1. Introducción

La motivación de realizar este proyecto ha venido marcada por el aumento del uso de los portales de compraventa y la explotación masiva del uso de las redes sociales en todo el mundo. La comodidad que te ofrece el hecho de poder comprar y vender desde casa con toda fiabilidad y el incremento del uso de las redes sociales han sido unos de los principales motivos por los que hemos decidido realizar este proyecto.

Se trata de facilitar las cosas a un usuario que decida desprenderse de un artículo y quiera sacar un beneficio, o bien, la necesidad por la que valga la pena intentar empezar a buscar mediante las nuevas tecnologías. El usuario que finalmente se decida a realizar una compra o venta desde esta página, tendrá el valor añadido de la fiabilidad que merece poder conocer a una persona en la misma página mediante el componente de red social al que se orienta a partir de una navegación sencilla e intuitiva marcada fuertemente mediante dos estilos diferentes en cuanto a presentación.

A nivel personal, creo que el hecho de crear un entorno, con dos funcionalidades distintas en un mismo lugar, puede abrir un camino de crecimiento personal en cuanto a experiencia y madurez en la creación y desarrollo del software como ingeniero. El reto que me ofrece este proyecto es el de integrar los conocimientos adquiridos en la universidad, con el conocimiento en los diferentes frameworks y lenguajes que se demandan en el mundo laboral. Por nombrar algunos, los más conocidos, Spring Framework, el cual te ofrece la posibilidad de incorporar algunas metodologías de trabajo demandas y la puesta en marcha de patrones de software J2EE, como la inyección de dependencias, IoC (inversión de control) y transaccionalidad. Otro de los frameworks utilizados y con gran demanda es Struts, que te permite implementar el patrón de diseño de software MVC (Modelo Vista Controlador). Y por nombrar uno más Ibatis, un framework que te permite tener relacionados los objetos del dominio con las tablas de la base de datos, lo que te permite desacoplar la parte de base de datos de la aplicación y alojas las consultas SQL (Structured Query Language) en uno o varios ficheros XML (eXtensible Markup Language).

2. Objetivos

El objetivo principal que este proyecto quiere abarcar es la posibilidad de que un usuario sea capaz de comprar, vender y hacer amigos de manera cómoda, rápida y sencilla sin necesidad de tener que estar más tiempo del necesario, si el usuario no lo desea, para completar las tareas básicas para las que se ha desarrollado la aplicación.

Un usuario anónimo, por anónimo se entiende desconocido de la aplicación que bien puede ser que este registrado aunque no identificado, podrá consultar en todo momento los productos que estén a la venta pudiendo realizar búsquedas de los productos mediante un filtro o criterios de búsqueda. También se le permitirá poder consultar un producto. Tendrá visible la opción de vender y de ir a la red social, simplemente para que el usuario vea que la pagina dispone de más funcionalidades pero hay más opciones que no podrá ver si no se identifica o si no lo está, se registra.

En cuanto un usuario se identifica, previo relleno de formulario de alta de la aplicación, podrá acceder a la completa funcionalidad de la aplicación, tanto de la parte de la red social como del portal de compraventa, pudiendo pasar de un entorno a otro gracias a una opción de menú de la barra superior. Las acciones que podrá realizar en la parte de compraventa será a grandes rasgos, comprar, vender, reservar (deberá realizar una petición al dueño para ello), consultar, modificar productos propios, entre otras acciones. En la parte de la red social podrá consultar perfiles de otros usuarios, consultar sus productos (tanto los que dispone de venta como los comprados), consultar sus noticias en el mismo perfil, ver las estadísticas de votaciones de otros usuarios, buscar usuarios, enviar mensajes privados, añadir fotos, ver fotos de otros usuarios, etc.

Como se ve, se realizará un aplicativo completo para las necesidades que pueda tener un usuario y además cabe la posibilidad de que se pueda ampliar el proyecto añadiendo funcionalidades nuevas, ya que puede abarcar un sinfín de funcionalidades para que el usuario se ‘enganche’ a la página web y sea al fin y al cabo, una red social más con otras funcionalidades que otras páginas no tengan.

3. Estudio de mercado

Antes de realizar el análisis de requerimientos, es conveniente realizar un estudio del mercado puesto que ya existen páginas de compraventa y también redes sociales, algunas muy potentes. En este punto, tratamos de hacer el estudio para ver las fortalezas y debilidades que presentan otras páginas para intentar aportar algo novedoso al mercado y poder, así, obtener oportunidades.

Antes de nada, decir que algunas aplicaciones web existentes, están desarrolladas por equipos grandes de trabajo, empresas en las que participa mucha gente, y por lo tanto, tienen una amplia gama de funcionalidades, unas mejores y otras peores, que podremos aprovechar para realizar este proyecto.

Algunas de las páginas que hemos estudiado han sido: www.ebay.es, www.ciao.es y www.telocompro.es

3.1. Web www.ebay.es

Imagen 1. Web Ebay

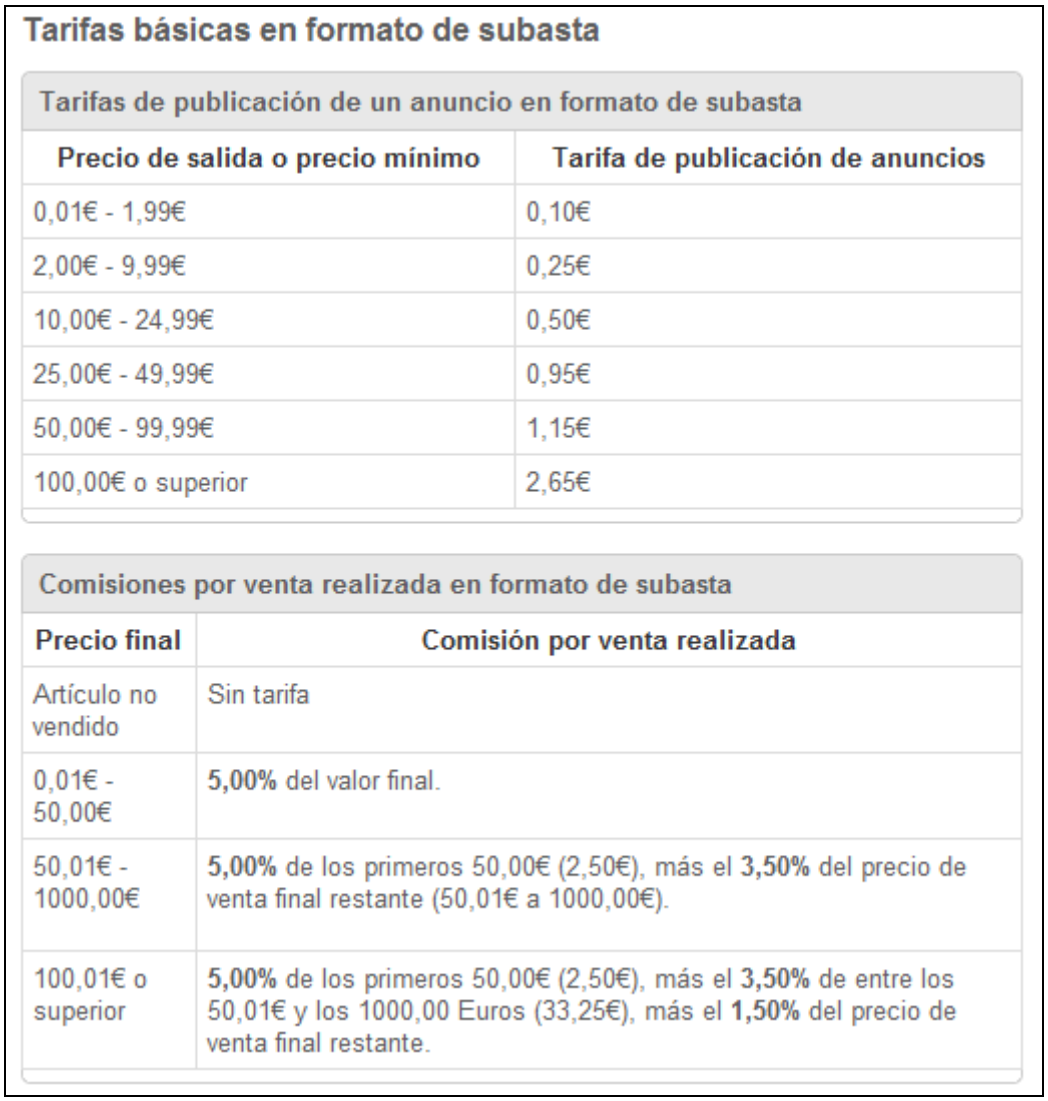
En la página de inicio tienes la posibilidad de escoger dos opciones, si ir a ebay clásico o bien ir a la página dedicada a los anuncios. Nosotros contemplaremos la parte de ebay clásico que es la que nos afecta en este caso ya que es la que permite comprar y vender artículos.

A la hora de registrarte te pide los datos típicos, como el nombre, apellidos, fecha de nacimiento, etc. En el siguiente paso te ofrece la posibilidad de crear una cuenta pay-pal i enlazarla con tu cuenta en ebay para los pagos a través de VISA.

Los menús están divididos por menú y submenú. En la categoría de menú podemos escoger entre las opciones de comprar, vender, comunidad y perfil. Dentro de cada opción accederemos a los submenús de esas categorías. El menú que puede influirnos a la hora de enfocar la parte de la red social de nuestro proyecto, encontramos que lo que contiene ese menú o categoría, es la parte de los foros y artículos de ayuda. Y por otro lado está la parte del perfil del usuario en el que puedes comunicarte con otros usuarios mediante mensajes privados, añadir a favoritos y votar/comentar sobre las ventas con este usuario.

A la hora de vender, los pasos a seguir serian los siguientes:

a) Introducir los datos del artículo.
b) Cuenta en la que recibiremos el pago (o la cuenta paypal si a la hora del registro la hemos dado de alta).
c) Forma de envío, cuanto se tardara en enviar y si los costes del envío van a cargo del vendedor.
d) Política de devoluciones (si se admiten o no).
e) Finalizados estos pasos te ofrecen la posibilidad de marcar el articulo en negrita para más visibilidad sobre otros, con lo que te cobran una comisión.
f) Una vez vendido el articulo te cobran una serie de comisiones dependiendo de varios factores, en la siguiente imagen veremos la tabla de comisiones por venta.



Tarifas básicas en formato de subasta

Tarifas de publicación de un anuncio en formato de subasta

Precio de salida o precio mínimo	Tarifa de publicación de anuncios
0,01€ - 1,99€	0,10€
2,00€ - 9,99€	0,25€
10,00€ - 24,99€	0,50€
25,00€ - 49,99€	0,95€
50,00€ - 99,99€	1,15€
100,00€ o superior	2,65€

Comisiones por venta realizada en formato de subasta

Precio final	Comisión por venta realizada
Artículo no vendido	Sin tarifa
0,01€ - 50,00€	**5,00%** del valor final.
50,01€ - 1000,00€	**5,00%** de los primeros 50,00€ (2,50€), más el **3,50%** del precio de venta final restante (50,01€ a 1000,00€).
100,01€ o superior	**5,00%** de los primeros 50,00€ (2,50€), más el **3,50%** de entre los 50,01€ y los 1000,00 Euros (33,25€), más el **1,50%** del precio de venta final restante.

Imagen 2. Tabla de comisiones

Para la compra, el usuario se dirige al apartado de categorías y subcategorias, escoge una del listado i se muestra una lista de productos. Escoge uno y le sale la información del producto, una pequeña tabla con la información básica del vendedor y para entrar en su perfil a mirar sus estadísticas. No entramos en el apartado de subastas ya que no nos centraremos en ello.

3.2. Web www.priceminister.es

Imagen 3. Web de priceminister

Las características para vender y comprar siguen el mismo estilo de ebay. Tanto para la venta como la compra tienen las mismas características. Una funcionalidad destacable es la de apadrinar amigos, y ganar así bonos de descuentos de 7 euros para la próxima compra al que apadrina y al apadrinado.

La página hace de intermediaria entre el comprador y el vendedor en cuanto a pago. El proceso de compra difiere del de ebay siendo el siguiente:

a) El comprador hace el encargo del artículo.
b) El vendedor valida la venta y se compromete a enviarlo.
c) Priceminister hace el cargo en tu tarjeta de crédito.
d) Cuando el artículo llega al comprador y opinas sobre la venta, priceminister hace el ingreso en la cuenta del vendedor. Si no opina, el vendedor puede pedirte que votes para recibir el producto. En el caso de que pasadas seis semanas desde la venta, no ha llegado el voto, el vendedor recibe el pago.

Para el vendedor el caso es el mismo. Para el pago tienes dos posibilidades, hacer vía paypal o hacer el pago mediante una funcionalidad de la misma página. Consiste en un monedero, en el cual, el crédito lo ingresas a priceminister. Puedes ir recargando el monedero a través de transferencias bancarias, o con lo que ganes en la misma página.

La web tiene un sistema de comisiones similar a ebay por el que te cobran cuando realizas una venta de un artículo.

3.3. Web www.telocompro.es

Imagen 4. Web telocompro.es

Esta página sí tiene un marcado carácter de compra-venta estilo ebay. Tienes similares características a ebay con la diferencia es que para hacer el pago, tienen una página de intermediario de pagos llamada www.telopago.com. Aunque dispongan de un sistema de pago intermediario, no significa que un usuario no pueda comprar si no se registra en la otra página sino que tiene diferentes opciones de pago como por ejemplo, contra rembolso.

La navegación es sencilla e intuitiva ya que es fácil comprar y vender. A la hora de vender no te permite añadir tantas características como otras páginas ni tampoco hacer subasta. Ésta es una página que podríamos usar como referencia para nuestro proyecto.

A partir de aquí, se han encontrado numerosas páginas de cartel de anuncios como por ejemplo:

www.tablondeanuncios.com

www.trueque-online.com

Estas páginas se basan en colocar un anuncio y el usuario interesado en el, pide los datos del vendedor a la página para ponerse en contacto con él, o dependiendo del nivel de visibilidad que el vendedor haya dado a sus datos, se ven directamente con el artículo.

3.4. Tabla resumen

Funcionalidades/web	EBAY	PRICEMINISTER	TELOCOMPRO
Posibilidad de compra venta	Si	Si	Si
Posibilidad de valorar al usuario por articulo	Si	Si	No hay visibilidad hasta pasado un periodo de prueba
Envío de mensajes privados	Si	Si	No hay visibilidad hasta pasado un periodo de prueba
Poner fotos en el perfil del usuario	No	No	No
Publicar anuncios en el perfil de usuario	No (solo al votar)	No (solo al votar)	No hay visibilidad hasta pasado un periodo de prueba
Visualizar compras de otro usuario	No	No	No hay visibilidad hasta pasado un periodo de prueba
Visualizar productos de un usuario	Si	Si	No hay visibilidad hasta pasado un periodo de prueba
Componente de red social	No	No	No
Posibilidad de añadir amigos o favoritos	Si (favorito)	Si (favorito)	No hay visibilidad hasta pasado un periodo de prueba
Añadir fotos al	Si	Si	Si

articulo			

Como vemos en la comparativa, la parte de la compraventa está más que trabajada, mientras que la interacción entre usuarios es reducida, te permiten, en general, realizar pocas acciones con otros usuarios, como por ejemplo enviarse mensajes privados, añadirse entre ellos como favoritos, etc.

Nosotros hemos enfocado la página mas como un portal de compraventa que como red social, y para poder aportar algo novedoso a los sitios ya existentes hemos decidido incorporar el componente de la red social. Esto nos aporta una ventaja principal, y es que no es lo mismo comprar a un amigo o conocido que a un usuario totalmente desconocido para nosotros. Siempre tendremos la fiabilidad que nos da poder escoger productos de un conocido que de otro usuario que no sabemos siquiera si el artículo existe. A parte de esta ventaja, siempre podremos valorar a un usuario para que el resto de usuarios puedan ver las valoraciones que este ha recibido.

4. Tecnologías y metodología de desarrollo.

4.1. Tecnologías

4.1.1. Elección

El lenguaje de programación escogido para el desarrollo de la aplicación ha sido Java y para la parte de vistas JSP y Javascript.

En cuanto a los frameworks escogidos para el desarrollo han sido Spring, Struts e Ibatis. A grandes rasgos podemos decir que Spring facilita el desarrollo de aplicaciones web en la que su estructura es por capas. Añade un conjunto de funcionalidades que junto con Struts, hacen la tarea al desarrollador mucho más fácil. En cuanto a Struts podemos decir que es un marco de trabajo especialmente dedicado a aplicaciones J2EE que nos provee del patrón MVC (Modelo-Vista-Controlador). Para la parte de la persistencia, hemos decidido usar Ibatis, un framework que se sitúa entre las capas de negocio y persistencia y su principal función es la de relacionar los objetos del dominio con las tablas de la base de datos. También nos proporciona el patrón de diseño web DAO (Data Access Object).

4.1.2. Spring framework

Spring es un marco de trabajo, que aporta facilidad al desarrollo de aplicaciones J2EE y que promueve buenas prácticas en el diseño de la aplicación. Permite implementar patrones de diseño. Una gran ventaja de Spring es su modularidad, permite usar varias funcionalidades sin necesidad de aplicar el resto:

a) Tenemos la inversión del control, responsable de la creación y configuración de objetos mediante inyección de dependencias. Estos objetos se configuran en archivos XML mediante beans. No hay que confundir los beans de clases con los de Spring ya que simplemente es una indicación a la clase que inyectas ese bean para que tenga acceso a la clase en cuestión. Para tener una idea más clara veamos un ejemplo:

```
<bean id="ventaManager" parent="txProxyTemplate">
    <property name="target">
        <bean class="org.friendlybuy.service.impl.VentaManagerImpl">
            <property name="ventaDAO" ref="ventaDAO"/>
        </bean>
    </property>
</bean>
```

Imagen 5. Bean declarado en el contexto de Spring

```
<bean id="ventaDAO" class="org.friendlybuy.dao.impl.VentaDAOImpl">
   <property name="sqlMapClient"><ref bean="sqlMapClient"/></property>
</bean>
```

Imagen 6. Bean declarado en el contexto de Spring

```
public class VentaManagerImpl implements VentaManager {

   Logger logger = Logger.getLogger( this.getClass());

   private VentaDAO ventaDAO;

   public void setVentaDAO(VentaDAO ventaDAO) {
       this.ventaDAO = ventaDAO;
   }
```

Imagen 7. Objeto que se inyectara mediante la inyección de dependencias de Spring.

