacto vía e-mail a informatica1@rozema.es o llamando al número de teléfono +34 961 516 464.

Una vez obtenida la licencia, el equipo de TCPline le proporcionará un usuario y contraseña con la que podrá entrar a nuestra aplicación desde la pantalla principal (mostrada a continuación).

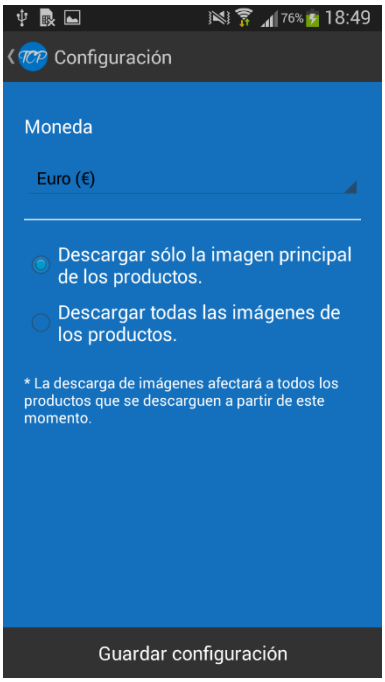
2. Listado de tiendas

Una vez realizado el inicio de sesión con éxito, encontrará un listado de tiendas a las que se le ha proporcionado acceso previamente. Al entrar en esta pantalla se recargarán automáticamente, aunque podrá hacerlo de forma manual pulsando sobre el botón .Pulse sobre una tienda para seleccionarla.



3. Configuración de la tienda

Una vez seleccionada la tienda a la que quiere entrar, deberá establecer unos parámetros por defecto. Éstos se podrán modificar una vez abierta la tienda desde el elemento de menú *"Configuración"*.



En esta pantalla podrá elegir la moneda en la que aparecerán los precios de todos los productos de la tienda. Los presupuestos se almacenarán con la moneda que tenía seleccionada en el momento de su guardado.

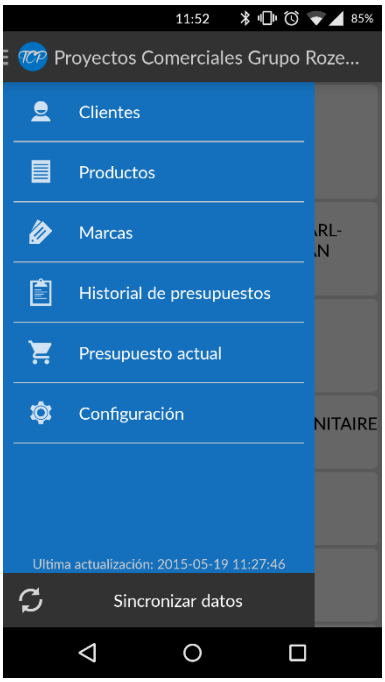
También se deberá elegir la cantidad de imágenes por producto que vamos a descargar. Las opciones son las siguientes:

- *"Descargar sólo la imagen principal de los productos".* Sólo se descargará la imagen principal de cada producto. Es la opción más rápida.
- *"Descargar todas las imágenes de los productos".* En caso de que las tuvieran, se descargarán todas las imágenes de los productos. Consume algo más de recursos.

En caso de cambiar esta opción de forma posterior a una sincronización, sólo afectará a los nuevos productos que se descarguen en sincronizaciones posteriores.

4. Menú principal

Una vez seleccionada la configuración inicial de la tienda, se encontrará un menú desplegable lateralmente, que tendrá una apariencia tal que así:



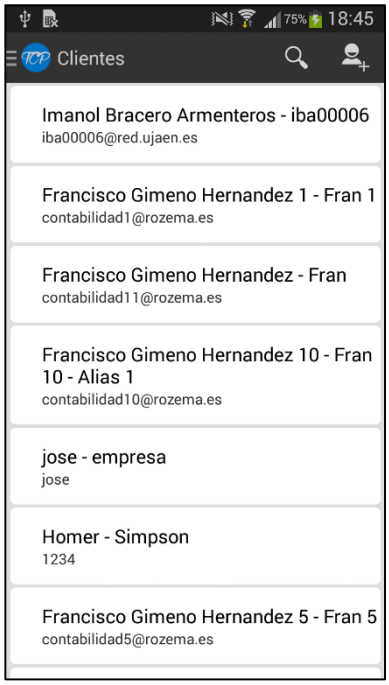
Cada una de las opciones a las que podemos acceder será explicada en los apartados siguientes.

