

*Creación de una página Web
con Java Empresarial*

Innovación aplicada

*Desarrollo de aplicaciones
Multiplataforma*

*I.E.S Valle del Jerte
Plasencia - Curso 2016*

*Laura Irina
Huaycama
Mendoza*



*Fecha
13 / 12 / 2016*

Indice de Memoria

1. Descripción.....	3
2. Objetivos	3
3. Justificación.....	3
4. Descripción técnica.....	3
4.1 Descripción general.....	3
4.2 Instalación de herramientas en el servidor.....	4
a) Conexión con el servidor:	4
b) Instalación instalación apache tomcat en el servidor:.....	4
c) Instalación instalación mysql en el servidor:	5
conexión con el Mysql:.....	6
Creación de base de datos y tablas.....	6
4.3 Crear Proyecto Web dinamic.....	7
a) Configuración a la base de datos.....	8
b) Configurar JPA.....	10
4.4 Convertir proyecto a maven.....	13
4.5 Configurar servlet y jsp.....	14
4.6 Creación de clases.....	16
1) Clase JpaConfig.....	16
2) Creación GenericDao y GenericDaoImpl.....	17
3) Clase UsuarioController.....	19
4) Clase AdminController.....	22
5) Clase UsuarioService.....	26
6) Clase UsuarioDao.....	28
7) Clase RecetaDao.....	29
8) Clase ComentarioDao.....	30
9) Clase BlogException.....	31
4.6 Contenido WebContent.....	32
1) Jsp cabecera.....	33
2) Jsp pie.....	34
3) Jsp Inicio usuario.....	35
4) Jsp Mostrar receta.....	36
5) Jsp Registrarse.....	37
6) Jsp Iniciar sesión.....	38
7) Jsp Pagina usuario.....	38
8) Jsp ingresar receta.....	39
5. Explicación del funcionamiento del controller y el jsp.....	40
5.1 Notas importantes.....	42
6. Medios utilizados	42
7. BIBLIOGRAFIA.....	43

1. Descripción

Se crea este proyecto con el fin de aplicar algunas de las herramientas que se usan en la creación de las páginas web es: ver lo que hay detrás de ellas. En la cual esta página se pueda simular alguna de la funcionalidades que tiene Facebook, Twitter, Viky, comunidades de blogs, etc, como son subir una foto o registrarse o iniciar sesión o dejar un comentario. Y para conseguir esto se necesitará una base de datos para almacenar la información y luego recuperarlo para poder mostrarlo en nuestra página web.

2. Objetivos

Esta página web está diseñado con el objetivo de llegar a todas las personas que deseen compartir sus recetas, y se creará un entorno en el cual los usuarios puedan subir, comentar, consultar recetas, registrarse, iniciar sesión, eliminar comentarios y eliminar recetas. Y que lo puedan usar usuarios registrados como no registrados. Es aprender a crear este tipo de páginas web y todas las funcionalidades que contiene.

3. Justificación

Esta idea ha sido por un proyecto que realice en una empresa. En la cual era crear un blog para poder simular una página web, que aparte pueda conectarse una base de datos en la cual sube datos, borran, registran, iniciar sección, etc.

Es algo muy interesante la creación de este tipo de páginas web porque actualmente en el mercado las empresas necesitan este tipo de páginas web sea para la venta de productos o registrar una reserva en hospedaje o simples blogs en los cuales sus usuarios puedan usarlo.

Considero que es un campo amplio en este tema y además para la creación de estas páginas se puede usar diferentes herramientas y lenguajes de aplicación. Como usar PHP para la recuperación o envío con la conexión a la base de datos que en este proyecto no se usará, sino a cambio usaremos JPA (Java Persistence API, es la API de persistencia desarrollada para la plataforma Java EE). También veremos cómo configurar un servidor para poder desplegar nuestro proyecto y la conexión con una base de datos que se encontrara en el servidor.

4. Descripción técnica

4.1 Descripción general

Se usa un servidor Ubuntu, y se instalará el Apache Tomcat para poder desplegar proyectos .war y MySQL para el almacenamiento de la base de datos.

El proyecto se realizará en Eclipse con el sistema operativo OS X El Capitan (MAC). Para Java EE, siendo un conjunto de módulos de aplicación Web completa (empaquetada archivo .war) o un conjunto de objetos distribuidos EJBs (empaquetados en archivo .jar). Y en este proyecto se usará el archivo .war para que luego se despliegue en el servidor. Se usa Java Web Dynamic Project para la creación de este tipo de proyecto, aparte de diferentes lenguajes de programación: Java, Maven, JPA, HTML, CSS, Hibernate, JavaScript, SQL, etc.

Será un buen resumen de todo lo aprendido en el curso y algunos temas nuevos que son usados actualmente.

4.2 Instalación de herramientas en el servidor

El Servidor es Ubuntu 12.04.

Nombre de dominio: alumnos.iesvjp.es

Transcripción de puertos:

16180 :80(Ahora mismo corre ahí apache, que no apache tomcat)

16181 : 8080 (Para que corra apache)

16122 : 22 (Para ssh)

16133 : 3306 (Para Mysql)

a) Conexión con el servidor:

Para la conexión con el servidor es abrir un terminal y escribir:

ssh -p 16122 root@alumnos.iesvjp.es

a continuación pedirá una contraseña que es P@ssw0rd. Con esto ya estaríamos conectados al servidor.

b) Instalación instalación apache tomcat en el servidor:

- sudo install java-8-openjdk-amd64/jre
- sudo apt-get update
- sudo apt-get install default-jdk
- sudo groupadd tomcat8
- sudo useradd -s /bin/false -g tomcat -d /opt/tomcat8 tomcat
- wget http://www-us.apache.org/dist/tomcat/tomcat-8/v8.0.33/bin/apache-tomcat-8.0.33.tar.gz
- sudo mkdir /opt/tomcat8
- sudo tar xvf apache-tomcat-8*tar.gz -C /opt/tomcat8—strip-components=1
- cd /opt/tomcat8
- sudo chgrp -R tomcat conf
- sudo chmod g+rwx conf
- sudo chmod g+r conf/*
- sudo nano /etc/init/tomcat.conf

```
laurairinamendoza — root@Laura-Proyecto-Jesus: ~ — ssh -p 16122 root@alumnos.iesvjp.es
GNU nano 2.2.6          File: /etc/init/tomcat.conf

[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking
Environment='JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre'
Environment='CATALINA_PID=/opt/tomcat8/temp/tomcat.pid'
Environment='CATALINA_HOME=/opt/tomcat8'
Environment='CATALINA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'
Environment='JAVA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
ExecStart=/opt/tomcat8/bin/startup.sh
ExecStop=/opt/tomcat8/bin/shutdown.sh
User=tomcat
Group=tomcat

[Install]
WantedBy=multi-user.target
description "Tomcat Server"
start on runlevel [2345]
stop on runlevel [!2345]
respawn
respawn limit 10 5
setuid tomcat
setgid tomcat
env JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64/jre
env CATALINA_HOME=/opt/tomcat8
```

- sudo initctl reload-configuration

Y con esto ya estaría ejecutando tomcat en el servidor. Hay que dar permisos en el tomcat para poder usar el modo de interfaz.

