29-11-2018

Documentación

ArtisanBeer - Proyecto final DAW

Miguel Ángel García y Julián Rodenas

2º DAW

INDICE

**DOCUMENTACION PROYECTO FINAL**

**Miguel Ángel García Martínez**

**Julián Díaz Rodenas**

**Fecha 2018**

**Ciclo de Grado Superior y Grupo Informática**

**IES LEONARDO DA VINCI**

**DESARROLLO DE APLICACIONES WEB**

# Información acerca del proyecto

## Software utilizado

Para el desarrollo de este proyecto hemos utilizado:

* Lenguaje principal Java enfocado a desarrollo web
* Otros lenguajes utilizados: Bootstrap (HTML5 y CSS3), JavaScript, AJAX
* Entorno de desarrollo NetBeans en su versión 8.2
* Base de datos: MYSQL
* Gestor de base de datos: HeidiSQL
* Servidor de Aplicaciones: Glassfish Server 4.1.1
* Hibernate 4.1
* GitHub
* GitHub Desktop

## Descripción del proyecto

Aplicación web enfocada a la compraventa de cerveza artesanal. La idea surge de las dificultades que existen a la hora de distribuir este tipo de producto, ya que suele estar elaborado por pequeños productores con dificultades de publicitarse.

La aplicación permite ver un amplio catálogo de productos subidos por los propios usuarios vendedores. La aplicación permite registrarse tanto como usuario consumidor o como usuario vendedor, de forma que un usuario vendedor pueda acceder a sus pedidos y realizarlos, y un usuario vendedor puede acceder a un panel de control para poder poner a la venta productos y categorizarlos a su gusto.

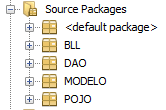
La aplicación está implementada con Java como lenguaje de programación principal utilizando el framework Hibernate para gestionar el acceso a la base de datos.

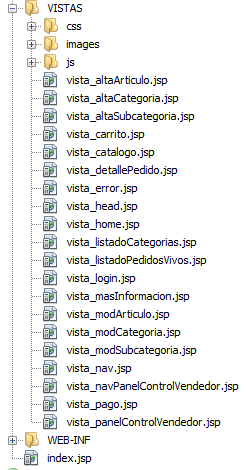
Las vistas han sido realizadas con HTML5 y hemos utilizado las librerías de Bootstrap 4 junto con CSS3 para realizar el diseño de la web.

El proyecto al realizarlo de forma conjunta ha tenido que ser actualizado utilizando un repositorio común, en nuestro caso utilizamos Git-Hub y usamos su aplicación de escritorio Git-Hub Desktop para sincronizar nuestro trabajo.

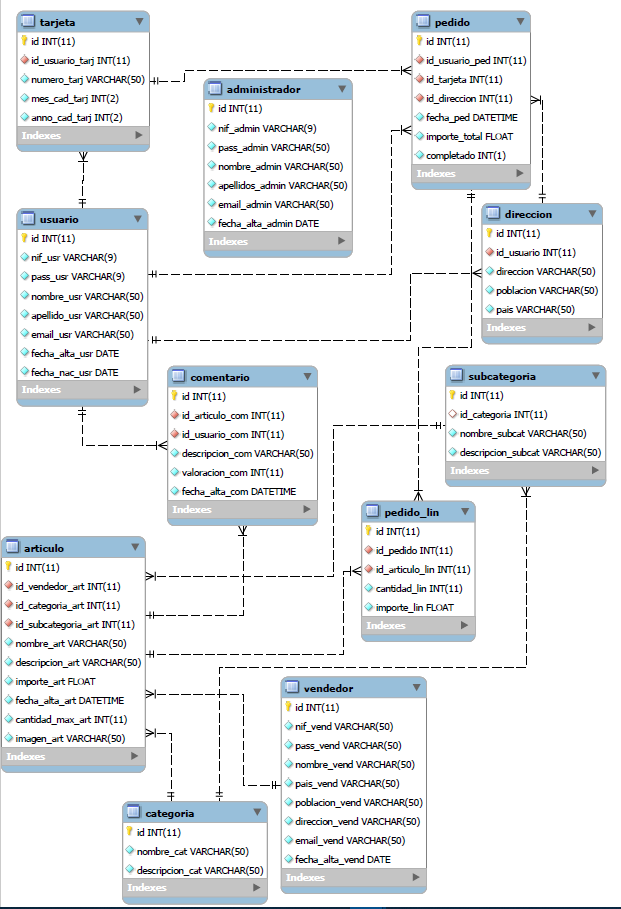
## Modelo de organización

Se ha utilizado un modelo de organización modelo-vista-controlador (MVC):





# Diagrama de tablas



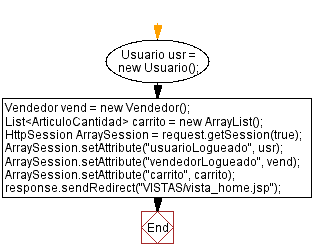
# API de la aplicación

Se adjunta junto a la documentación del proyecto.

