TFG Java EE: Sistema TPV con módulo de fidelización de clientes.

Memoria

Juan Andrés Bravo Díaz

16 Junio 2014

ÍNDICE

1- Introducción

El presente proyecto Trabajo Fin de Grado, tiene como objetivo la realización de un sistema de fidelización de clientes para comercio, con módulo de TPV (Terminal Punto de Venta), todo ello realizado con tecnología JavaEE. La motivación principal ha sido la aplicación de los conocimientos adquiridos a lo largo del itinerario de Ingeniería del Software así como adquirir nuevas competencias durante todo el proceso de desarrollo del proyecto.

2- Justificación

En el ámbito comercial actual, en tiempos de crisis, es fundamental atraer a los clientes, pero aún más importante es retener a estos clientes ofreciendoles beneficios tangibles que les haga regresar al comercio para seguir consumiendo y aumentar las ventas. Para ello, una de las herramientas más eficaces es la fidelización de clientes que consiste en ofrecer recompensas por cada compra realizada, ya sea en descuentos, promociones o regalos.

En concreto, el proyecto que nos ocupa va a permitir gestionar un sistema de fidelización basado en la acumulación de un porcentaje por cada compra realizada por el cliente. Se pretende además que el cliente pueda consultar sus recompensas desde Internet y utilizarlas como mejor crea conveniente.

Por otro lado, es necesario facilitar la tarea al encargado de la tienda física, evitando cualquier tipo de carga administrativa a la hora de aceptar o asignar recompensas. Para ello, el sistema desarrollado descargará totalmente de trabajo adicional al vendedor, el cual solo validará cheques descuento y procederá a realizar ventas desde el TPV, el cual se encargará de realizar todos los procesos de suma y descuento de saldos de cliente.

El proyecto ha sido llamado ECOMICS ya que actualmente, por circustancias laborales, regento un comercio dedicado al mundo del comic, juegos de mesa y merchandise. Una de las principales motivaciones a la hora de desarrollar este proyecto, es la aplicación práctica que va a tener el sistema, ya que pretendo hacer uso de el, implantándolo en la página web y en el tpv de la tienda.

3- Objetivos

Desarrollar una aplicación en JavaEE que sea accesible vía Internet, tanto por clientes como por los vendedores del comercio. Los clientes podrán registrarse online, consultar y modificar sus datos y, además, consultar su saldo de puntos y la posibilidad de imprimir cheques descuento que podrán utilizar a posteriori en el comercio físico.

Los vendedores podrán acceder al listado de usuarios, dar de alta nuevos artículos y acceder al TPV desde el cual realizarán las ventas del comercio físico. El principal reto es la gestión de cheques descuento, que han de poder ser usados en el TPV como un artículo cualquiera, para ello será necesario que el cheque descuento impreso por el cliente disponga de un código de barras único que pueda ser leido durante la venta.

Como se ha comentado en el punto anterior, pretendo dar un uso práctico al software desarrollado, por lo tanto un objetivo fundamental es disponer de un software totalmente funcional, que pueda ser utilizado en el local comercial e integrado en una página web que actualmente se está desarrollando. Además se pretenden implementar otras utilidades de servicio al cliente y a las actividades lúdicas que se desarrollan en el comercio, esto implica que el diseño deba ser altamente escalable permitiendo el desarrollo de nuevos módulos de forma transparente para los usuarios del sistema.

4- Planificación

La planificación inicial del proyecto ha sido la siguiente:

Nombre	Duracion	Inicio	Fin
TFG Java EE	68 días	13/03/2014	16/06/2014
PAC2 – Análisis y diseño	35 días	13/03/2014	17/04/2014
PAC3 – Implementación	46 días	18/04/2014	02/06/2014
Entrega Final	14 días	03/06/2014	16/06/2014

Los tiempos de planificación se han cumplido totalmente, aún incluso habiendo existido ciertas dificultades técnicas durante el inicio del proceso de implementación que han retrasado trabajo. Afortunadamente el proceso de implementación ha permitido recuperar ese tiempo perdido en tareas de configuración del entorno de desarrollo.

5- Análisis

Requisitos

Se necesita una aplicación de fidelización de clientes para un comercio. Se han de tomar en cuenta las siguientes consideraciones:

- Cada compra realizada por un cliente registrado le aporta a su saldo un 5% de dicha compra.

- El cliente podrá acceder a una web para consultar su saldo.

- El cliente puede imprimir vales de descuento (desde 5€) que podrá usar o regalar a otras personas. Para que esto sea así, a cada vale generado se le asigna un código de barras y, a todos los efectos, se considera un artículo en venta con valor negativo. En el terminal de ventas, el vendedor escaneará el vale descuento y quedará reflejado en la lista de venta.

- El sistema restará/sumará automáticamente los puntos al finalizar cada venta.

- El vendedor/administrador puede acceder a los listados de clientes y ver sus compras y saldos.

Análisis

Identificación de actores del sistema

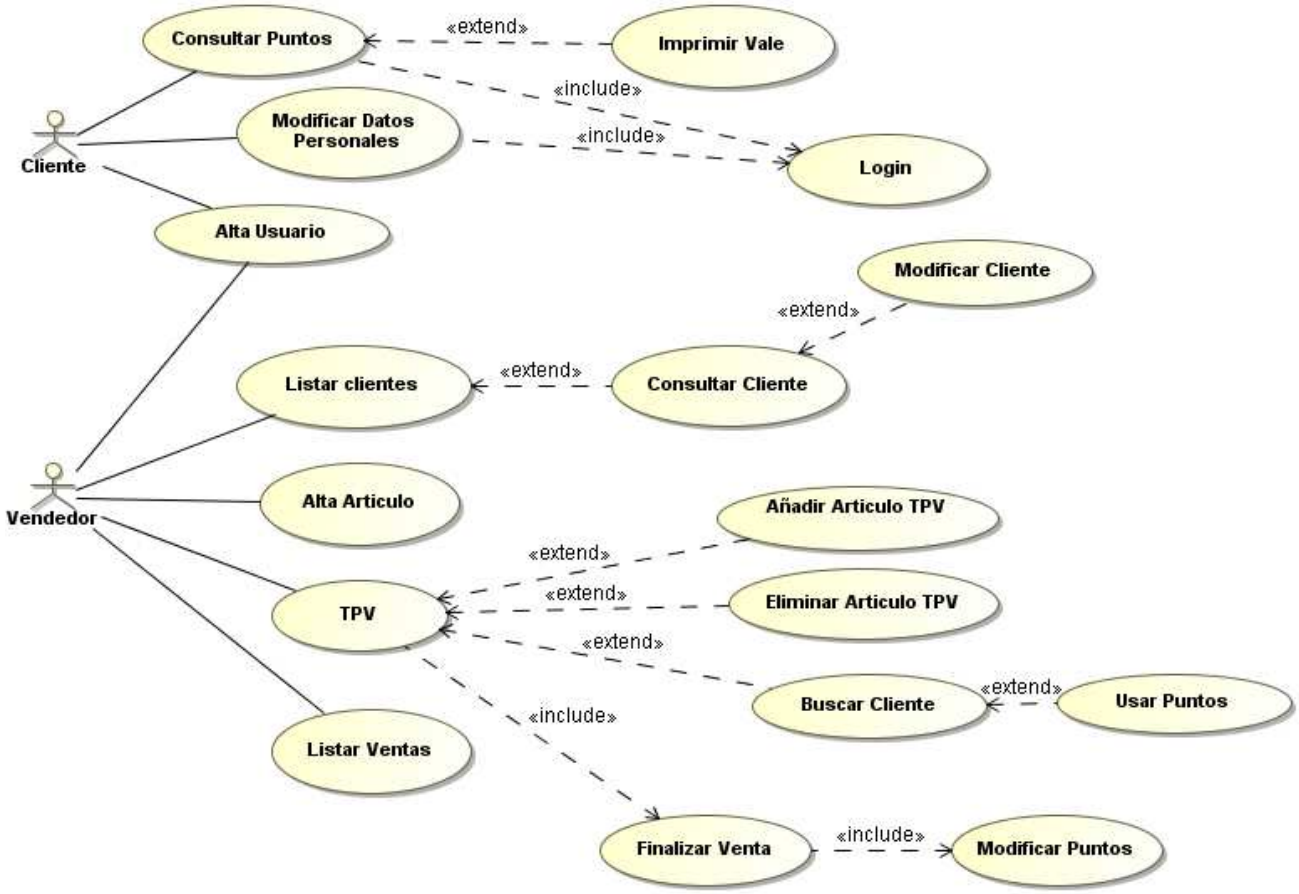
Se han identificado los siguientes actores que harán uso de la aplicación:

- **Cliente**: Cualquier usuario del comercio que esté registrado en el sistema. Las acciones que podrá realizar son las siguientes:
    - Registrarse en el sistema
    - Identificarse en el sistema
    - Modificar sus datos personales
    - Consultar puntos de fidelización
    - Imprimir vale descuento

- **Vendedor**: usuario del sistema con permisos de acceso al módulo de ventas TPV. Las acciones que podrá realizar son las siguientes:
    - Identificarse en el sistema
    - Registrar clientes
    - Consultar clientes
    - Alta de articulos en venta
    - Realizar ventas

Casos de uso

Se han identificado los siguientes casos de uso:



Para mejorar la legibilidad del diagrama se han omitido los “include” hacia el caso de uso **login** del actor **Vendedor**, obviamente la mayoría de acciones a realizar por este actor necesitan que esté se encuentre autenticado en el sistema.

Casos de uso textuales

1. Caso de uso **Login**
   1.1. Actor principal: **Cliente** y **Vendedor**
   1.2. Precondición: Ninguna
   1.3. Postcondición: El usuario está autenticado en el sistema
   1.4. Casos de uso relacionados: Todos aquellos que necesiten al actor autenticado en el sistema.
2. Escenario principal:
   2.1. El usuario introduce su usuario y contraseña
   2.2. El sistema comprueba si las credenciales son válidas
3. Flujos alternativos:
   3.1. Si el usuario no existe o la contraseña no es correcta, el sistema muestra mensaje de error en el login

1. Caso de uso **Consultar Puntos**
   1.1. Actor principal: **Cliente**
   1.2. Precondición: Cliente autenticado en el sistema
   1.3. Postcondición: Ninguna
   1.4. Casos de uso relacionados: **login**, **imprimir vale**
2. Escenario principal:
   2.1. El usuario se conecta al sistema
   2.2. El sistema muestra el saldo en puntos de fidelización del cliente

1. Caso de uso **Modificar Datos Personales**
   1.1. Actor principal: **Cliente**
   1.2. Precondición: Cliente autenticado en el sistema
   1.3. Postcondición: Ninguna
2. Escenario principal:
   2.1. El usuario solicita la modificación de datos en la vista principal de cliente
   2.2. El sistema muestra los datos del cliente
   2.3. El cliente modifica sus datos
3. Flujos alternativos:
   3.1. El cliente cancela la modificación de datos

1. Caso de uso **Alta usuario**
   1.1. Actor principal: **Cliente**
   1.2. Precondición: Ninguna
   1.3. Postcondición: Ninguna
2. Escenario principal:
   2.1. El usuario accede al formulario de registro de nuevo cliente
   2.2. El sistema muestra el formulario de registro
   2.3. El cliente introduce sus datos
   2.4. El sistema registra el nuevo usuario
3. Flujos alternativos:
   3.1. Si el usuario existe o no ha introducido correctamente todos los datos el sistema muestra mensaje de error

1. Caso de uso **Listar Clientes**
   1.1. Actor principal: **Vendedor**
   1.2. Precondición: el usuario debe estar autenticado en el sistema
   1.3. Postcondición: Ninguna
2. Escenario principal:
   2.1. El sistema muestra un listado con todos los clientes registrados en la aplicación

1. Caso de uso **Alta Articulo**
   1.1. Actor principal: **Vendedor**
   1.2. Precondición: el usuario debe estar autenticado en el sistema
   1.3. Postcondición: Ninguna
2. Escenario principal:
   2.1. El sistema muestra el formulario de alta de artículo nuevo
   2.2. El vendedor introduce los datos del artículo nuevo
   2.3. El sistema registra el nuevo artículo
3. Flujos alternativos:
   3.1. Si el artículo existe o no se han introducido correctamente todos los datos el sistema muestra mensaje de error

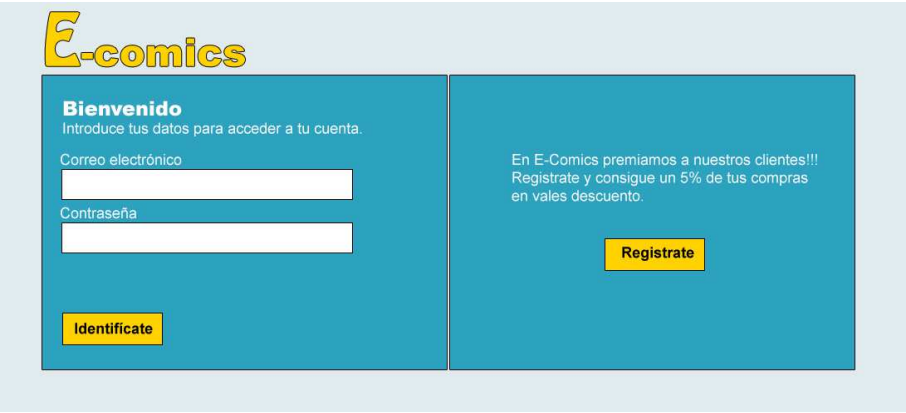
1. Caso de uso **Listar Ventas**
   1.1. Actor principal: **Vendedor**
   1.2. Precondición: el usuario debe estar autenticado en el sistema
   1.3. Postcondición: Ninguna
2. Escenario principal:
   2.1. El sistema muestra un listado con todos las ventas realizadas en el sistema

1. Caso de uso **TPV**
    1.1. Actor principal: **Vendedor**
    1.2. Precondición: el usuario debe estar autenticado en el sistema
    1.3. Postcondición: Ninguna
2. Escenario principal:
    2.1. El vendedor añade artículos a la lista de compra
    2.2. El vendedor finaliza venta
3. Flujos alternativos:
    3.1. El vendedor busca a un cliente
        3.1.1. El vendedor selecciona los puntos de fidelización que va a usar el cliente
        3.1.2. El vendedor finaliza venta
        3.1.3. El sistema suma/resta puntos del saldo del cliente
    3.2. El vendedor añade vale descuento a la lista de la compra
        3.2.1. El vendedor finaliza venta
        3.2.2. El sistema elimina vale pendiente del cliente

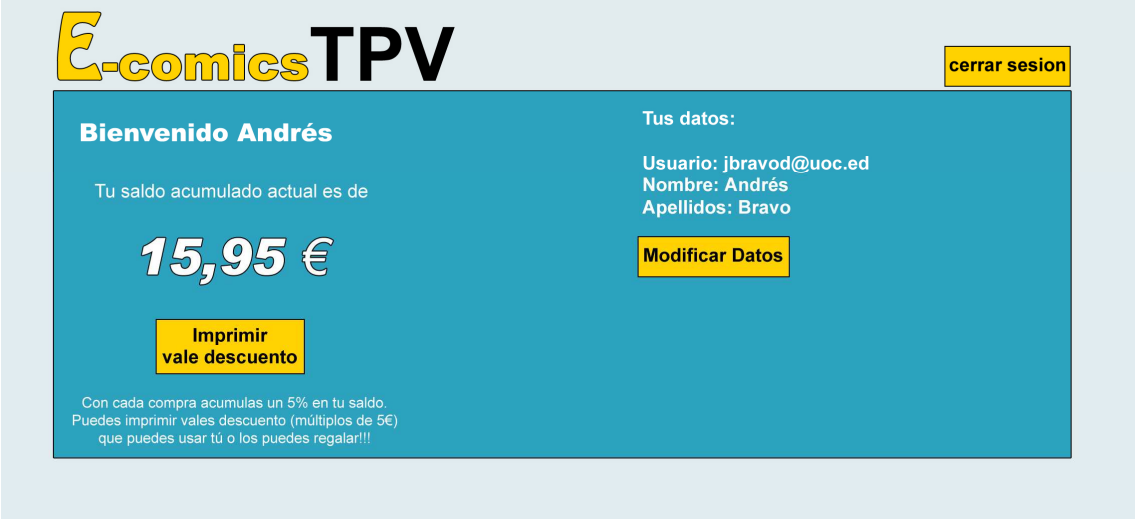
Cabe destacar, que el sistema permite que el vendedor pueda aplicar descuentos al cliente desde el terminal de ventas, sumando o restando automáticamente los puntos de la venta (en principio un 5% del gasto). Sin embargo, si un cliente imprime un vale descuento, se le descuenta directamente de su saldo. Un vale descuento se instancia en el sistema como un artículo más con coste negativo y queda cancelado en el momento que es utilizado en una venta.

