PROYECTO: ALL THAT I WANT.

MIGUEL ANGEL NAVARRO VERA

I.E.S CAMP DE MORVEDRE, SAGUNTO

INDICE

INTRODUCCIÓN

El proyecto se basa en la realización de una plataforma web que realice la función de aunar la oferta de diferentes proveedores y democratizar la figura del *personal shopper* dadas las corrientes sociales actuales.

Tras este paso la página dispondrá como función principal, una tienda online que aglutina toda la oferta de moda de las principales marcas de moda. También dispondremos de espacios a la venta/alquiler para empresas/particulares que deseen tener a la venta sus artículos y a la vez tener visibilidad desde una gran plataforma.

Como servicios anexos, la página web dispondrá de un blog de tendencias así como de une espacio en el que tanto las usuarias como *bloggers* podrán crear sus propias *love list* de prendas disponibles en nuestra web tanto para comercializarlas como para crear rankings o *"toptens*" de diferentes aspectos.

La función de la creación de estas *love list* es dotar al mundo *blogger* que no tiene medios de un espacio acoplable a sus propios blogs que sirva como pasarela hacia nuestra tienda online.

Para completar los servicios en un fururo se creará una base de datos, que tras ser parametrizados a través de un test inicial, el sistema sugerirá al cliente: prendas, accesorios,... en definitiva, ofrecerá diferentes looks que más se adecuen a las preferencias de los clientes. Está parte aunque de sentido al proyecto como atractivo, lo consideramos parte auxiliar.

MÓDULOS RELACIONADOS

- Desarrollo Aplicaciones Web.
- Desarrollo Aplicaciones Cliente.
- Despliegue De Aplicaciones Web.
- Diseño De Interfaces.
- Bases De Datos.
- Sistemas.

PLA DE TRABAJO

ETAPAS

ETAPA 1:

Estudio del mercado, lluvia de ideas y planteamiento de los requerimientos del usuario final.

ETAPA 2:

Selección de tecnologías.

ETAPA 3:

Diseño de la base de datos.

ETAPA 4:

Ideas para el diseño de la página, sus elementos.

ETAPA 5:

Creación de un proyecto en GITHUB, y dos ramas diferentes, una en la cual desarrollar y otra rama para subir los cambios satisfactorios realizados en la rama de desarrollo.

ETAPA 6:

Desarrollo.

ETAPA 7:

Maquetación.

ETAPA 8:

Recopilación de problemas.

ETAPA 9:

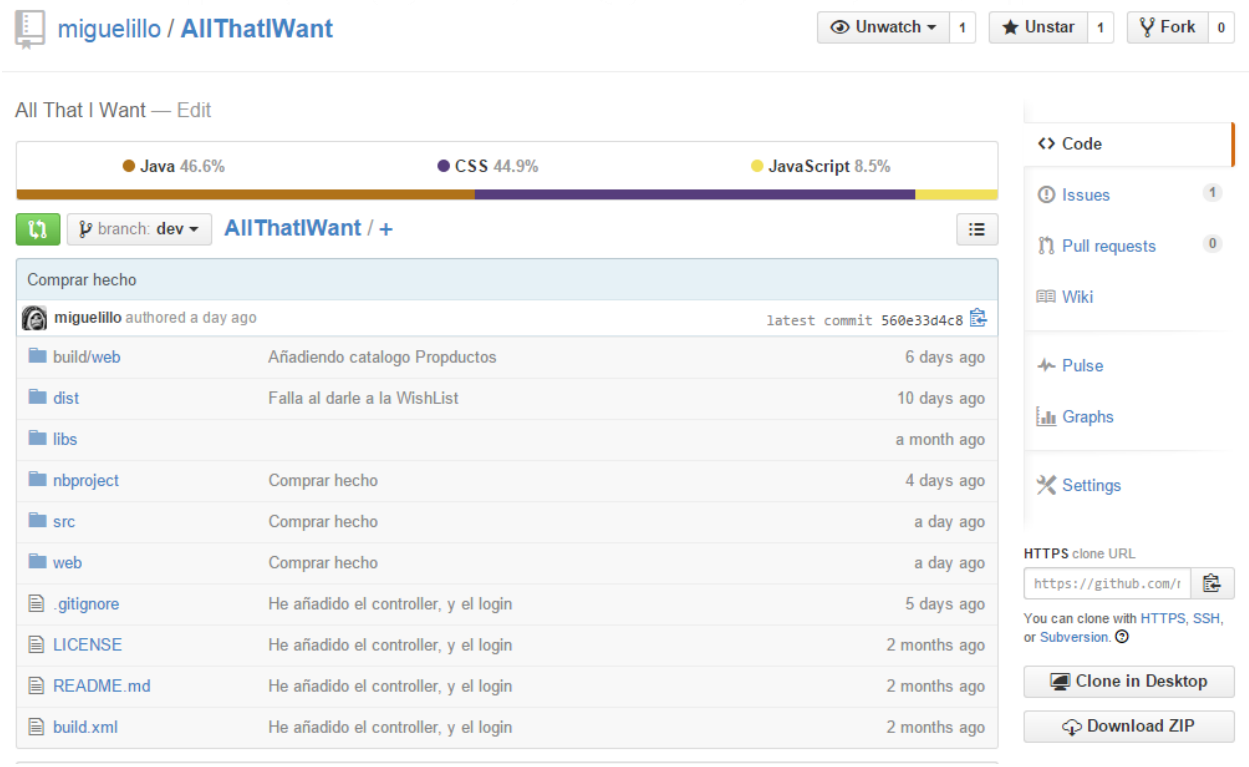
Resolución de errores.

VERSIONADO

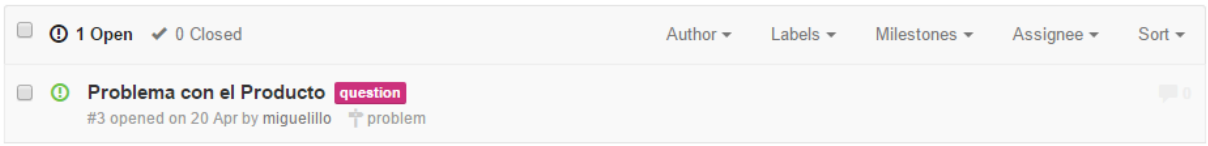
Durante todo el proceso de desarrollo he usado GITHUB, como plataforma para tener un control de versiones del mismo.

GITHUB permite realizar un control de versiones, así como modificar el código en la misma web, también permite dar anotaciones en los *commits*, y apuntarte como en un diario de *"issues"* todos los problemas que han surgido y no se han podido solucionar por diversos problemas, y que quedan pendientes para su posterior desarrollo.