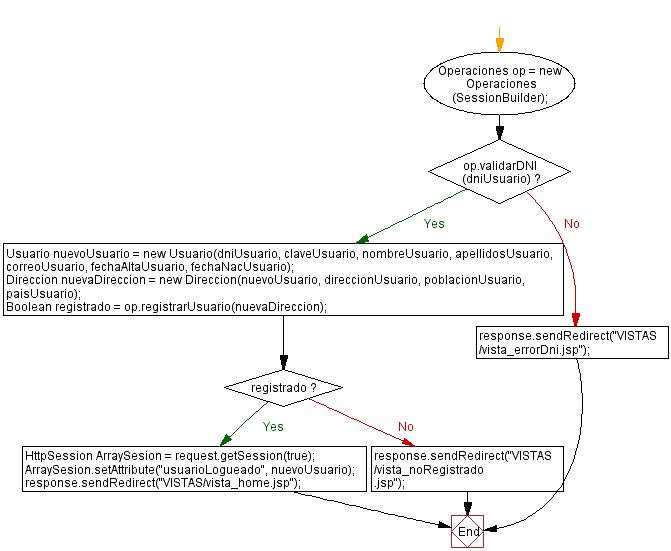
# Casos de uso

## Abrir aplicación.

La aplicación al abrir crea en la sesión un carrito vacío (List de ArticuloCantidad), un usuario (que servirá como usuario logueado) y un vendedor (que servirá como vendedor logueado).

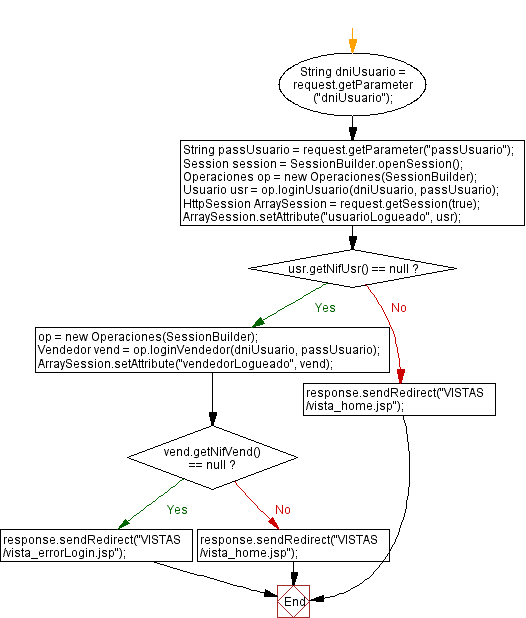


## Registrar Usuario

Para registrar un usuario validamos que el DNI sea correcto y después comprueba que no exista ya un usuario con ese DNI. El registro de vendedor es similar, por lo que lo incluyo en el mismo apartado

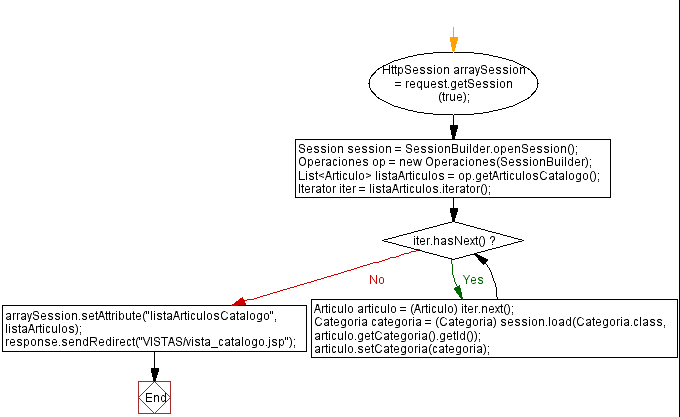
## Registrar vendedor

Se comprueba si existe un usuario, y si es así inicia sesión guardándolo en la sesión, si no es así comprueba si existe ese vendedor, si existe lo guarda en la sesión, si no muestra un aviso de error.