Prototipo

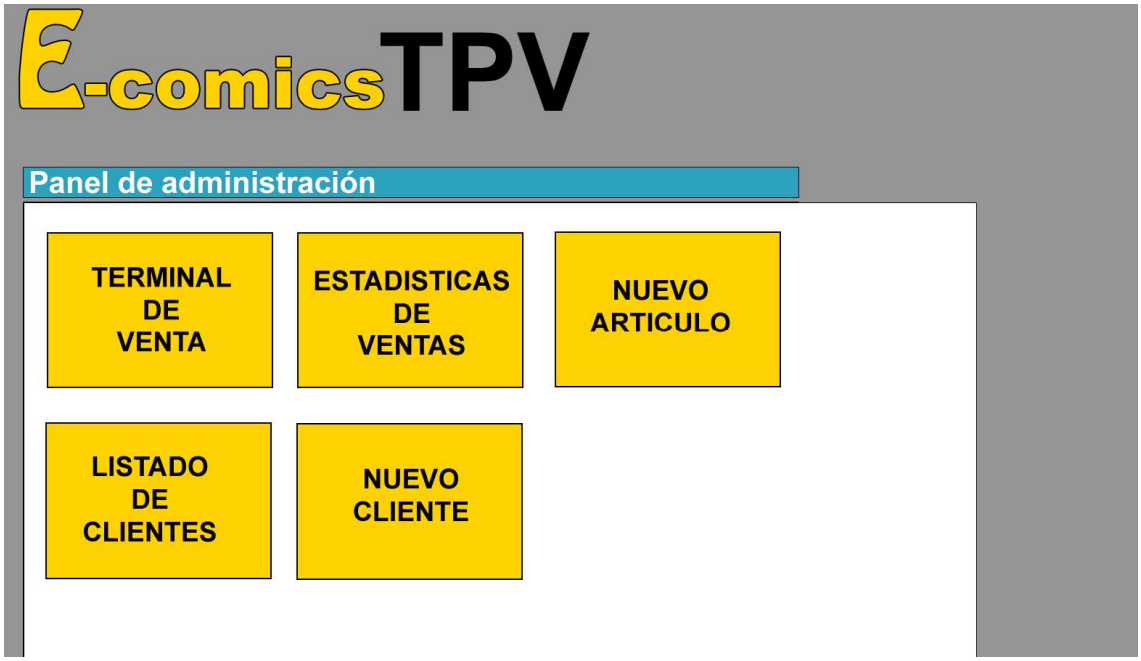
Se desea realizar una aplicación visualmente sencilla en la que además, el cliente no tenga que introducir un gran número de datos personales. En el acceso a traves de navegador web, el usuario llega a la pantalla de identificación en el sistema. En el caso de no disponer de usuario, podrá realizar el registro desde esta misma pantalla:



Una vez que el usuario ha proporcionado correctamente sus credenciales de acceso a la aplicación, es informado del saldo acumulado por compras que dispone en ese momento. Así mismo, puede modificar sus datos, imprimir un vale descuento y, por último, cerrar sesión:

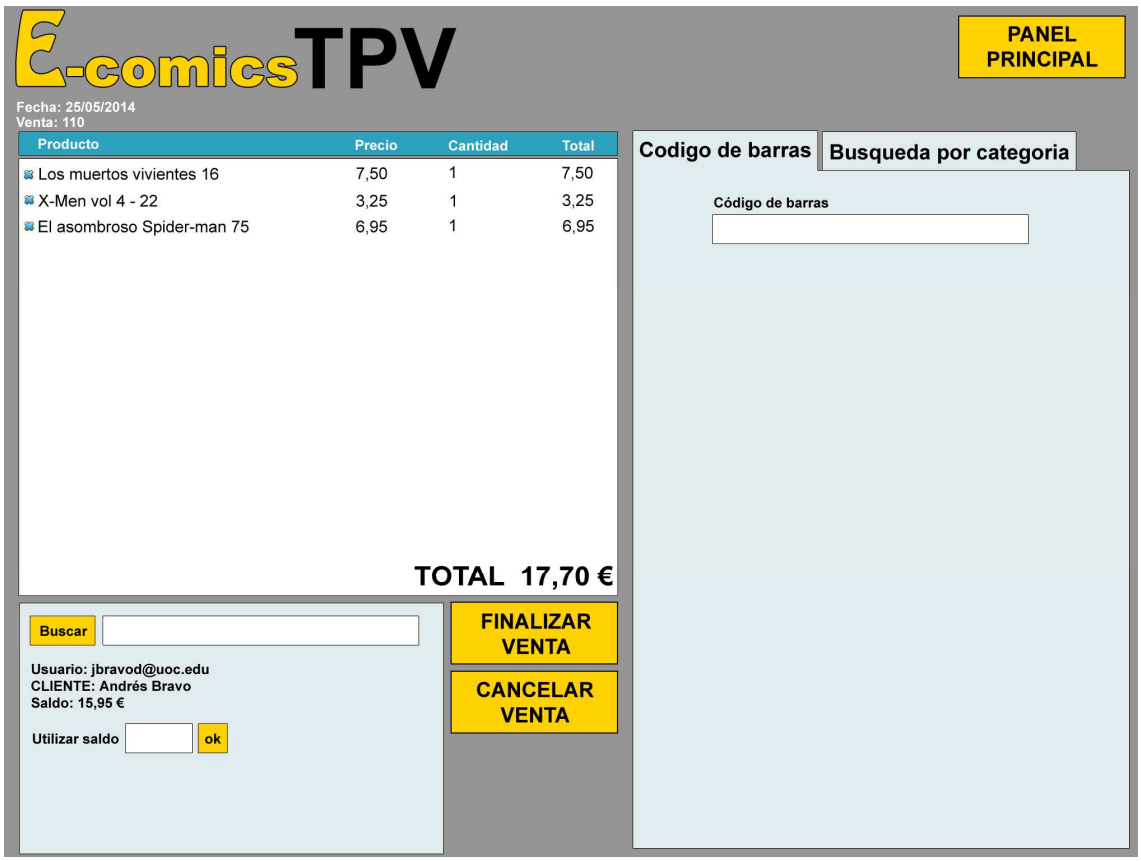


Si el usuario que accede al sistema es el administrador/vendedor, accederá directamente al panel de administración desde el cual podrá acceder a todas las funcionalidades previstas:



Se han omitido los prototipos de funcionalidades del tipo listados o formularios al considerarse triviales y, en todo caso, seguirán la misma estética elegida para el resto de la aplicación.