- sudo nano /opt/tomcat8/conf/tomcat-users.xml

```
GNU nano 2.2.6      File: /opt/tomcat8/conf/tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
               version="1.0">

<role rolename="manager-gui" />
<user username="laura" password="secreto12" roles="manager-gui" />
```

- root@Laura-Proyecto-Jesus:/opt/tomcat8/bin# ./startup.sh

Para entrar en la interfaz del apache tomcat solo hay que poner el usuario : laura y contraseña: secreto12.

Tomcat Web Application Manager					
Message: <input type="text" value="OK"/> Manager					
List Applications		HTML Manager Help		Manager Help	
Server Status					
Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions with idle ≥ 30 minutes"/>
/FotoWeb	None specified		true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions with idle ≥ 30 minutes"/>
/Imagenes	None specified		true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions with idle ≥ 30 minutes"/>
/ProyectoBlog	None specified	ProyectoBlog	true	1	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions with idle ≥ 30 minutes"/>

c) Instalación instalación mysql en el servidor:

- sudo apt-get update
- sudo apt-get install mysql-server-5.1
Contraseña: 123
- sudo netstat -tap | grep mysql
- mysql -uroot -p
- dpkg -l | grep mysql
- Hay que dar un password al root para conexión con la base de datos: mysql>
UPDATE mysql.user SET Password=PASSWORD('admin') WHERE User='root';
mysql> FLUSH PRIVILEGES;
- root@Laura-Proyecto-Jesus:~# sudo /etc/init.d/mysql start

ci) conexión con el Mysql:

Para la comenzar sesión con el Mysql: mysql -u root -p123

Ya estamos dentro del mysql y comenzamos a la creación de la base de datos.

cii) Creación de base de datos y tablas

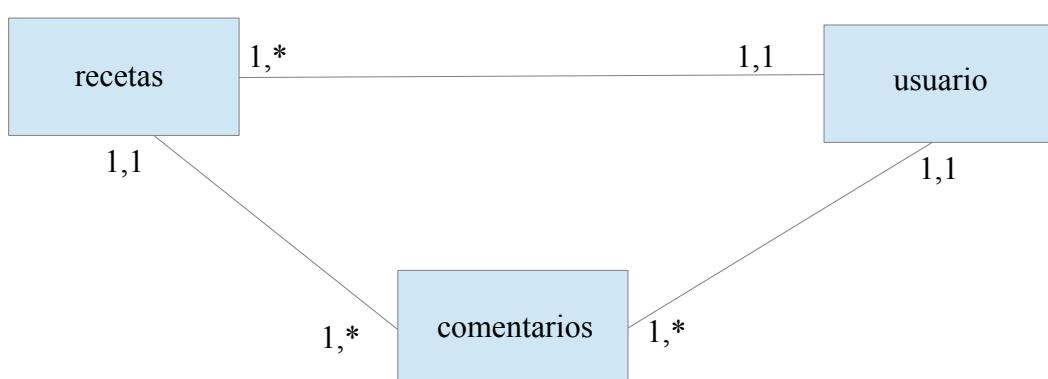
Primero hay que crear la base de dato que se llamará BDblog.

mysql> CREATE database BDblog;

Entramos en la base de datos.

mysql> use BDblog;

- Diseño de tablas



- Crear tablas:

```

CREATE TABLE usuario(
    idusu INTEGER NOT NULL AUTO_INCREMENT,
    nombreusuario VARCHAR(20) NOT NULL,
    clave VARCHAR(50) NOT NULL,
    PRIMARY KEY(idusu),
    UNIQUE (nombreusuario) );
  
```

```

CREATE TABLE recetas(
    idrect INTEGER NOT NULL AUTO_INCREMENT,
    nombrerect VARCHAR(500) NOT NULL,
    ingredientes VARCHAR(4000) NOT NULL,
    resumen VARCHAR(4000) NOT NULL,
    fechapubli DATE,
    descripcion VARCHAR(5000) NOT NULL,
    categoria VARCHAR(100) NOT NULL,
    usuarioid INTEGER NOT NULL ,
    imagen VARCHAR(2000),
    PRIMARY KEY(idrect),
  
```

FOREIGN KEY (usuarioid) REFERENCES usuario (idusu) ON DELETE CASCADE);

CREATE TABLE comentarios(

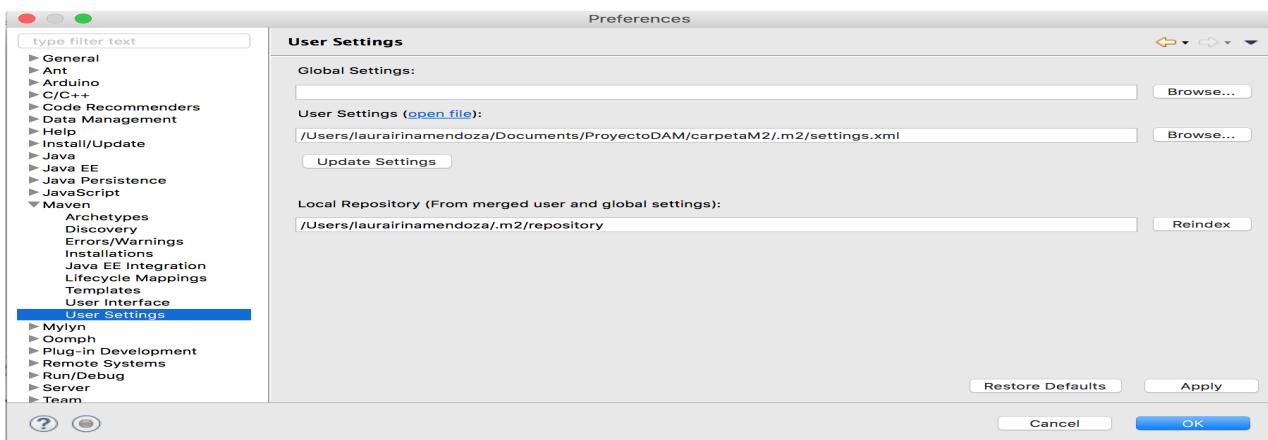
```
  idcoment INTEGER NOT NULL AUTO_INCREMENT,
  descripcoment VARCHAR(500) NOT NULL,
  personaid INTEGER NOT NULL,
  recetaid INTEGER NOT NULL,
  PRIMARY KEY(idcoment),
  FOREIGN KEY (personaid) REFERENCES usuario (idusu) ON DELETE CASCADE,
  FOREIGN KEY (recetaid) REFERENCES recetas (idrect) ON DELETE CASCADE);
```

Solo para los usuarios no registrados se crea el usuario anónimo para poder reconocerlos cuando dejen comentarios y acceso limitado a la pagina web.

```
INSERT INTO usuario(nombreusuario, clave) VALUES('laura',
'296ef2014ffd08ea17aef007044f4f7851daa33d'),
INSERT INTO usuario(nombreusuario, clave) VALUES('anonimo',
'b665e217b51994789b02b1838e730d6b93baa30f');
```

4.3 Crear Proyecto Web dinamic

Primero hay que configurar el proxy maven en eclipse, porque se usara maven para este proyecto. Se abre eclipse → Preferencias → Use settings → añadimos el fichero que se encuentra en la carpeta carpetaM2/.m2/settings.xml.



Con esto ya comenzaríamos con la creación del proyecto.