En este ejemplo vemos en la [imagen 5], como mediante el contexto de Spring, estamos inyectando al objeto VentaDAO la propiedad ventaDAO en la clase VentaManagerImpl. El contexto sabe de qué tipo es el objeto que está inyectando porque se lo estamos indicando en otro bean [imagen 6]. Para el uso de inyecciones, en la clase Java en la que inyectemos esa propiedad, no se debe declarar, la implementación, sino que se tiene que declarar un objeto de la interfaz y proveer a esta clase de un método 'setter' para que el contexto pueda inyectar esta dependencia. **Con esto controlamos el acceso de objetos entre capas y desacoplamos las capas al máximo.**

b) Data Acces Framework, que te permite trabajar con diferentes API de persistencia como Ibatis, Hibernate, JDBC, etc.
c) Control de transacciones, el cual permite que si en una transacción se ejecutan diversas operaciones sobre una base de datos, y por algún motivo, la última

operación lanza una excepción, automáticamente hace un 'Rollback' de las operaciones anteriores. La manera en que controlaríamos la transacción, es indicándole en el contexto donde declaramos los beans, que para un bean inyectado, si el método al que accedemos tiene diferentes interacciones con la base de datos, si ocurre una excepción, tire todas las operaciones que se han ejecutado desde la entrada en ese método, hacia atrás. En la siguiente imagen veremos cómo se declara tal función:

```
<bean id="articuloManager" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
    <property name="target">
        <ref bean="articuloManagerTarget" />
    </property>
    <property name="transactionManager"><ref bean="transactionManager"/></property>
       <property name="transactionAttributes">
        <props>
            <prop key="comprarArticulo">PROPAGATION_REQUIRED, -Exception</prop>
        </props>
       </property>
 </bean>
 <bean id="articuloManagerTarget" class="org.friendlybuy.service.impl.ArticuloManagerImpl">
    <property name="articuloDAO" ref="articuloDAO"/>
    <property name="ventaDAO" ref="ventaDAO"/>
    <property name="usuarioDAO" ref="usuarioDAO"/>
 </bean>
```

Imagen 8. Control de transacción mediante contexto de Spring

En esta imagen podemos ver que el bean declarado en cuestión es una clase en que inyectamos tres objetos, y a su vez se declara otra propiedad que indica que si el método al que accedemos es comprarArticulo en este caso, lanza una excepción, esta se propague por el resto de operaciones que hemos realizado a los DAOS que utilicemos dentro de este método.

d) Cabe la posibilidad de usar el patrón MVC mediante Spring en vez de Struts pero quitaríamos las posibilidades que tiene Struts. Hay que decir que un framework no se pisa con el otro, sino que se complementan a la perfección.

e) Spring Web Services, para lanzar servicios web, o peticiones a otras apliaciones web mediante xml's.

4.1.3. Framework Struts

Struts es una herramienta de soporte para el desarrollo de aplicaciones que implementa el patrón de diseño MVC (Modelo Vista Controlador).

Para ser puristas, habría que decir que la parte del modelo, se usa para manejar la lógica de negocio e interacción con la base de datos. Esto no es del todo cierto ya que puedes delegar en una capa más la tarea de lógica de negocio y utilizar la parte del modelo simplemente para recoger datos de las peticiones y retornar otros datos con las respuestas haciendo que la lógica de negocio quede totalmente desacoplada de la vista. Con esto ganamos, que si un día quisiéramos cambiar el marco de trabajo, no haya que tocar gran parte del código o migrarlo a otras clases para poder implantar otro o quedarnos, por ejemplo en nuestro caso, con Spring, que nos proporciona el mismo patrón de diseño.

La vista es la parte en la que presentamos nuestra aplicación al usuario. Se suele utilizar paginas JSP que, mediante una serie de librerías (taglibs) podemos enviar y recoger la información que nos llega des del modelo y representarla gráficamente para que el usuario vea su respuesta. Existen cantidad de plugins y frameworks para complementar la parte de la vista, por ejemplo JQuery, etc.

La parte del controlador es la parte más importante de este marco de trabajo ya que es la que, a partir de un servlet, struts-config, nos permite delegar las peticiones a una clase u otras y saber con exactitud, la pantalla que tiene que mostrar en cada caso, pudiendo así, configurar el flujo de nuestra aplicación en un único fichero xml de forma fácil y rápida sin demasiadas líneas de código.

El uso de este framework, facilita al desarrollador el intercambio de información entre las pantallas y el modelo mediante Actions y ActionForms.

4.1.4. Framework Ibatis

Ibatis es un marco de trabajo de Apache, de código libre, que se utiliza en la capa de persistencia como intermediario entre la capa de la lógica de negocio y la base de datos.

Ibatis permite que el desarrollador únicamente se tenga que preocupar en el desarrollo y la correcta configuración del entorno.

La principal ventaja de iBatis es la simplicidad. Este marco de trabajo carga los objetos en sentencias SQL mediante un descriptor XML.

El Data Mapper de ibatis nos proporciona un modo simple de mover datos entre objetos java y una base de datos relacional, es decir, mapeo de objetos. De esta manera, se obtiene todas las ventajas del SQL sin poner ni una sola línea de JDBC. Este framework, mapea las sentencias a partir de un XML simple.

Si profundizamos un poco más en este framework, vemos que nos permite definir múltiples ResultSet a partir de los ResultMap o ResultClasses, te permite crear trazas de log con Log4J, te permite especificar timeout a nivel de consulta, permite optimización de sentencias, ya que es el propio programador el que crea y mantiene la sentencia SQL.

4.1.5. Entorno de desarrollo Eclipse

Eclispe es un IDE (Integration Development Environment) disponible para diferentes lenguajes de programación.

Se ha escogido este entorno de desarrollo por varios motivos, el primero porque es un aplicativo open-source, es decir, de código abierto con licencias gratuitas. Otro motivo es que en el mundo laboral es uno de los IDE más solicitados. Una de las ventajas de Eclipse, es que aparte de poder descargar una versión más completa u otra, dispones de una gran cantidad de plugins para desarrollo como podría ser el plugin de UML, etc. Este entorno de desarrollo nos da también la libertad de configurar a nuestro antojo los servidores de aplicaciones, según nuestras necesidades. En nuestro caso hemos configurado un Tomcat 6.0.29 y mediante una tarea de ant, nos permite compilar el código y desplegarlo en el servidor de manera automática.

4.2. Metodología

Cada metodología tiene unas ventajas y unos inconvenientes y todas siguen un patrón específico a la hora del desarrollo del software. Nosotros hemos seguido el iterativo e incremental cuya forma podría verse como la siguiente imagen:

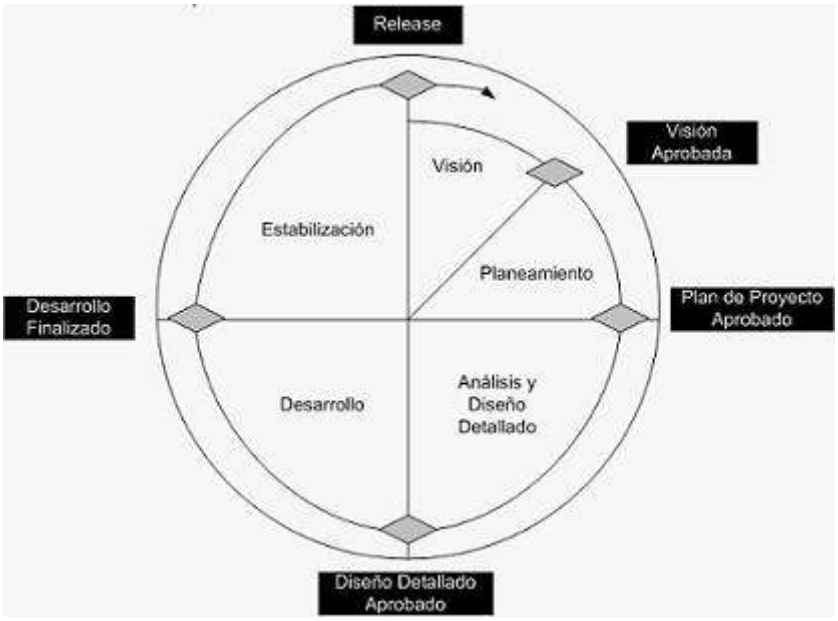


Imagen 9. Metodología iterativa incremental.

Es un enfoque para construir software en el cual todo el ciclo de vida está compuesto por varias iteraciones. Las iteraciones son pequeños proyectos compuestos de varias actividades cuyo objetivo es entregar una parte de un sistema parcialmente completo, probado, integrado y estable.

Esta metodología nos permite poder evaluar los riesgos durante todo el ciclo de vida del proyecto y también para obtener una mayor facilidad de cambio durante el proceso de desarrollo. La metodología iterativa incremental para por cuatro fases para cada iteración. La primera es la de determinar objetivos, la segunda para identificar riesgos o problemas que puedan surgir y resolverlos en un análisis previo, la tercera viene marcada por el desarrollo de los objetivos marcados en la primera fase y el testeo de estos, y por último, se planifica la siguiente iteración. Cada ciclo de cuatro fases acaba con una revisión para comprobar si se han conseguido los objetivos.

5. Presupuesto y planificación

5.1. Presupuesto del desarrollo de la aplicación

Para el cálculo del presupuesto se han destacado los siguientes costes:

Hardware:

- Ordenador portátil Acer 5700PX con un coste de 569 €
- Hosting para un mes 12€

Software:

- El software utilizado es open-source, por lo tanto es gratuito.

Conexiones:

- Telefónica ADSL 10 megas + llamadas a fijos 70 € * 4 meses

Recursos:

- Un analista programador con un coste de 35 € / hora y con duración de 560 horas

Tabla de costes:

Concepto	Cantidad	Precio	Subtotal
Ordenador portátil Aspire 5738ZG	1	569	569
Hosting	1	12	12
Software	0	0	0
Telefónica ADSL 10 megas + llamadas a fijos	4	70	280
Horas analista programador	560	35	19600
TOTAL			20461

Imagen 10. Tabla de Costes

5.2. Planificación

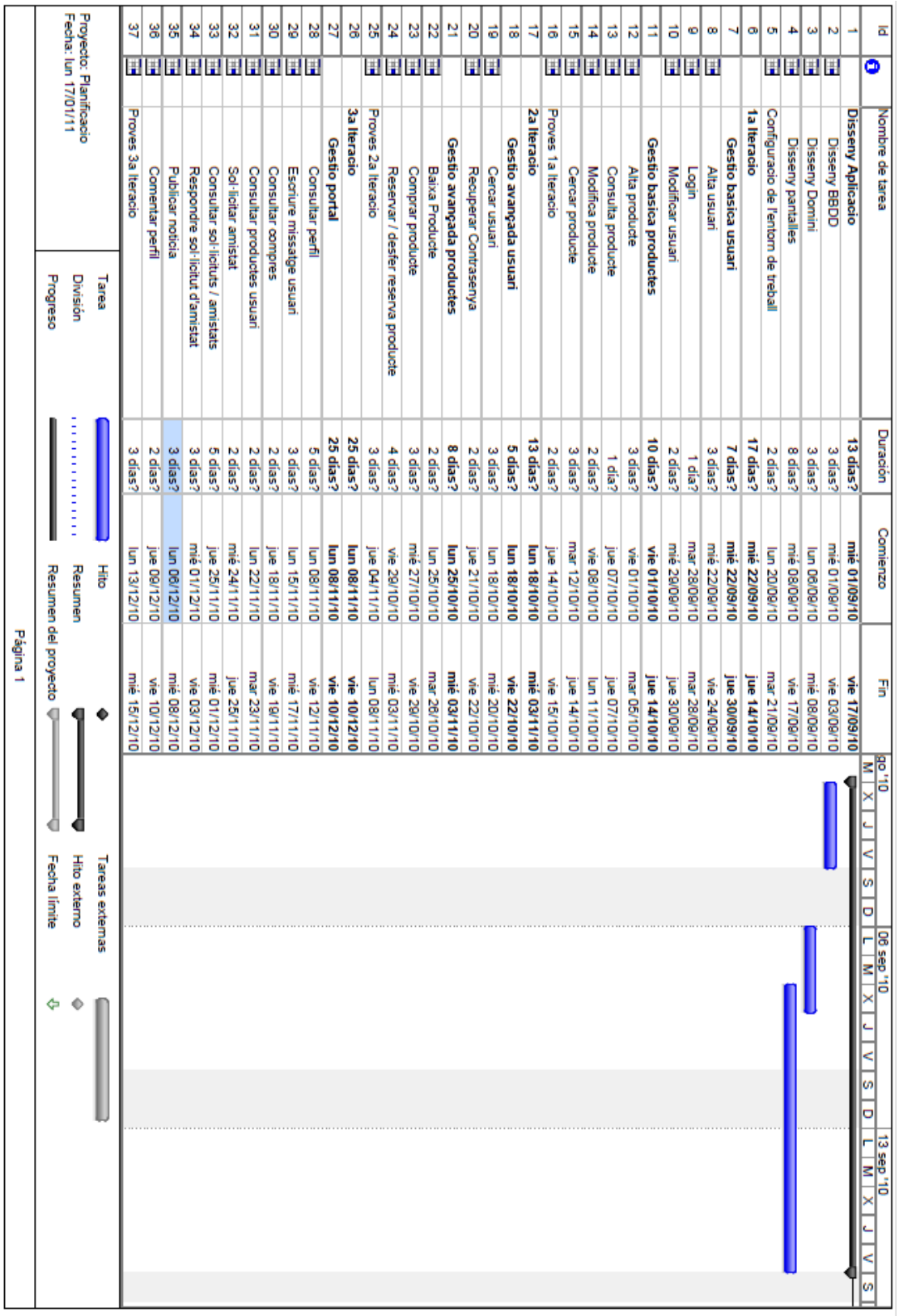


Imagen 11. Planificación parte 1.

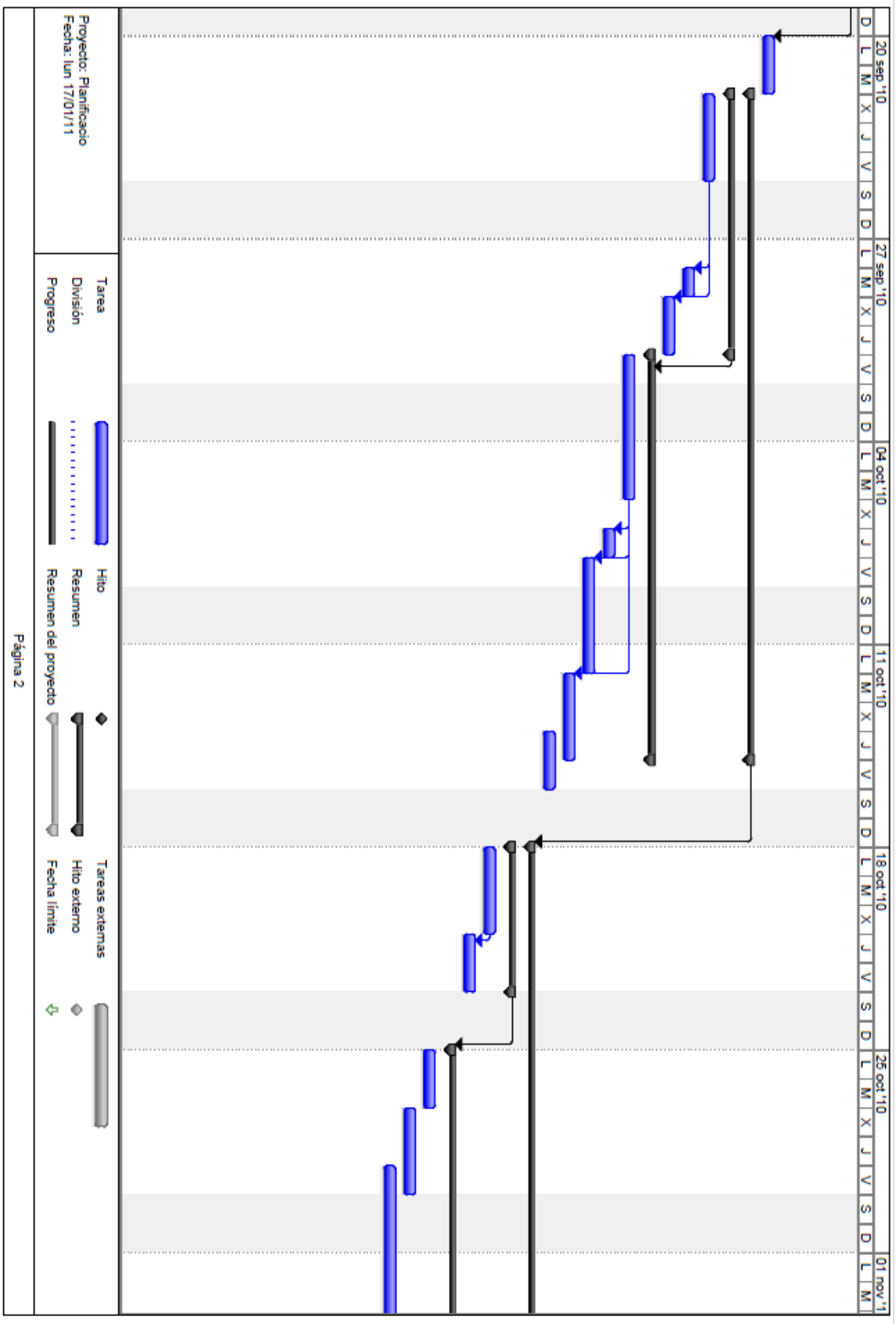


Imagen 12. Planificación parte 2.

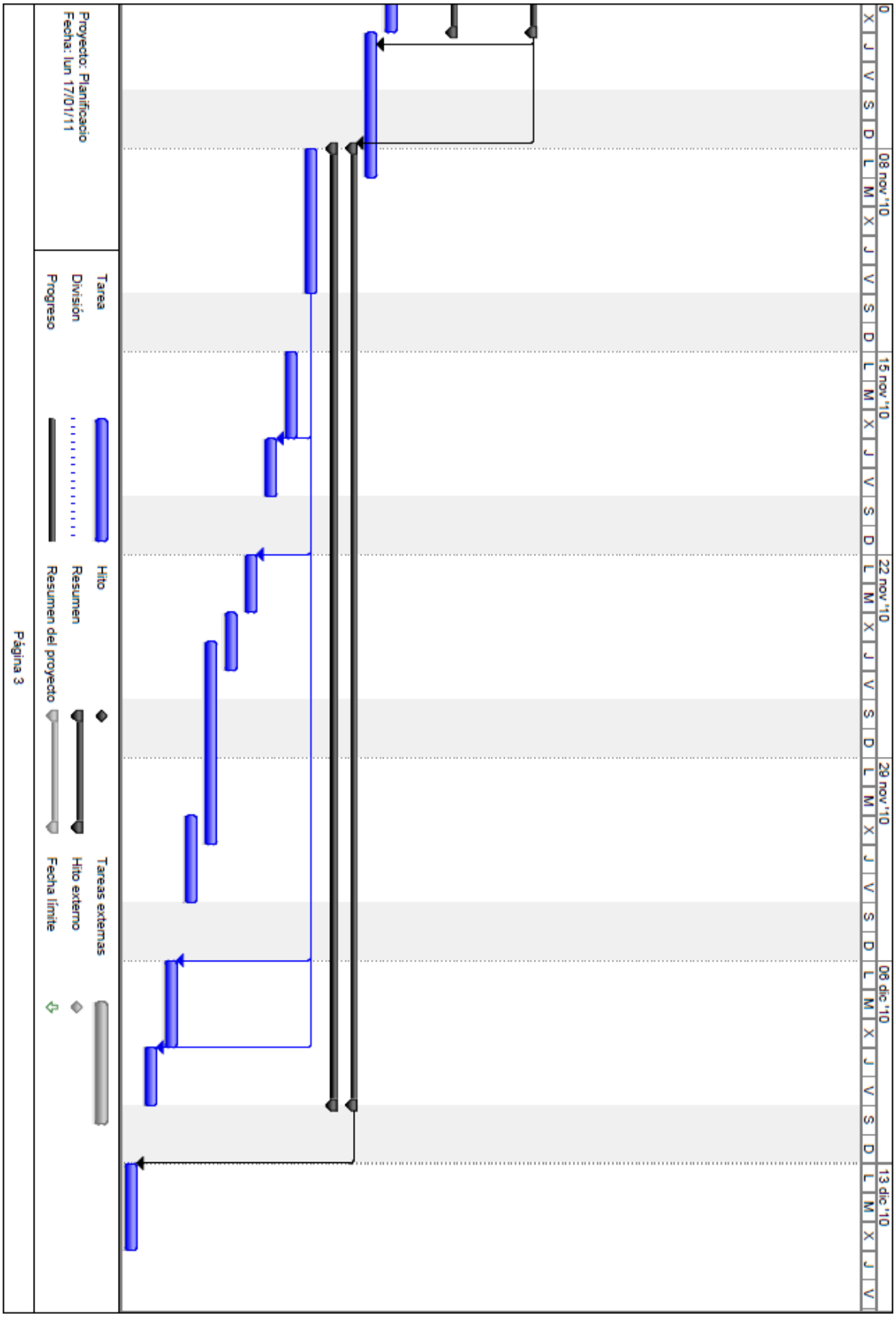


Imagen 12. Planificación parte 3.