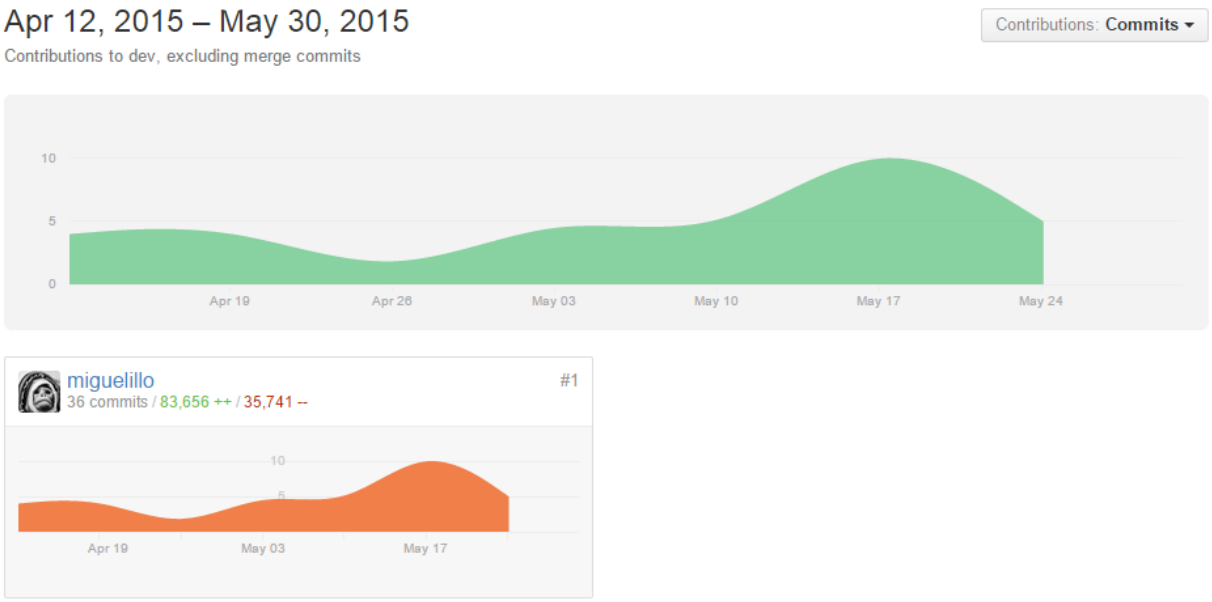
PANTALLA PRINCIPAL DEL PROYECTO.



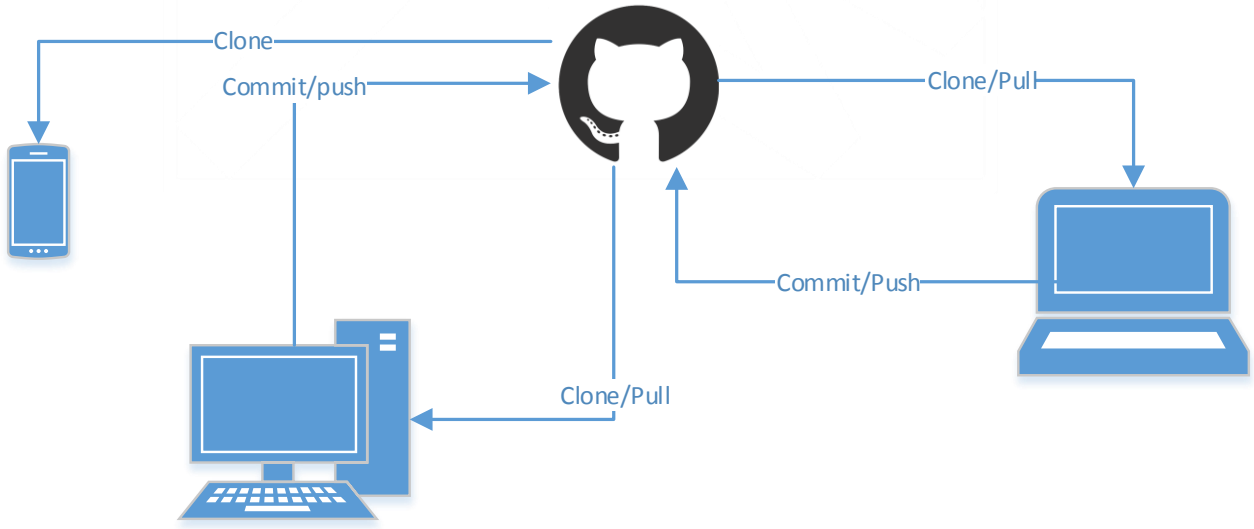
PANTALLA DE LA GESTION DE PROBLEMAS



PANTALLA DE LOS CONTRIBUYENTES AL PROYECTO.



Pero el objetivo de GITHUB es mucho más que un sistema de versionado. La potencia de GITHUB viene en tener el proyecto en la nube, de manera que desde cualquier ordenador, podemos acceder a él.



DESARROLLO

TECNOLOGÍAS

BASE DE DATOS:

La base de datos ha sido construida en MYSQL debido a su bajo coste económico. Además de todas las herramientas que hay en los mercados capaces de gestionar tanto los datos como la estructura de la base de datos.

LENGUAJES DE PROGRAMACIÓN (SERVIDOR):

El programa estará desarrollado como parte principal en J2EE. La tecnología java es una tecnología abierta y se basa en gran parte en estándares de organizaciones de normalización y estándares empresariales "de facto". Esto posibilita que los desarrolladores puedan conocer y entender completamente cómo hace las cosas java y aprovecharlo para sus aplicaciones y, por otro lado, al basarse en estándares empresariales, simplifica la integración con productos de múltiples compañías.

La tecnología java goza ya de un gran recorrido, y ha probado su eficacia en muchos entornos y situaciones empresariales distintas.

J2EE puede adquirirse de diferentes compañías y no requiere de una licencia para poder utilizarse, esto hace que el coste del proyecto con respecto a otras tecnologías se abarate considerablemente.

LENGUAJES DE PROGRAMACIÓN (CLIENTE):

Es importante el desarrollo en entorno cliente, puesto que facilita muchas acciones agiliza mucho la interacción del usuario con la aplicación, tanto en usabilidad, como rapidez.

Para esto se han usado tanto el FRAMEWORK basado en JAVASCRIPT, JQUERY, como JAVASCRIPT puro.

DISEÑO:

Para el diseño se ha usado tecnologías de última generación, basadas EN HTML5, CSS3 y JQUERY/JAVASCRIPT. Debido a su facilidad y flexibilidad a la hora de realizar trabajo.

Se ha desarrollado sobre el FRAMEWORK de diseño, BOOTSTRAP.

DESPLIEGUE:

El proyecto se desplegara bajo un servidor APACHE TOMCAT V7, sobre JAVA VIRTUAL MACHINE V8, sobre un sistema DEBIAN (WHEEZY) 64 BITS, y hospedado en un servidor VPS.