Luego crear el File → Other → Server → Tomcat v8.0 server → Finish

File → New → Project → Dynamic Web Project → Apache Tomcat v8.0 → Finish

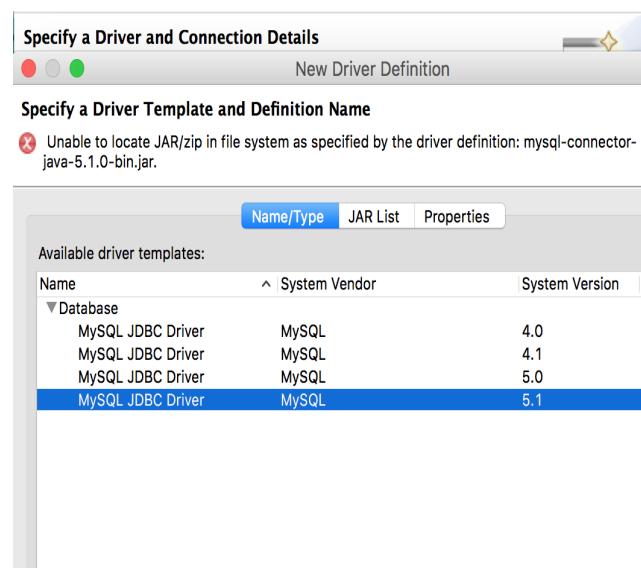
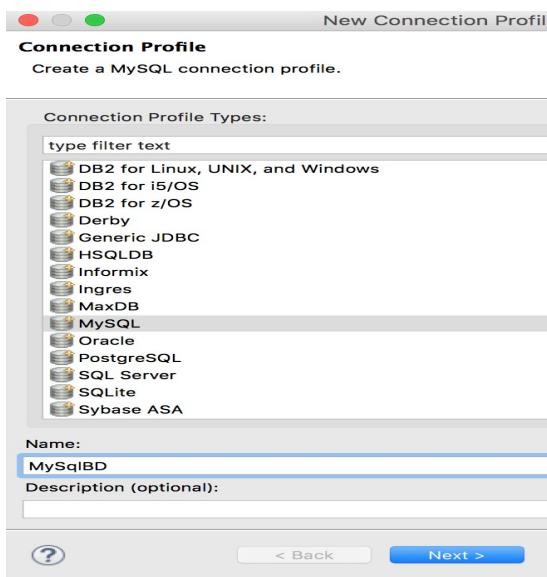
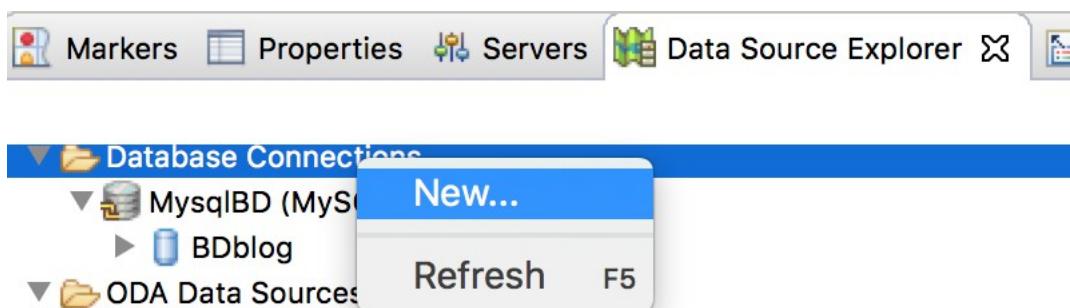


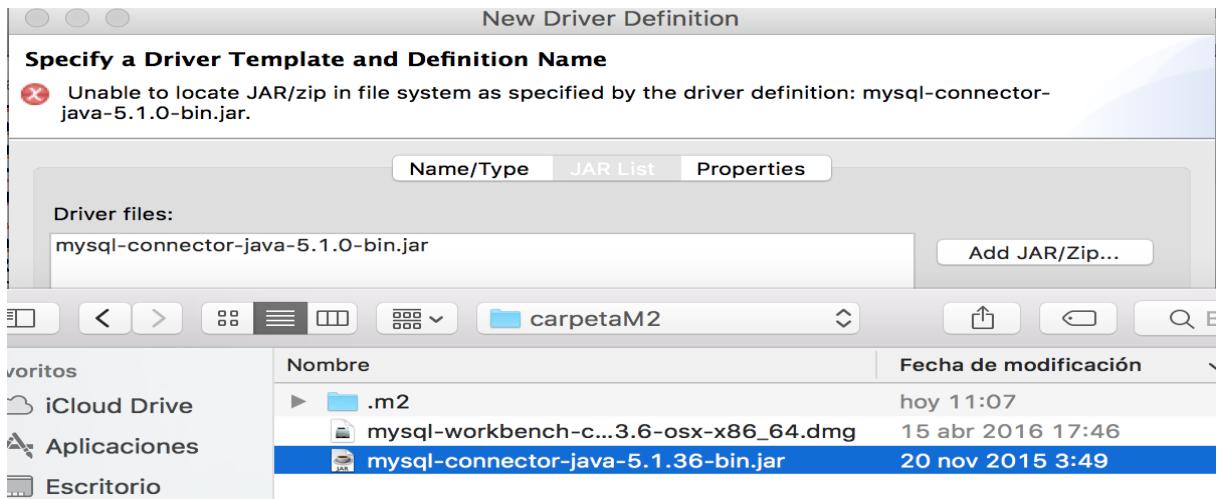
- **Java Resources:** En esta carpeta estará los archivos *.java, eclipse se encarga automáticamente de publicar los *.class en donde corresponda.
- **WebContent:** En esta carpeta estará los archivos Html, Jsp, Css,etc. dentro de esta carpeta podemos ver la estructura estándar de J2EE para un proyecto Web con la carpeta WEB-INF, el archivo web.xml, etc.

En este proyecto se usara Hibernate que es una librería ORM (Object Relational Mapping) para Java, que provee un framework para el mapeo de modelos de dominio OO (Object Oriented) a Bases de Datos Relacionales. Y que implementa la especificación JPA.

a) Configuración a la base de datos

Hay que crear la conexión de la base de datos que ya existe en el servidor con el proyecto. Y para eso creamos una Database Connections y quitamos el driver que viene por defecto y añadimos el driver de mysql que se encuentra en la carpetaM2.