En este menú se encontrará también la información referente a la última sincronización, así como el botón de sincronización.

Este menú estará accesible desde la mayoría de las pantallas de la aplicación, no estando disponible en las que el usuario requiere una mayor atención, como por ejemplo, guardar un presupuesto o añadir un cliente nuevo.

5. Clientes

Accediendo a la opción *"Clientes"* desde el menú principal se encontrará la siguiente pantalla:



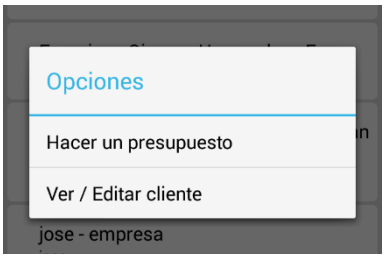
Encontraremos un listado con los clientes. En cada línea, encontraremos los siguientes datos:

- En la primera línea: nombre completo – alias.
- En la segunda línea: usuario / e-mail.

En la barra superior encontraremos dos iconos:

- Buscador. Filtrará por los campos que están visibles en este listado.
- Añadir cliente. Abrirá una nueva pantalla para rellenar un formulario de alta de cliente. Se especificará en el siguiente apartado.

Mediante una pulsación larga sobre algún cliente se abrirá el siguiente diálogo:

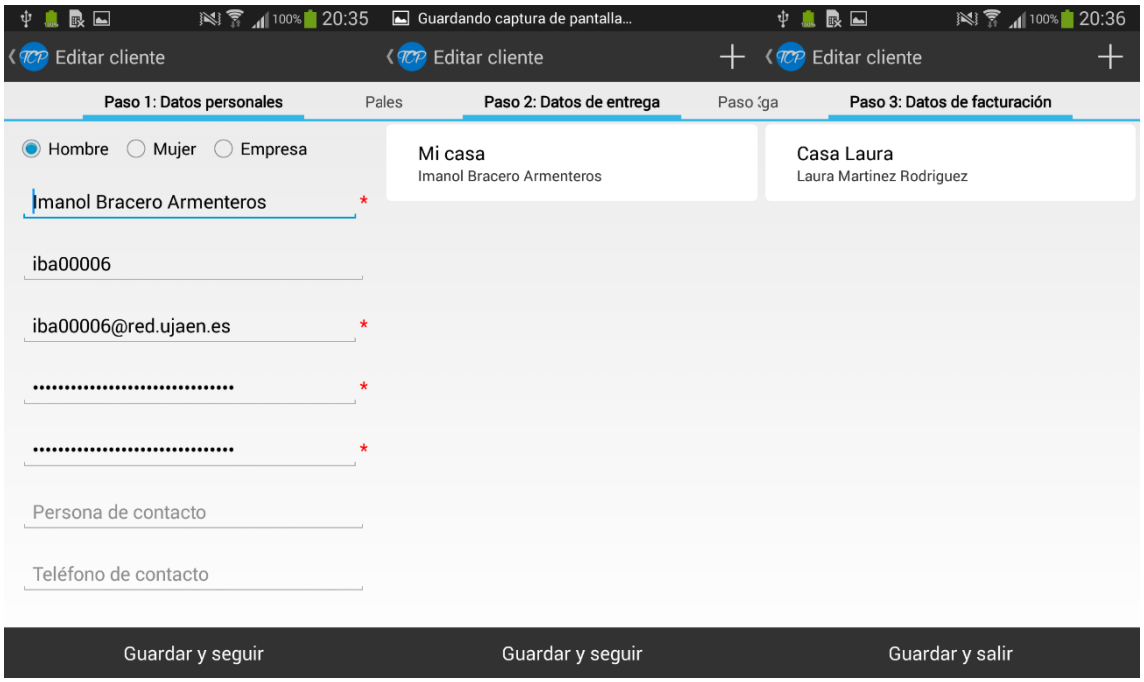


En este diálogo se ofrecen dos opciones:

- Hacer un presupuesto. Esta opción seleccionará al cliente para realizarle un presupuesto. En caso de que el presupuesto abierto ya tenga productos añadidos, el cliente seleccionado será vinculado a este presupuesto ya comenzado. En caso de que ya haya seleccionado un cliente para el presupuesto actual, se mostrará un mensaje de advertencia para confirmar estas reasignación.
- Ver / Editar cliente. Esta opción abrirá el formulario con los datos rellenos del cliente, para verlos o modificarlos.

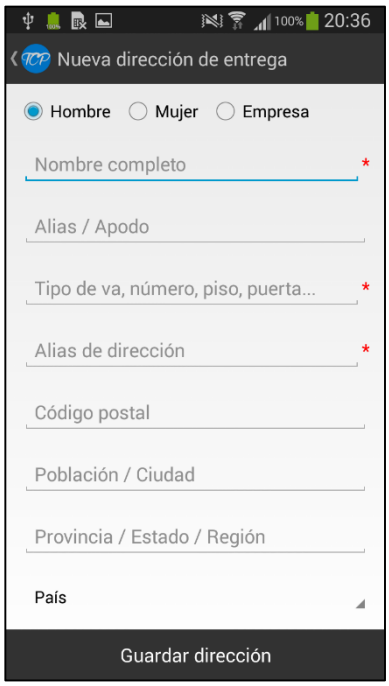
5.1. Añadir / Editar cliente

- Al pulsar sobre *"Añadir cliente"* () o sobre la opción *"Ver / Editar cliente"*, se abrirá un formulario con la siguiente apariencia.



En estas pantallas se podrá cumplimentar toda la información referente al cliente, así como sus direcciones de entrega y facturación. Los campos que tienen * son campos obligatorios, el resto serán opcionales.

Se podrán añadir tantas direcciones como se quieran, pulsando sobre el icono . Al tocar sobre él, se abrirá el siguiente formulario.



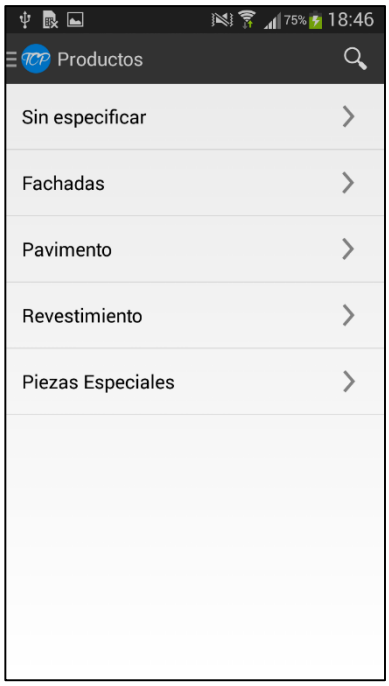
6. Productos

Se podrá acceder a cada una de las fichas de los productos a través de dos de las opciones del menú principal:

- A través de *"Productos"* se visualizará un árbol de categorías por el que se podrá navegar.
- A través de *"Marcas"* se visualizará un listado de marcas que contendrán todos los productos referentes a cada marca.