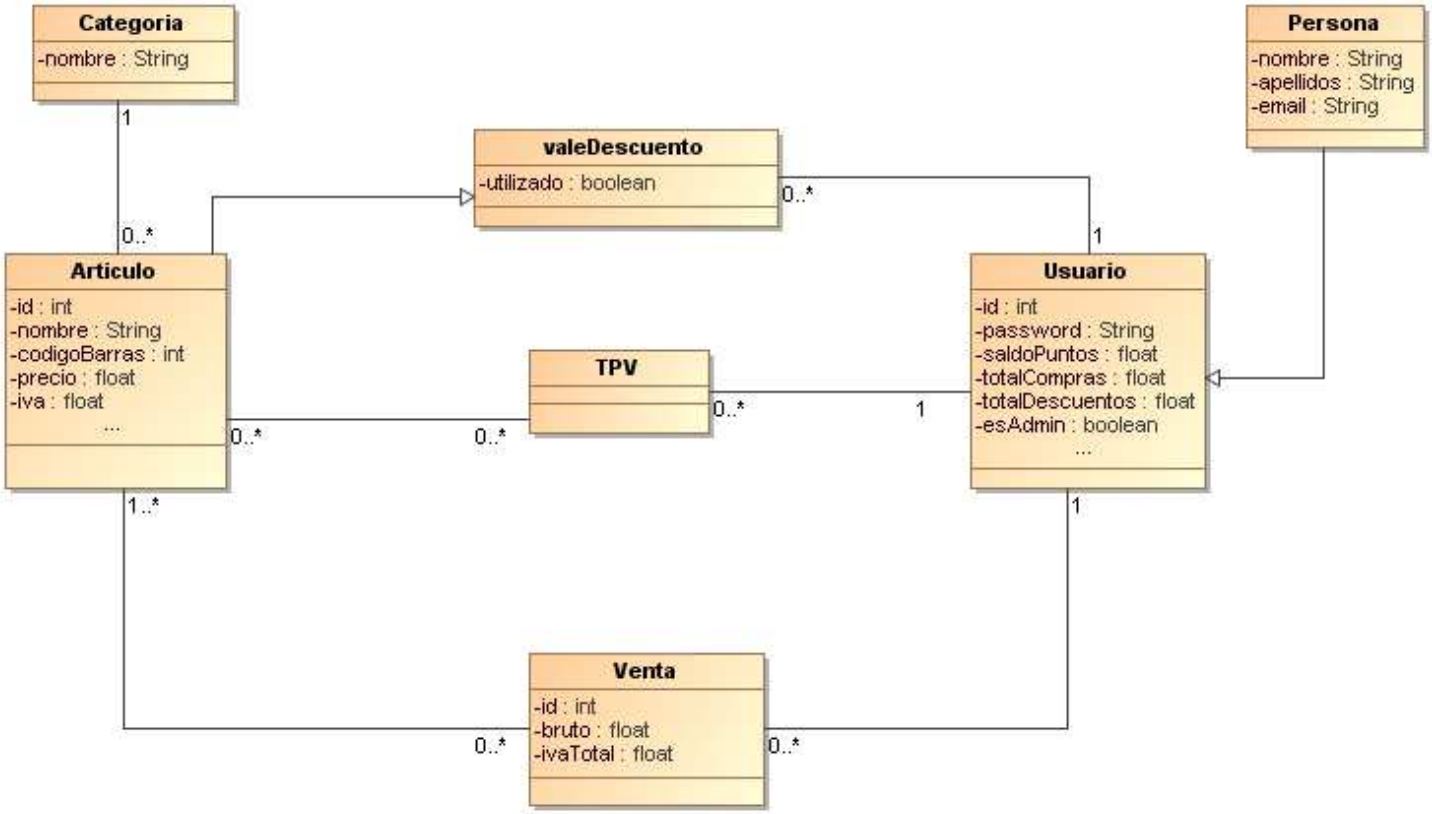
El panel Terminal de venta está dividido en: lista de compra, buscador de artículos (por código de barras o categoría) y busqueda de cliente:



6- Diseño

Diagrama de clases

En el punto de vista de la información se describe en detalle la información que tratará el sistema, básicamente las estructuras de datos, valores posibles y relaciones entre entidades. De todo ello, resulta el siguiente esquema invariante mediante diagrama de clases UML:



Base de datos

Para diseñar la base de datos, primero se deben determinar las entidades (tablas) que formarán la base de datos. Basando la decisión en el diagrama UML se obtienen las siguientes entidades:

- Categoria
- Artículo
- Usuario
- Venta
- ValeDescuento

Los atributos de cada tabla serán los siguientes (las claves principales aparecen subrayadas):

- Categoría:
  - **Nombre**

- Artículo:
  - **Id** (no todos los artículos tienen código de barras, por ello es necesaria esta id)
  - Nombre
  - **Código de barras**
  - Precio
  - Iva

- Usuario:
  - **Id** (número de serie asignado por el sistema, puede servir como clave aparte del mail)
  - Nombre
  - Apellidos
  - **email**
  - Password
  - saldoPuntos
  - totalCompras
  - totalDescuentos
  - esAdmin

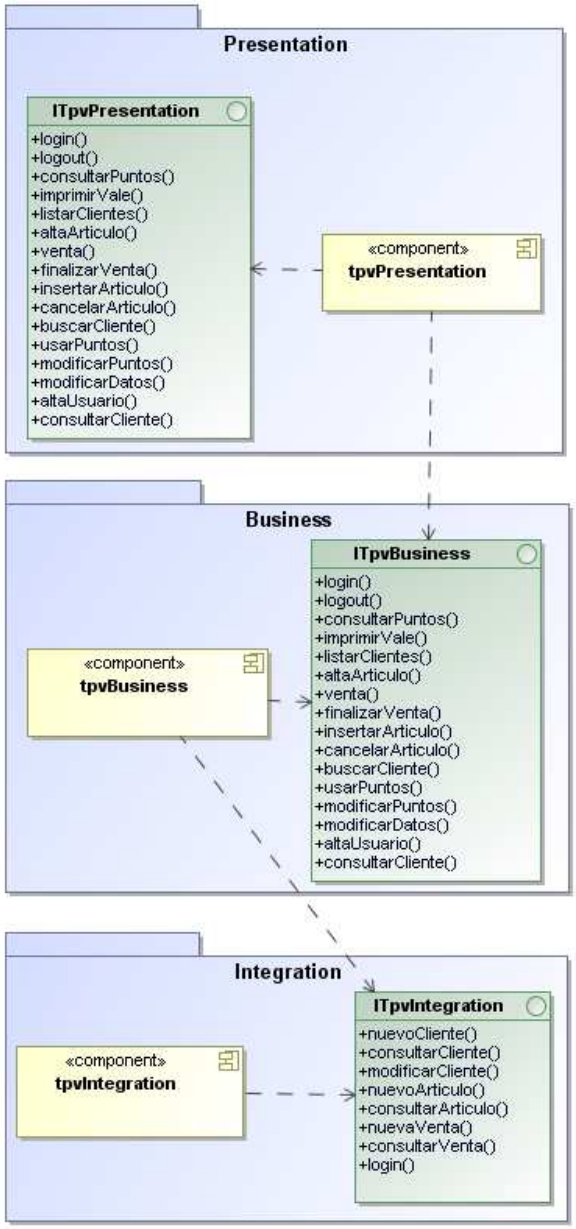
- Venta:
  - **Id**
  - Bruto
  - ivaTotal

- ValeDescuento:
  - Utilizado
  - **Código de barras** (clave foránea de artículo)

Arquitectura

La aplicación se va a desarrollar en **JavaEE** con lo cual se obtendrá una aplicación web que sigue el **modelo cliente/servidor**. La aplicación constará de tres capas:

1- **Capa de presentación**: permitirá la interacción de los usuarios con la aplicación.
2- **Capa de negocio**: implementará la lógica de negocio, es decir, la funcionalidad de la aplicación.
3- **Capa de integración** (o persistencia): permite el acceso a los datos que almacenan la información persistente.



Decisiones de diseño para el **componente de presentación**:

- Se utilizará el framework **Java Server Faces** (JSF) de la especificación JavaEE
- Controlador de la capa el servlet **Faces Servlet**
- Acciones definidas con **Managed Bean**
- Vistas implementadas mediante **Facelets**

Decisiones de diseño para el **componente de negocio**:

- Implementación con **EJB de sesión sin estado con acceso remoto**.
- Sigue el patrón **Facade**