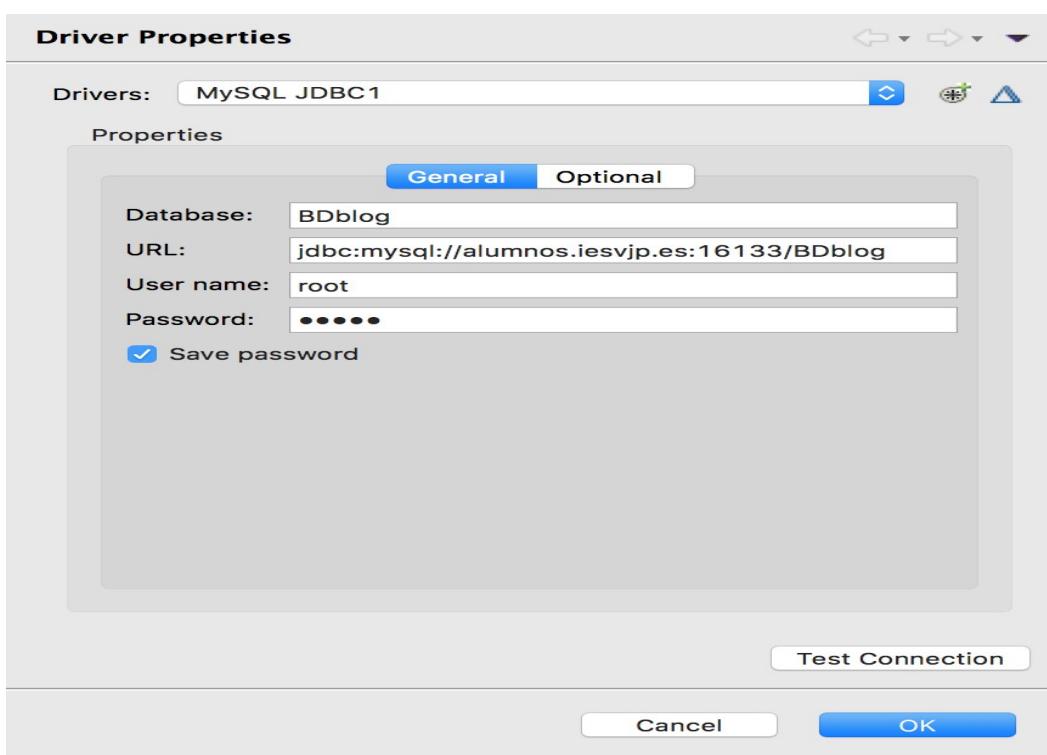


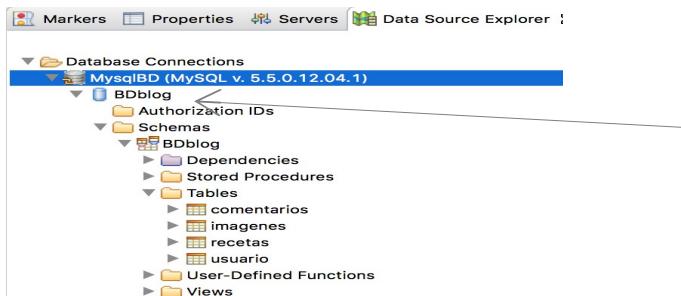


El siguiente paso a rellenar es como se muestra en la imagen, para conectarse a la base de datos BDblog que esta en el servidor alumnos.iesvjp.es con puerto 16133.

El usuario root y contraseña admin. Indicar guardar password.

Se puede verificar si la conexión es correcta haciendo click en Test Connection y si es correcta te confirmara que la conexión se hizo. Y finalizamos dando ok.





Esto es lo que se creara cuando se conecte con la base de datos. Se puede ver las tablas que tiene BDblog.

Las actividades básicas de programación del JDBC que utiliza:

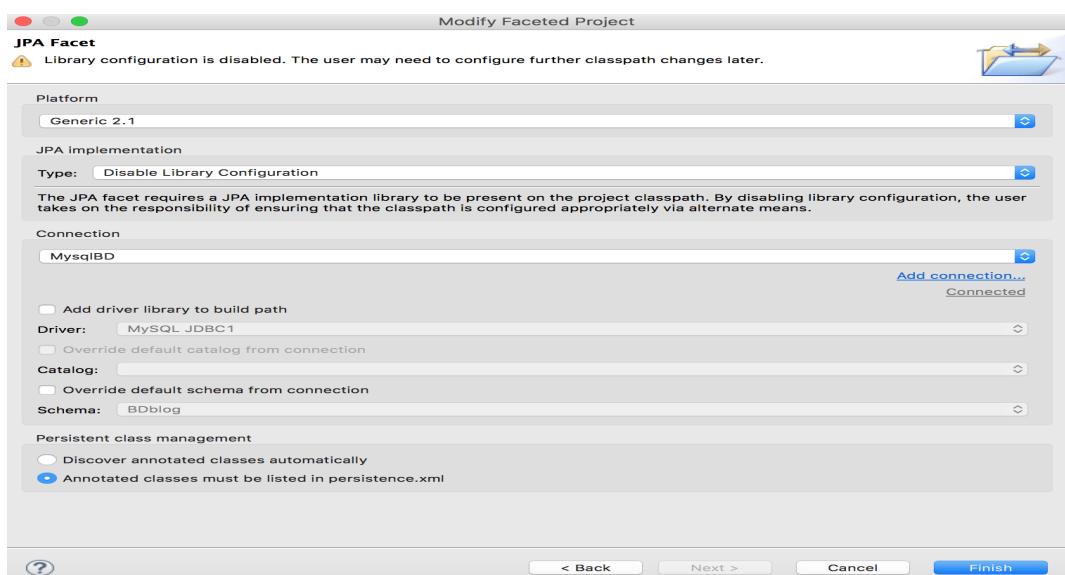
- Conectarse a la fuente de datos, a la base de datos MySQL.
- Enviar Querys y Updates a la base de datos.
- Recuperar y procesar los resultados obtenidos de la base de datos en respuesta a la query enviada.

b) Configurar JPA

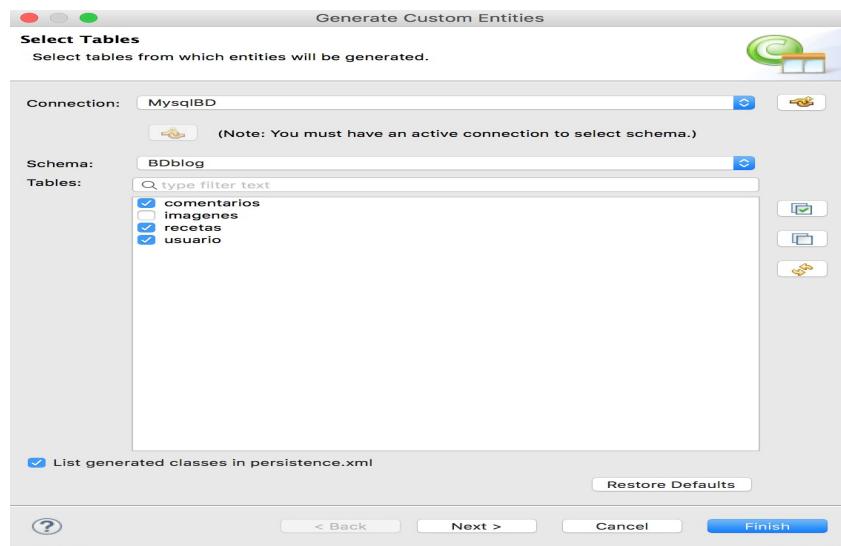
Con el JPA podemos tener la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad (por el mapeo), por lo que no se requieren ficheros descriptores XML.

Para crear las entidades hacemos click derecho en el proyecto → configure → convert to JPA Project → hacemos click en next.