6.1. Categorías

A través de la opción *"Productos"* se accederá a un árbol de categorías o familias de productos. Se visualizará una pantalla similar a la siguiente:

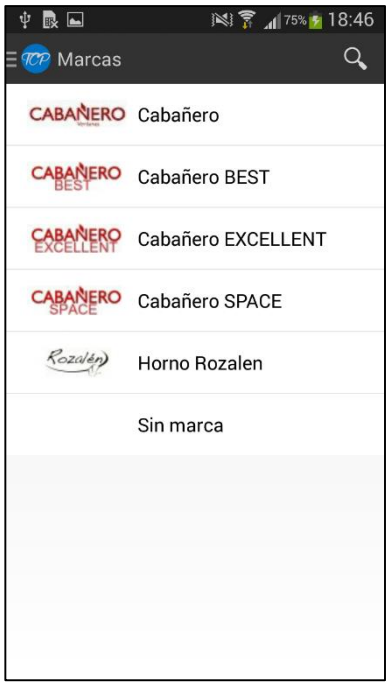


Desde aquí, se podrá acceder a la ficha de un producto de dos formas:

- Navegando entre las distintas familias.
- A través del buscador (). Este buscador encontrará los productos entre toda la base de datos filtrando por ID, referencia, nombre de producto o descripciones. Si introducimos una cadena de caracteres que se encuentre en cualquiera de estos campos, este quedará visible en el listado.

6.2. Marcas

Al pulsar sobre la opción *"Marcas"* del menú principal, se verá una pantalla como esta:

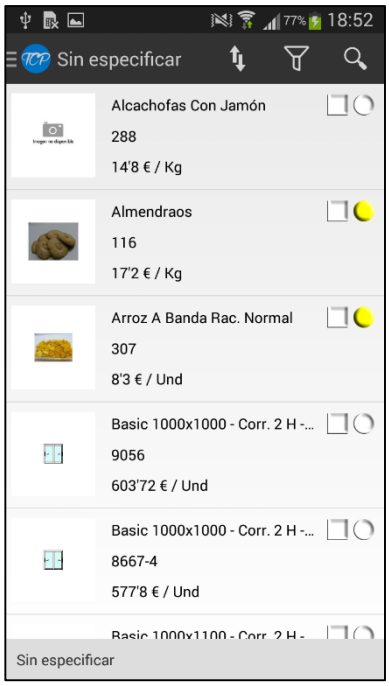


Al pulsar sobre cualquiera de las marcas visibles, se verán todos los productos de la marca en cuestión.

También se incorpora un buscador en esta pantalla, que filtrará únicamente por los nombres de marcas.

6.3. Listado de productos

Una vez pulsado sobre una de las opciones vistas anteriormente (*"Marcas" o "Categorías")* se verá el listado de productos. En este encontraremos la siguiente información:



Para cada línea del listado visto, encontraremos en orden:

- Nombre del producto
- Referencia del producto
- PVP (moneda seleccionada / Unidad de venta)
- Disponibilidad tienda y proveedor ( ). Las disponibilidades seguirán un código de color que se corresponde con el stock existente del producto. Los colores / estados posibles son:
    - Verde: en stock.
    - Amarillo: pocas unidades.
    - Rojo: sin stock, con previsión de reponer.
    - Negro: sin stock, descatalogado.
    - Blanco: stock desconocido.