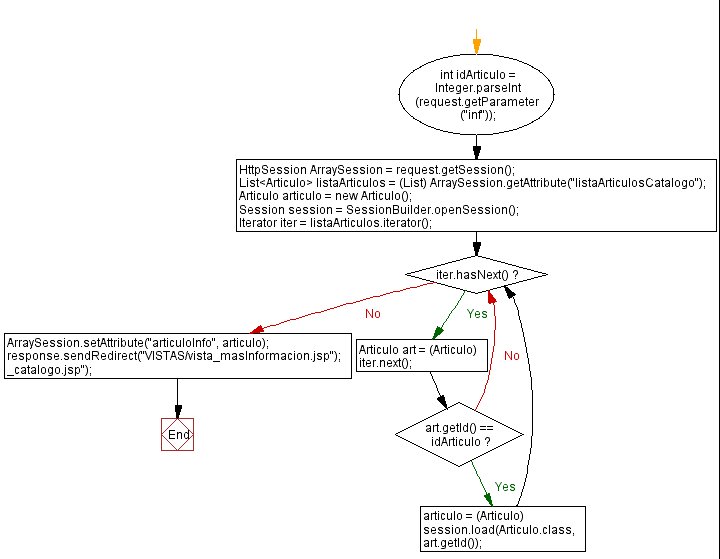
## Mostrar catálogo

Extrae los artículos disponibles en la base de datos y los guarda en una variable de sesión como List.



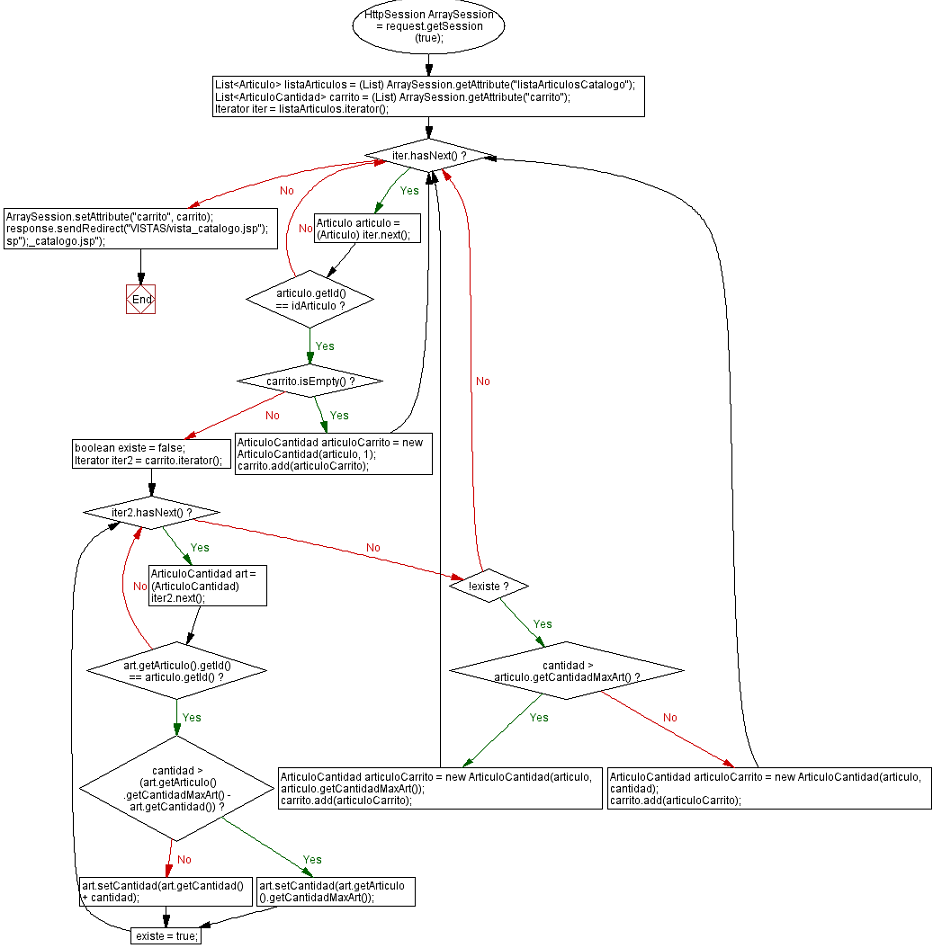
## Mostrar más información sobre un articulo

Busca en la lista de artículos de la sesión el articulo seleccionado y hace un load para cargarlo en la sesión de Hibernate y hacer visibles los datos de los objetos que contiene.