6. Análisis

6.1. Requerimientos funcionales

El desarrollo de este proyecto pasa por el estudio previo de unos requisitos para que un usuario tenga acceso a las funcionalidades descritas más adelante.

El objetivo del proyecto es que un usuario pueda vender, comprar, consultar productos, comunicarse con otros usuarios, valorar las compras a otros usuarios, etc.

Un usuario anónimo, podrá acceder a la página sin ningún problema pudiendo consultar los productos y hacer búsquedas filtradas por una serie de criterios predefinidos, y todo esto sin la necesidad de estar identificado ni registrado. Únicamente será necesario el registro y la posterior identificación del usuario para realizar las compras, introducir nuevos artículos, entrar en su perfil de usuario, introducir fotos, y una larga lista que veremos más adelante gracias a la descripción de los casos de uso.

El sistema deberá proveer al usuario la capacidad de registrarse en la aplicación.

Si el usuario se ha registrado y no recuerda la contraseña, el sistema deberá proveer de una funcionalidad para recordar la contraseña al usuario.

El usuario podrá modificar sus datos de usuario mediante la opción modificar perfil en el que podrá cambiar los datos de su registro.

Como ya hemos comentado, un usuario sin estar identificado, podrá acceder al listado de artículos y realizar búsquedas acotadas según los criterios de búsqueda. También podrá seleccionar un producto para consultarlo y poder ver así, las fotos y características del artículo y datos del usuario vendedor.

Para comprar un producto, el usuario, ahora sí, deberá estar registrado e identificado en la aplicación, si no lo esta se llevara al usuario al formulario de identificación con posibilidad

de registrarse. También podrá reservar un artículo con previa petición al usuario vendedor. Esta petición se realizará mediante mensaje privado al vendedor, para que sea el mismo vendedor, por cuanto tiempo decida tener reservado el artículo al posible comprador. Cuando un artículo queda reservado, otros usuarios podrán consultarlo pero no podrán comprarlo.

Una vez el usuario a comprado un producto, la aplicación deberá enviar un correo electrónico al comprador y otro al vendedor informado que la compra o venta, dependiendo del rol del usuario, se ha efectuado correctamente e informando el numero de artículo que se ha comprado o vendido.

Para vender un producto, el usuario (identificado) deberá dirigirse al menú de 'Vender' y rellenar el formulario de venta. Los datos del producto podrán ser modificados siempre y cuando el producto no esté reservado ni vendido. De igual forma, no podrá ser borrado si el producto está en dichos estados.

Para acceder a la parte de red social, el usuario deberá estar registrado e identificado, si no lo está, se le redirige al formulario de login o de alta.

En la parte del perfil un usuario podrá ver en el menú todas las opciones de las que dispone, ver e introducir noticias propias, consultar los productos comprados o propios a partir de cuadros informativos a la derecha y ver sus estadísticas en cuanto a votos.

Para añadir un amigo nuevo, deberá ir al listado de usuarios, buscar al usuario usando los filtros y añadirlo. En este momento deberá esperar a que el otro usuario se conecte y lo acepte. Si al usuario le han realizado una petición de amistad, en el menú amigos, podrá aceptar o rechazar la petición de amistad.

El usuario podrá enviar mensajes privados a otro usuario. Para ello deberá ir al menú de mensajes privados, y una vez ahí crear un nuevo mensaje privado. Los datos requeridos son el nombre de usuario, el asunto y el cuerpo del mensaje. Si no sabe el nombre de usuario, la aplicación deberá proporcionarle un método de búsqueda de usuario, lo que redirigirá al usuario al listado de usuarios para que lo busque y a partir de la imagen de

mensaje privado pueda enviarle un mensaje. Si el usuario tiene mensajes privados por leer, al lado del menú mensajes privados, y entre paréntesis, le saldrá el número de mensajes sin leer que tiene. El usuario debe poder consultar, responder y borrar mensajes.

Asimismo, el usuario podrá introducir fotos para su perfil que podrán ser consultadas por otros usuarios en cuanto entren al perfil de este usuario, para ello deberá dirigirse al menú de fotos de usuario y añadir fotos mediante el formulario.

Una vez se ha realizado una compra, el usuario podrá votar o valorar, el estado del producto, o más bien, de la venta. Todo el sistema de votación se activa cuando el usuario ha comprado un artículo. En el menú de votación, le saldrá un número conforme tiene votaciones pendientes que realizar. La aplicación mostrara las votaciones pendientes y las realizadas por un usuario en el menú de votaciones.

6.2. Especificación de casos de uso.

6.2.1. Diagrama de casos de uso

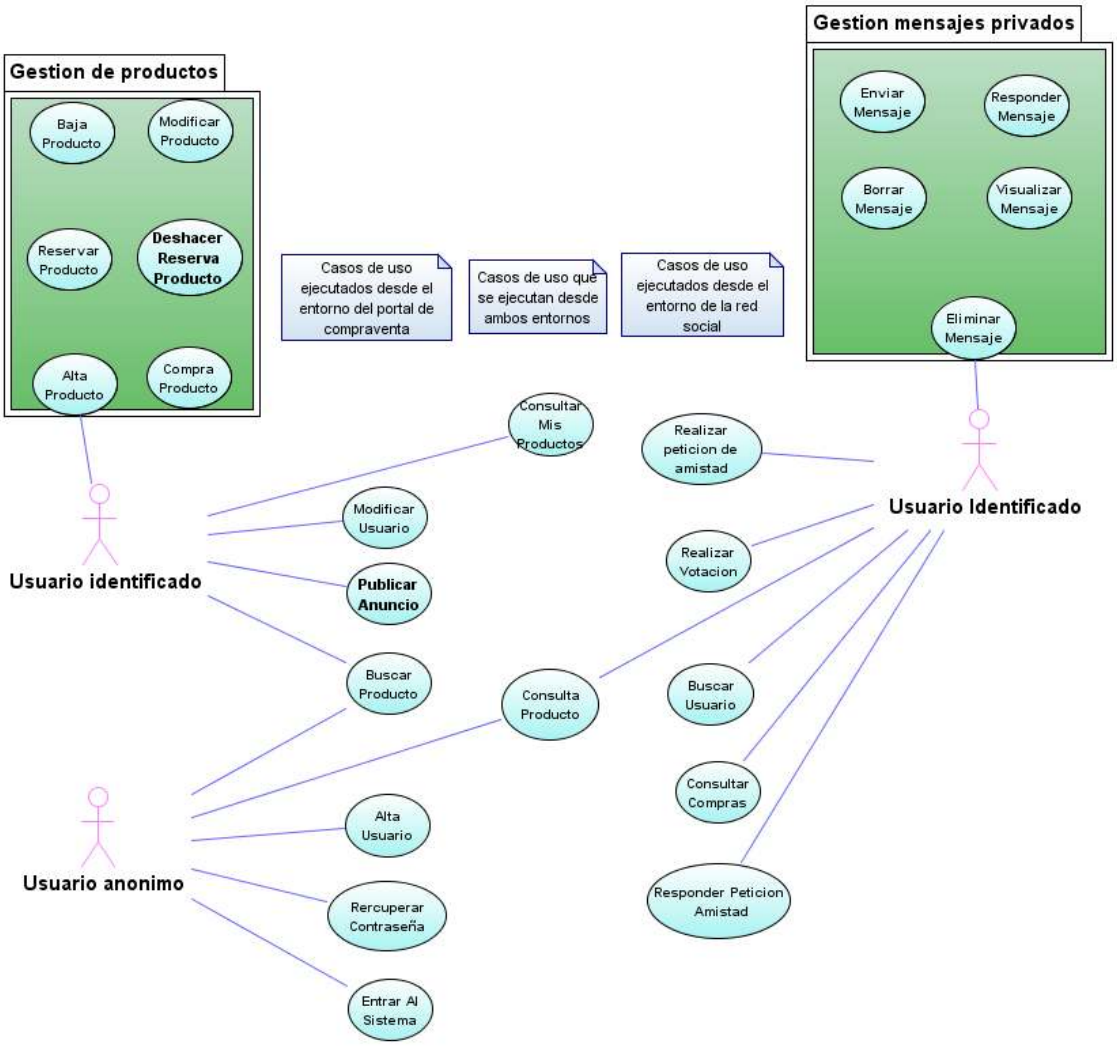


Imagen 13. Diagrama de casos de uso.

6.2.2. Actores que intervienen en el sistema

Los actores que intervienen en el sistema, como podemos ver en el diagrama son dos. El usuario invitado que podrá realizar una serie de acciones y el usuario registrado que podrá ejecutar la mayoría de acciones de la aplicación.

6.2.3. Casos de uso

Como podemos comprobar en el diagrama de casos de uso, se han divido los casos de uso en dos partes, la parte del portal de compraventa y la parte de la red social.

Parte portal

- Alta usuario
- Modificar usuario
- Buscar usuario
- Entrar al sistema
- Recuperar contraseña
- Alta producto
- Modifica producto
- Consulta producto
- Buscar producto
- Baja producto
- Compra producto
- Reserva producto
- Deshacer reserva producto

Parte red social

- Enviar mensaje
- Responder mensaje
- Borrar mensaje

- Ver mensaje
- Consultar compras
- Consultar mis productos
- Realizar petición de amistad
- Aceptar / rechazar amistad
- Publicar anuncio

Como la lista de casos de uso es larga y algunos son triviales, definiremos los casos de uso que son más significativos en la aplicación.

Nombre	Alta usuario
Descripción	Dar de alta un nuevo usuario al sistema
Actores	Usuario, sistema
Pre-condiciones	No estar identificado
Post-condiciones	El usuario quedará registrado pero no identificado.
Flujo normal	1. El usuario pide ingresar como nuevo usuario 2. El sistema muestra el formulario 3. El usuario rellena datos y acepta 4. El sistema valida datos y registra el nuevo usuario en la aplicación.
Flujo alternativo	4.1. El sistema detecta anomalías en los datos, vuelve al paso 3.

Nombre	Comprar producto
Descripción	La compra de un artículo por un usuario.
Actores	Usuario, sistema
Pre-condiciones	El producto debe tener estado ‘En venta’
Post-condiciones	El usuario finaliza la compra del producto.
Flujo normal	1. El usuario ejecuta el caso de uso ‘Consultar producto’. 2. Una vez consultado el producto pulsa el botón de

	comprar. 3. El sistema valida el estado del producto. 4. El sistema introduce nueva venta 5. El sistema cambia el estado del producto 6. El sistema envía un mail al comprador indicándole la compra. 7. El sistema envía un mail al vendedor.
Flujo alternativo	3.1 El producto no tiene el estado correcto, muestra mensaje al usuario.

Nombre	Enviar mensaje privado
Descripción	Enviar un mensaje privado a un usuario.
Actores	Usuario, sistema
Pre-condiciones	El usuario debe estar registrado e identificado en la aplicación.
Post-condiciones	El mensaje será enviado al usuario receptor.
Flujo normal	1. El usuario pulsa nuevo mensaje privado 2. Rellena los datos del formulario y envía. 3. El sistema valida los datos. 4. El sistema introduce nuevo mensaje para el usuario destinatario. 5. El sistema devuelve al usuario al listado de mensajes privados.
Flujo alternativo	3.1. El usuario introducido no existe, vuelve al paso 2 informando al usuario que no existe.

Nombre	Reservar producto (desde usuario interesado)
Descripción	Un usuario desea reservar un producto.
Actores	Usuario comprador, usuario vendedor, sistema
Pre-condiciones	El usuario debe estar registrado e identificado en la

	aplicación.
Post-condiciones	-
Flujo normal	1. El usuario consulta un producto y quiere reservarlo. 2. El usuario ejecuta el caso de uso 'Enviar mensaje privado' 3. El usuario vendedor recibe el mensaje y acepta. 4. El usuario vendedor se dirige al menú del producto. 5. El usuario introduce el nombre del usuario comprador y reserva. 6. El sistema modifica el estado del producto.
Flujo alternativo	6.1 El usuario introducido no existe, vuelve al paso 4.

7. Diseño de la aplicación

7.1. Diseño de la interfaz.

El diseño de la interfaz de la aplicación es sencillo e intuitivo, dirigido para todo tipo de clase de usuarios, desde mayores hasta los más jóvenes. La intención es que cualquier usuario pueda pasar un rato agradable, interactuar con otros usuarios mediante el componente de red social y tener la posibilidad de comprar, vender, consultar o reservar artículos de otros usuarios. Para ello se ha decidido escoger unos colores adecuados para cada entorno y que ambos queden claramente diferenciados.

En cuanto a la navegabilidad de la aplicación, se ha separado en dos partes, la primera parte es la parte del portal de compraventa en el que tenemos un menú particular con las típicas opciones que un portal de este tipo puede disponer, más una pestaña que nos dirige a la parte de la red social.

En la segunda parte, el diseño sigue siendo igual, con la principal diferencia que el diseño cambia el color, pero no el estilo. Sigue siendo un entorno sencillo e intuitivo en el que el usuario podrá saber en cada caso donde deberá dirigirse según sus necesidades sin tener que perder tiempo en averiguar dónde debe ir.

En la siguiente imagen podemos ver la página principal a la que accedemos cuando ponemos la url de la aplicación en el navegador sin habernos identificado aun como registrados:

Imagen 14. Diseño de la pantalla principal.

A continuación veremos la diferencia de color que comentábamos para distinguir entre entornos. Para poder acceder a la parte de red social nos hemos tenido que identificar con un usuario que teníamos para probar.

El color elegido ha sido el azul ya que muchas redes sociales han escogido también el azul. El color azul es un color fresco, da tranquilidad y se le asocia con la mente. Según estudios psicológicos, el azul ayuda a tener claras las ideas y a tener claridad de mente.

Imagen 15. Diseño de la parte de red social.

Como podemos comprobar, el cambio entre entornos está claramente diferenciado por el color pero no supone un cambio drástico para la vista ya que el tono elegido ha sido más bien un tono claro, en el que un usuario pueda pasar bastante tiempo sin que se canse de la tonalidad, pues el mismo color denota calma y tranquilidad.

7.2. Diseño del modelo físico de bases de datos

La base de datos será la encargada de persistir los datos de la aplicación y de proveer los datos a los usuarios que requieran dicha información.

La base de datos es relacionar, es decir, podemos relacionar tablas y columnas de tablas. Hemos escogido MySql ya que es de código libre y porque es suficientemente potente para nuestra aplicación en cuanto a optimización de velocidad de consultas o transacciones.

A continuación podemos ver el modelo físico de la base de datos:

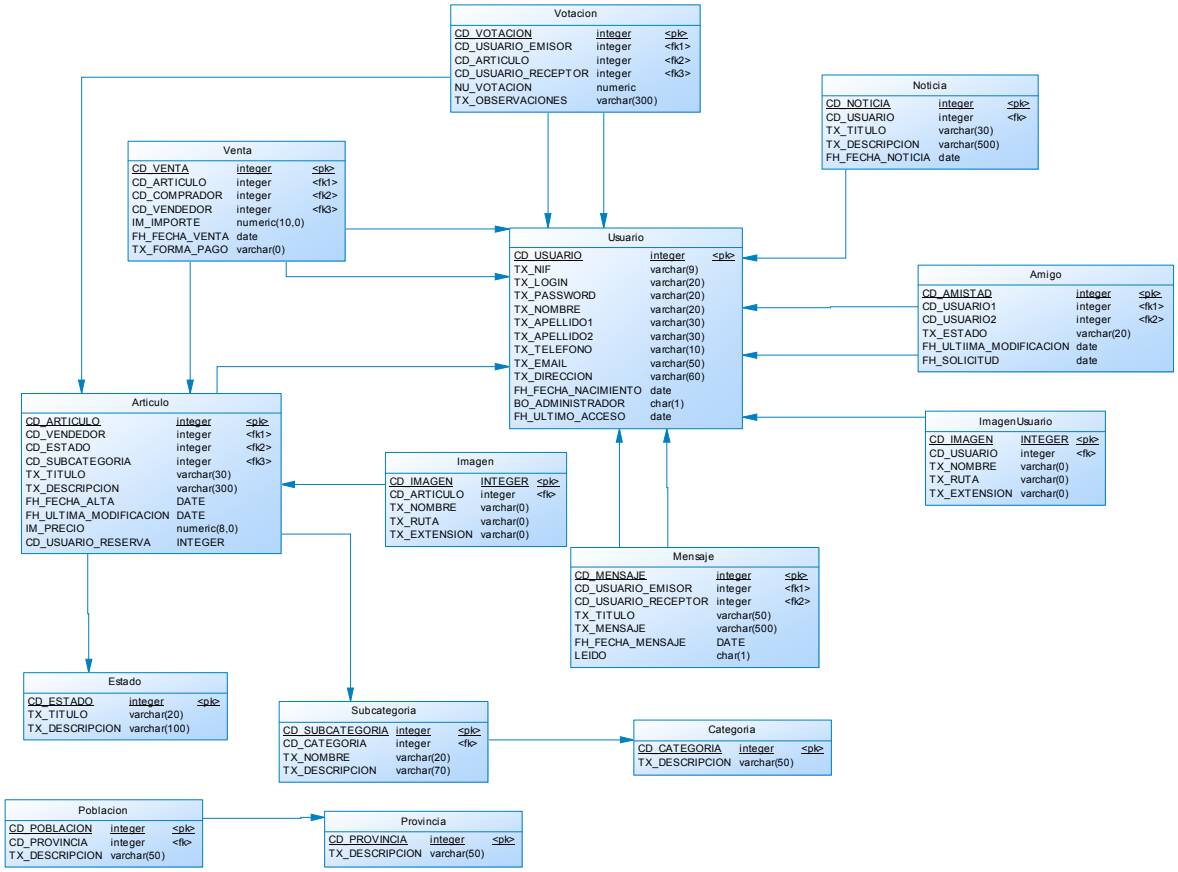


Imagen 16. Modelo físico de la base de datos.