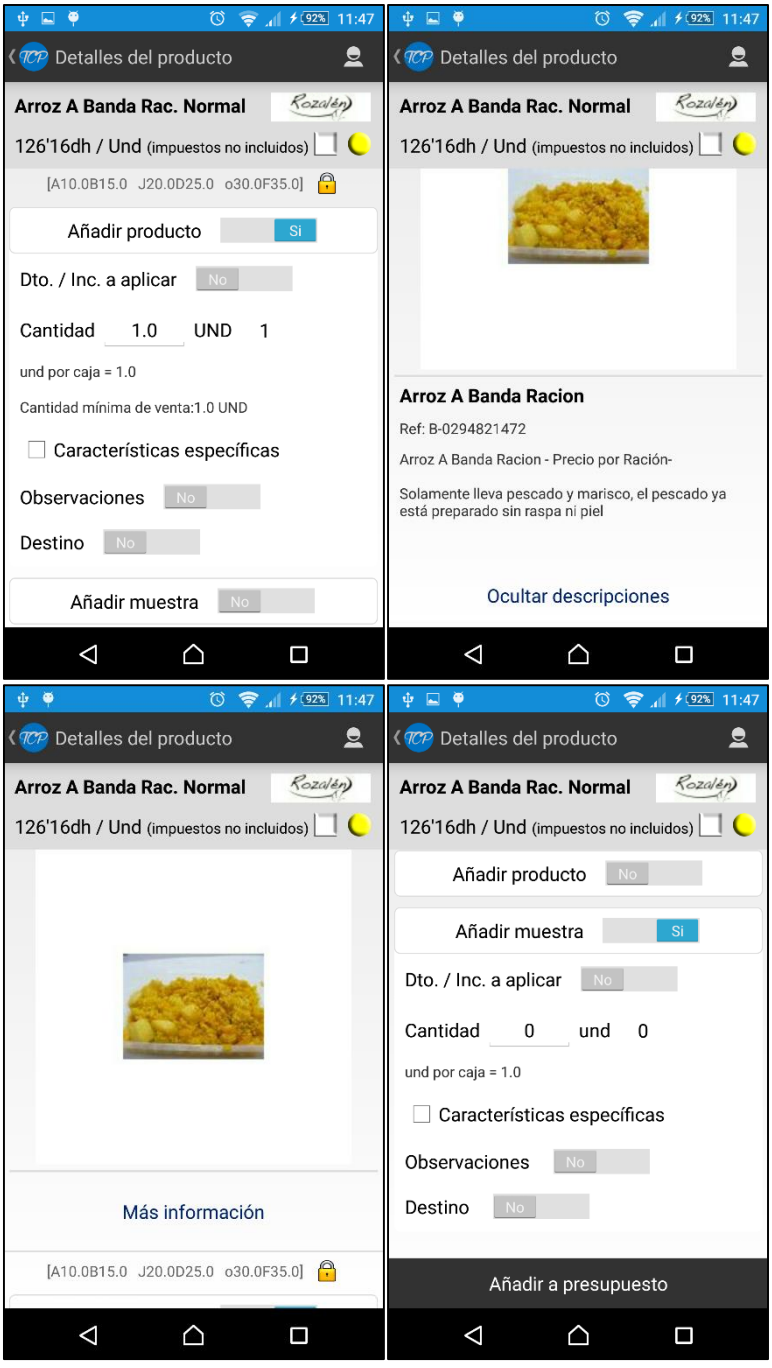
En la barra superior se podrá observar el nombre de la categoría / marca a la que pertenecen los productos y los siguientes iconos:

- Ordenación. Se podrá ordenar el listado obtenido por nombre (ascendente o descendente) y precio (ascendente o descendente)
- Filtro por disponibilidad. Reducir el listado mostrando únicamente los productos con la disponibilidad seleccionada.
- Buscador. Filtra entre los resultados que contengan los caracteres introducidos.

Al pulsar sobre cualquiera de estos productos, se accederá a la ficha de producto.

6.4. Ficha de producto

En la ficha de producto podremos encontrar toda la información relacionada con el producto que se ha seleccionado en el listado anterior.



Los datos que podemos ver en estas pantallas son:

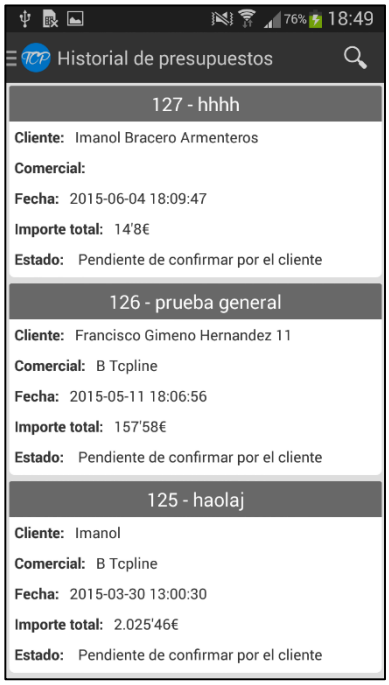
- Condiciones del cliente seleccionado para el presupuesto
- Nombre del producto
- Imagen de la marca
- Precio del producto (moneda / unidad de venta).