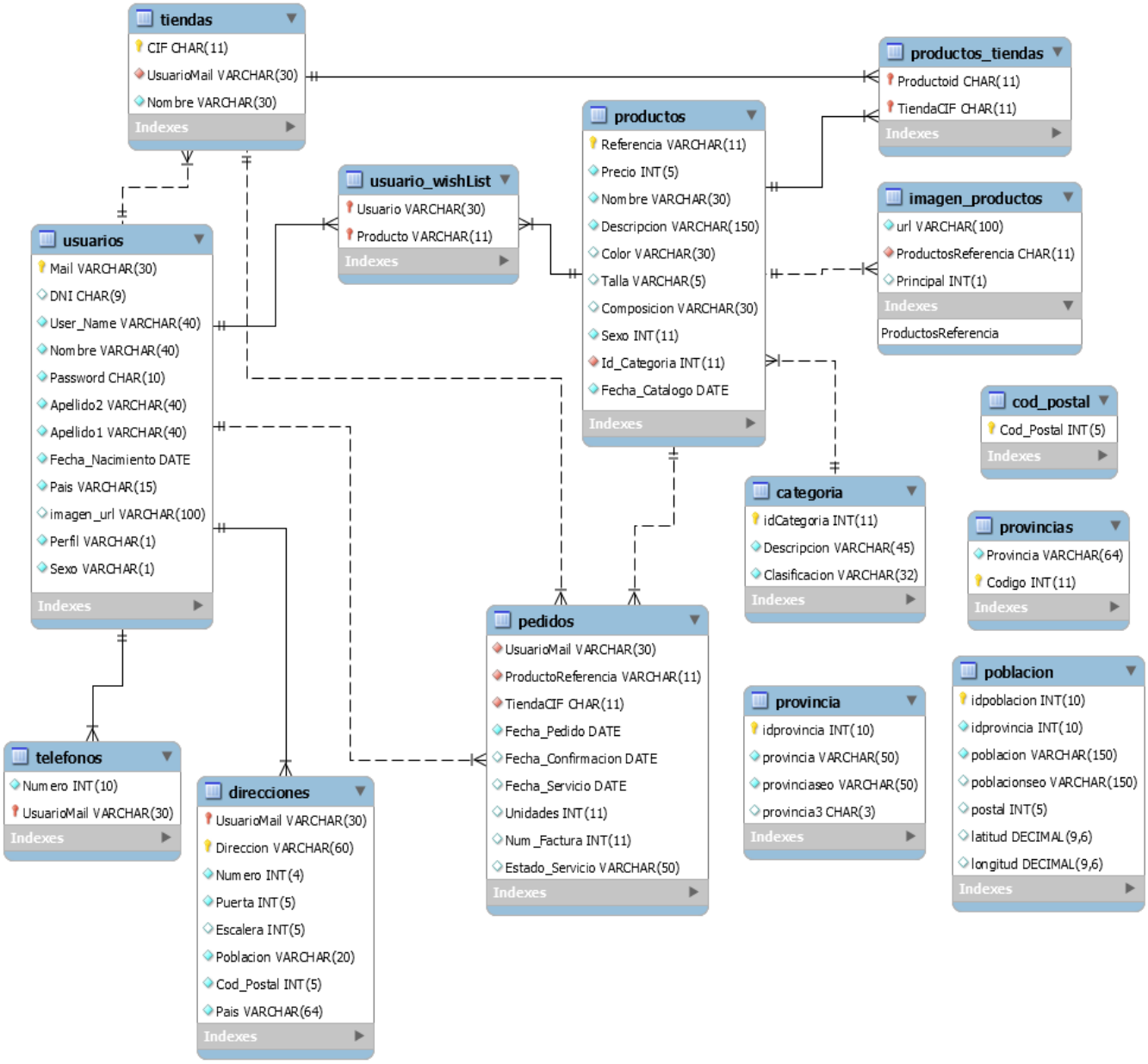
La empresa en la que esta hospedado el servidor es OVH. OVH es una empresa de soluciones informáticas francesa, que debido a su alta calidad de soporte técnico y sus precios competentes, era una de las mejores opciones a tener en cuenta en el despliegue.

VERSIONING:

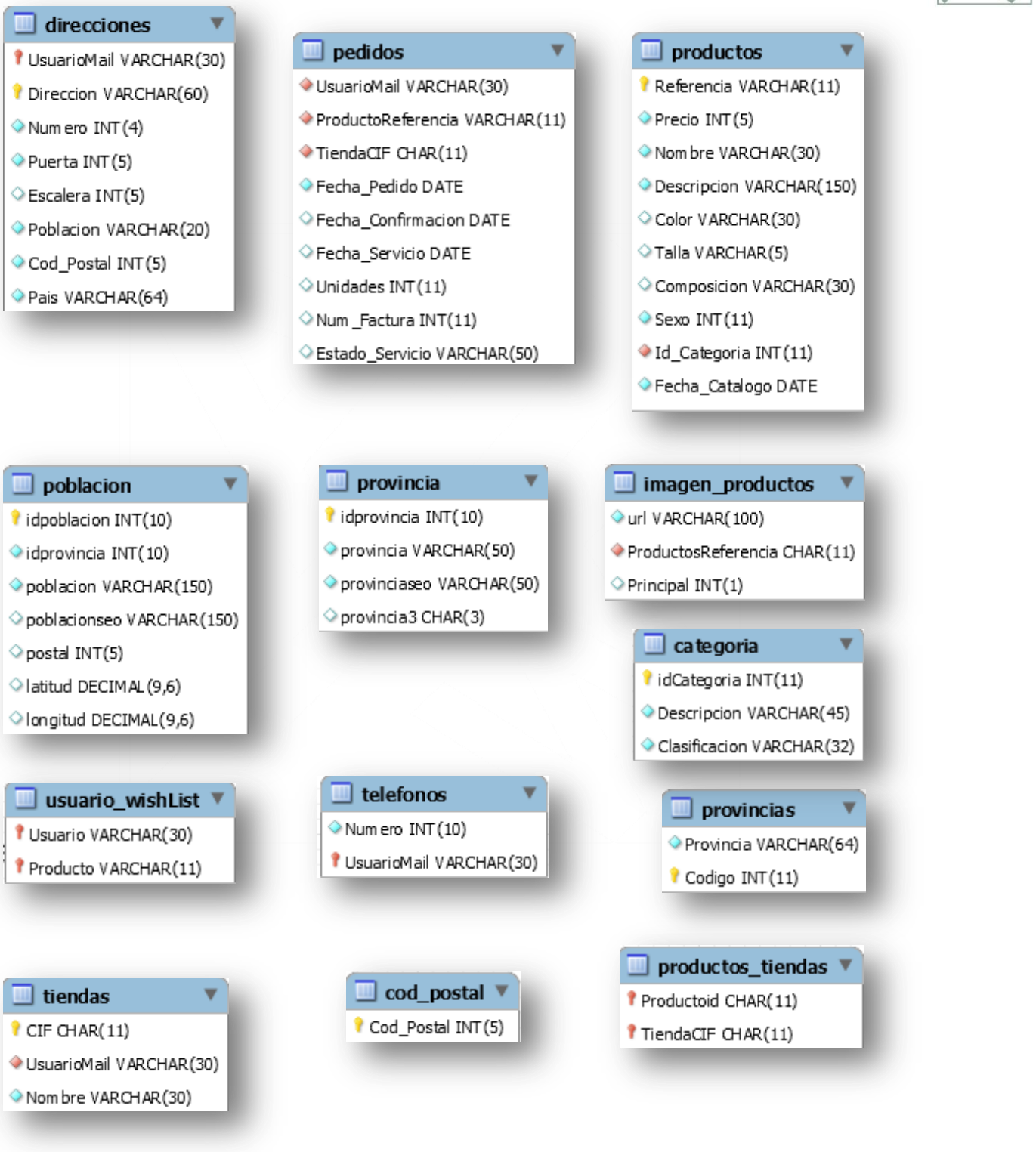
Para el control de versiones, flujos de trabajo y clonado desde cualquier entorno, he usado GITHUB.