Como podemos comprobar, nuestra aplicación se mueve en torno a un usuario, y este, puede acceder a toda la información relacionada con ese usuario. Así pues vamos a definir las relaciones existentes:

- Votación – artículo: Una votación es de un artículo y ese artículo solo puede tener una votación.
- Artículo – estado: Un artículo tiene un estado y solo uno, en cambio un estado puede estar en ninguno o en muchos productos.
- Artículo – Sub-categoría: Un artículo solo puede pertenecer a una sub-categoría y esa sub-categoría puede pertenecer a ninguno o a muchos artículos.
- Sub-categoría – categoría: Una sub-categoría pertenece a una categoría y una categoría puede tener una o varias sub-categorías.
- Venta – articulo: Una venta es de un y solo un artículo y ese artículo solo puede estar una vez vendido.
- Artículo – usuario: Un artículo es de un usuario y ese usuario puede tener cero o muchos artículos.
- Artículo – imagen: Un artículo puede tener cero o muchas fotos y una imagen es de un y solo un artículo.
- Venta – usuario: Una venta tiene un comprador y un vendedor (y solo uno en ambos casos) en cambio un usuario puede ser comprador cero o muchas veces y un vendedor puede serlo cero o muchas veces.
- Votación – usuario: Una votación tiene un usuario emisor y otro receptor (y solo uno en ambos casos) y un usuario puede tener cero o muchas votaciones como emisor y cero o muchas como receptor.
- Mensaje – usuario: Un mensaje tiene un usuario emisor y otro receptor (y solo uno en ambos casos) y un usuario puede tener cero o muchos mensajes como emisor y cero o muchos como receptor.
- ImagenUsuario – Usuario: Una imagen de usuario solo es de un usuario y nada más, y un usuario puede tener ninguna o muchas imágenes de usuario.
- Noticia – Usuario: Una noticia solo puede ser publicada por un solo usuario y un usuario puede tener publicadas cero o muchas noticias.
- Usuario – Amigo: Una amistad está compuesta por dos usuarios, el usuario uno y el usuario 2 y solo uno en cada caso. Y un usuario puede estar como amistad en ambos casos cero o muchas veces.

7.3. Diseño del dominio

El modelo del dominio, al usar Ibatis, ha sido realizado como una especie de espejo del modelo físico de la base de datos. Esto es debido a que el modelo del dominio y el de la base de datos tiene que ser igual para poder corresponder objetos relacionales persistentes con objetos del modelo de datos de la aplicación. Las relaciones son las mismas solo que en código aplicamos la orientación a objetos que nos aporta Java.

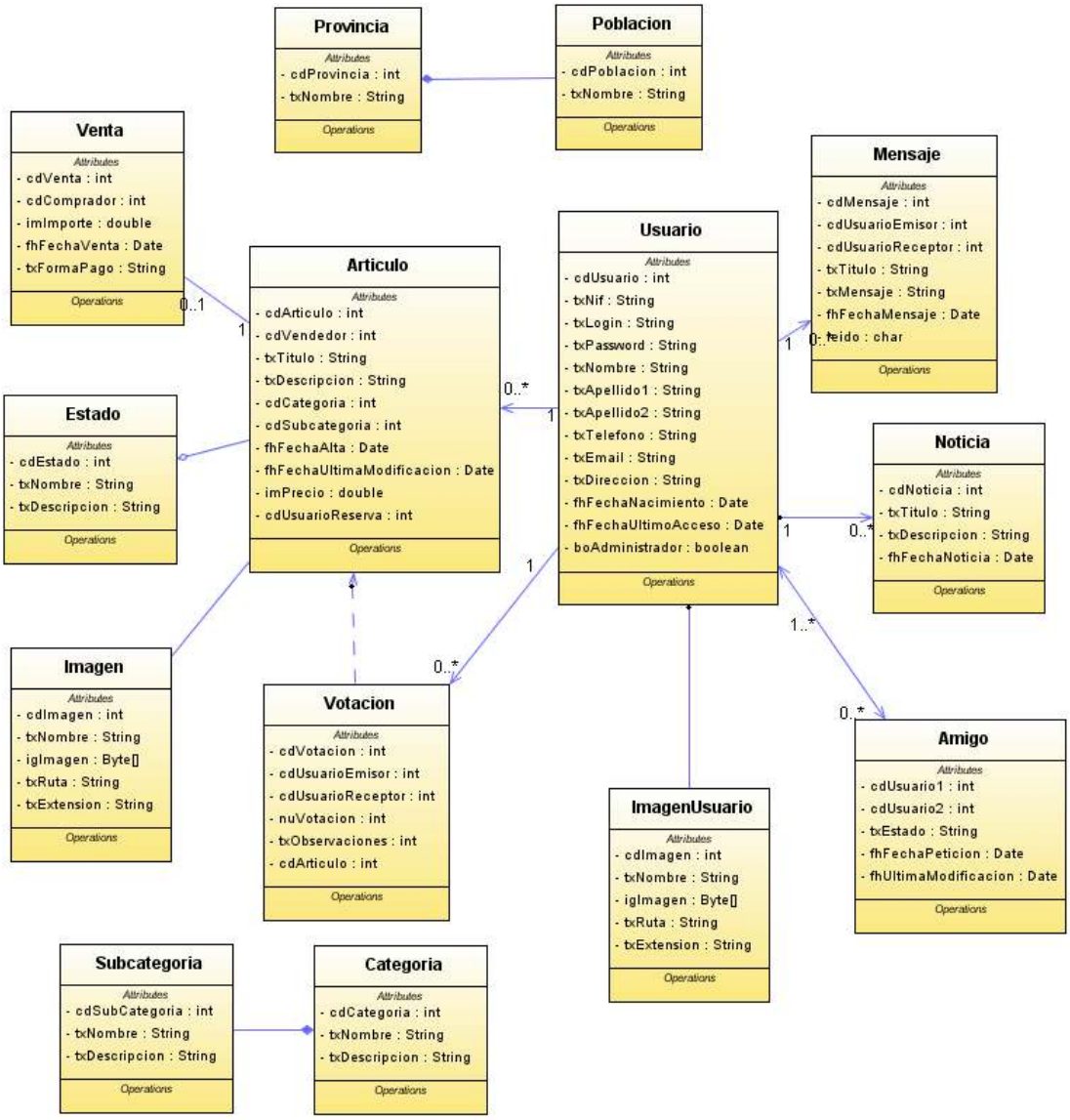


Imagen 17. Diagrama de clases.

Como podemos comprobar en el diagrama de clases del dominio, las clases que moverán la aplicación nos reflejan el tipo de aplicación que será. La aplicación se moverá en torno a los usuarios y los artículos. Las relaciones las hemos comentado en el modelo físico de la base de datos ya que es una copia calcada y de ahí hemos podido deducir gran parte del diagrama de clases. Las relaciones que comentaremos, y muy superficialmente ya que no tienen una gran importancia en la influencia sobre las demás, son las de agregación.

La provincia contendrá una lista de poblaciones y esa población únicamente estará contenida en un único objeto provincia.

De igual forma una categoría está compuesta de subcategorias. Estas subcategorias es un nivel por debajo de la categoría puesto que un usuario, cuando desee especificar el tipo de producto que está dando de alta lo indicará por la subcategoria siendo más descriptiva que la categoría a la que pertenece. Una categoría, por lo tanto, estará contenida en una lista de subcategorias del objeto del dominio categoría.

8. Desarrollo de la aplicación

En este apartado mostraremos como se ha montado el entorno de desarrollo desde cero. Un entorno de desarrollo totalmente diferente de todo lo que se ha impartido en la universidad, lo que ha implicado un esfuerzo inicial enorme para adaptarnos a estas tecnologías, que si bien una es difícil, complementar varias a la vez para que resuelvan nuestra problemática tiene un coste de adaptación alto. Una vez superado el coste de adaptación y se ha aprendido a utilizar estos marcos de trabajo, el desarrollo (tanto la velocidad como la calidad) ha aumentado considerablemente.

Mostraremos como se ha montado el IDE sobre el que desarrollaremos la aplicación, qué servidor de aplicaciones se ha utilizado para desplegar la aplicación, y los diferentes marcos de trabajo que no se han utilizado en las asignaturas impartidas y que tienen gran demanda en el mercado laboral.

Puesto que las tecnologías en las que se ha puesto más hincapié, son las que resuelven problemáticas en prácticamente todos los casos de uso, explicaremos los casos de uso más complejos para mostrar el funcionamiento de estas tecnologías.

8.1. Configuración del entorno de desarrollo

8.1.1. Configuración del IDE Eclipse Galileo.

El Entorno de desarrollo o IDE (de las siglas *Integrated Development Environment*) que hemos utilizado ha sido el Eclipse y la versión más reciente en el momento de la descarga era el Galileo.

Se ha descargado de la página de eclipse en la que se muestran las diferentes versiones disponibles: http://www.eclipse.org/downloads/packages/release/galileo/r.

Una vez descargado se instala siguiendo las instrucciones.

8.1.2. Configuración del servidor de aplicaciones Tomcat 6.0.29

Si bien Tomcat es considerado por muchos un servidor de aplicaciones, no lo es realmente ya que únicamente es un contenedor de servlets. El servidor de aplicaciones se descarga desde la página de apache: http://tomcat.apache.org/download-60.cgi.

Una vez descargado se tiene que descomprimir en el directorio deseado sin tener que modificar nada más.

Para integrarlo en el entorno de desarrollo debemos ir a la pestaña de Servers y hacer click derecho --> New --> Server.

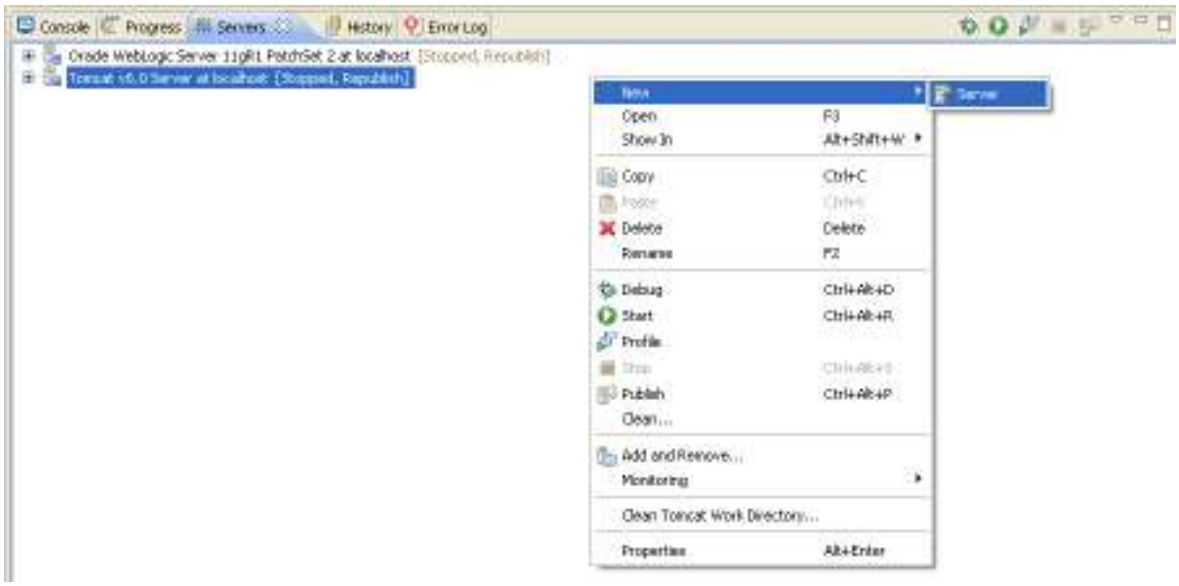


Imagen 18. Pestaña de servers en Eclipse.

Saldrá la ventana para añadir un nuevo servidor. En esta le debemos indicar el nombre en el paso 1. El siguiente paso es configurar el entorno de ejecución del servidor. Para ello hacemos click en el link que hay al lado de la caja de texto y nos abre una nueva ventana en la que le indicamos la ruta donde está instalado nuestro Tomcat y posteriormente la JRE que utilizará (podemos seleccionar una o dejar la que utiliza el Eclipse por defecto).

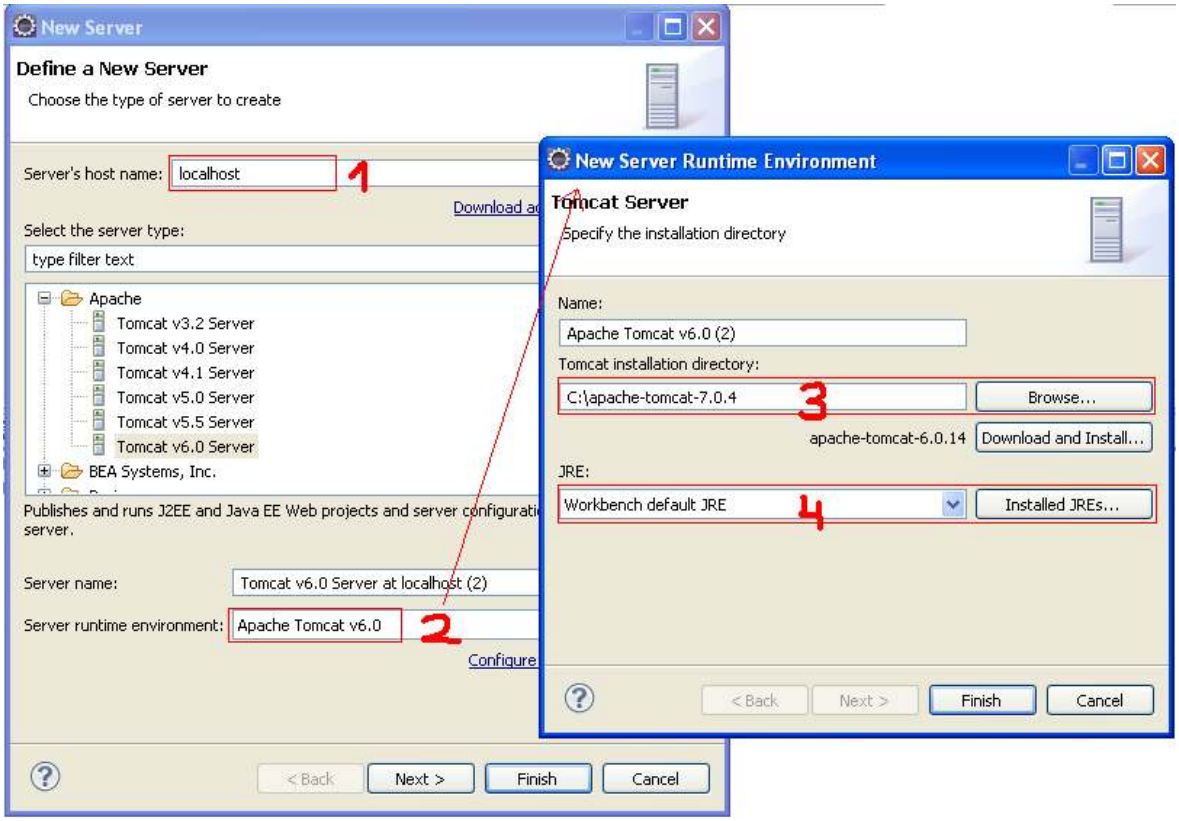


Imagen 19. Configuración de nuevo servidor.

Una vez configurado el servidor le añadimos el modulo del proyecto web (se puede hacer posteriormente). Le damos a next y nos aparece una ventana como la siguiente:

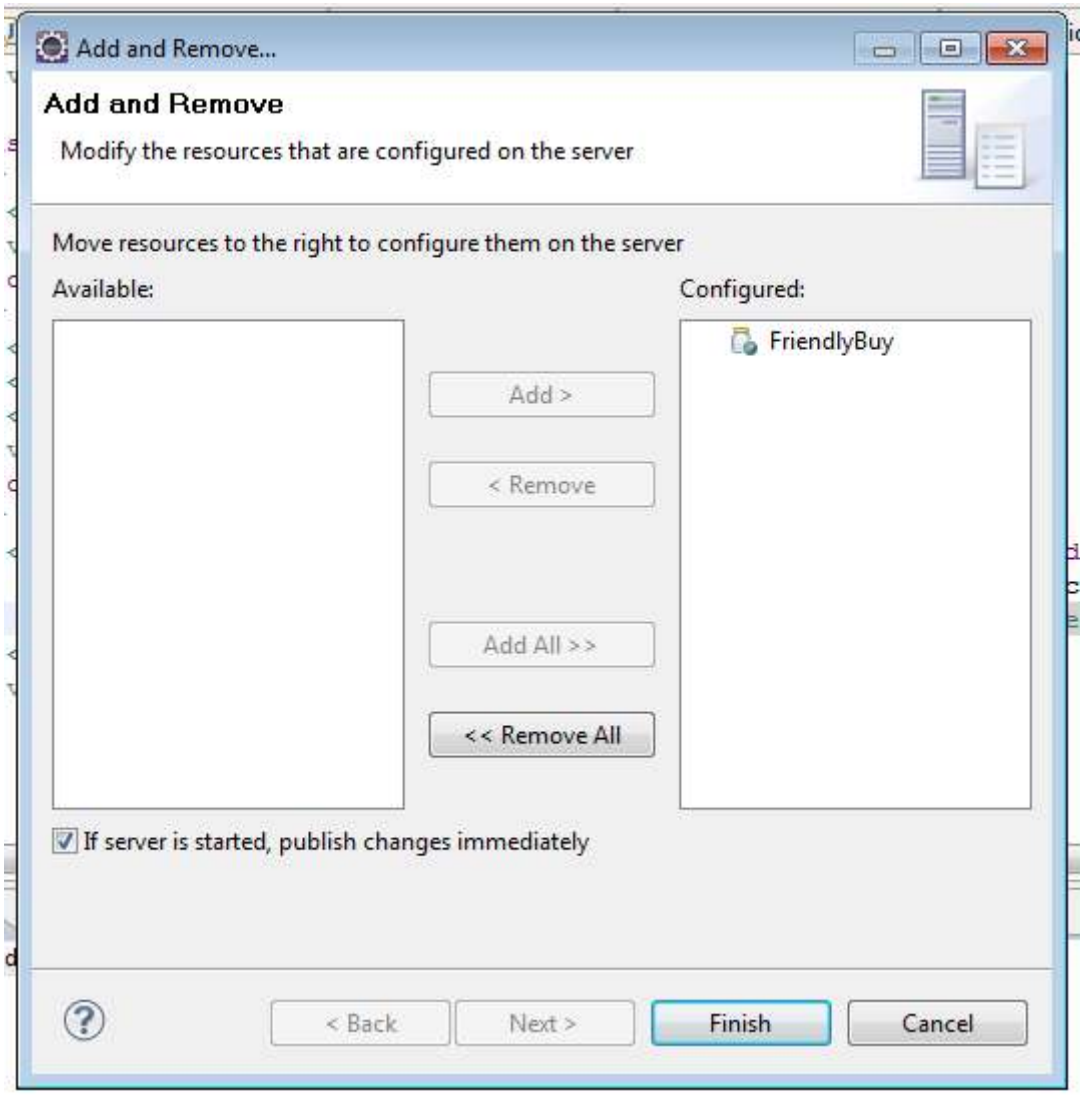


Imagen 20. Configuración de servidor en Eclipse.