- Disponibilidad en tienda y proveedor.
- Imágenes del producto. Se mostrará primero la imagen principal. Se podrán ver las demás (en caso de que las tenga) deslizando el dedo sobre la imagen de derecha a izquierda.
- Botón *"Más información"*. Al pulsar sobre este botón, se harán visibles los campos de las descripciones.
- Descuentos. Códigos guía para el comercial, mediante estos códigos sabrá que descuento / incremento podrá realizar en este producto.
- Bloque añadir producto. En este bloque el usuario podrá aplicar descuentos / incrementos por cantidad o porcentaje. Aquí se seleccionará la cantidad de producto a añadir al presupuesto, posibles observaciones al añadir éste y un destino (los informes se agruparán por destinos; por ejemplo en un proyecto de una reforma, podrá haber productos agrupados por baño, cocina, comedor, etc.).
- Bloque añadir muestra. Igual que el anterior. Al añadir una línea de presupuesto como muestra quedará indicado en el título de la línea.

Una vez rellenados los campos, al pulsar sobre *"Añadir al presupuesto"* se añadirán los productos y/o muestras seleccionadas y se llevará a la pantalla *"Presupuesto actual"*.

7. Historial de presupuestos

Si se pulsa sobre a opción *"Historial de presupuestos"*, se verá una pantalla como esta:



Para cada presupuesto mostrado, se verá la información siguiente:

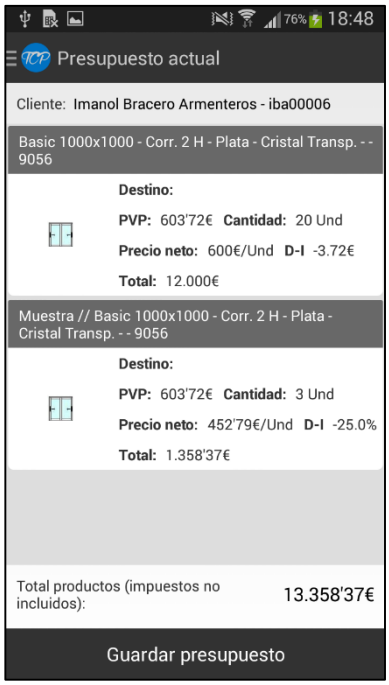
- Cliente
- Comercial asignado
- Fecha en la que se guardó el presupuesto
- Importe total del presupuesto en la moneda en que se guardó
- Estado en que se encuentra el presupuesto actualmente

Si realizamos una pulsación larga sobre cualquier presupuesto podremos realizar dos acciones:

- Ver / Editar presupuesto. Con esta opción podremos ver / editar el listado de productos del presupuesto, cambiar la dirección de entrega o facturación y los datos generales del presupuesto. (Se verá con más detalle en el apartado "*Presupuesto actual – Guardar presupuesto"*).
- Cambiar estado. Se mostrará un diálogo mediante el que se podrá cambiar el estado del presupuesto. (p.e. Anulado por el cliente)