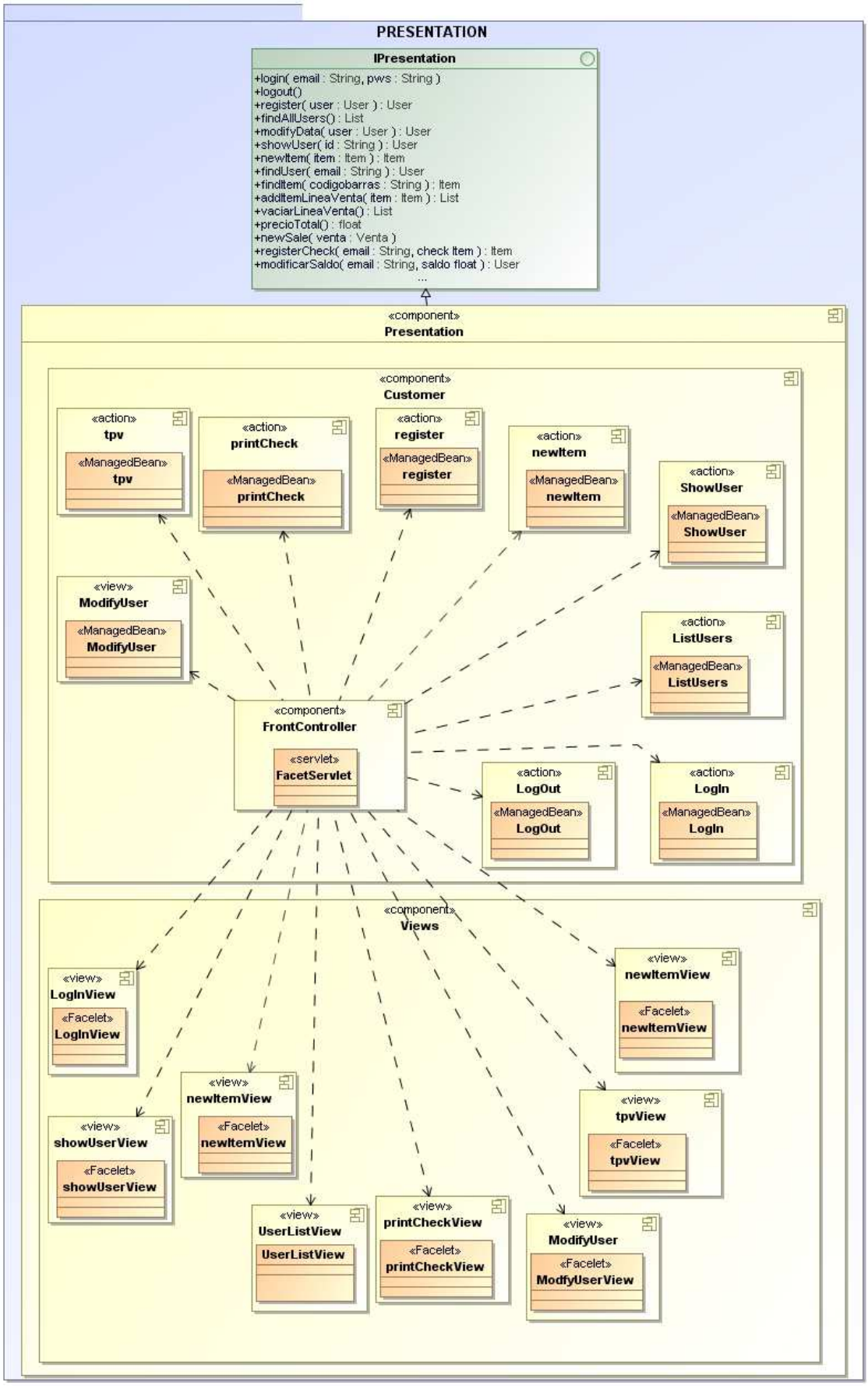
Decisiones de diseño para el **componente de integración**:

- Las entidades se almacenarán en una base de datos relacional.
- Se utilizarán entidades **Java Persistence API** (JPA)

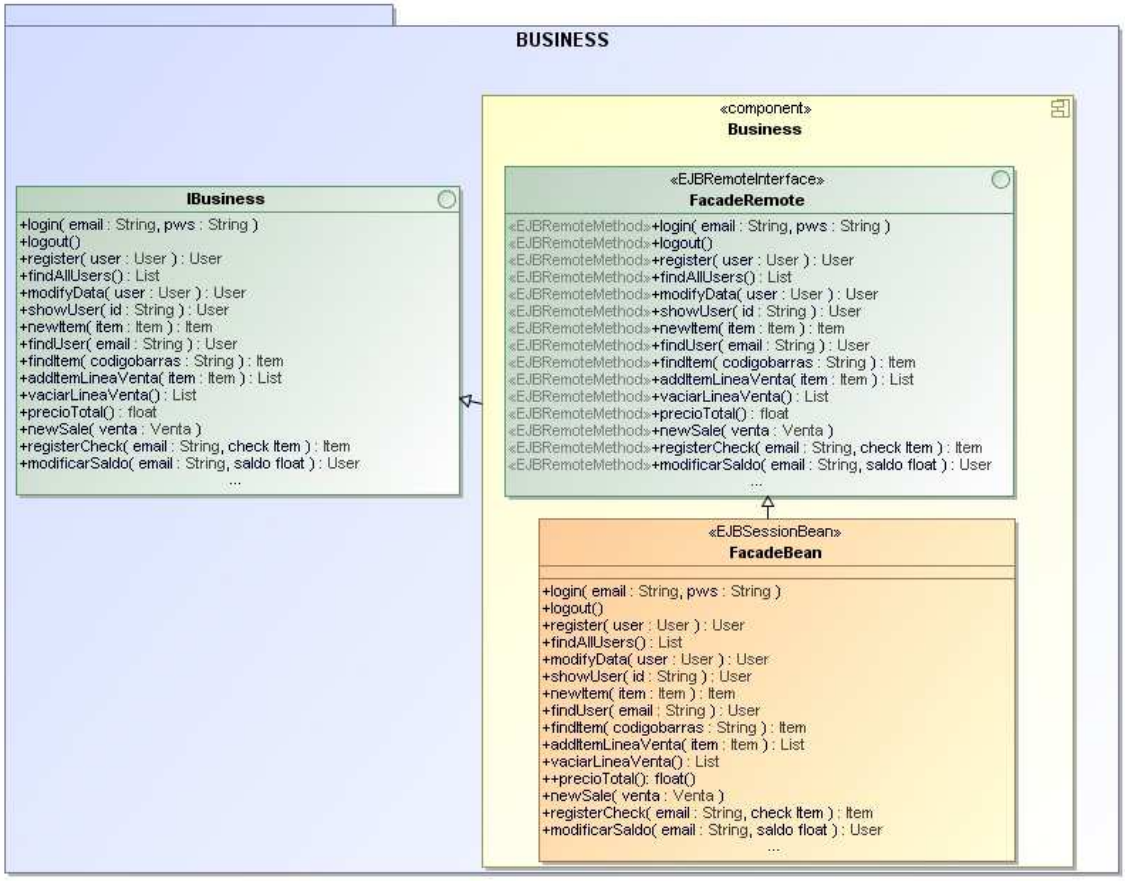
Refinamiento de las capas

Partiendo del diagrama de componentes base, se ha procedido a hacer un refinamiento por capas hasta conseguir un diagrama de componentes software dependiente de la tecnología elegida para la implementación.

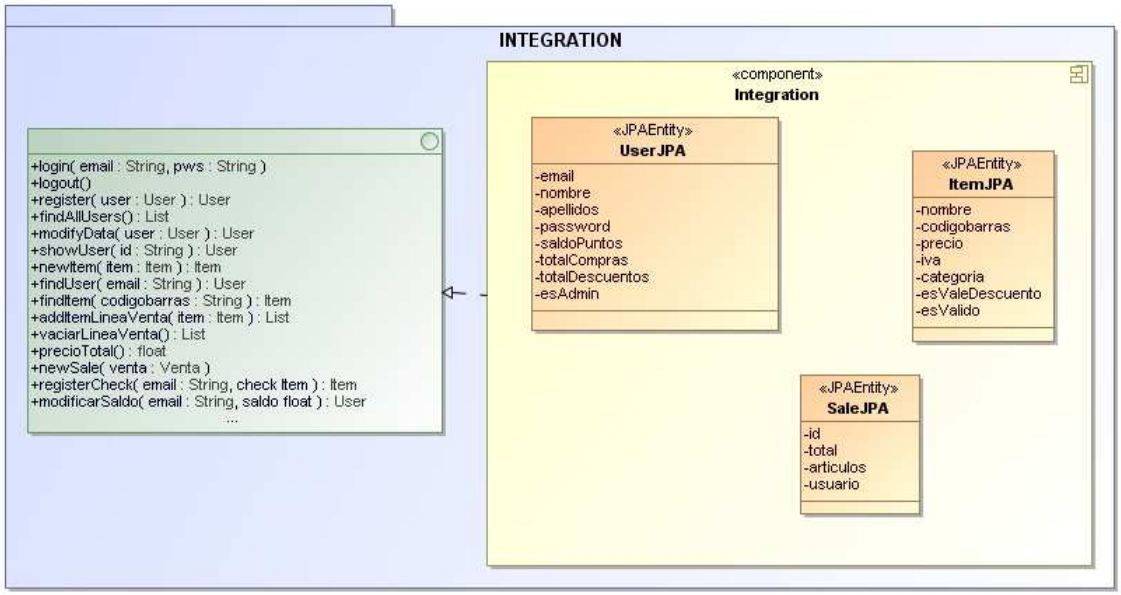
En primer lugar, tenemos el gráfico correspondiente al diagrama de componentes pertenecientes a la capa de presentación, en el que se puede observar el controlador de la capa (el servlet Faces Servlet), las acciones definidas con Managed Bean y las vistas implementadas mediante Facelets.