Lo único que hay que hacer es seleccionar el modulo que queremos y darle a Add>. Después le damos a finalizar y ya tenemos configurado el servidor.

En la pestaña de servers podremos darle a arrancar, parar, arrancar en modo de depuración, etc.

El proyecto lo importamos al servidor a partir de una tarea Ant. El proyecto, en la raíz, deberá tener un archivo que se llame build.xml en el que configuraremos las tareas como clean, build, compile, deploy, etc. Estas tareas se definen en ese archivo XML. Un ejemplo de una tarea para compilar:

```
<target name="compile" description="Compile main source tree java files">
    <mkdir dir="${build.dir}/classes"/>
    <javac destdir="${build.dir}/classes" debug="true" optimize="false"
        deprecation="false" failonerror="true">
        <src path="${src.dir}"/>
        <classpath>
            <path refid="compile.classpath"/>
        </classpath>
    </javac>
    <copy todir="${build.dir}/classes">
        <fileset dir="${src.dir}" includes="**/*.xml"/>
    </copy>
</target>
```

Imagen 21. Tarea 'Compile' del build.xml para ejecutar mediante Ant.

8.1.3. Configuración del marco de trabajo Ibatis

Para utilizar iBatis Framework, lo primero que debemos hacer es descargarnos las librerías de la página de iBatis: http://code.google.com/p/mybatis/downloads/list?can=4 Esta URL ha cambiado recientemente ya que antes se bajaba desde la página de Apache.

Una vez descargadas las librerías se añaden al proyecto y pasamos a la configuración de los archivos xml que nos permitirán la integración y utilización de iBatis en nuestro proyecto.

Imagen 22. Jar de Ibatis añadido al espacio de trabajo.

En uno de los contextos de Spring, más concretamente el que utilicemos para la parte de persistencia (mas adelante explicaremos porque y como lo separamos), debemos definir los beans con los que accederemos al Framework de Ibatis:

```
<!-- Transaction manager for iBATIS DAOs -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>

<!-- SqlMap setup for iBATIS Database Layer -->
<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation" value="classpath:/org/friendlybuy/web/resources/SqlMapConfig.xml"/>
</bean>

<!-- Anadir DAOs adicionales aqui -->

<bean id="usuarioDAO" class="org.friendlybuy.dao.impl.UsuarioDAOImpl">
    <property name="sqlMapClient"><ref bean="sqlMapClient"/></property>
</bean>
```

Imagen 23. Archivo de configuración de Spring Framework.

El primer bean de la imagen anterior quiere decir que las transacciones las haremos a partir del datasource definido por el Framework de Spring, que en realidad lo tendremos definido nosotros mediante otro bean, o mejor dicho, redefinido:

```
<!-- Database Configuracion -->
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    <property name="url" value="jdbc:mysql://localhost:3306/PROYECTO"/>
    <property name="username" value="root"/>
    <property name="password" value=""/>
</bean>
```

Imagen 24. Configuración del Datasource de conexión a base de datos.

El siguiente bean está indicando que utilizamos los mapeos que están definidos en el SqlMapConfig.xml que está en ese path:

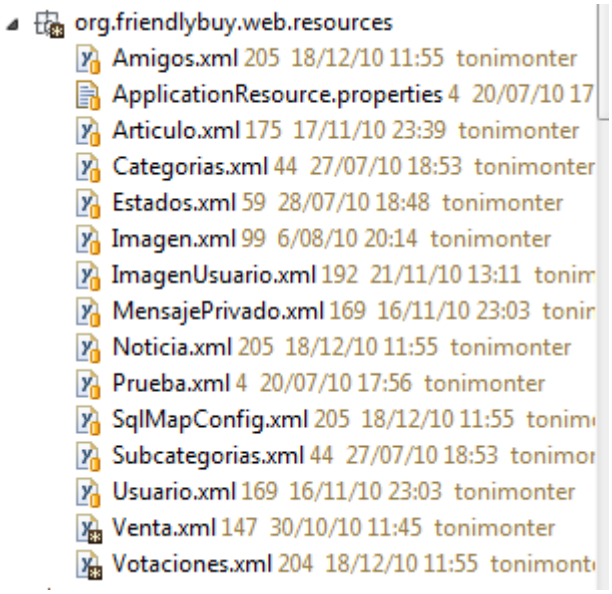


Imagen 25. Archivos xml donde se mapean los objetos con la base de datos.

Lo que contiene ese archivo es la importación de los diferentes archivos xml, los cuales mapean los objetos de la base de datos con los objetos del dominio. Contienen también las sentencias SQL que ejecutaremos, tanto sentencias de inserción, como de modificación, borrado o consulta.

8.1.4. Configuración del marco de trabajo de Spring

Spring, como habíamos explicado, dispone de varios módulos que pueden ser utilizados de manera independiente como por ejemplo, Spring Security, Spring Web Services, etc. Nosotros solo necesitaremos el módulo funcional de la inyección de dependencias y del control de transaccionalidad.

Para ello nos dirigimos a la página de Spring y nos bajamos el paquete de Spring Framework en la siguiente URL: http://www.springsource.org/download. Una vez descargadas las librerías, debemos hacer la configuración de los archivos pertinentes para la implementación e integración de Spring en nuestra aplicación.

Empezamos con el web.xml, donde se define, por ejemplo, el tipo de navegación que se va a usar, cual es la página inicial de nuestra aplicación, y se importan las dependencias de las librerías que usaremos. Ahí tenemos que añadir las siguientes líneas:

```
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<listener>
  <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>
</listener>
```

Imagen 26. Configuración del listener de Spring en el archivo web.xml

Como en nuestra aplicación, el controlador que utilizaremos será el de Struts, pero la inyección de dependencias para los action las manejaremos con Spring, debemos indicarle, que la acción que realizará en cada caso, está contenida en el contexto (archivo XML) de Spring. En el archivo Struts-config.xml añadiremos las siguientes líneas:

```
<plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
    <set-property property="contextConfigLocation"
    value="/WEB-INF/action-servlet.xml"/>
</plug-in>
```

Imagen 27. Importacion del contexto de Spring en el archivo struts-config.xml

Y para cada acción, le indicamos que el listener que tiene que utilizar está en el contexto de Spring, recordemos que el contexto se carga cuando arrancamos el servidor, por lo tanto todos los beans estarán disponibles desde el mismo momento que levantamos el servidor. Se lo indicamos en los actions de struts-config de la siguiente manera:

```
<action path="/Votacion"
        type="org.springframework.web.struts.DelegatingActionProxy"
        scope="request"
        parameter="method"
        validate="false">
    <forward name="correcto" path="/WEB-INF/jsp/nuevaVotacion.jsp"/>
</action>
```

Imagen 28. Delegación de la acción al contexto de Spring

Podemos poner todos los beans definidos en la aplicación en el mismo archivo de contexto, pero eso hace que sea menos mantenible ya que tendríamos un archivo xml de muchísimas líneas de código. Por ello definimos varios archivos XML, uno por cada capa. Lo único que hay que añadir en el web.xml, que cuando arranque el servidor, cargue todos los

archivos XML cuyo nombre contenga el siguiente formato: applicationContext*.xml, siendo los asteriscos, el nombre de la capa a la que pertenece cada archivo de contexto.

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/applicationContext*.xml</param-value>
</context-param>
```

Imagen 29. Carga de archivos xml para el contexto.

8.2. Implementación caso de uso 'Comprar producto'

Debido a la gran cantidad de casos de uso que hay en la aplicación hemos decidido explicar y mostrar los casos de uso de mayor complejidad que hay en esta aplicación.

Uno de los casos de uso que presentan una mayor dificultad a la hora de implementarlo es el de comprar un producto, más que nada por la cantidad de acciones que realiza y no por la complejidad funcional puesto que la palabra comprar se define por sí misma.

Para realizar esta acción, se deben cumplir una serie de premisas, la primera es que el usuario comprador esté identificado en la aplicación. La segunda es que el producto que queremos comprar tenga estado en venta y no esté reservado.

Si el usuario no está identificado no se le mostrará el botón de comprar producto. En cambio si está identificado podrá visualizar tal botón. Veámoslo en una imagen:

Imagen 30. Diferenciación de pantalla con usuario identificado – no identificado.

En el mismo producto vemos como si no se está identificado, no se visualiza el botón que nos permite comprar el producto, mientras que si estamos identificados, la aplicación nos muestra el botón.

Para la siguiente premisa, concluimos que nos mostrará el botón si el producto que queremos comprar está en venta, o bien, si está reservado para nosotros.

Una vez se cumplen las precondiciones o premisas anteriores, podremos acceder a la acción de comprar. Para empezar a comprar debemos consultar previamente el artículo. En la pantalla de consulta nos mostrarán los datos del artículo, los datos del vendedor y las fotos que haya disponibles.

Imagen 31. Pantalla de consulta de artículo.

Cuando pulsamos sobre el botón de 'Comprar' este hace un envío del formulario, el cual tiene una acción ligada que en este caso es ComprarArticulo.do:

```
<form id="ComprarArticuloForm" name="ComprarArticuloForm" action="ComprarArticulo.do" method="post">
<input type="hidden" id="cdArticulo" name="cdArticulo" value="${articulo.cdArticulo}" />
```

Imagen 32. Acción que se realiza al pulsar el botón comprar.

Esta acción está ligada a un Action del Framework Struts y definida en el archivo struts-config.xml la cual nos dirá a qué clase java deberá ir aunque no será exactamente el controlador de Struts el que nos indicará la clase, sino que delegaremos la faena al Framework de Spring.

A continuación vemos la referencia de la acción a la que llamamos desde la página:

```
<action path="/ComprarArticulo"
        type="org.springframework.web.struts.DelegatingActionProxy"
        name="ComprarArticuloForm"
        scope="request"
        parameter="method"
        validate="false">
    <forward name="correcto" path="/Portal.do"/>
</action>
```

Imagen 33. Acción definida en Struts-config.xml

Si nos fijamos en el atributo 'type', nos indica que el que decidirá la clase a la que iremos será el propio contexto de spring en el que hemos creado un bean. Esto lo haremos para evitar que la clase action, pueda acceder a todas las clases de la capa de negocio. Vemos el ejemplo a continuación:

```
<bean name="/ComprarArticulo" class="org.friendlybuy.web.actions.portal.ComprarArticuloAction" singleton="false">
   <property name="articuloManager" ref="articuloManager"/>
</bean>
```

Imagen 34. Acción recogida en el contexto de Spring

Como decíamos, este bean está definido en el action-servlet.xml. Este nos indica la clase que debemos ir una vez hemos invocado esa acción. Dentro declaramos las propiedades (clases de negocio) a las que podremos invocar desde ese action. A eso le llamamos inyección de dependencias. Esta propiedad, está a su vez declarada en otro bean dentro del contexto de la capa de negocio, y hace referencia a la implementación de la interfaz a la que nosotros estamos llamando desde el action. Miramos primero el atributo declarado en el action al que llamamos:

```
private ArticuloManager articuloManager;

public v                                                    nager articuloManager) {
    this                                                    r;
}
```

org.friendlybuy.service.ArticuloManager

Press 'F2' for focus

Imagen 35. Creación del bean inyectado desde Spring.

Vemos que estamos declarando una propiedad de la interfaz, y gracias a un método setter en la clase action y a la inyección que hemos puesto desde el contexto, la aplicación será capaz de entender que debe ir a la clase que implementa esta interfaz. Veamos ahora el bean declarado que hace referencia a la implementación:

```
<bean id="articuloManager" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
   <property name="target">
       <ref bean="articuloManagerTarget" />
   </property>
   <property name="transactionManager"><ref bean="transactionManager"/></property>
      <property name="transactionAttributes">
       <props>
           <prop key="comprarArticulo">PROPAGATION_REQUIRED, -Exception</prop>
       </props>
      </property>
  </bean>
<bean id="articuloManagerTarget" class="org.friendlybuy.service.impl.ArticuloManagerImpl">
   <property name="articuloDAO" ref="articuloDAO"/>
   <property name="ventaDAO" ref="ventaDAO"/>
   <property name="usuarioDAO" ref="usuarioDAO"/>
</bean>
```

Imagen 36. Inyección de dependecias desde el contexto de Spring

En la imagen anterior hemos declarado la propiedad articuloManager, y en esta imagen vemos que esa propiedad hace referencia a la clase que implementa la interfaz.

Veamos el siguiente paso, que es la recogida de información del formulario y enviada a la clase de negocio:

```
public ActionForward doPerform(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
    logger.info(this.getClass().getName());

    try {
        //Formulario de la pantalla
        ComprarArticuloForm formulario=(ComprarArticuloForm)form;

        //identificador del articulo a comprar
        String cdArticulo=formulario.getCdArticulo();

        //Recogemos la sesion y de ella extraemos el usuario identificado en la sesion
        HttpSession sesion = request.getSession();
        Usuario usuarioSesion=(Usuario) sesion.getAttribute("usuario");

        //llamada al metodo que nos permitira comprar el articulo para el usuario de sesion
        articuloManager.comprarArticulo(cdArticulo, usuarioSesion);

        return mapping.findForward(SUCCESS);

    } catch (Exception e) {
        logger.info("Error en el action comprar: "+e.getMessage());
        SUCCESS="error";
        return mapping.findForward(SUCCESS);
    }
}
```

Imagen 37. Acción específica para el comprar artículo.

Vemos como se recoge la información del formulario y la envía a la clase de negocio a través de la llamada al método comprarArticulo.

```
public synchronized void comprarArticulo(String cdArticulo, Usuario usuarioComprador)
        throws Exception {
    try
    {
        //recogemos el articulo
        Articulo articulo=articuloDAO.getArticulo(new Integer(cdArticulo));

        if(articulo.getCdEstado()==Integer.parseInt(Constantes.ESTADO_VENDIDO)){
            //Se ha vendido mientras esperaba en cola
            throw new Exception("Este articulo no se puede comprar");
        }
        else if(articulo.getCdEstado()==Integer.parseInt(Constantes.ESTADO_RESERVADO)
                && articulo.getCdUsuarioReserva()!=usuarioComprador.getCdUsuario()){
            //miramos si lo han reservado mientras esperaba en cola
            throw new Exception("Este articulo no se puede comprar");
        }
        else{
            //todo correcto, procedemos con la compra
            //Actualizamos el estado del articulo
            articulo.setFhUltimaModificacion(Utilidades.getFechaActual());
            articulo.setCdEstado(Integer.parseInt(Constantes.ESTADO_VENDIDO));
            articuloDAO.updateArticulo(articulo);

            //Creamos una nueva venta
            Venta nuevaVenta= new Venta();
            nuevaVenta.setCdArticulo(articulo.getCdArticulo());
            nuevaVenta.setCdComprador(usuarioComprador.getCdUsuario());
            nuevaVenta.setCdVendedor(articulo.getCdVendedor());
            nuevaVenta.setFhFechaVenta(Utilidades.getFechaActual());
            nuevaVenta.setImImporte(articulo.getImPrecio());
            nuevaVenta.setTxDescArticulto(articulo.getTxDescripcion());
            nuevaVenta.setTxFormaPago("Contrareembolso");
            ventaDAO.insertVenta(nuevaVenta);

            //Enviaos los mails correspondientes, uno al vendedor y otro al comprador
            Usuario usuario=usuarioDAO.getUsuarioDelArticulo(Integer.toString(articulo.getCdArticulo()));

            HashMap parametros=new HashMap();
            parametros.put("correoDestinatario", usuario.getTxEmail());
            parametros.put("sujeto", Constantes.SUJETO_COMPRA_VENDEDOR);
            parametros.put("mensaje", Constantes.TEXTO_COMPRA_VENDEDOR+articulo.getCdArticulo());

            MailUtil.enviarMail(parametros);

            parametros=new HashMap();
            parametros.put("correoDestinatario", usuarioComprador.getTxEmail());
            parametros.put("sujeto", Constantes.SUJETO_COMPRA_COMPRADOR);
            parametros.put("mensaje", Constantes.TEXTO_COMPRA_COMPRADOR+articulo.getCdArticulo());

            MailUtil.enviarMail(parametros);
        }

    } catch (Exception e) {
        logger.info("EXCEPCION COMPRANDO UN ARTICULO: "+e.getMessage());
        throw e;
    }

}
```

Imagen 38. Método comprar en la capa de lógica de negocio.

Como estamos en la capa de negocio, es aquí donde delegaremos toda la lógica de negocio. En esta capa es donde se crearán los objetos que insertemos en la base de datos o modificaremos la información que necesitemos. En el primer párrafo actualizamos el estado y la fecha de última modificación del artículo. En el segundo párrafo de código creamos la venta y la insertamos. En el tercero enviamos los correos electrónicos al comprador y al vendedor para informar de la compra o venta.

Todas estas transacciones se realizan una vez más mediante la inyección de dependencias. Esta clase, en la capa de negocio, tendrá unas dependencias específicas para sus necesidades, por lo tanto se le inyectan los DAOs necesarios mediante el contexto, en la imagen 23 podemos ver la inyección realizada. Los correos se envían mediante una clase de utilidades llamada MailUtil, que a partir de una librería, nos permite enviar correos electrónicos, veamos la clase:

```java
package org.friendlybuy.web.utils;

import java.util.HashMap;

public class MailUtil {


    /**
     * Metodo que envia un email desde mi cuenta de correo..xD
     * Hay que meter tres parametros en el hm en modo clave-valor
     * @param parametros - correoDestinatario, sujeto, mensaje
     * @throws Exception
     */
    public static void enviarMail(HashMap parametros)throws Exception{
        try{
            Properties props = new Properties();

            // Nombre del host de correo, es smtp.gmail.com
            props.setProperty("mail.smtp.host", "smtp.gmail.com");
            // TLS si está disponible
            props.setProperty("mail.smtp.starttls.enable", "true");
            // Puerto de gmail para envio de correos
            props.setProperty("mail.smtp.port","25");
            // Nombre del usuario
            props.setProperty("mail.smtp.user", "tonimonter@gmail.com");
            // Si requiere o no usuario y password para conectarse.
            props.setProperty("mail.smtp.auth", "true");

            // Get session
            Session session = Session.getDefaultInstance(props);
            //session.setDebug(true);

            // Define message
            MimeMessage message = new MimeMessage(session);
            // Quien envia el correo
            message.setFrom(new InternetAddress("tonimonter@gmail.com"));

            // A quien va dirigido
            message.addRecipient(Message.RecipientType.TO,
                    new InternetAddress((String) parametros.get("correoDestinatario")));
            message.setSubject((String) parametros.get("sujeto"));
            message.setText((String) parametros.get("mensaje"));