GITHUB es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones GIT.

MODELO RELACIONAL



ENTIDADES


direcciones
UsuarioMail VARCHAR(30)
Direccion VARCHAR(60)
Numero INT(4)
Puerta INT(5)
Escalera INT(5)
Poblacion VARCHAR(20)
Cod_Postal INT(5)
Pais VARCHAR(64)
pedidos
UsuarioMail VARCHAR(30)
ProductoReferencia VARCHAR(11)
TiendaCIF CHAR(11)
Fecha_Pedido DATE
Fecha_Confirmacion DATE
Fecha_Servicio DATE
Unidades INT(11)
Num_Factura INT(11)
Estado_Servicio VARCHAR(50)
productos
Referencia VARCHAR(11)
Precio INT(5)
Nombre VARCHAR(30)
Descripcion VARCHAR(150)
Color VARCHAR(30)
Talla VARCHAR(5)
Composicion VARCHAR(30)
Sexo INT(11)
Id_Categoria INT(11)
Fecha_Catalogo DATE
poblacion
idpoblacion INT(10)
idprovincia INT(10)
poblacion VARCHAR(150)
poblacionseo VARCHAR(150)
postal INT(5)
latitud DECIMAL(9,6)
longitud DECIMAL(9,6)
provincia
idprovincia INT(10)
provincia VARCHAR(50)
provinciaseo VARCHAR(50)
provincia3 CHAR(3)
imagen_productos
url VARCHAR(100)
ProductosReferencia CHAR(11)
Principal INT(1)
categoria
idCategoria INT(11)
Descripcion VARCHAR(45)
Clasificacion VARCHAR(32)
usuario_wishList
Usuario VARCHAR(30)
Producto VARCHAR(11)
telefonos
Numero INT(10)
UsuarioMail VARCHAR(30)
provincias
Provincia VARCHAR(64)
Codigo INT(11)
tiendas
CIF CHAR(11)
UsuarioMail VARCHAR(30)
Nombre VARCHAR(30)
cod_postal
Cod_Postal INT(5)
productos_tiendas
Productoid CHAR(11)
TiendaCIF CHAR(11)

CONSULTAS SQL

MOSTRAR_USUARIO *= "SELECT * FROM USUARIOS WHERE MAIL = ?";*

REGISTRO_USUARIO *= "INSERT INTO USUARIOS (`DNI`,`USER_NAME`, `MAIL`, `NOMBRE`,`PASSWORD`,`APELLIDO2`,`APELLIDO1`,`FECHA_NACIMIENTO`,`PAIS`,`IMAGEN_URL`,`PERFIL`,`SEXO`)VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";*

LOGIN_USUARIO *= "SELECT * FROM USUARIOS WHERE MAIL = ? AND PASSWORD = ?";*

REGISTRAR_TIENDA *= "INSERT INTO PROYECTOFINALDAW.TIENDAS (`CIF`, `USUARIOMAIL`, `NOMBRE`) VALUES (?, ?, ?)";*

PROPIETARIO_TIENDA *= "SELECT * FROM TIENDAS WHERE USUARIOMAIL = ?";*

MOSTRAR_TIENDA_PRODUCTO *= "SELECT * FROM `PRODUCTOS_TIENDAS` WHERE `PRODUCTOID` = ?";*

BUSCAR_TIENDA *= "SELECT * FROM `TIENDAS` WHERE `CIF` =?";*

INSERTAR_USUARIO *= "INSERT INTO `USUARIOS`("`MAIL`,`DNI`,`USER_NAME`,`NOMBRE`,`PASSWORD`,`APELLIDO2`,`APELLIDO1`,`FECHA_NACIMIENTO`,`PAIS`,`IMAGEN_URL`,`PERFIL`,`SEXO`)VALUES (?,?,?,?,?,?,?,?,?,?,?,?)";*

INSERTAR_DIRECCION_USUARIO *= "INSERT INTO `DIRECCIONES`("`USUARIOMAIL`, `DIRECCION`,`NUMERO`,`PUERTA`,`ESCALERA`,`POBLACION`,`COD_POSTAL`,`PAIS`)"VALUES (?,?,?,?,?,?,?,?)";*

INSERTAR_TELEFONO_USUARIO *= "INSERT INTO `TELEFONOS`("`NUMERO`,`USUARIOMAIL`) "VALUES (?,?)";*

MOSTRAR_PRODUCTO *= "SELECT * FROM PRODUCTOS WHERE REFERENCIA = ?";*

INSERTAR_PRODUCTOS *= "INSERT INTO PROYECTOFINALDAW.PRODUCTOS ("`REFERENCIA`,`PRECIO`,`NOMBRE`,`DESCRIPCION`,`COLOR`,`TALLA`,`COMPOSICION`,`SEXO`,`ID_CATEGORIA`"`FECHA_CATALOGO`)VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";*

INSERTAR_PRODUCTOS_TIENDA *= "INSERT INTO PROYECTOFINALDAW.PRODUCTOS_TIENDAS (`PRODUCTOID`,`TIENDACIF`)"VALUES (?, ?);";*

WISH_LIST_BORRAR_PRODUCTO *= "DELETE FROM USUARIO_WISHLIST WHERE USUARIO = ? AND PRODUCTO = ?";*

CANTIDAD_WISH_LIST *= "SELECT COUNT(*) FROM USUARIO_WISHLIST WHERE USUARIO = ?";*

WISH_LIST *= "SELECT * FROM USUARIO_WISHLIST INNER JOIN PRODUCTOS ON USUARIO_WISHLIST.PRODUCTO = PRODUCTOS.REFERENCIA WHERE USUARIO_WISHLIST.USUARIO = ? LIMIT 0 , 30";*

INSERT_WISH_LIST *= "INSERT INTO `USUARIO_WISHLIST`(`USUARIO`, `PRODUCTO`) VALUES (?,?)";*

MOSTRAR_PRODUCTOS *= "SELECT * FROM `PRODUCTOS` ORDER BY `FECHA_CATALOGO` LIMIT 1,7";*

MOSTRAR_PRODUCTOS_TIENDA *= "SELECT * FROM `PRODUCTOS_TIENDAS` WHERE TIENDACIF = ?";*

MOSTRAR_PRODUCTOS_PEDIDOS *= "SELECT PRODUCTOS.* FROM PRODUCTOS INNER JOIN PEDIDOS ON PEDIDOS.PRODUCTOREFERENCIA = PRODUCTOS.REFERENCIA WHERE PEDIDOS.USUARIOMAIL = ? AND PEDIDOS.ESTADO_SERVICIO = 'PEDIDO'";*