PRESENTATION
IPresentation
+login(email : String, pws : String)
+logout()
+register(user : User) : User
+findAllUsers() : List
+modifyData(user : User) : User
+showUser(id : String) : User
+newItem(item : Item) : Item
+findUser(email : String) : User
+findItem(codigobarras : String) : Item
+addItemLineaVenta(item : Item) : List
+vaciarLineaVenta() : List
+precioTotal() : float
+newSale(venta : Venta)
+registerCheck(email : String, check Item) : Item
+modificarSaldo(email : String, saldo float) : User
...
«component» Presentation
«component» Customer
«action» tpv
«ManagedBean» tpv
«action» printCheck
«ManagedBean» printCheck
«action» register
«ManagedBean» register
«action» newItem
«ManagedBean» newItem
«action» ShowUser
«ManagedBean» ShowUser
«view» ModifyUser
«ManagedBean» ModifyUser
«action» ListUsers
«ManagedBean» ListUsers
«component» FrontController
«servlet» FacetServlet
«action» LogOut
«ManagedBean» LogOut
«action» LogIn
«ManagedBean» LogIn
«component» Views
«view» LogInView
«Facelet» LogInView
«view» newItemView
«Facelet» newItemView
«view» showUserView
«Facelet» showUserView
«view» newItemView
«Facelet» newItemView
«view» tpvView
«Facelet» tpvView
«view» UserListView
UserListView
«view» printCheckView
«Facelet» printCheckView
«view» ModifyUser
«Facelet» ModfyUserView

En el siguiente gráfico, se encuentra el diagrama de componentes perteneciente a la capa de negocio, que sigue un patrón Fachada (Facade). Las operaciones a realizar son soncronas, no tienen estado y el contenedor JavaEE será quien gestione las transacciones. Por lo tanto, se utilizará un EJB de sesión sin estado para que el contenedor Java EE resuelva la complejidad de las comunicaciones remotas, comportamiento trasaccional y la gestión de la seguridad.



El último gráfico representa la capa de integración que es la que va a resolver la persistencia del sistema desarrollado. En esta capa representamos con entidades JPA los datos que han de ser almacenados en la base de datos relacional.



7- Implementación

Entorno

- Eclipse Java EE IDE for Web Developers Kepler Release
- Servidor Jboss 7.1.1
- PostgreSQL 9.3
- Control de versiones con Git

Decisiones de implementación

Como se ha indicado en el análisis, se ha utilizado Java Server Faces (JSF), framework para aplicaciones Java basadas en web que utiliza el patrón Modelo-Vista-Controlador (MVC).

Diseñar un entorno gráfico como el propuesto en el prototipo puede llegar a consumir una gran cantidad de tiempo de implementación, por lo tanto se ha optado por hacer un desarrollo en html incorporando **Bootstrap** (getBootstrap)**,** un framework visual popular por su uso en Twitter, el cual proporciona una serie de herramientas CSS para modelar la estética de las páginas en desarrollo.

Un cambio importante en la implementación es la posibilidad de generación de códigos de barra propios del comercio. Uno de los principales requisitos es la creación de cheques descuento como si fueran artículos para poder ser incorporados a la linea de venta, por lo tanto, es necesaria la creación secuencial de códigos de barras para ser asignados a los nuevos artículos. Para evitar recorrer toda la base de datos de artículos cada vez que se genera un cheque descuento, se ha creado una tabla que mantiene el último código de barras generado.

Implementación de la capa de presentación

Managed Beans:

- **tpvMBean**: Este Managed Bean es el que representaba mayor dificultad en su desarrollo, ya que en su vista se representaban diferentes datos simultaneamente. Se debían hacer llamadas a busqueda de artículos y a busqueda de clientes, así como representar una lista de venta, total de venta y ser capaz de soportar una entrada continua de articulos para la linea de venta.

- **showUserMBean**: Managed Bean para muestra de datos de usuario.
- **registerMBean**: Managed Bean para registro de nuevo usuario.
- **printCheckMBean**: Managed Bean para muestra e impresión de puntos de usuario.
- **newItemMBean**: Managed Bean dedicado al alta en el sistema de un nuevo artículo.
- **modifyUserMBean**: Managed Bean que controla la modificación de los datos de un usuario.
- **loginMBean**: Managed Bean de autenticación en el sistema.
- **logoutMBean**: Managed Bean de salida del sistema.
- **listUsersMBean**: Managed Bean que controla el listado de usuarios para su representación.

Vistas:

- **tpvView**: Como se comentaba en el Managed Bean correspondiente (tpvMBean), esta vista debe representar varios tipos de datos y proporcionar al usuario vendedor una serie de botones para interactuar con el sistema. Mediante etiquetas en html, se han implementado diferentes areas:
    - **Linea de Venta**: En este area se genera un listado con todos los artículos introducidos para su venta. La ventana dispone de un autoscroll para poder desplazarse a lo largo de la linea de venta.
    - **Insertar artículo**: En esta ventana se puede realizar la busqueda (por código de barras) de cualquier artículo del sistema. Bajo el cuadro de inserción de códigos aparecerán mensajes en caso de error en la busqueda, como por ejemplo cuando no existe un artículo, un cheque ha sido utilizado anteriormente o está duplicado en la misma venta.

    Para poder usar un lector laser de códigos de barras, es necesario que después de cada inserción de artículo, el "focus" permanezca en la caja de entrada de texto. Para ello se ha optado por añadir una funcionalidad en javascript que permite mantener el cursor en la caja de entrada de texto requerida.

```
<script type="text/javascript">
      window.onload = function() {
      document.getElementById('inputitem:input1').focus();
      }
</script>
```

- **showUserView**: Vista que muestra los datos del usuario.
- **registerView**: Esta vista permite el registro de un nuevo usuario en el sistema. No son necesarias credenciales lo cual permite que cualquier usuario pueda crear un usuario sin supervisión del administrador del sistema.
- **printCheckView**: Esta vista muestra los datos del posible cheque descuento que puede imprimir el cliente.
- **printedCheckView**: Una vez que el cliente confirma que desea imprimir un cheque descuento, esta vista será la encargada de representar en pantalla los datos del cheque descuento, con su correspondiente código de barras. El cliente podrá guardar un "pantallazo" para imprimir posteriormente o hacer uso del boton "imprimir" el cual enviará la vista a impresora.

Para la impresión del código de barras ha sido necesario investigar de que manera se representa un código de barras. Inicialmente el problema parecía trivial y la solución se limitaba a una simple inserción de fuentes de código de barras, pero en realidad no hay una correspondencia simple entre código de barras y fuente tal cual, ya que entra en juego una matriz de datos que selecciona los gráficos en función de los elementos del string de código recibido. Finalmente se utilizó una librería JQuery (barcodegenerator de bytescout) en la que un javascript recoge el String correspondiente al código de barras y lo transforma en el gráfico correcto correpondiente imprimiendolo en pantalla.

- **newItemView**: Vista para alta de nuevo artículo, solo visible para el administrador.
- **modifyUserView**: Vista de modificación de datos de usuario.
- **loginView**: Vista para autentificación de usuario en el sistema. Tal cual se ha implementado la aplicación, añadiendo el framework Bootstrap, no es necesario acceder a esta vista ya que existe un acceso a login desde el encabezado de la página (headerView).
- **userListView**: Listado de todos los usuarios del sistema. Accesible por el administrador.
- **adminView**: Pantalla principal con las funcionalidades accesibles solo para el administrador.
- **cuentaUserView**: Pantalla principal con las funcionalidades accesibles para el usuario cliente del sistema.
- **headerView**: Plantilla común para la mayoría de vistas.
- **homeView**: Pantalla principal de acceso a la aplicación.
- **ventacanceladaView**: Vista de venta no válida.