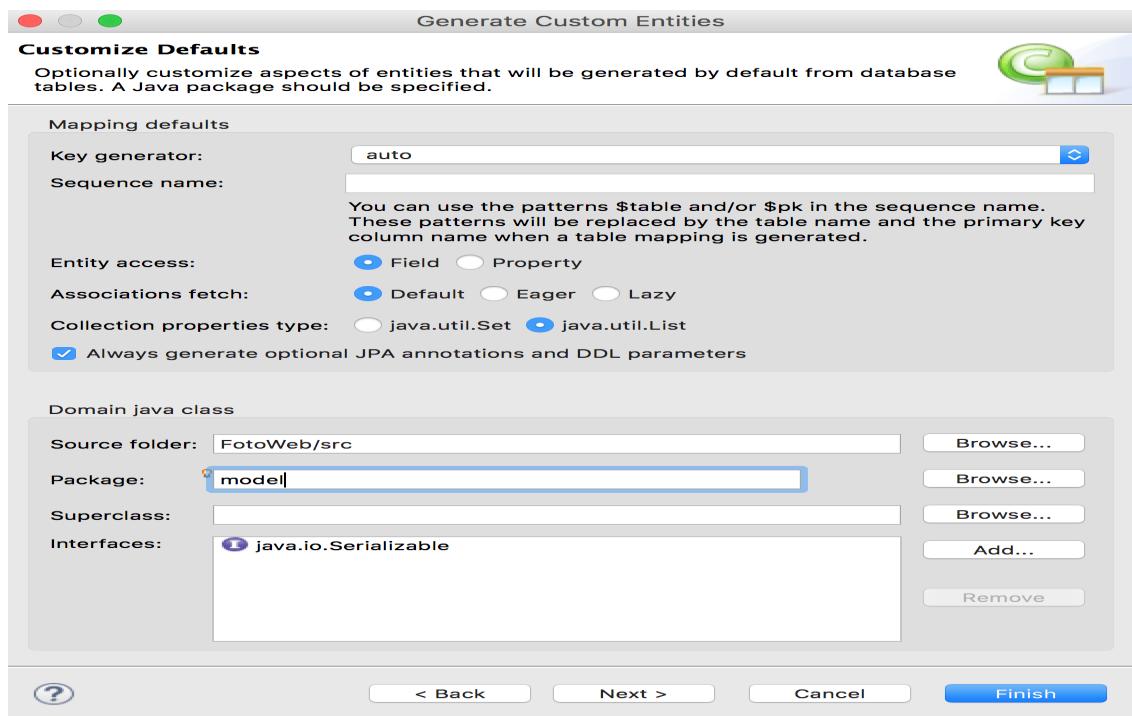
Indicamos todo igual a la imagen. Seleccionamos el MysqlBD creado que tiene la conexión con la base de datos BDblog. Y finalizamos.



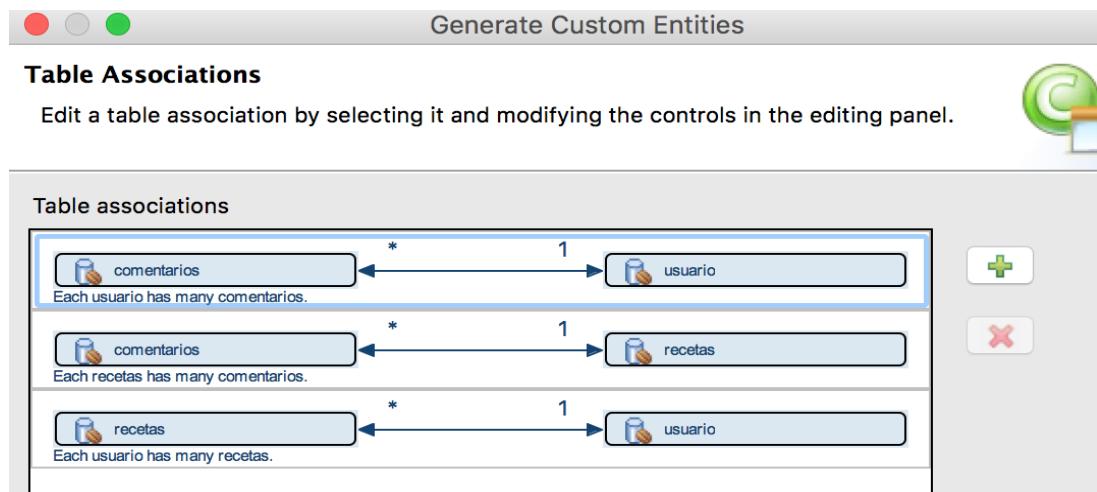
Lo siguiente es crear las entidades desde las tablas para ello, volvemos hacer click derecho en el proyecto → JPA Tools → Generates Entities from tables. Seleccionamos las tablas usuario, recetas, comentarios y damos a next.



Indicamos todo igual a la imagen, las tablas se guardarán en el paquete model y damos next.



Se podrá apreciar la asociación que tienen las tablas. Damos next.



En cada una de las tablas cambiaremos el tipo que tienen por defecto en las claves principales, en el Mapping type tienen int esto lo cambiaremos a Integer que seria por el objeto. Luego damos finalizar después de los cambios. Y se habrá creado el paquete model con las clases.

Generate Custom Entities

Customize Individual Entities

Tables and columns

recetas
usuario
idusu
clave

Generate this property

Column mapping

Property name: idusu

Mapping type: Integer

Mapping kind: id

Column is updatable

Column is insertable

Domain Java Class

Getter scope: public

Setter scope: public

Project Explorer

- FotoWeb
- ProyectoBlog
 - Deployment Descriptor: ProyectoB
 - JAX-WS Web Services
 - JPA Content
 - Java Resources
 - src
 - config
 - controller
 - controller.session
 - dao
 - exception
 - model
 - Comentario.java
 - Receta.java
 - Usuario.java
 - service
 - META-INF
 - Libraries
 - JavaScript Resources
 - Deployed Resources
 - build
 - codigotablas
 - BaseD.txt
 - Imagenes
 - target
 - WebContent
 - pom.xml
 - Servers
 - Servlet1

Usuario.java

```

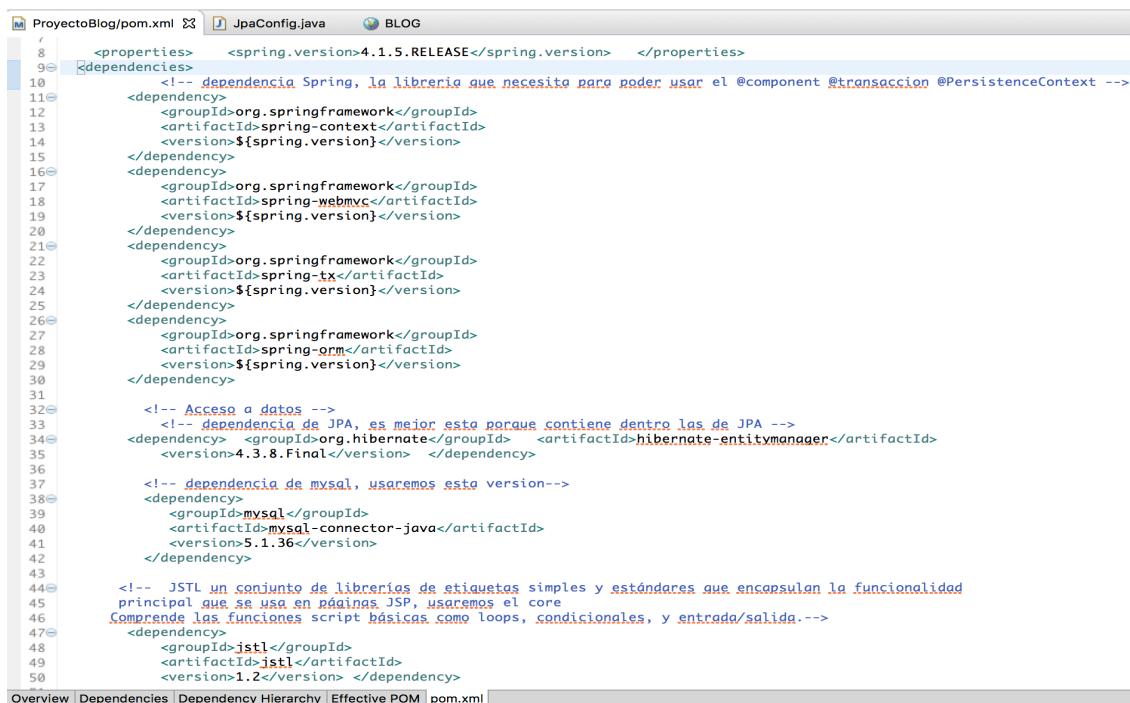
1 package model;
2
3 import java.io.Serializable;
4
5 /**
6  * The persistent class for the usuario database table.
7  */
8
9 @Entity
10 @Table(name="usuario")
11 @NamedQuery(name="Usuario.findAll", query="SELECT u FROM Usuario u")
12 public class Usuario implements Serializable {
13     private static final long serialVersionUID = 1L;
14
15     @Id
16     @GeneratedValue(strategy=GenerationType.AUTO)
17     @Column(unique=true, nullable=false)
18     private Integer idusu;
19
20     @Column(nullable=false, length=50)
21     private String clave;
22
23     @Column(nullable=false, length=20)
24     private String nombreusuario;
25
26     //bi-directional many-to-one association to Comentario
27     @OneToMany(mappedBy="usuario")
28     private List<Comentario> comentarios;
29
30     //bi-directional many-to-one association to Receta
31     @OneToMany(mappedBy="usuario")
32     private List<Receta> recetas;
33
34     public Usuario() {
35     }
36
37     public Integer getIdusu() {
38         return this.idusu;
39     }
40
41     public void setIdusu(Integer idusu) {
42         this.idusu = idusu;
43     }
44
45 }
  