MOSTRAR_CATEGORIAS_PRODUCTOS *= "SELECT * FROM `CATEGORIA`";*

AGRUPAR_PRODUCTOS_CLASIFICACION *= "SELECT `CLASIFICACION` FROM `CATEGORIA` GROUP BY `CLASIFICACION`";*

MOSTRAR_PRODUCTOS_CLASIFICACION *= "SELECT * FROM PRODUCTOS INNER JOIN CATEGORIA ON PRODUCTOS.ID_CATEGORIA = `IDCATEGORIA` WHERE CATEGORIA.IDCATEGORIA = ?";*

MOSTRAR_CATEGORIA_PRODUCTO *= "SELECT * FROM `CATEGORIA` WHERE IDCATEGORIA = ?";*

REALIZAR_PEDIDO *= "INSERT INTO `PEDIDOS`("`USUARIOMAIL`,`PRODUCTOREFERENCIA`,`TIENDACIF`,`FECHA_PEDIDO`,`FECHA_CONFIRMACION`,`FECHA_SERVICIO`,`UNIDADES`,`NUM_FACTURA`, `ESTADO_SERVICIO`)VALUES (?,?,?,?,?,?,?,?,?)";*

MOSTRAR_PEDIDOS *= "SELECT * FROM `PEDIDOS` WHERE `USUARIOMAIL` = ? AND `ESTADO_SERVICIO` = 'PEDIDO'";*

BORRAR_PRODUCTO_PEDIDO *= "DELETE FROM `PEDIDOS` WHERE `PRODUCTOREFERENCIA` = ? AND `USUARIOMAIL` = ?";*

ACTUALIZAR_PEDIDOS_COMPRADO *= "UPDATE `PEDIDOS` SET `FECHA_CONFIRMACION`=CURRENT_DATE(),`NUM_FACTURA`=?,`ESTADO_SERVICIO`= 'COMPRADO' WHERE `PRODUCTOREFERENCIA`= ?";*

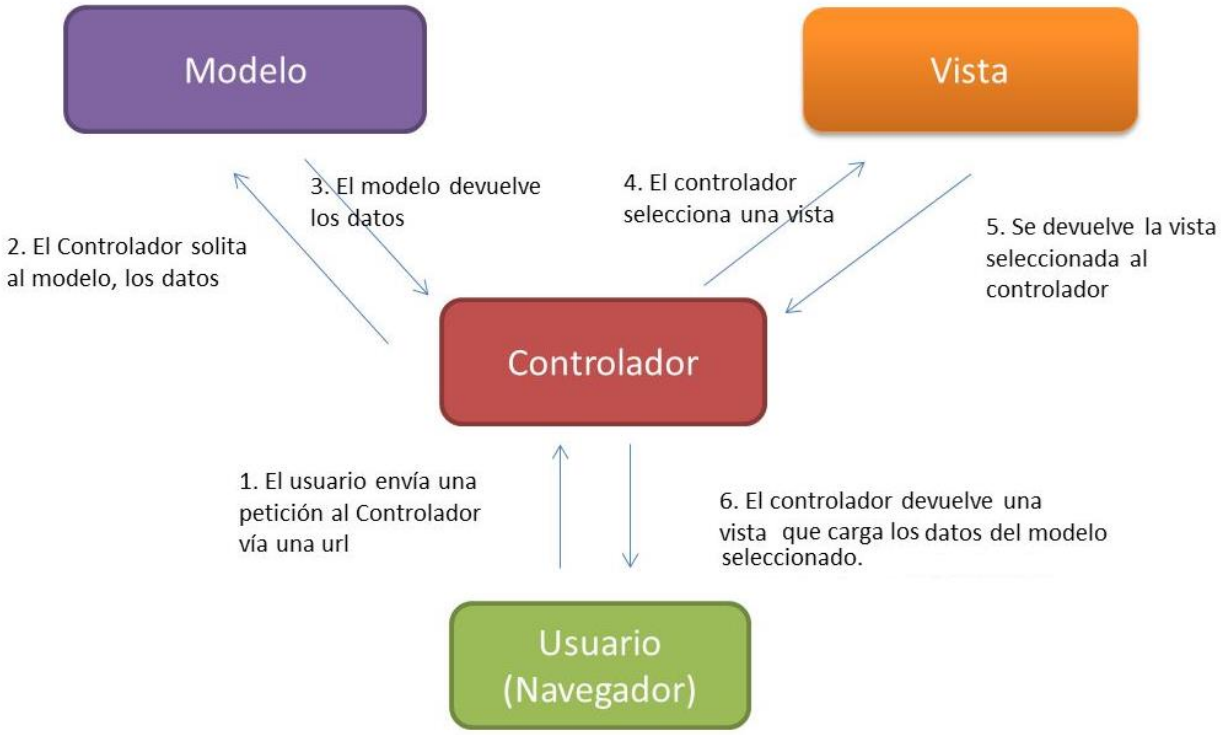
LENGUAJES

PATRONES DE DISEÑO

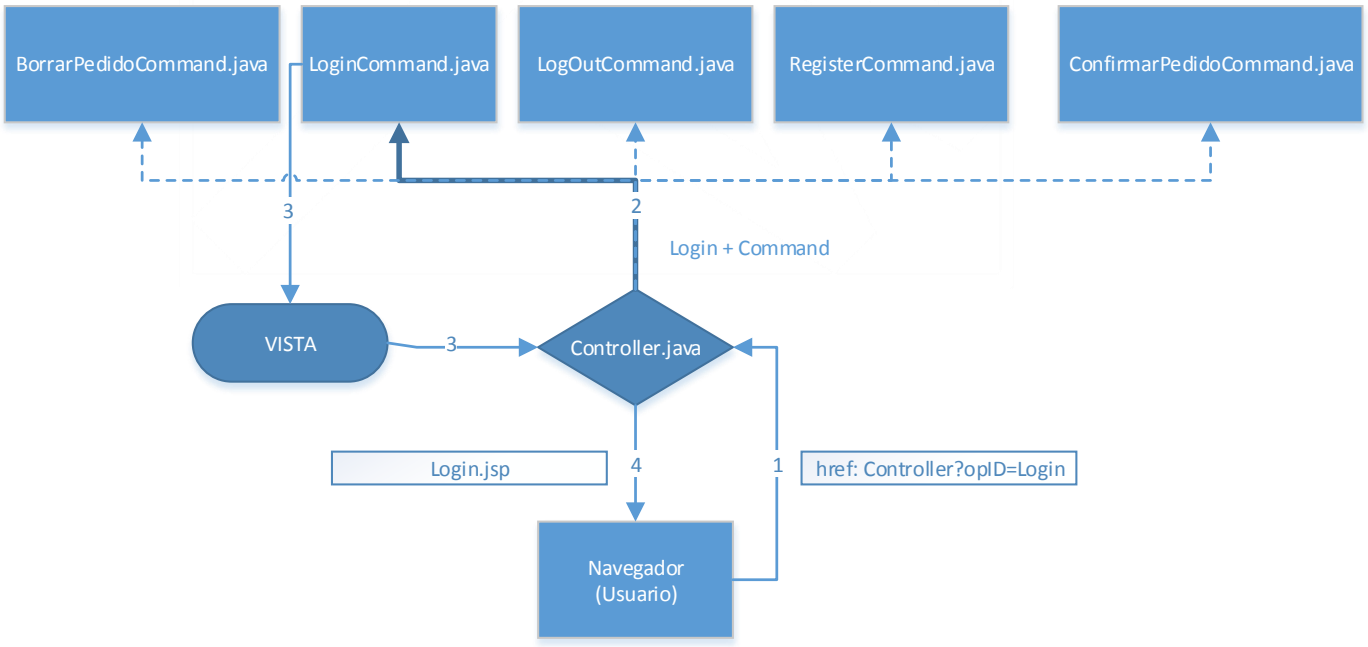
Modelo Vista Controlador – MVC

Es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.