            // Send message
            Transport t = session.getTransport("smtp");
            t.connect("smtp.gmail.com","tonimonter@gmail.com","contraseña");
            t.sendMessage(message,message.getAllRecipients());
        }

        catch(Exception e){
            throw e;
        }
    }
}
```

Imagen 39. Método que envía los correos electrónicos.

Como vemos, a partir de unos objetos que nos proporciona la interfaz, rellenamos los datos necesarios y enviamos el correo electrónico pertinente.

Para acabar con la explicación de las acciones que realiza esta clase de negocio, hay que tener en cuenta la concurrencia y la transaccionalidad. Para el control de concurrencia

hemos decido que lo mejor y más optimo seria poner un synchronized en la cabecera del método para que no puedan alterar el estado dos usuarios a la vez. Si dos usuarios acceden a la vez al método, uno de ellos se pone en cola. Cuando el usuario primero ha acabado entra el siguiente y realiza las acciones. Y por si el artículo es el mismo, se realizan las previas comprobaciones para lanzar la excepción en caso de que sea necesario.

Para la transaccionalidad, Spring Framework nos proporciona la capacidad de hacer rollback si alguna transacción dentro de un mismo método da algún tipo de excepción en tiempo de ejecución. En la siguiente imagen veremos un claro ejemplo:

```
<bean id="articuloManager" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
   <property name="target">
       <ref bean="articuloManagerTarget" />
   </property>
   <property name="transactionManager"><ref bean="transactionManager"/></property>
      <property name="transactionAttributes">
       <props>
           <prop key="comprarArticulo">PROPAGATION_REQUIRED, -Exception</prop>
       </props>
      </property>
  </bean>
<bean id="articuloManagerTarget" class="org.friendlybuy.service.impl.ArticuloManagerImpl">
   <property name="articuloDAO" ref="articuloDAO"/>
   <property name="ventaDAO" ref="ventaDAO"/>
   <property name="usuarioDAO" ref="usuarioDAO"/>
</bean>
```

Imagen 40. Bean inyectado con control de transacciones.

En la imagen vemos como definimos una propiedad de transacción en la que indicamos que si el método comprarArticulo genera una excepción, todas las transacciones que se hayan realizado dentro de ese método, no persistan en la base de datos.

Una vez se ha llamado al método de insertar una nueva venta, por ejemplo, hemos llamado al método de la implementación. Esta implementación hereda de una clase de la librería de Ibatis que nos permitirá ejecutar la sentencia que le digamos mediante un nombre de la query y en el que se le mete también los parámetros que necesite la query, en caso de que necesite alguno. Veámoslo en la imagen:

```
public class VentaDAOImpl extends SqlMapClientDaoSupport  implements VentaDAO {

    Logger logger = Logger.getLogger( this.getClass());
    public void insertVenta(Venta venta)throws Exception{
        try {
            getSqlMapClientTemplate().insert("insertVenta", venta);
        } catch (DataAccessException e) {
            logger.debug("EXCEPCION insertando venta: "+e.getMessage());
            throw e;
        }
    }
```

Imagen 41. Método de inserción en la capa de persistencia.

Cuando arranca el servidor, igual que con los beans de Spring, todas las sentencias (que se identifican a partir de un id) se cargan en el contexto de la aplicación. De esta manera, cuando desde el código se llama a un método con un identificado, o una clave, se dirige al contexto y ejecuta la sentencia. No distingue por fichero XML ya que eso es solo una forma de ponerlo nosotros para que sea más claro y mantenible. Por lo tanto esa sentencia de la imagen anterior llamará a la siguiente función:

```
<insert id="insertVenta" parameterMap="ventaParameter">
    INSERT INTO T_M_VENTAS (
        CD_COMPRADOR,
        CD_VENDEDOR,
        CD_ARTICULO,
        IM_IMPORTE,
        FH_FECHA_VENTA,
        TX_FORMA_PAGO
    )
    VALUE(?,?,?,?,SYSDATE(),?)
</insert>
```

Imagen 42. Método de inserción en el archivo xml correspondiente.

En el que le estamos pasando un parámetro que se llama ventaParameter, que está definido al principio del fichero XML y que hace referencia a una clase del dominio:

```
<typeAlias alias="Venta" type="org.friendlybuy.model.Venta"/>

<parameterMap class="Venta" id="ventaParameter">
     <parameter property="cdComprador" jdbcType="INT"/>
    <parameter property="cdVendedor" jdbcType="INT"/>
    <parameter property="cdArticulo" jdbcType="INT"/>
    <parameter property="imImporte" jdbcType="NUMBER"/>
    <parameter property="txFormaPago" jdbcType="VARCHAR"/>
</parameterMap>
```

Imagen 43. Parametrizacion del objeto Venta con el de la capa de dominio.

8.3. Implementación del control de acceso.

Para el control de acceso de usuarios identificados y no identificados hemos creído conveniente implementar una fachada en la capa de presentación para controlar la visibilidad de unos usuarios y otros.

La finalidad de este control es que un usuario que no se haya identificado en la aplicación no pueda acceder a la parte de red social, ni acceder a las diferentes partes de la aplicación que estén en el entorno del portal de compraventa que requiera estar identificado.

Para ello, se ha decidido implementar un Action 'padre', del que extiendan todos los actions que tenemos en la aplicación. En ese action padre se definen una serie de acciones para las que se requiere estar identificado y ese mismo action padre será el que nos lleve a la pantalla de identificación en el caso de que no lo estemos ya.

De esta manera evitamos tener que implementar un control de visibilidad en todos los actions, delegando esta función en la clase de la que extienden los actions hijos. Lo veremos más claro con un diagrama de estados y mostrando el código:

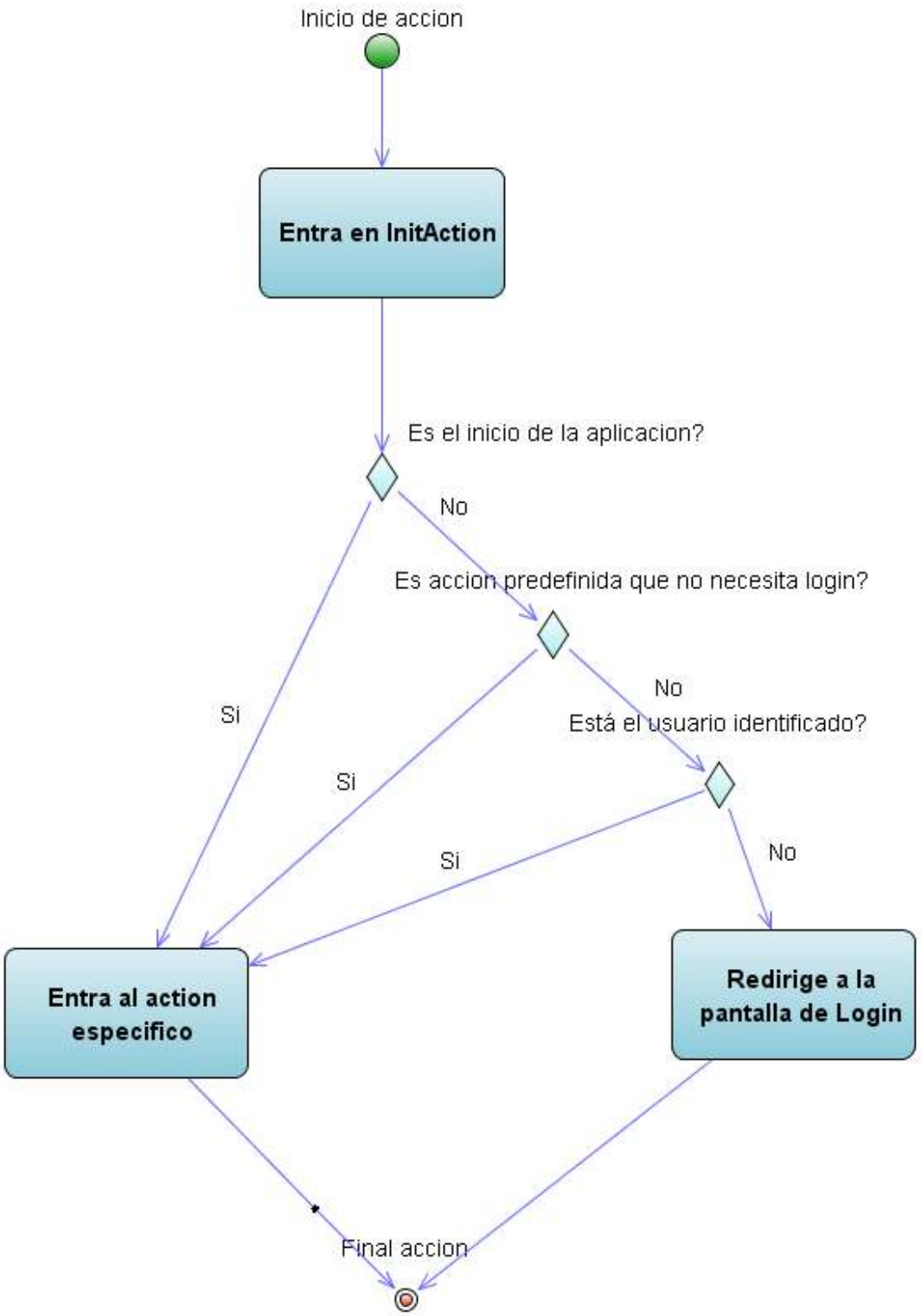


Imagen 44. Diagrama de estados de la llamada a un action. Fachada de la vista.

Como vemos, la clase InitAction, que extiende de la clase Action del Framework de Struts, bloquea las peticiones, comprueba que se puede pasar al siguiente nivel y delega la faena

en el action correspondiente, si no puede pasar al siguiente nivel redirige a la pantalla de login.

Veamos ahora el código y explicaremos paso a paso:

```
public abstract class InitAction extends Action {

    Logger log = Logger.getLogger( this.getClass());

    public ActionForward execute(ActionMapping mapping, ActionForm form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        ActionForward actionForward = null;
        HttpSession session = request.getSession();
        log.info(this.getClass().getName());

        try{
            //Es el inicio de la aplicacion, cuando llamamos al indice
            if(this.getClass().getName()
                    .equals("org.friendlybuy.web.actions.IndexAction")){
                return mapping.findForward(Constantes.INDICE);
            }
            //Las siguientes acciones no necesitan estar identificados
            else if(this.getClass().getName()
                    .equals("org.friendlybuy.web.actions.usuario.AltaUsuarioAction")
                    || this.getClass().getName()
                    .equals("org.friendlybuy.web.actions.portal.PortalAction")
                    || this.getClass().getName()
                    .equals("org.friendlybuy.web.actions.portal.ConsultarArticuloAction")
                    || this.getClass().getName()
                    .equals("org.friendlybuy.web.actions.usuario.LoginAction")
                    || this.getClass().getName()
                    .equals("org.friendlybuy.web.actions.usuario.RecuperarPasswordAction")){
                actionForward = doPerform( mapping, form, request, response );
            }
            //Esta logueado, puede acceder a cualquier accion
            else if(session.getAttribute("usuario")!=null){
                actionForward = doPerform( mapping, form, request, response );
            }
            //No esta logeado, redirige a la pantalla de login
            else{
                String accion=this.getClass().getName();
                session.setAttribute("accion", accion);
                log.info("Sesion no iniciada, redirigiendo a pantalla de login");
                return mapping.findForward(Constantes.NO_SESION);
            }
        }catch ( Exception e ){
            log.info(e.getMessage());
        }

        return actionForward;
    }
```

Imagen 45. Action de la que heredan las demás actions de la aplicación. Control de acceso.

Como podemos comprobar, esta clase InitAction extiende de Action. Cuando se ejecuta una acción entra al método execute y en nuestro caso, sigue una serie de condicionales. El

primer caso es cuando entra en la aplicación por primera vez, al índice. El segundo condicional nos indica, que si el usuario requiere una de las acciones que están descritas dentro de los paréntesis, al no ser necesario estar identificado, hacemos la llamada a la acción que inicialmente estábamos llamando a partir del doPerform, que es un método abstracto definido en la clase:

```
public abstract ActionForward doPerform
        (ActionMapping mapping, ActionForm form, HttpServletRequest request,
        HttpServletResponse response) throws Exception;
```

Imagen 46. Método abstracto que esta implementado en las demás actions.

Este método abstracto buscará el método que le implemente entre las clases que extiendan de InitAction a partir del ClassName que le pasamos mediante el contexto de Spring en el archivo Action-servlet.xml.

El tercer caso, únicamente comprueba que el usuario está identificado (si no ha entrado en las anteriores dos), si lo está, ejecuta la acción. Y por último, el ultimo condicional, si después de haber comprobado que ni es la llamada al índice, no se ha llamado a una acción que no requiere identificación, ni está el usuario identificado, redirige a la pantalla de identificación.

9. Prueba significativa.

Como prueba significativa del funcionamiento de la aplicación haremos un circuito completo de la compraventa de un producto, así como utilizar todas las funcionalidades de las que dispone la red social, desde añadir fotos, añadir amigos, aceptar solicitudes de amistad, rechazarlas, enviar mensajes, responder, borrar, realizar valoraciones, etc.

9.1. Portal de compraventa.

Empezamos viendo la página principal de la aplicación:

Imagen 47. Pantalla principal de la aplicación.

Desde aquí podemos hacer varias cosas, por ejemplo podemos consultar un artículo, hacer una búsqueda acotada de productos, o navegar por las opciones de menú, que son pocas porque no estamos registrados y aun así nos pedirá que nos identifiquemos. También

podemos identificarnos, pedir a la aplicación que nos recuerde la contraseña o registrarnos. Consultamos algún producto que este dado de alta y la pantalla resultante de seleccionar un producto y darle al botón es:

Imagen 48. Consulta de artículo con un usuario no identificado.

Podemos pulsar sobre una foto y ver la galería de imágenes para ese producto:

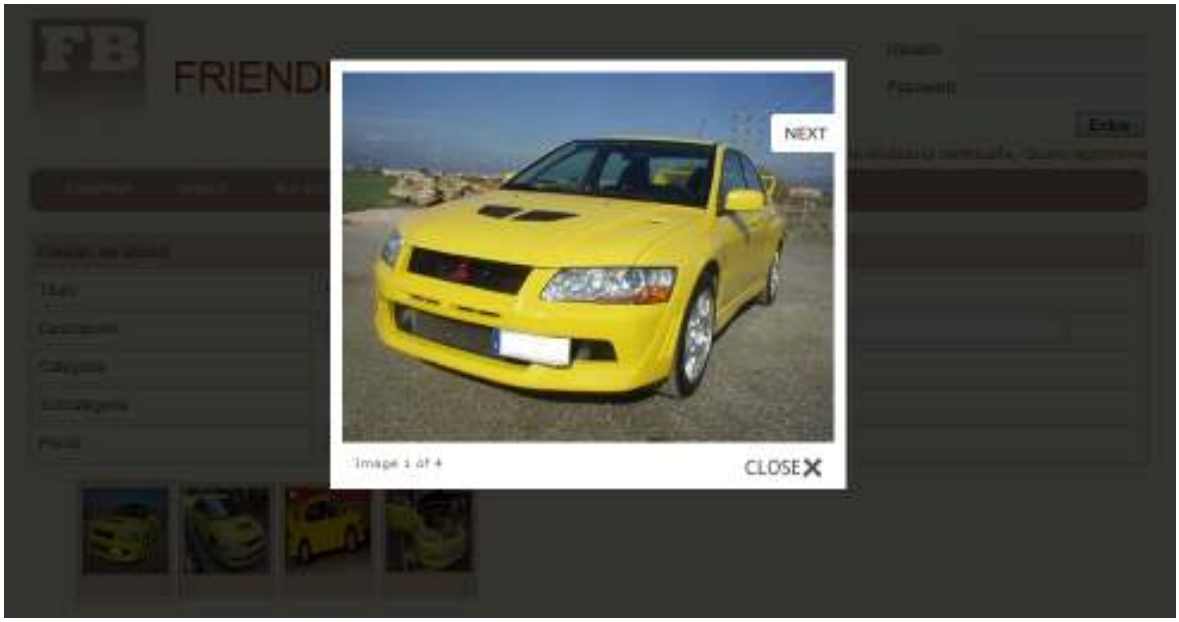


Imagen 49. Consulta de imágenes de un producto.

Volviendo a la página principal por las opciones de menú, vamos a hacer una búsqueda por nombre, utilizando los criterios de búsqueda. Para el ejemplo buscaremos todos los productos que de titulo tengan Mitsubishi y que su estado sea 'En venta':

Imagen 50. Criterios de búsqueda.

Vemos que para esos criterios de búsqueda nos aparece un solo producto.

Ahora, si quisiéramos comprarlo deberíamos consultar el artículo y dentro de la consulta pulsar el botón de comprar, pero como no tenemos visibilidad sobre ese botón porque no estamos identificados, vamos a registrarnos para poder comprar o reservar ese producto. Para ello, en la esquina superior derecha, pulsamos sobre el enlace que poner 'Quiero registrarme' y accedemos al formulario de alta:

Imagen 51. Formulario de alta de usuario

Una vez hemos pulsado al botón, si al repetir la contraseña introducida no es igual, nos saldrá una ventana emergente informándonos que no es correcto:

Imagen 52. Mensaje de alerta de la verificación de contraseña.

De igual manera, si el email introducido ya existe, la aplicación informa que esa dirección ya se utiliza y te ofrece la posibilidad de recuperar la contraseña:

Imagen 53. Mensaje de alerta de correo electrónico repetido.

Vamos a recuperar la contraseña, pulsamos sobre el enlace y nos lleva a la pantalla de recuperación de contraseña:

Imagen 54. Pantalla de recordatorio de contraseña.

Una vez hemos introducido el correo electrónico al que nos habíamos registrado y pulsamos el botón de 'Recuperar' la aplicación nos informa que se han enviado los datos a nuestro correo electrónico.

Imagen 55. Email de recordatorio de contraseña.

Una vez recuperada la contraseña, nos identificamos en la aplicación. El próximo paso va a ser pedirle al usuario del artículo que nos lo reserve para poder pensárnoslo mediante la imagen que nos permite enviar un mensaje privado al usuario:

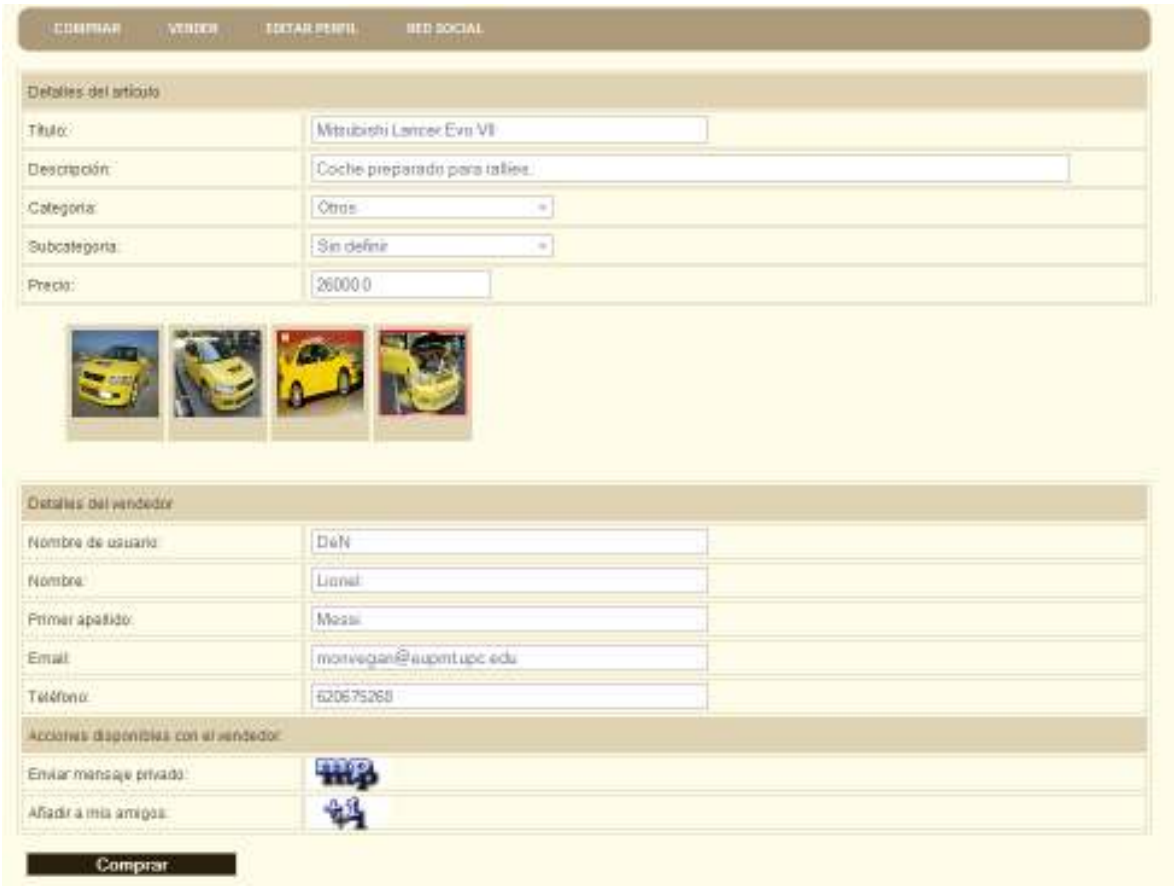


Imagen 56. Consulta de artículo con usuario identificado.

Y el hecho de pulsar encima de la imagen nos lleva a la parte de la red social y más concretamente, a la pantalla de nuevo mensaje privado:

Imagen 57. Redacción de mensaje privado.

Si el usuario acepta reservarnos el artículo, éste ira al menú de su artículo y lo reservara para nosotros. Ningún otro usuario podrá comprar el artículo pero si podrán verlo, solo nosotros podremos comprar el artículo.

Ahora seguiremos el flujo de la aplicación dando de alta un producto. Para ello debemos ir a la opción de menú de vender y nos muestra la siguiente pantalla:

Imagen 58. Pantalla de listado de productos del usuario.

En esta pantalla podemos hacer todas las acciones que se pueden hacer desde la principal y además gestionar nuestros productos. El listado de productos mostrará nuestros productos y el buscador los filtrará según nuestras necesidades.

El siguiente paso va a ser dar de alta un producto, para ello pulsamos el botón de 'Alta'. La aplicación nos muestra el siguiente formulario:

Imagen 59. Formulario de alta de artículo.

Una vez rellenados los datos y puesto a la venta nuestro producto, la aplicación nos da la oportunidad de dar de alta imágenes para ese producto, la pantalla que muestra es la siguiente:

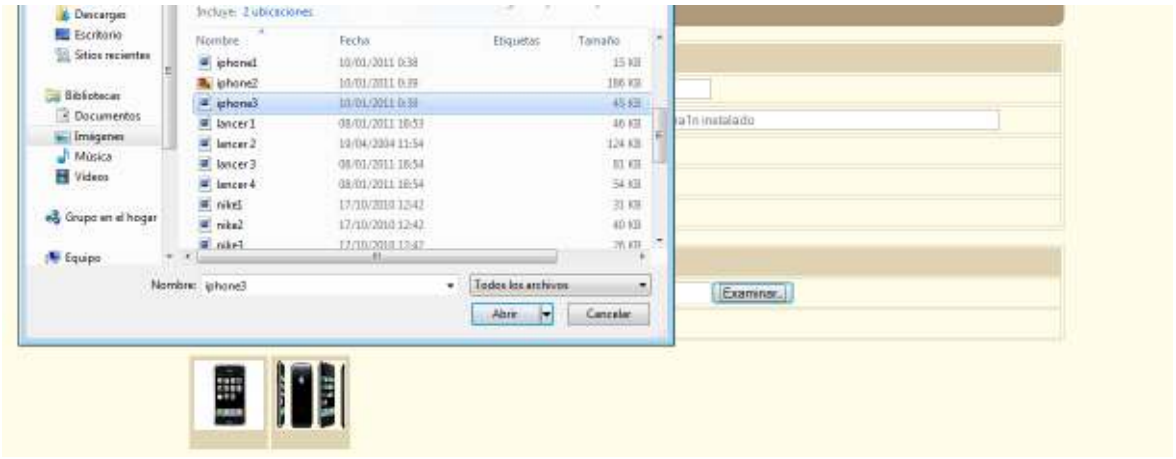


Imagen 60. Pantalla alta artículo con formulario para imágenes.

Le damos a examinar y nos permite buscar la foto en nuestro ordenador. La seleccionamos, le damos a abrir y la subimos pulsando el botón. Una vez hecho esto, ya lo podremos ver en nuestro menú de vender, en el listado de nuestros artículos:

Imagen 61. Listado de productos de usuario.

Para modificar el artículo tendríamos que seleccionarlo y pulsar el botón de modificar, la pantalla que nos saldría será la misma que la del formulario de alta pero con los campos informados:

Imagen 62. Pantalla de modificación de artículo.

Modificamos por ejemplo, el importe y le ponemos que cuesta 5 € más y aceptamos:

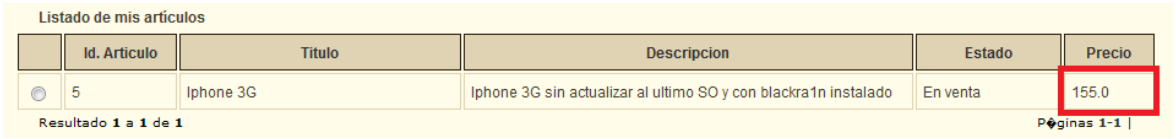


Imagen 63. Artículo modificado.

Como podemos ver, el precio ha sido modificado y ahora, en el listado de los productos del usuario, ya sale el registro modificado. La pantalla de consulta es idéntica a la que tenemos en el listado del inicio, la de los productos para comprar.

Si un usuario nos quisiera reservar un articulo, debería seguir los pasos que hemos seguido nosotros para reservar un articulo. En el caso de que alguien nos lo pidiera vía mensaje privado, nosotros iríamos a la pantalla para gestionar la reserva mediante el listado de nuestros productos, seleccionarlo y pulsando el botón 'Gestionar reserva'. Ahí deberíamos poner el nombre de usuario del usuario que nos lo ha pedido y pulsar aceptar. Veamos la pantalla:

Imagen 64. Formulario de reserva.

Cuando ponemos el nombre y confirmamos la reserva nos aparecerá un mensaje de confirmación para que confirmemos la reserva:

Imagen 65. Alerta de confirmación de reserva.

Ahí aceptamos y ya queda el producto reservado. Si por cualquier razón el nombre introducido no existiera saldrá un mensaje en rojo informando de que el usuario introducido no existe:

Imagen 66. Alerta de usuario inexistente.

Una vez hemos logrado poner el producto en reserva para el usuario que nos lo pidió, solo este usuario podrá comprar el producto y ningún otro podrá, aunque si podrá consultarlo. Si nosotros decidimos ponerlo de nuevo a la venta, desde la pantalla del listado de nuestros productos, volvemos a seleccionar el producto y pulsamos 'Gestionar reserva'. Esta vez la pantalla que nos muestra es igual solo que el nombre del usuario para el que está reservado se muestra no editable y el botón muestra el texto de 'Poner en venta':

Imagen 67. Formulario de reserva.

Para deshacer la reserva pulsamos el botón y nos saldrá un mensaje que nos pedirá confirmación para ponerlo de nuevo a la venta. Aceptamos o cancelamos según nuestras necesidades.

Para comprar un producto nos vamos al apartado de menú de comprar y nos aparece el listado de productos a la venta y para reservar. Seleccionamos el artículo y consultamos. Para comprar hay que pulsar el botón 'Comprar'. Nos aparecerá un mensaje de confirmación informándonos que la forma de pago es contrareembolso ya que no se ha implementado ninguna pasarela de pago y que la política de devoluciones hay que acordarla con el vendedor, solo nos queda aceptar o cancelar:

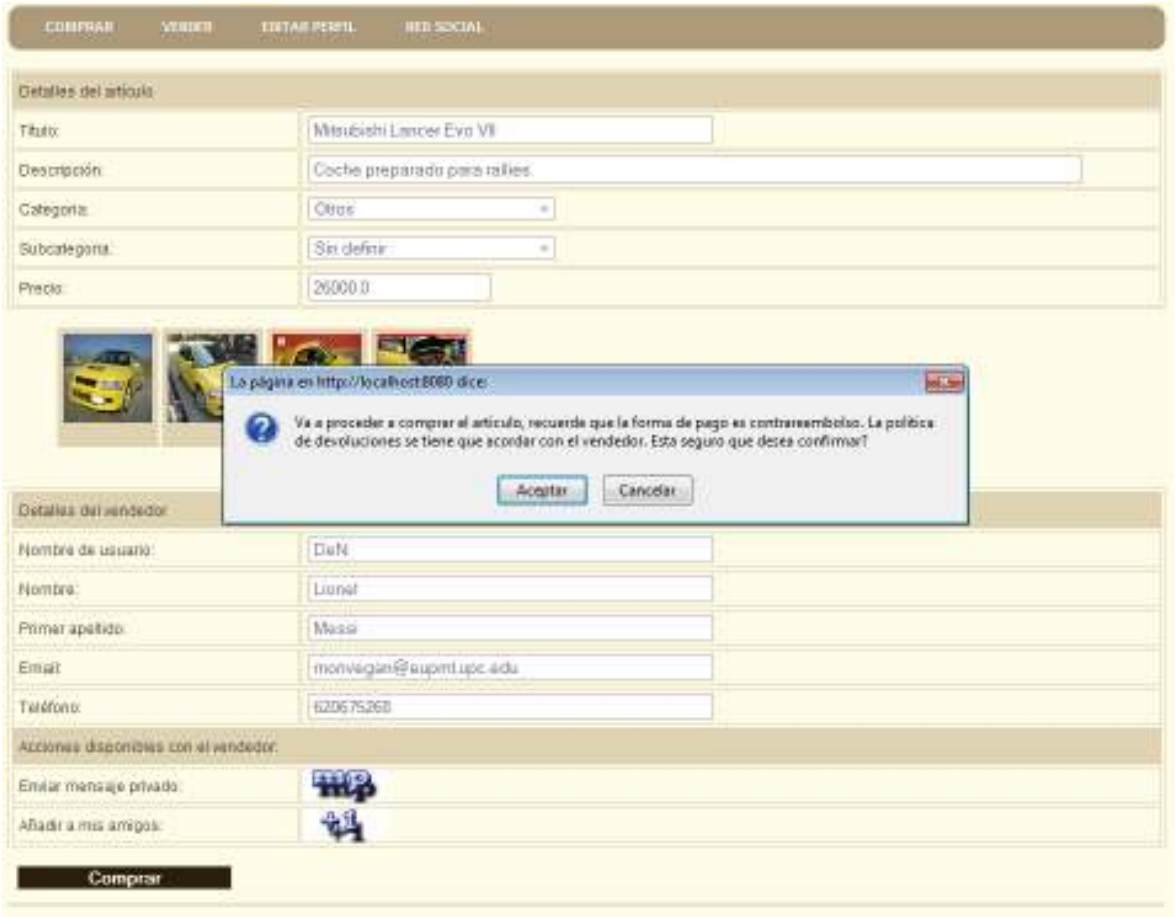


Imagen 68. Mensaje de confirmación de compra de producto.

En cuanto aceptamos, nos llega un correo electrónico a la dirección que dimos al registrarnos informándonos que la compra se ha efectuado con éxito. Al vendedor le llega otro mensaje informándole que se ha vendido el producto y que debe enviárselo al usuario comprador. Este es el mensaje que recibe el comprador:

Imagen 69. Correo de confirmación de compra de artículo.

Y el vendedor recibirá un correo como el siguiente:

Imagen 70. Confirmación de venta de artículo.

En este momento, el usuario vendedor deberá ponerse en contacto con nosotros para poder saber la dirección de envío. De esta manera siempre sabremos cuando el vendedor decidirá enviarlo y no habrá compromisos de pago si el vendedor se demora en el envío etc.

Por último, desde la parte del portal podremos modificar nuestros datos de usuario. Para ello, desde el portal, debemos dirigirnos a la opción de menú que poner 'Editar perfil'. Ahí nos saldrá un formulario con nuestros datos:

Imagen 71. Formulario de edición de usuario.

Nos lo permitirán modificar todo a excepción del NIF (que se supone que no cambia) y el nombre de usuario puesto que es identificativo para otros usuarios.

9.2. Red social.

Toda esta funcionalidad es la que podemos utilizar en la parte del portal de compraventa, para disfrutar del entorno de la red social, nos podemos dirigir desde el menú, en la pestaña de 'Red social'. Cuando presionamos sobre la opción de menú, nos cambia el entorno y nos llevará a la página principal del perfil del usuario conectado:

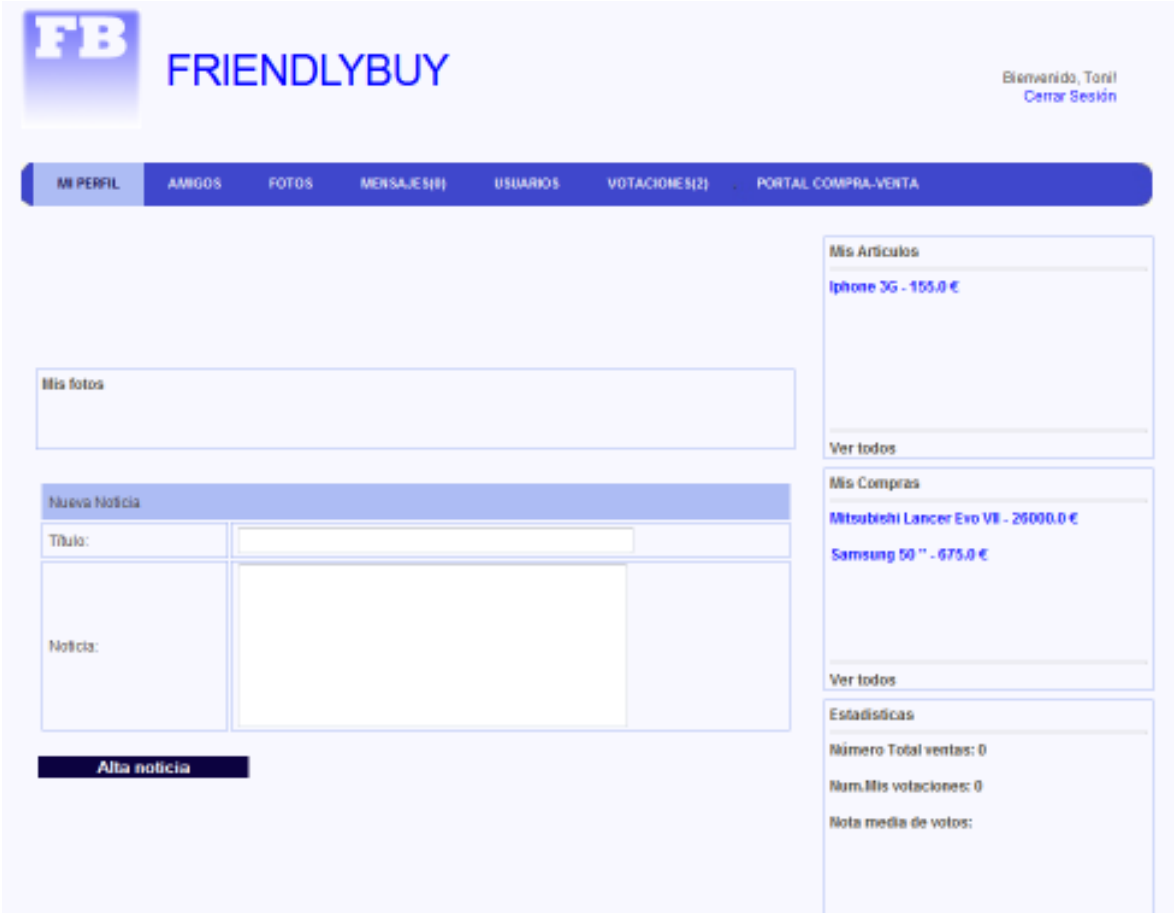


Imagen 72. Pantalla principal de la red social.

Desde el menú principal podremos acceder a todas las opciones que dispone la red social. Como en el entorno de compraventa, en la barra superior encontramos las diferentes opciones de las que disponemos para movernos por este entorno. En el cuerpo de la página

podemos encontrar en la parte de la izquierda superior un cuadro donde irán apareciendo las fotos que el usuario vaya poniendo para su perfil. Debajo está el formulario de alta de noticias, en las que el usuario podrá poner publicaciones y que otros usuarios las vean que por ejemplo, podrían ser futuros productos en venta, estados del usuario, etc. A la derecha vemos tres recuadros, el superior nos indica los productos que tenemos en venta, el del medio los productos que hemos comprado y el último las estadísticas de las valoraciones que otros usuarios nos han puesto por la venta de nuestros productos.