8. Presupuesto actual

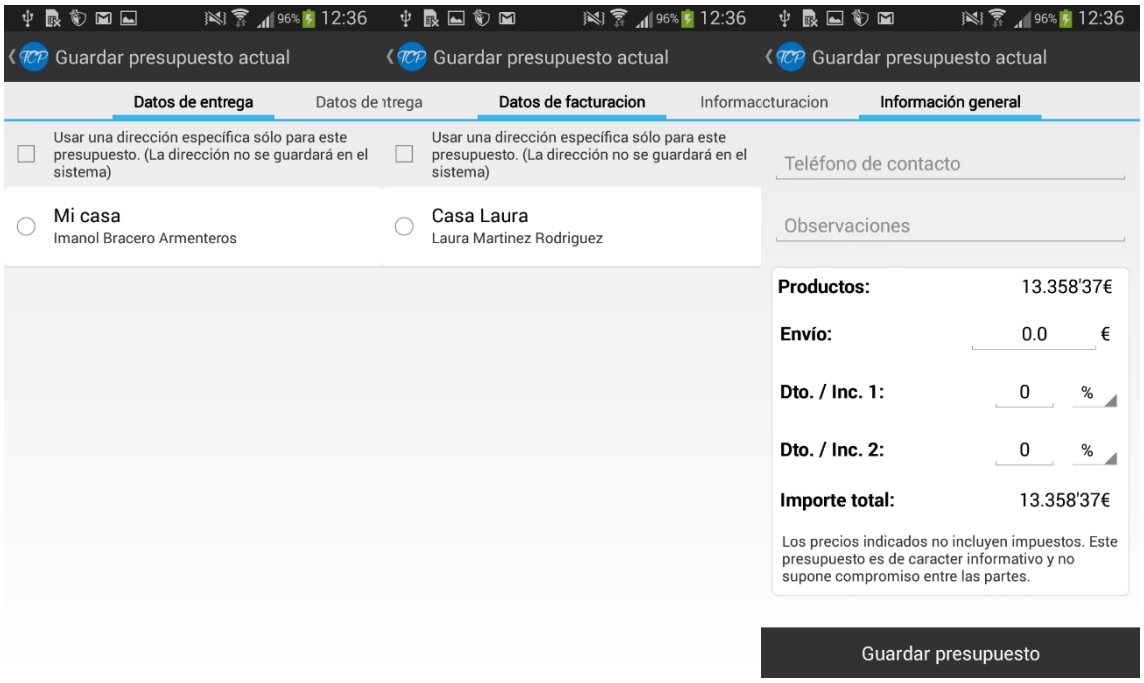
Esta pantalla se corresponde con los productos que han sido añadidos hasta el momento. La apariencia de esta pantalla es similar a esta:



Desde esta pantalla podremos:

- Editar una línea añadida (mediante una pulsación larga). Se mostrará un diálogo con los mismos campos que la ficha de producto.
- Guardar el presupuesto actual (mediante una pulsación en el botón *"Guardar presupuesto"*).

Al pulsar sobre este último botón, se verán las siguientes pantallas:



- En la primera pantalla se podrá seleccionar una dirección de entrega existente en la ficha de cliente o bien introducir una dirección específica para este presupuesto (la dirección no se almacenará en la ficha del cliente).
- En la primera pantalla se podrá seleccionar una dirección de facturación existente en la ficha de cliente o bien introducir una dirección específica para este presupuesto (la dirección no se almacenará en la ficha del cliente).
- En la última pantalla se podrán introducir los datos generales del presupuesto.
    - Nombre del proyecto
    - Forma de pago
    - Persona de contacto
    - Teléfono de contacto
    - Observaciones generales del presupuesto
    - Gastos de envío
    - Dos Descuentos / Incrementos generales

Para cerrar el presupuesto y poder verlo en el histórico de presupuestos, se deberá pulsar sobre *"Guardar presupuesto"*.

9. Sincronización de datos

La sincronización de datos forma una parte fundamental en el sistema, por lo que es importante conocer las nociones de su funcionamiento.

9.1. ¿Cuándo se debe sincronizar?

Para sincronizar los datos se debe contar con conexión a Internet (Wi-fi o datos móviles). Cada vez que se editen / añadan presupuestos o clientes, se tienen que sincronizar los datos (si se quiere que éstos estén disponibles en la versión web).

9.2. ¿Qué pasa si se edita un mismo cliente / presupuesto desde la aplicación móvil y la aplicación web?

Los datos de la versión prevalecerán siempre sobre los datos de la versión web, es decir, si un presupuesto ha sido editado desde ambas partes y posteriormente se realiza una sincronización, prevalecerán los datos de la aplicación móvil.