APLICADO AL PROYECTO



Durante el proceso de desarrollo, surgió la necesidad de hacer un *login* usando el MVC integrado en el proyecto.

FUNCIÓN JAVA, PARA LLAMAR AL SERVLET:

```
$(document).ready(function () {
    $(".form-signin").submit(function () {
        $(".help-block-Login").html('<img src="images/loading.gif"/>');
        $.ajax({
            url: 'LoginUsuario',
            type: 'POST',
            dataType: 'json',
            data: $('.form-signin').serialize(),
            success: function (data) {
                if (!data.loginValido) {
                    $(".help-block-Login").hide().html('ERROR DE LOGIN').slideDown(500);
                } else {

                    document.location.href = "Controller?opID=Login";
                }
            }
        });
        return false;
    });
```

Básicamente esta clase lo que hace es detectar si en el formulario asignado se ha realizado una acción “SUBMIT”, la función AJAX recoge la información serializada del formulario. La URL del SERVLET al que va a llamar, y el método en el que le pasara la información.

Si el SERVLET le ha devuelto información, lo que hará es cargar mediante el modelo vista controlador, la página a la cual vamos a llamar.

SERVLET QUE RECOGERÁ LA INFORMACIÓN AJAX:

LIBRERÍAS NECESARIAS:

COM.GOOGLE.GSON.GSON;

JAVA.UTIL.HASHMAP;

JAVA.UTIL.MAP;

Es necesario poner la anotación al SERVLET, ya que debido a esta anotación AJAX sabrá que tiene que llamar a este SERVLET y no a otro.

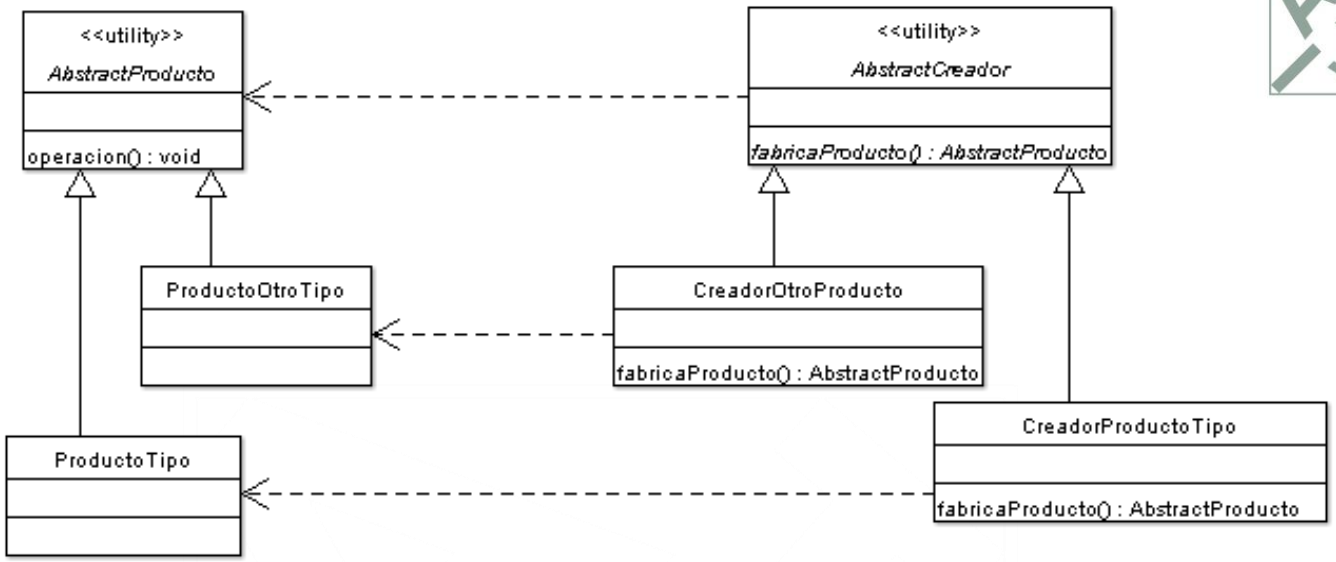
```java
@WebServlet("/LoginUsuario")
public class LoginUsuarioGson extends HttpServlet {

    public LoginUsuarioGson() {
        super();
    }
```

Una vez el SERVLET haya realizado sus acciones, mediante la clase MAP devolverá la información al AJAX en caso de que esta haya sido exitosa.

```java
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    Map<String, Object> map = new HashMap<>();
    boolean loginValido = false;
    UsuarioBLL usuarioBll = new UsuarioBLL();
    Usuarios usuario = new Usuarios();
    Usuarios sesion = null;
    String mail = request.getParameter("mail");
    String password = request.getParameter("password");

    usuario.setMail(mail);
    usuario.setPassword(password);

    try {
        sesion = usuarioBll.loginUsuario(usuario);
    } catch (Exception ex) {
        sesion = null;
    }

    if (sesion != null) {
        loginValido = true;
        map.put("mail", mail);
        map.put("password", password);
        request.getSession().setAttribute("usuarioValido", sesion);
    }
    map.put("loginValido", loginValido);
    write(response, map);
}

private void write(HttpServletResponse response, Map<String, Object> map) throws IOException {
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(new Gson().toJson(map));
}
```

Factory Method

El patrón de diseño FACTORY METHOD consiste en utilizar una clase constructora abstracta con unos cuantos métodos definidos y otro(s) abstracto(s): el dedicado a la construcción de objetos de un subtipo de un tipo determinado.



Las clases principales en este patrón son el creador y el producto. El creador necesita crear instancias de productos, pero el tipo concreto de producto no debe ser forzado en las subclases del creador, porque las posibles subclases del creador deben poder especificar subclases del producto para utilizar.

Define una interfaz para crear objetos pero deja que sean las subclases las que deciden qué clases instanciar.