```

4.4 Convertir proyecto a maven

El proyecto lo convertimos a maven con el fin de poder usar mas fácil los jar que necesitemos en el transcurso del desarrollo del proyecto. Y después de crear el JPA habrá diferentes errores porque se necesita las dependencias que tiene el JPA.

Click derecho en el proyecto → Configure → Convert to Maven Project → dejamos todo los datos por defecto y finish.

En el pom.xml es donde guardaremos las dependencias que se necesiten.



```

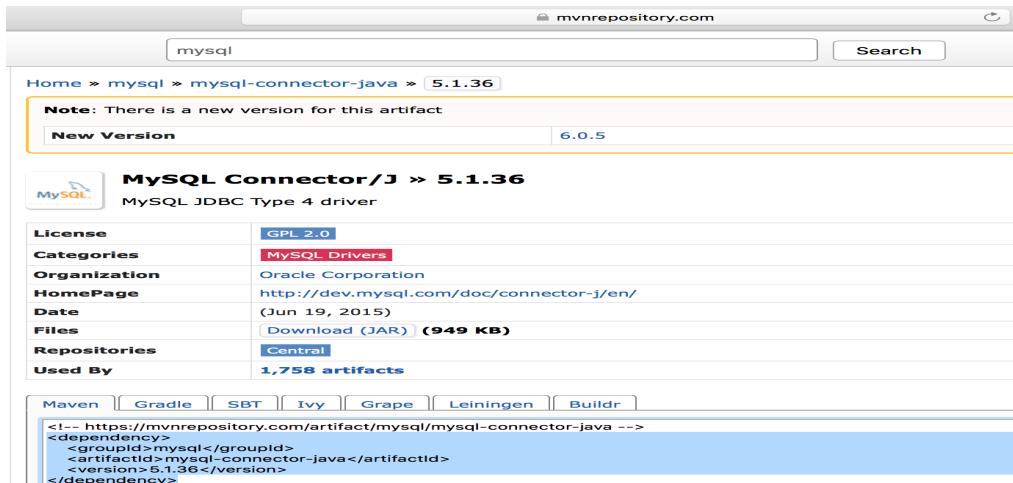
<properties>
    <spring.version>4.1.5.RELEASE</spring.version>
</properties>

<dependencies>
    <!-- dependencia Spring, la librería que necesita para poder usar el @component @transaction @PersistenceContext -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <!-- Acceso a datos -->
    <!-- dependencia de JPA, es mejor esta porque contiene dentro las de JPA -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
        <version>4.3.8.Final</version>
    </dependency>
    <!-- dependencia de mysql, usaremos esta version-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.36</version>
    </dependency>
    <!-- JSTL un conjunto de librerías de etiquetas simples y estándares que encapsulan la funcionalidad principal que se usa en páginas JSP, usaremos el core Comprende las funciones script básicas como loops, condicionales, y entrada/salida.-->
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>

```

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

Para encontrar las dependencias es ir a esta pagina.



mvnrepository.com

mysql

Home > mysql > mysql-connector-java > [5.1.36]

Note: There is a new version for this artifact

New Version 6.0.5

MySQL Connector/J >> 5.1.36

MySQL JDBC Type 4 driver

License	GPL 2.0
Categories	MySQL Drivers
Organization	Oracle Corporation
HomePage	http://dev.mysql.com/doc/connector-j/en/
Date	(Jun 19, 2015)
Files	Download (JAR) (949 KB)
Repositories	Central
Used By	1,758 artifacts

Maven Gradle SBT Ivy Grape Leiningen Buildr

```

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.36</version>
</dependency>

```

4.5 Configurar servlet y jsp

Donde se configura el servlet y el jsp, para habilitar Spring en su aplicación web, es en el web.xml , pero este xml no crea por defecto.

Para crearlo es click derecho en Deployment Descriptor → Generate Deployment Descriptor Stub.

Web.xml define un nombre para el servlet y especifica la clase compilada que ejecuta el servlet. O también en lugar de especificar una clase de servlet, puede especificar un JSP.