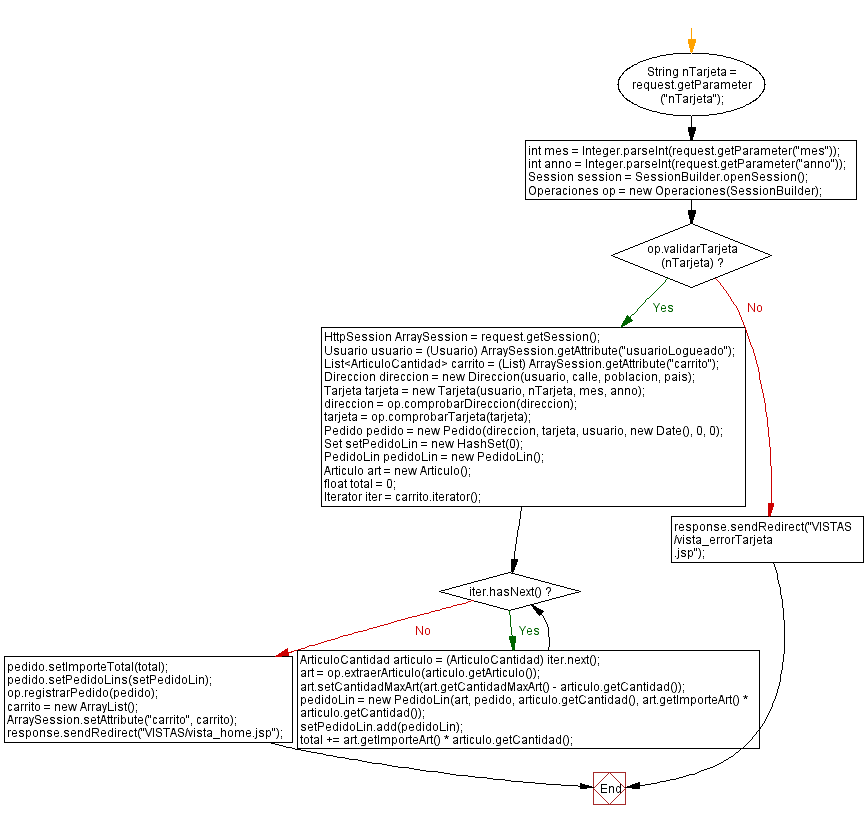
## Añadir articulo a carrito

Comprueba si el articulo a añadir existe en el carrito, si existe modifica la cantidad que hay en él, también comprueba que la cantidad a añadir no sea mayor al stock disponible, si es así establece la cantidad en la máxima disponible en stock.



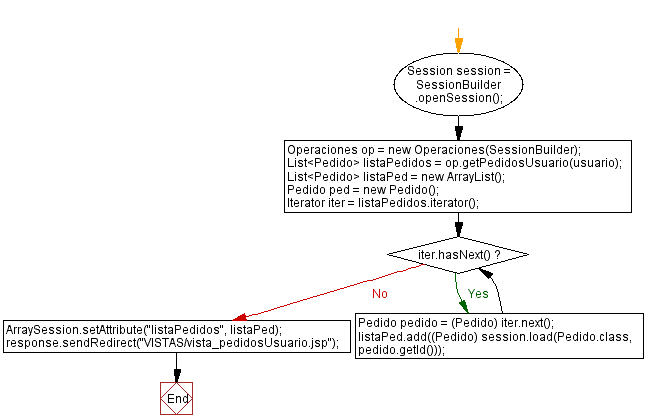
## Realizar pedido

Valida que la tarjeta sea válida. Luego comprueba si la dirección y la tarjeta existen en la base de datos, y si es así la extrae y la añade al objeto pedido. Luego extrae de la base de datos los artículos que lleva el pedido (Ya que los objetos que tenemos al haberlos cargado anteriormente son transitorios y para insertar en la base de datos deben ser persistentes), crea un Set de líneas de pedido para insertarlo en el pedido. Hecho todas estas comprobaciones, inserta pedido en la base de datos.



## Listado Pedidos

Extrae los pedidos del usuario logueado y lo mete en sesión



## Listado pedidos completados

## .

# Variables de sesión

1. usuarioLogueado: (Usuario) sirve para almacenar el usuario que hace log-in.
2. vendedorLogueado: (Vendedor) sirve para almacenar el vendedor que hace log-in.
3. Carrito: (List<ArticuloCantidad>) sirve para almacenar los artículos que se añaden al carrito.
4. listaArticulosCatalogo: (List<Articulo>) sirve para almacenar todos los artículos que se mostrarán en el catálogo.
5. arrayCategorias: (ArrayList<Categoria>) sirve para almacenar la lista de categorías
6. arraySubcategorias: (ArrayList<Categoria>) sirve para almacenar la lista de subcategorías.
7. arrayPedidoLin: (ArrayList<PedidoLin>) sirve para almacenar la lista de líneas de pedido.
8. correcto: (boolean) guarda si se ha realizado correctamente
9. error: (boolean) guarda si no se ha realizado
10. arrayPedidos: (ArrayList<Pedido>) sirve para almacenar los pedidos de un usuario
11. categoría: (Categoria): categoría que se va a modificar
12. subcategoria: (SubCategoría) subcategoria que se va a modificar