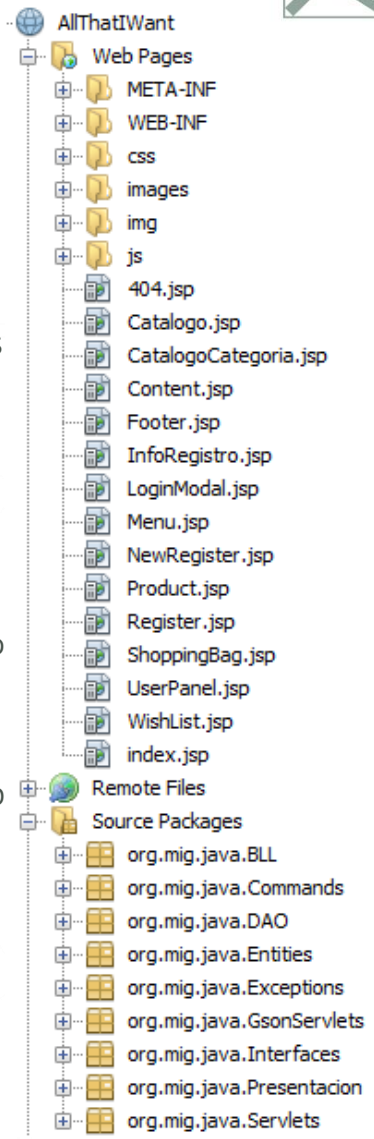
El patrón de diseño *factory method* consiste en definir una interfaz para crear objetos de tipo genérico permitiendo a las subclases decidir qué tipo de objetos concreto crear.

El patrón *factory method* permite escribir aplicaciones que son más flexibles respecto de los tipos a utilizar difiriendo la creación de las instancias en el sistema a subclases que pueden ser extendidas a medida que evoluciona el sistema. Permite también encapsular el conocimiento referente a la creación de objetos, hace que el diseño sea más adaptable a cambio de sólo un poco más de complejidad. Como inconveniente observamos que se obliga al cliente a definir subclases de la clase “creador” sólo para crear un producto concreto y esto puede no ser apropiado en todas las ocasiones a lo largo de la vida del proyecto.

ESTRUCTURA

La estructura del proyecto estará basada en:

- **WEB PAGES**: CARPETA RAÍZ DE TODOS EL CONTENIDO DEL PROYECTO.
  - **CSS**: CARPETA CONTENEDORA DE LAS HOJAS DE ESTILOS
  - **IMAGES**: CARPETA DONDE SE GUARDARAN LAS PROYECTO.
  - **IMG**: CARPETA DONDE SE GUARDAN IMÁGENES ESPECIFICAS DE JS'S
  - **JS**: CARPETA DONDE SE ALMACENAN TODOS LOS JQUERY'S.
  - **JSP**
- **SOURCE PACKAGES**
  - **ORG.MIG.JAVA.BLL** -> CLASES DE NEGOCIO
  - **ORG.MIG.JAVA.COMMANDS** -> CLASES VISTA DEL MVC
  - **ORG.MIG.JAVA.DAO** -> CLASES DAO
  - **ORG.MIG.JAVA.ENTITIES** -> ENTIDADES
  - **ORG.MIG.JAVA.EXCEPTIONS** -> CLASES RELACIONAD GESTION DE EXCEPCIONES
  - **ORG.MIG.JAVA.GSONSERVLETS** -> SERVLETS DE AJAX
  - **ORG.MIG.JAVA.INTERFACES**-> INTERFACES
  - **ORG.MIG.JAVA.PRESENTACION** -> CLASE DEL CO MVC
  - **ORG.MIG.JAVA.SERVLETS** -> SERVLETS INDIVIDUALES



DISEÑO

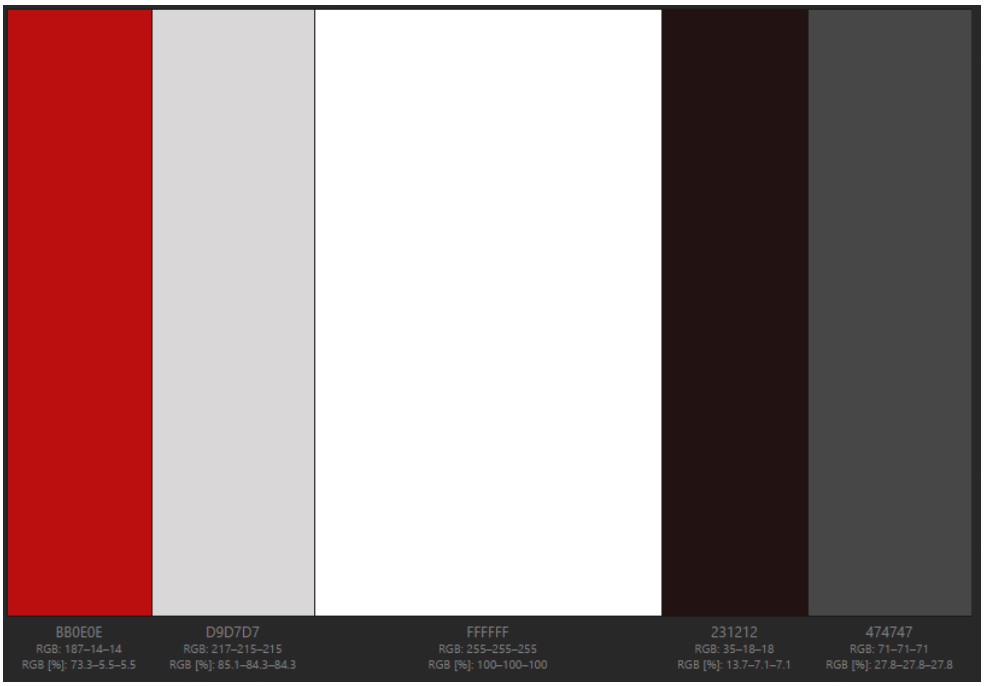
PALETA DE COLORES

A la hora de diseñarlo decidí no hacer algo excesivamente agobiante para el usuario, por eso decidí por unas tonalidades claras, con pocos elementos oscuros.

Estos elementos son elementos secundarios, que el usuario no tendrá un acceso directo a ellos.

Elementos como pueden ser el *footer*, la zona del *copyright*

.