Implementación de la capa de negocio

En la **capa de lógica de negocio** los métodos implementados son:

```
public Collection<?> findAllUsers();
public UserJPA showUser(String idUser);
public UserJPA login(String email, String password);
public UserJPA register(UserJPA user);
public void logout();
public UserJPA modifyData(UserJPA user);
public ItemJPA newItem(ItemJPA item);
public UserJPA findUser(String email);
public ItemJPA findItem(String codigobarras);
public Collection<ItemJPA> addItemLineaVenta(ItemJPA item);
public float preciototal();
public void newSale(SaleJPA venta);
public ItemJPA registerCheck(String email, ItemJPA check);
public Collection<ItemJPA> vaciarLineaVenta();
public UserJPA modificarSaldo(String email, float saldo);
```

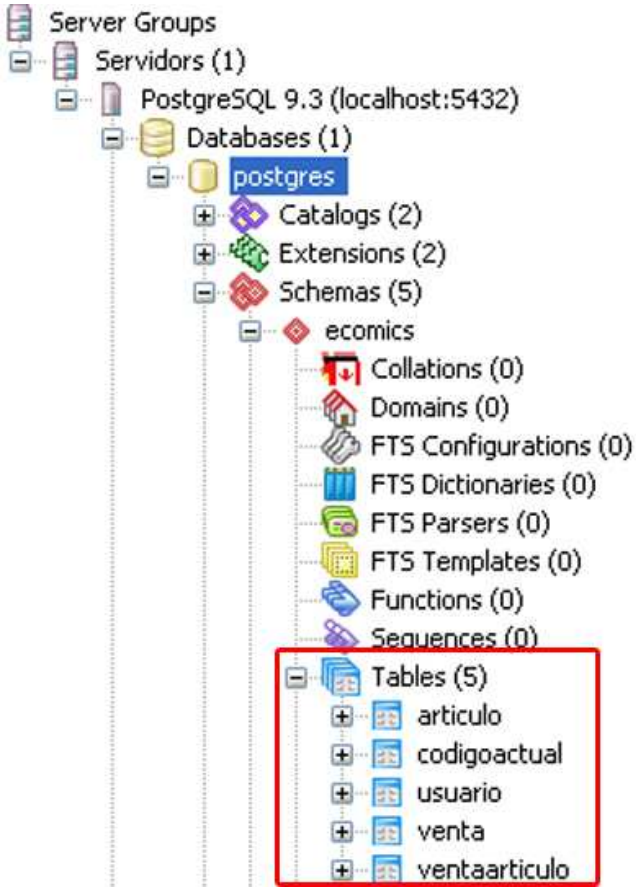
A continuación se describe el funcionamiento de dos de los métodos más importantes de la capa de negocio:

newSale es el método encargado de registrar una nueva venta, una vez que el vendedor pulsa el botón Finalizar Venta en el módulo TPV. Lo primero que comprueba es el uso de cheques descuento ya que, en caso afirmativo, debe encargarse de marcar dichos cheques descuento como “ya utilizados” para evitar su uso posterior en otras compras por parte del cliente. A continuación, calcula los puntos a sumar al saldo del usuario, guarda los datos modificados del usuario y registra la venta en la base de datos.

registerCheck es el encargado de generar un nuevo ItemJPA con el valor adecuado (según saldo del usuario) y con las características necesarias para que desde el TPV sea identificado como un cheque descuento. El código de barras es generado con un número consecutivo al encontrado en la tabla correspondiente de la base de datos (este número es actualizado en esta operación) para lograr mantener códigos de barra únicos en el comercio.

Implementación de la capa de integración (o persistencia)

Se han generado las tablas pertinentes en PostgreSQL y estas son accedidas mediante las entidades JPA implementadas en el sistema. Como se ha indicado en la fase de refinamiento de capas, finalmente se optó por utilizar persistencia para las entidades Usuario, Artículo y Venta. Ventaarticulo representa la relación One-to-Many entre las entidades Artículo y Venta y Codigoactual es utilizado para mantener un registro del último código de barras de la tienda utilizado.

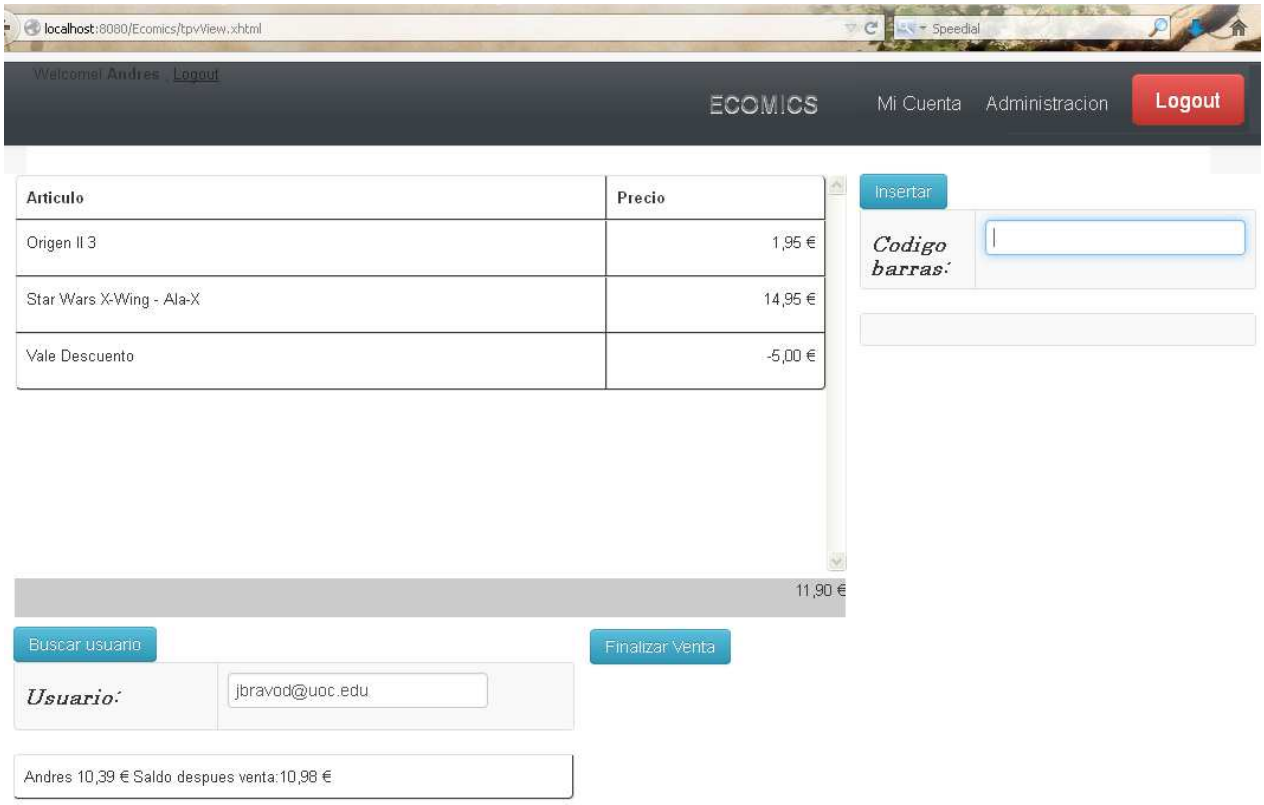


Entorno gráfico final

Como se comentaba inicialmente, se ha utilizado el framework bootstrap como ayuda en la apariencia del entorno gráfico. Unicamente han existido modificaciones de código CSS en la vista tpvView ya que era necesaria cierta maquetación de la vista para la distribución de los diferentes elementos necesarios.

En la vista tpvView se pueden observar las diferentes areas comentadas. A la derecha, la caja de texto de inserción de códigos de barras de artículos. En el lado izquierdo se encuentra la linea de venta en la que se listan los artículos introducidos y al final de dicha linea de venta, el total de la venta.

Por último, debajo se encuentra la entrada de busqueda de usuario, en el caso de que sea necesario asignar puntos y el botón de finalización de la venta.