Para ir por partes, vamos a empezar con la publicación de una nueva noticia en la que anunciamos que próximamente pondremos a la venta teléfonos móviles de una marca. El procedimiento a seguir seria informar los campos y pulsar el botón que aparece debajo del formulario:

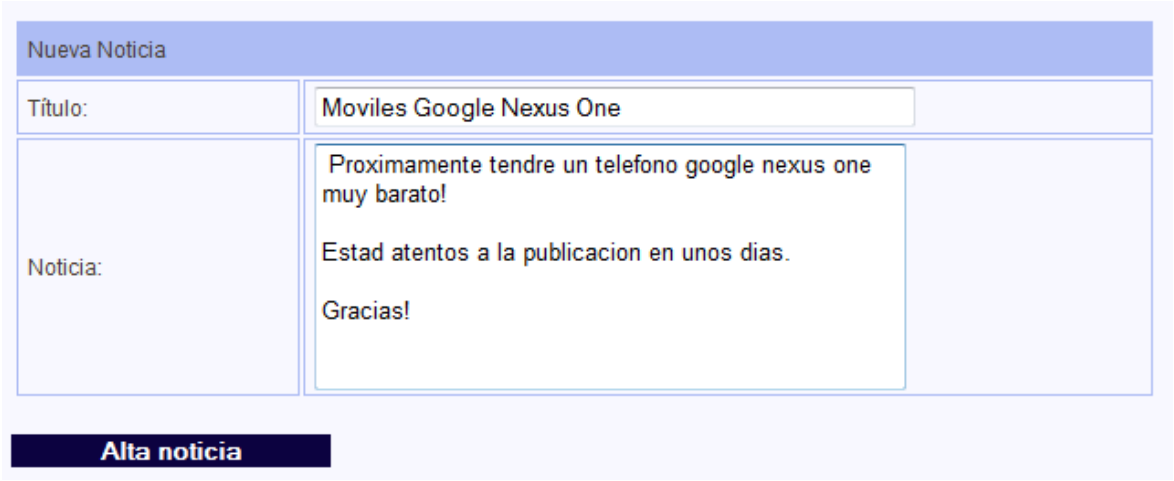


Imagen 73. Formulario de alta de noticia.

Una vez dada de alta la noticia podremos ver la publicación en nuestro perfil:

Imagen 74. Pantalla principal con noticia dada de alta.

También podremos dar de alta fotos para que otros usuarios puedan verlas. Estas fotos aparecerán en el recuadro que pone 'Mis fotos'. Para dar de alta fotos iremos a la opción del menú que pone 'Fotos', la pantalla que nos aparece será la siguiente:

Imagen 75. Formulario de alta de fotografías de usuario.

De la misma manera que dábamos de alta fotos para los productos, tenemos que darle al botón de examinar, nos aparecerá el navegador de carpetas, escogemos la foto y la subimos

mediante el botón de ‘Subir imagen’. Una vez subida la apariencia en esta pantalla es la siguiente:

Imagen 76. Formulario de alta de imágenes con fotografía insertada.

Y desde la pantalla principal de nuestro perfil, se mostrará, como ya hemos comentado, en el recuadro de ‘Mis fotos’:

Imagen 77. Pantalla principal de red social con imagen incluida.

Desde el perfil podemos consultar los productos que hemos puesto a la venta mediante el primer cuadro de la derecha del cuerpo de la página. Tenemos dos opciones, ir al listado o ir directamente a la pantalla de consulta del producto. Para la primera debemos pulsar el enlace que pone 'Ver todos' y para la segunda, pulsando directamente sobre el artículo.

Si pulsamos en el enlace de ver todos, la pantalla que nos muestra la aplicación es similar a la pantalla que disponemos en el entorno de compraventa en el que vemos todos nuestros productos, solo que no disponemos de buscador:

Imagen 78. Consulta del listado de artículos de usuario desde la red social.

Como vemos, el estilo es otra vez igual al del portal de compraventa, pues creemos que si consultamos un producto debe seguir una línea definida.

En el caso de querer consultar únicamente uno de los artículos que nos muestra el cuadro, pulsamos sobre el enlace y nos mostrará el detalle del artículo:

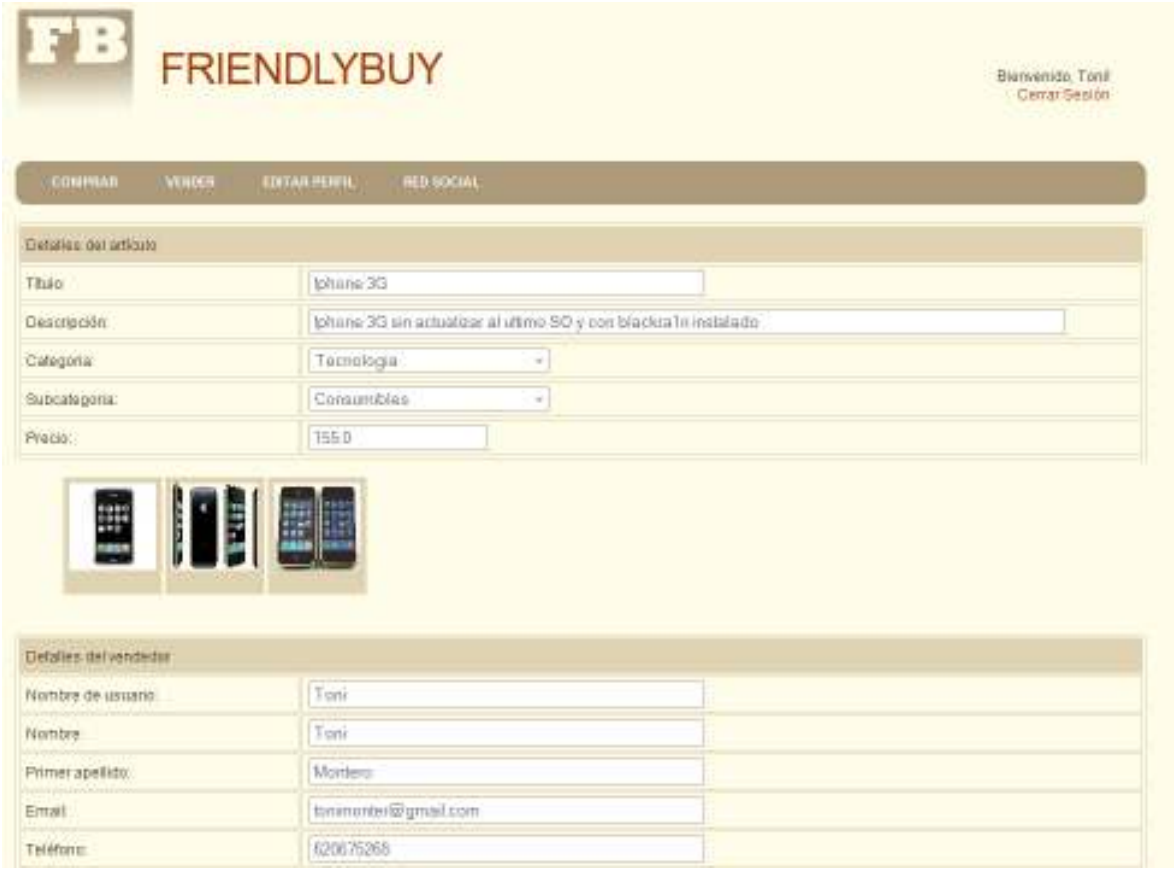


Imagen 79. Consulta de un artículo del usuario desde red social.

De igual manera con los artículos que hemos comprado, la consulta de ambos es igual y en el mismo entorno y estilo:

Imagen 80. Consulta de un artículo comprado desde la red social.

Y para todos los productos:

Imagen 81. Consulta de todos los productos comprados desde red social.

El último cuadro están las estadísticas de las valoraciones de otros usuarios, si nos fijamos en el cuadro veremos las ventas totales, las valoraciones totales y la nota media de estas valoraciones:

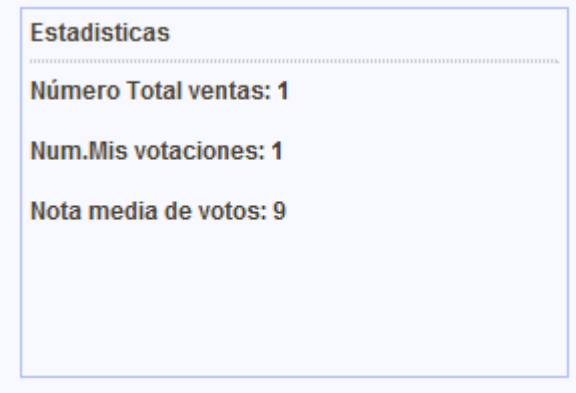


Imagen 82. Cuadro estadístico del perfil de usuario.

Si queremos ver los amigos disponibles o solicitudes de amistades debemos ir a la opción de menú 'Amigos' donde se mostrarán dos tablas con las solicitudes y las amistades confirmadas, en el caso que no haya de alguna saldrá un mensaje informando que no hay solicitudes o amistades:

Imagen 83. Pantalla de consulta de amistades.

Si quisiéramos añadir un usuario como amigo tendríamos que ir a la opción de menú usuarios y buscar un usuario mediante el buscador o, en el caso de que este en la lista utilizar las opciones disponibles que hay para ese usuario:

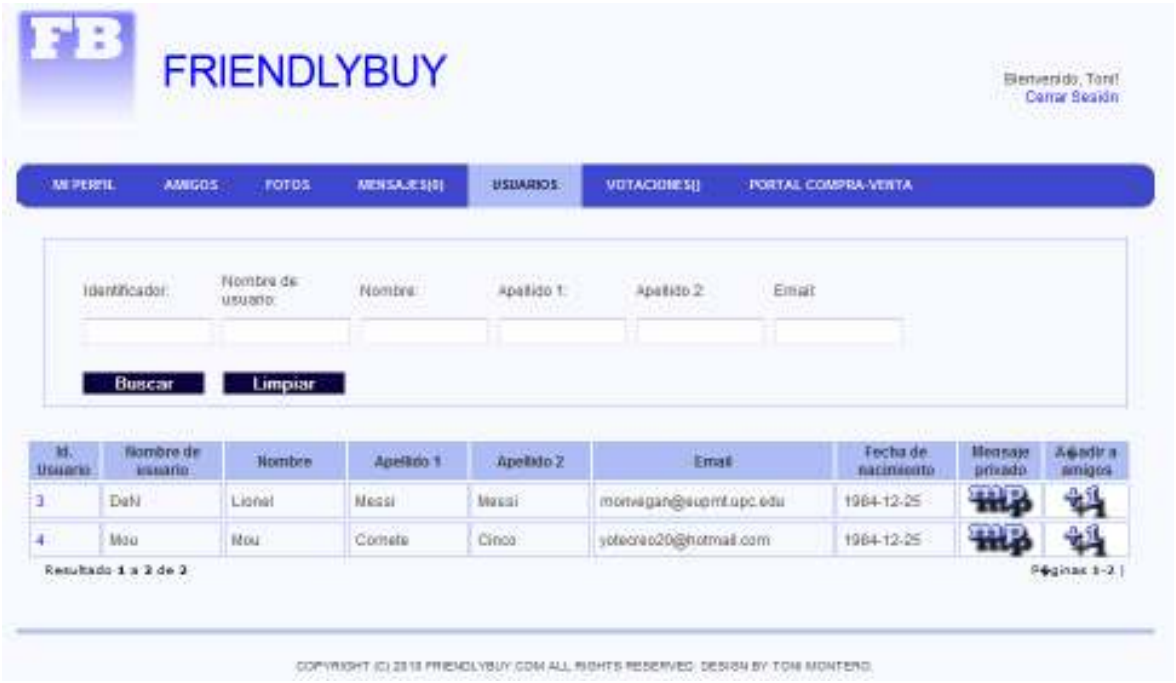


Imagen 84. Pantalla del listado de usuarios con buscador.

La primera imagen que muestra ‘mp’ sirve para enviar un mensaje privado al usuario y el siguiente es para crear una solicitud de amistad. El hecho de ser una solicitud significa que el otro usuario, el que añadimos, debe aceptar esa solicitud aunque también podría rechazarla. Si ya hemos realizado una solicitud a un usuario el sistema nos lo informa lo que quiere decir que el usuario destinatario aún no ha contestado. Cuando el otro usuario entre al menú de ‘Amigos’ podrá ver la solicitud que ha realizado:

Imagen 85. Pantalla de amistades con solicitud de amistad.

En este momento debe rechazar o aceptar la amistad. Si rechaza todo quedará como al principio, en cambio, si acepta, en nuestro listado de amigos quedará reflejado como amigo:

Imagen 86. Pantalla de amistades con usuario añadido.

La ventaja de tener a usuarios como amigos es simplemente el tenerlos localizados en todo momento. Puedes acceder a su perfil mediante el link que hay pulsando sobre el identificador, puedes consultar el email o enviarle un mensaje privado mediante la imagen.

Otra de las características de la aplicación, que realmente es muy interesante debido a que en los portales de compraventa, nunca puedes saber si te quieren engañar o cosas por el estilo, es el tema de las votaciones o valoraciones. Cuando has comprado un artículo puedes votar al vendedor/producto mediante el menú de 'Votaciones':

Imagen 87. Pantalla de votaciones.

Lo que deberíamos hacer seria seleccionar el producto, que se supone que nos ha llegado y queremos valorarlo para que otros usuarios vean la calidad de los productos de ese vendedor, y pulsar el botón de 'Valorar producto'. Este apartado, nos redirigirá a una pantalla similar a la pantalla de consulta del artículo con la excepción de que aquí nos mostrará al final del todo un pequeño formulario en el que podemos introducir un número mediante un combo y una caja de texto para observaciones.

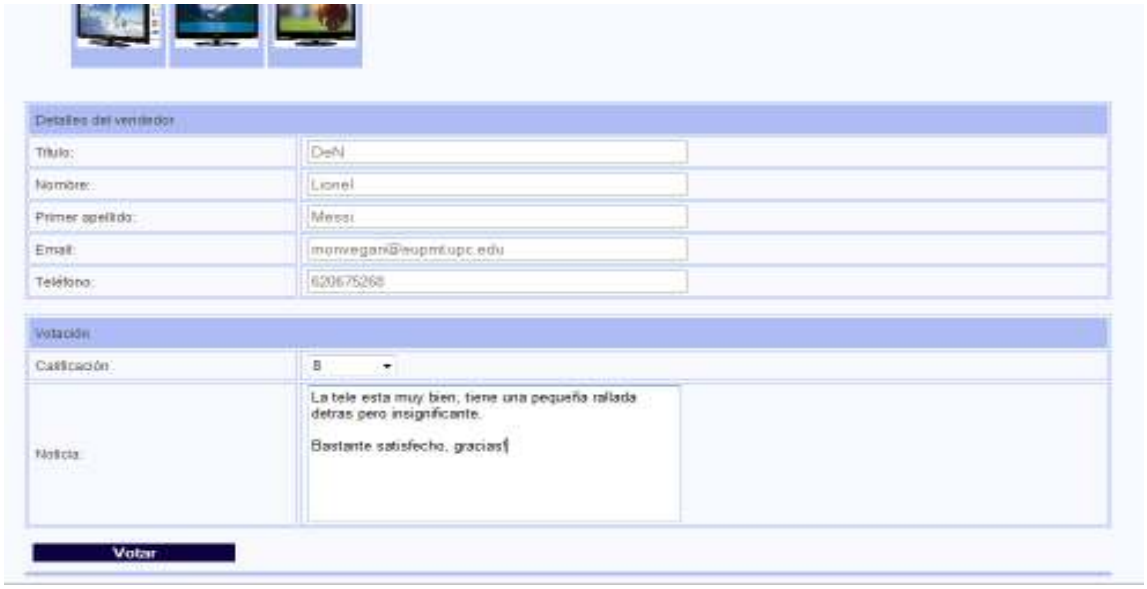


Imagen 88. Valoración de artículo comprado.

Una vez haya informado los datos necesarios, pulsamos el botón de 'Votar'. De esta manera, cuando un usuario entre en el perfil de otro, podrá ver las valoraciones en forma de estadística de este otro usuario y hacerse una idea, de si los productos que ha vendido tienen la calidad necesaria para ser comprados o ver si el usuario no es de fiar.

Por esto, vamos a ver si las votaciones que hemos realizado a otros usuarios, en concreto del usuario al que le hemos comprado el televisor, se le ha almacenado correctamente la valoración que le hemos hecho, para ello debemos ir a buscar al usuario, ya sea por el menú 'Amigos' o el de 'Usuarios' y vamos a su perfil:

Imagen 89. Pantalla principal de la red social.

Notamos que no podemos introducir noticias ya que esto es propio para cada usuario. A lo que íbamos, en el cuadro de estadísticas, pulsamos el enlace que pone ‘Ver Todas’ y nos aparecerá la ventana con las todas las votaciones que ha tenido este usuario:

Imagen 90. Pantalla de votaciones del usuario del perfil consultado.

Como podemos comprobar, nuestra opinión queda persistida en su perfil y cualquier usuario que quiera conocer las opiniones verá nuestra opinión sobre este otro usuario.

10. Conclusiones

Una vez finalizado el proyecto, podemos concluir que hemos conseguido los objetivos principales planteados inicialmente.

Un usuario podrá dar de alta productos, modificar, dar de baja, consultar, gestionar reservas y comprar productos. Hacer búsquedas de productos gracias a un filtro y dar de alta imágenes para un artículo.

Podrá contactar con otros usuarios mediante mensajes privados, podrá también añadir usuarios como amigos, consultar sus productos a partir de un perfil personal, sus compras, las fotografías que haya dado de alta en la aplicación, valorar a otros usuarios, consultar estadísticas de otros usuarios y propias y buscar usuarios.

A nivel personal, ha sido una experiencia muy enriquecedora, tanto personal como profesionalmente. A nivel personal, el hecho de marcar un objetivo y cumplirlo es muy gratificante, pero a nivel profesional, sobretodo, ha sido un reto el hecho de hacer un proyecto, integrando tecnologías que bien podrían ser todas las tecnologías que utiliza un proyecto en el mundo laboral, haberlo desarrollado desde cero, ha hecho que tecnológicamente haya madurado, a mi parecer, mucho.

11. Ampliaciones del proyecto

Decir que es un proyecto cerrado, sería una barbaridad ya que un proyecto de este tipo, dirigido al usuario, en un mundo tecnológico tan cambiante, siempre hay posibles ampliaciones y funcionalidades nuevas para añadir.

Si me propusieran seguir con el proyecto, yo mismo propondría nuevas funcionalidades como podrían ser incorporar un módulo de pago, ya sea propio o utilizando librerías ya hechas, por ejemplo el módulo de pago de PayPal. Otra funcionalidad que propondría e implementaría seria añadir la posibilidad de subastar artículos.

Todo esto en la parte del portal de compraventa, en cuanto a la parte de la red social se podría hacer un sistema de prestigio para usuarios, crear foros, añadir un chat interno, añadir juegos en formato flash, etc.

También podríamos añadir un panel de administración para gestionar usuarios, categorías, sub-categorías, o moderación de foros, chivato de productos/comentarios 'ilegales', entre otras cosas.

12. Bibliografía

[1] Craig Wells y Ryan Breidenbach, *Spring in action* (2004)

[2] http://static.springsource.org/spring/docs/2.0.x/reference/index.html (Noviembre 2010)

[3] http://www.developersbook.com/ibatis/iBatis-tutorials/iBatis-tutorials.php (Noviembre 2010)

[4] http://java.sun.com/products/javamail/javadocs/index.html (Noviembre 2010)

[5] http://leandrovieira.com/projects/jquery/lightbox/ (Noviembre 2010)

[6] Ted Husted, Cedric Dumoulin, George Franciscus, David Winterfeldt, Craig R. McClanahan, *Struts in action (2002)*

[7] Clinton Begin, Brandon Goodin and Larry Meadors, *Ibatis in action (2007)*

[8] William Crawford & Jonathan Kaplan *J2ee Design Patterns (2006)*

13. Anexo I (contenido del CD-ROM)

El contenido del CD-ROM es el siguiente:

- Documentación
  - Portada y resumen.
  - Índice.
  - Memoria.
- Scripts de bases de datos.
  - Script de creación de las tablas.
  - Script de inserción de datos de poblaciones y provincias.
  - Script de inserción de datos parametrizados (categorías, subcategorias, estados, etc.)
- Aplicación.
  - War de la aplicación lista para ser desplegada en servidor Tomcat 6.0.29
  - Código fuente de la aplicación.
  - Librerías usadas para el desarrollo y ejecución del proyecto.
- Software utilizado para el desarrollo.
  - Eclipse.
  - MySQL versión no instalable.
  - MySQL Administrator.
  - Servidor de aplicaciones Tomcat 6.0.29