FUENTE

- ROBOTO
- CHRISTIAN ROBERTSON
- APACHE LICENSE, VERSION 2.0

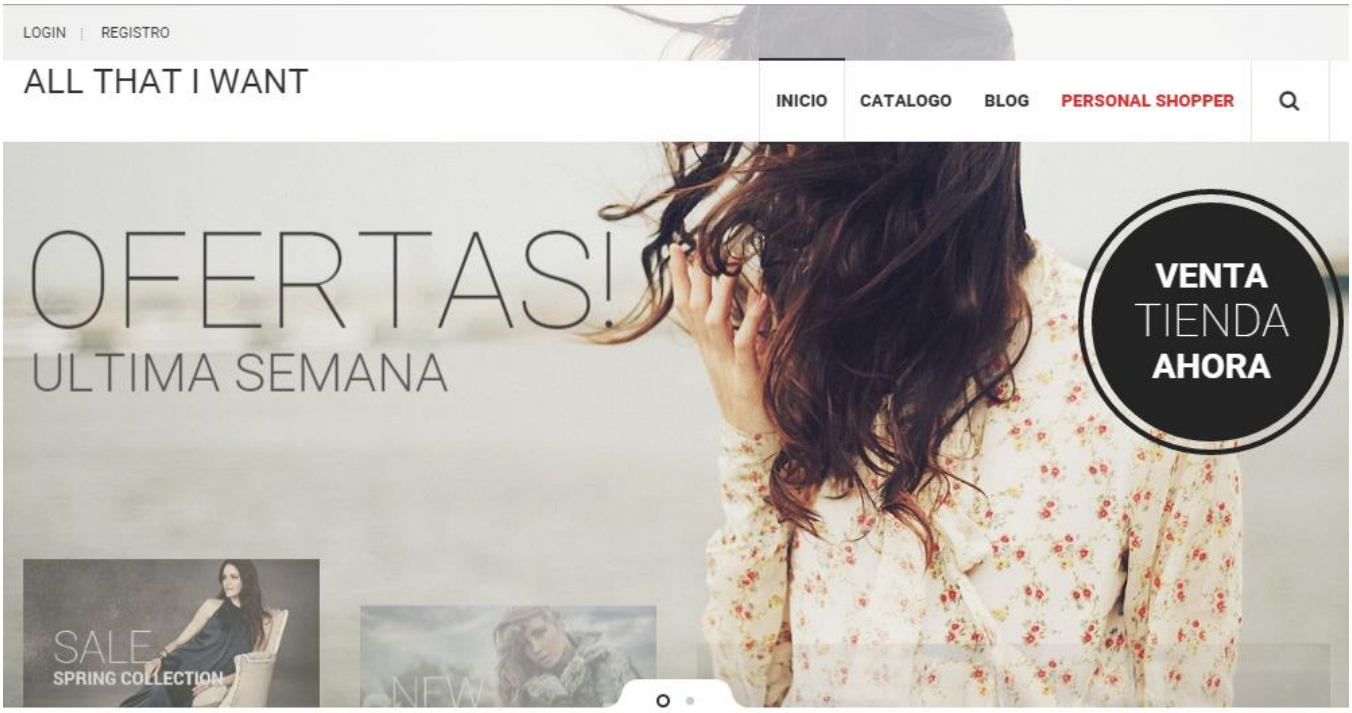
Roboto es una fuente con un esqueleto y una mecánica de geometría larga, a su vez también es amigable y tiene unas curvas abiertas, esto hace que sea fácil de leer, y no resulte pesada.

Además es de tipo sheriff, y es una de las fuentes más fáciles de leer para el ojo humano, según estudios realizados.

CARACTERÍSTICAS DEL SITIO

- RESPONSIVE
- HTML5
- CSS3
- BOOTSTRAP

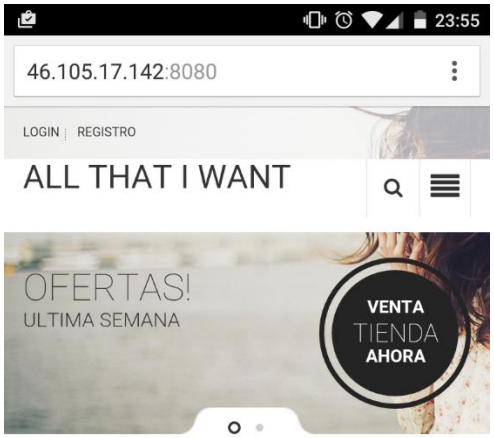
VERSION ESCRITORIO:

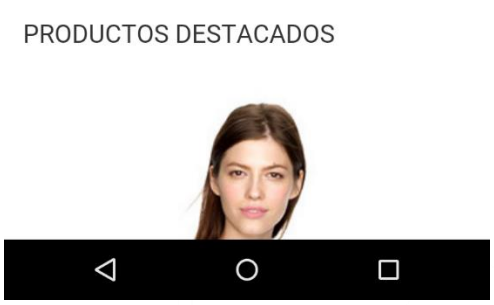


PRODUCTOS DESTACADOS




VERSION SMARTPHONE:



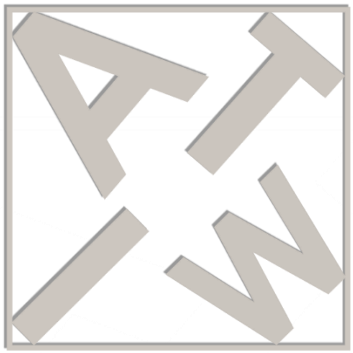


Toda la página esta creada con el FRAMEWORK BOOTSTRAP, debido a su "facilidad" para crear con poco esfuerzo logístico paginas *responsive* y con un resultado muy profesional.

BOOSTRAP cuenta con herramientas como BOOTSNIPPETS, que ofrecen soluciones con la tecnología BOOTSTRAP tales como, formularios de registro, inputs de selección de países, y demás utilidades.

LOGO

FAVICON



LOGO PRINCIPAL

IMPLANTACIÓN

SERVICIOS

El servidor VPS que he contratado se basa en un sistema DEBIAN “virgen”, en el he tenido que instalar todos los sistemas necesarios para poder desplegar satisfactoriamente el proyecto.

Mediante la herramienta PUTTY, he configurado los siguientes servicios.

JAVA VIRTUAL MACHINE 8: el proyecto está desarrollado con tecnología JAVA V8, es por ello que surgió la necesidad de instalar en el servidor la versión 8 de java, especialmente tedioso puesto que no hay una versión “libre” de la JDK y hay que bajárselo de terceros e instalarlo.

APACHE TOMCAT V7: para poder desplegar las aplicaciones J2EE es necesario APACHE TOMCAT, y su respectiva configuración, para integrarlo satisfactoriamente con el JDK8.

PROFTPD: para subir contenido al servidor web, es necesario un sistema de transferencia de archivos, en este caso he elegido PROTFPD.

APACHE2: con la necesidad de evitar que TOMCAT sirva las imágenes que se suben de los productos y usuarios, me vi con la necesidad de instalar APACHE2 como servidor web.

HERRAMIENTAS

- PUTTY: para la conexión al servidor.
- FILEZILLA: para subir archivos al servidor.

CONCLUSIONES

ALL THAT I WANT ha sido todo un reto al que todavía le queda mucho camino.

Durante el proceso de desarrollo el proyecto ha sufrido modificaciones, tanto estructuralmente, como ideológicamente.

Estos cambios forzados han sido efecto del estudio de nuevas tecnologías, nuevos patrones de diseño o simplemente efecto de carencias a la hora del desarrollo inicial del mismo.

Todos los problemas que han surgido por el camino han sido solucionados en función de la urgencia de este, y con una solución más o menos “limpia”.

FUTURO

- implementar el personal *shopper*.
- realizar un panel de usuario más avanzado.
- permitir realizar búsquedas pasándole unos filtros.
- implementar el registro de tiendas.
- entre otras cosas e ideas, que sin lugar a duda irán surgiendo durante el camino.

MEJORAS

- mejorar la experiencia del usuario durante el registro.
- mejorar la experiencia del usuario durante el *login*.

FUENTES

- www.google.com
- www.wikipedia.com
- http://www.tutorialspoint.com/
- http://tecadmin.net/
- toda documentación oficial, apache commons, oracle, debian…