Se puede observar, que la zona de busqueda de usuario dispone de una zona de mensajes en la que muestra el usuario actual seleccionado, el saldo del que dispone y el saldo que obtendrá si se finaliza la venta (este último dato se va actualizando cada vez que se inserta un artículo a la venta).

La caja de inserción de códigos de barra de artículos también dispone de un area de mensajes en las que el sistema avisará de contingencias durante la busqueda de artículos, como pueden ser: articulos no existentes, vales descuento repetidos en la misma venta o vales descuento ya utilizados anteriormente.

En la vista de usuario, el usuario encuentra la siguiente pantalla al solicitar la posibilidad de impresión de un cheque descuento. El sistema le informa del saldo disponible, del valor del cheque que se imprimirá y del saldo restante que le quedará:

Si el usuario decide imprimir el cheque, el sistema le enseñará la vista del cheque generado, con el correspondiente código de barras y un botón que le permitirá mandar a impresora la pantalla:

8- Testing

Se han realizado pruebas de integración de las principales funcionalidades del proyecto. Estas pruebas y sus resultados se han detallado en fichas de testing en las que se recoge la entrada utilizada, la acción o resultado esperado y si ha sido correcto o no, el funcionamiento de la funcionalidad probada. Debido al tiempo disponible se decidió no realizar pruebas automatizadas, ya que esto requería una inversión de tiempo bastante alta en comparación con el testeo final. Además, cabe indicar que en dicho testeo, se han obviado las funcionalidades mas triviales.

Sistema	Tpv		
Propósito :	Control de operaciones		
Nº	**Entrada**	**Acción Esperada**	**Verificar**
1	Realizar Venta	Actualizacion correcta de saldo de usuario	X
2	Realizar venta sin usuario seleccionado	Finalización de venta	Ok
3	Utilizar dos veces el mismo cheque descuento	Debe denegar la segunda inserción	Ok

Sistema	Alta de artículos		
Propósito :	Control de altas en la base de datos		
Nº	**Entrada**	**Acción Esperada**	**Verificar**
1	Nuevo artículo	Creación de artículo en base de datos	Ok
2	Nuevo artículo con código de barras ya existente	Error – denegación del alta	Ok
3	Nuevo artículo, formulario incompleto	Error – indicar datos obligatorios	Ok

Sistema	Registro de nuevo usuario		
Propósito :	Control de registro de usuario		
Nº	**Entrada**	**Acción Esperada**	**Verificar**
1	Nuevo usuario	Creación de usuario en base de datos	Ok
2	Nuevo usuario con email ya existente	Error – denegación del registro	Ok
3	Nuevo usuario, formulario incompleto	Error – indicar datos obligatorios	Ok

OK	8	**EVALUACIÓN**
FAIL	1	**89%**

Al observar el error en el resultado esperado en la suma de puntos de saldo de usuario, se procedió a su corrección. Efectivamente existía un error en el código que hacía que la suma de puntos de saldo no se realizase correctamente. Dicho error (y otros pequeños errores puntuales detectados) ha quedado corregido en la última versión del código que se entrega con esta memoria.

9- Objetivos conseguidos

Se ha conseguido desarrollar una aplicación que facilitará mucho la gestión de la fidelización de clientes en cualquier pequeño comercio que necesite implantar este tipo de solución. Obviamente se trata de un sistema muy básico en el que debe mejorar la apariencia gráfica y añadir otra serie de funcionalidades, pero basicamente resuelve una problemática que hasta ahora resolvía manualmente en hojas de cálculo.

Uno de los principales retos consistía en hacer funcionar el sistema TPV ya que ignoraba si la tecnólogia JavaEE era capaz de trabajar en una sola vista con diferentes tipos de llamadas (artículos, usuarios y linea de venta) así como una actualización continua de la vista al realizar inserciones de artículos, al buscar un nuevo usuario, etc. Además, también desconocía si sería posible hacer una lectura continúa de códigos de barras a través de una pistola lectora laser.

Teniendo en cuenta las consideraciones anteriores, el resultado ha sido totalmente satisfactorio, por lo tanto, considero el objetivo cumplido y continuaré desarrollando otras funcionalidades ya que, como he comentado anteriormente, este software con mejoras puede resultar de gran ayuda para ciertos tipos de comercio.

10- Trabajo futuro y mejoras

Desde el punto de vista estético, una futura mejora consistiría en la personalización de todo el entorno gráfico (actualmente dependiente del framework Bootstrap), adaptandolo al logotipo y colores corporativos del comercio que haga uso del proyecto. Incluso sería posible añadir funcionalidades que permitiesen este tipo de cambios desde una página de configuración, accesible por el administrador del sistema.

Una funcionalidad que me hubiera gustado incorporar al módulo de fidelización de clientes es la posibilidad de que los clientes no utilicen todo su saldo para imprimir un cheque. Es decir, incorporar una entrada en la que el cliente pueda indicar la cantidad de saldo con la que quiere generar el cheque. También podría realizarse graficamente con unos botones “+” y “-“ que modificarían el cheque en múltiplos de 5€.

Una vez que el usuario imprime un cheque con su saldo, su saldo queda reducido (ya que ahora consta en el cheque impreso) y no tiene más visión en el sistema de que ha pasado con su saldo anterior. Sería interesante añadir un listado, visible por el usuario, en el que aparecerían todos sus cheques generados y el estado en el que se encuentran (utilizados y pendientes). Esto podría ser visualizado también por el vendedor con la posibilidad de, durante una venta, utilizar cheques pendientes de usuario. Esto último podría implementarse poniendo la lista de cheques pendientes bajo el nombre del usuario, cuando este es buscado desde la vista tpv.

La vista tpv debería ser finalizada desde el punto de vista de una venta real, es decir, proporcionar acceso al tipo de pago que realiza el cliente (efectivo, tarjeta), indicación del cambio, apertura de cajón portamonedas e impresión de ticket de venta.

En un comercio del tipo que nos ocupa, a menudo los clientes hacen pedidos de artículos que no se encuentran en stock en ese momento. El vendedor habitualmente anota los datos, la fecha y tiene que estar pendiente de la llegada del artículo e incluso gestionar plazos. Una posible funcionalidad a desarrollar sería la de pedidos de clientes, en la que en una ficha se introducirían todos los datos y el sistema, mediante unas alarmas, nos indicaría la llegada de artículos, fin de plazos de espera, etc.

En comercios de comics y juegos de tablero, es habitual la realización de actividades para los clientes tales como presentaciones y torneos lo cual aporta otra futura funcionalidad muy interesante. La posibilidad de registrar clientes participantes en estas actividades, introducción de resultados, mantenimiento de clasificaciones y sorteos aleatorios.

11- Conclusiones

Cuando me decanté por realizar el itinerario de desarrollo de software, lo hacía motivado por conocer un área de la informática que siempre me había llamado la atención pero desde una perspectiva externa. En mis años de experiencia laboral, siempre he trabajado en la parte de sistemas y nunca en la parte de desarrollo. Mi aliciente principal era comprender el ciclo de vida de una aplicación, desde que se idea, se analiza, se implementa y finalmente se implanta.

Este proyecto me ha hecho comprender la razón de ser de la ingenieria del software en todas sus fases, consiguiendo entender la importancia de todos los pasos necesarios en un desarrollo. Durante la implementación he descubierto que pueden surgir problemas derivados del análisis y diseño, y de ahí la gran importancia que tiene realizar un ánalisis y diseño exhaustivos.

Por otro lado, ha sido muy interesante comprender como funciona un desarrollo en JavaEE con JSP y como permite un desarrollo modular de aplicaciones que pueden ser tan simples o complejas como se necesite.

Como conclusión final solo me queda comentar que he disfrutado mucho, aprendiendo como se realiza un proyecto software y aplicando los conocimientos adquiridos durante todos estos semestres.

Bibliografía

getBootstrap. (s.f.). Recuperado en junio de 2014, de http://getbootstrap.com/

JSF. (s.f.). Recuperado el junio de 2014, de http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.jst.jsf.doc.user%2Fhtml%2Fgettingstarted%2Ftutorial%2FJSFTools_tutorial_JSF20.html

Código de barras (s.f.) Recuperado en junio de 2014, de http://www.gomaro.ch/Specifications/EAN13e.htm

Barcodegenerator (s.f.) Recuperado en junio de 2014, de https://bytescout.com/products/enduser/misc/barcodegenerator.html