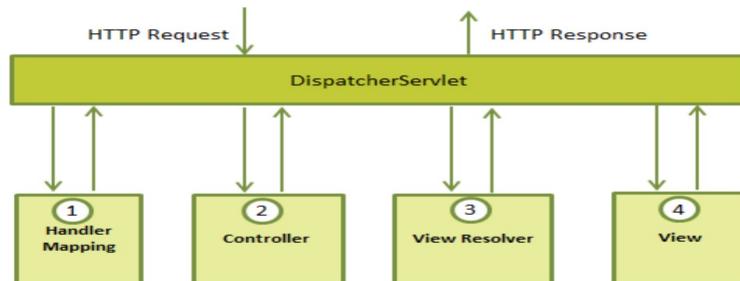
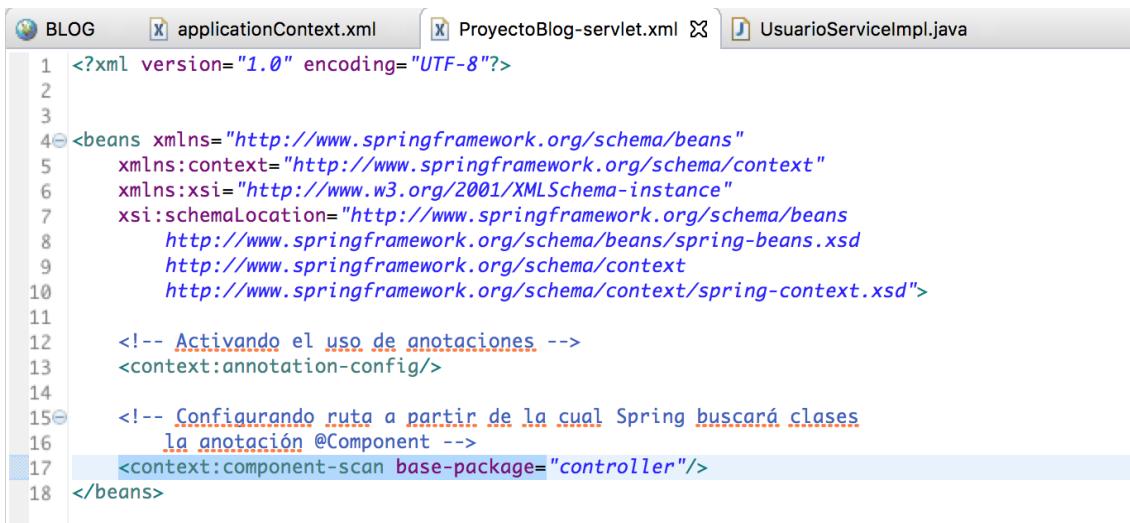
```

BLOG web.xml ProyectoBlog-servlet.xml applicationContext.xml

4  <!-- BEGIN: Contexto Spring Global -->
5    <listener> <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class> </listener>
6    <context-param> <param-name>contextConfigLocation</param-name>
7      <param-value>/WEB-INF/applicationContext.xml</param-value> </context-param>
8    <!-- END: Contexto Spring Global -->
9
10   <!-- BEGIN: Spring MVC -->
11   <servlet>
12     <servlet-name>ProyectoBlog</servlet-name>
13     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
14     <load-on-startup>1</load-on-startup>
15   </servlet>
16
17   <servlet-mapping>
18     <servlet-name>ProyectoBlog</servlet-name>
19     <url-pattern>/action/*</url-pattern> <!--Este es el nombre en localhost mas con el mapping -->
20   </servlet-mapping>
21   <!-- END: Spring MVC -->
22
23
24   <!-- BEGIN: Filtro de Codificacion -->
25   <filter>
26     <filter-name>encoding-filter</filter-name>
27     <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
28     <init-param>
29       <param-name>encoding</param-name>
30       <param-value>UTF-8</param-value>
31     </init-param>
32   </filter>
33   <filter-mapping> <filter-name>encoding-filter</filter-name> <url-pattern>/*</url-pattern> </filter-mapping>
34   <!-- END: Filtro de Codificacion -->
35
36   <filter> <!-- BEGIN: Open Session in View -->
37     <filter-name>OpenEntityManagerInViewFilter</filter-name>
38     <filter-class>org.springframework.orm.jpa.support.OpenEntityManagerInViewFilter</filter-class>
39   </filter>
40   <filter-mapping>
41     <filter-name>OpenEntityManagerInViewFilter</filter-name>
42     <url-pattern>*</url-pattern>
43
44   </filter-mapping> <!-- END: Open Session in View -->
45
46   <!-- parte por defecto y modificada del web.xml -->
47   <welcome-file-list> <welcome-file>action/usuario</welcome-file> </welcome-file-list>

```

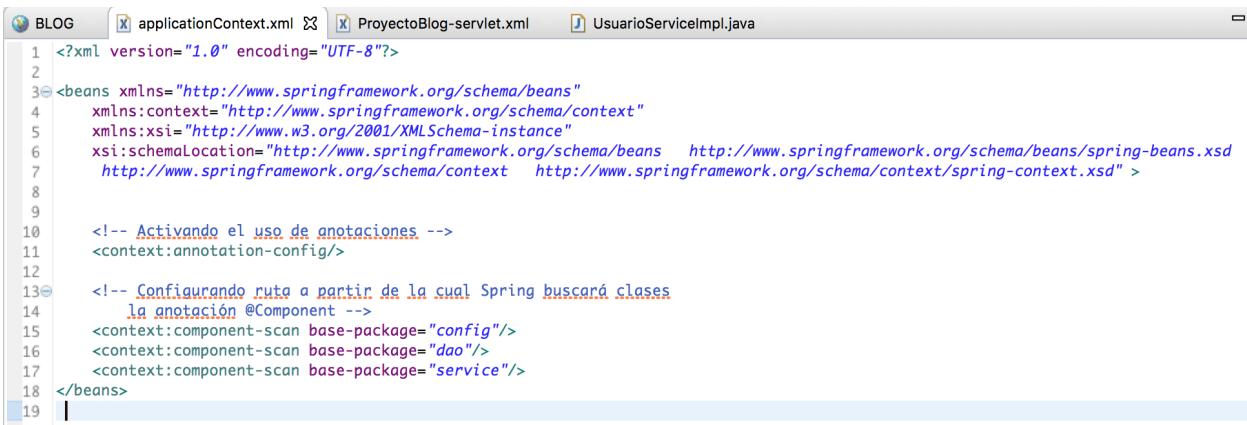
El framework MVC (model-view-controller) de Spring Web está diseñado alrededor de un DispatcherServlet que maneja todas las peticiones y respuestas HTTP.

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd">
  <!-- Activando el uso de anotaciones -->
  <context:annotation-config/>
  <!-- Configurando ruta a partir de la cual Spring buscará clases
       la anotación @Component -->
  <context:component-scan base-package="controller"/>
</beans>
  
```

applicationContext.xml es donde ContextLoaderListener va a buscar la declaración de beans para desplegarlos en el root ServletContext (padre) de la aplicación. Podemos configurar aquí, como crear y conectar algunos beans Spring.



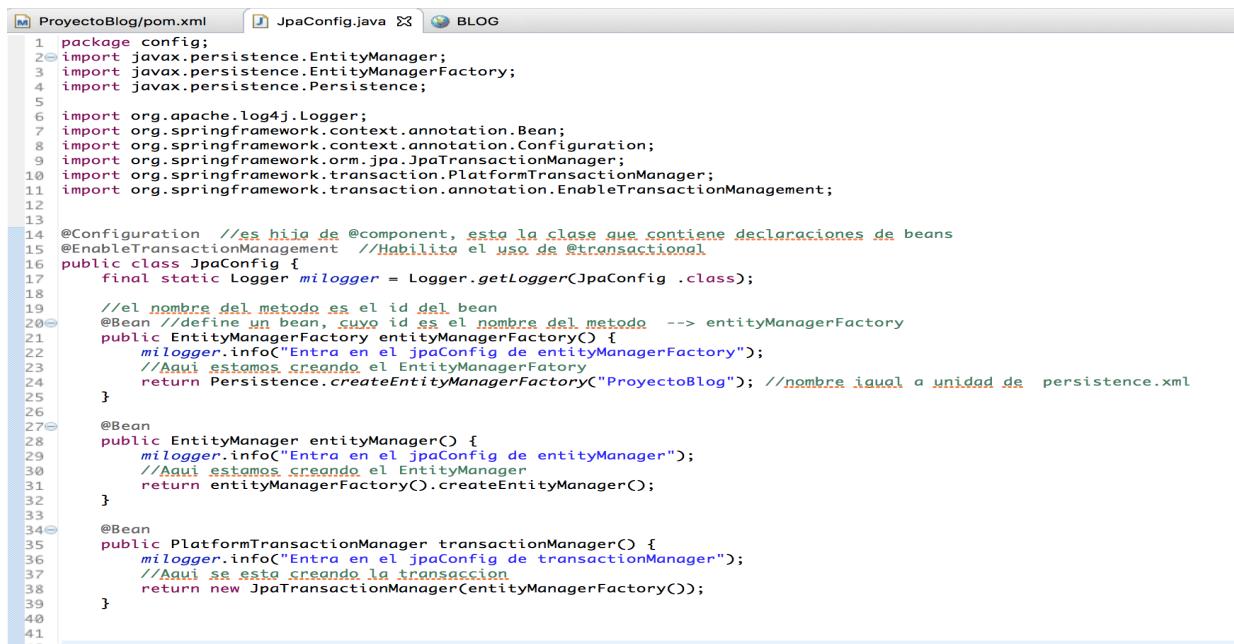
```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd">
  <!-- Activando el uso de anotaciones -->
  <context:annotation-config/>
  <!-- Configurando ruta a partir de la cual Spring buscará clases
       la anotación @Component -->
  <context:component-scan base-package="config"/>
  <context:component-scan base-package="dao"/>
  <context:component-scan base-package="service"/>
</beans>
  
```

4.6 Creación de clases

1) Clase JpaConfig

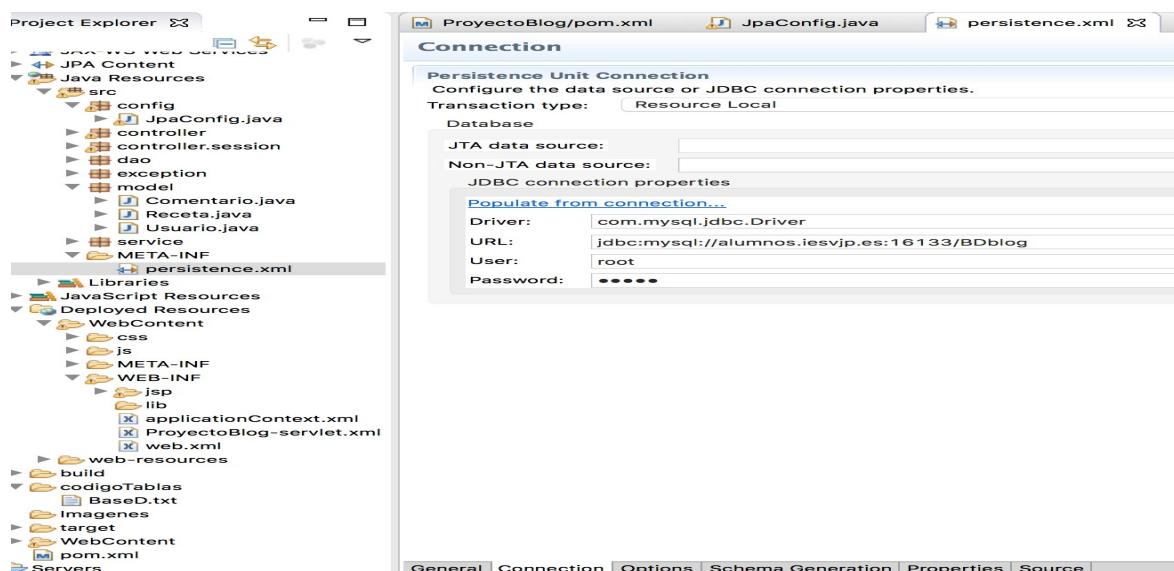
Un EntityManager será una instancia de EntityManagerFactory en JPA. Dicha factoría lo que hace es representar la configuración para acceder a la base de datos que utilice nuestra aplicación.



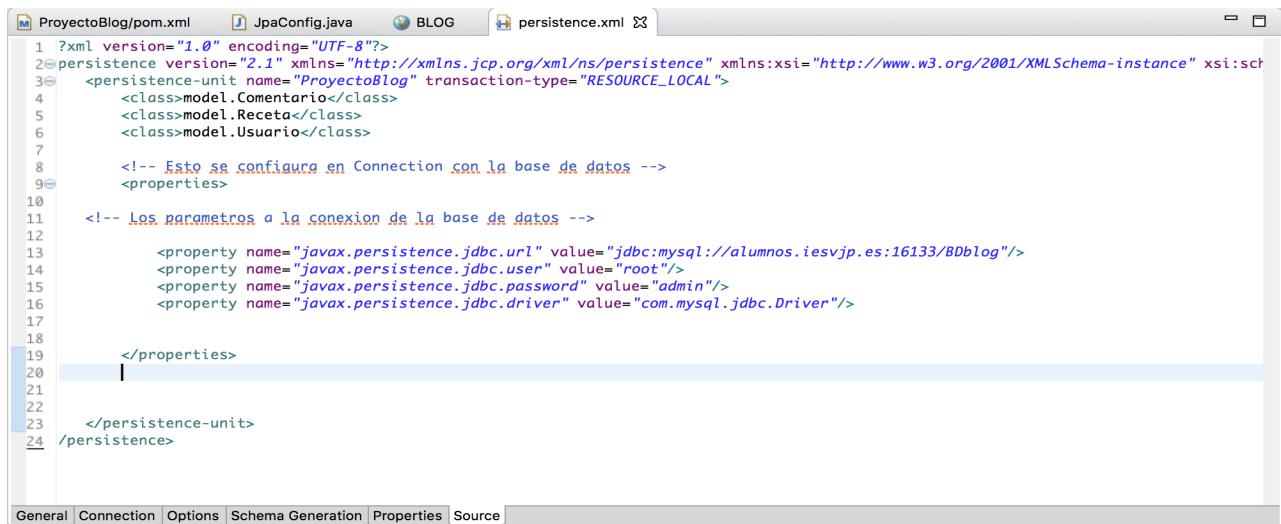
```

1 package config;
2 import javax.persistence.EntityManager;
3 import javax.persistence.EntityManagerFactory;
4 import javax.persistence.Persistence;
5
6 import org.apache.log4j.Logger;
7 import org.springframework.context.annotation.Bean;
8 import org.springframework.context.annotation.Configuration;
9 import org.springframework.orm.jpa.JpaTransactionManager;
10 import org.springframework.transaction.PlatformTransactionManager;
11 import org.springframework.transaction.annotation.EnableTransactionManagement;
12
13
14 @Configuration //es hijo de @component, esta la clase que contiene declaraciones de beans
15 @EnableTransactionManagement //Habilita el uso de @transactional
16 public class JpaConfig {
17     final static Logger milogger = Logger.getLogger(JpaConfig .class);
18
19     //el nombre del metodo es el id del bean
20     @Bean //define un bean, cuyo id es el nombre del metodo --> entityManagerFactory
21     public EntityManagerFactory entityManagerFactory() {
22         milogger.info("Entra en el jpaConfig de entityManagerFactory");
23         //Aqui estamos creando el EntityManagerFatory
24         return Persistence.createEntityManagerFactory("ProyectoBlog"); //nombre igual a unidad de persistence.xml
25     }
26
27     @Bean
28     public EntityManager entityManager() {
29         milogger.info("Entra en el jpaConfig de entityManager");
30         //Aqui estamos creando el EntityManager
31         return entityManagerFactory().createEntityManager();
32     }
33
34     @Bean
35     public PlatformTransactionManager transactionManager() {
36         milogger.info("Entra en el jpaConfig de transactionManager");
37         //Aqui se esta creando la transaccion
38         return new JpaTransactionManager(entityManagerFactory());
39     }
40
41 }
```

Donde podemos ver la unidad de persistencia es en el persistence.xml, que necesita el JPA, y el nombre que tiene es el que se usara. También hay que configurar los parámetros de conexión en la base de datos.



Y si vemos en el source del persistence.xml podemos ver que se ha creado así.



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://www.w3.org/2001/XMLSchema-instance">
3   <persistence-unit name="ProyectoBlog" transaction-type="RESOURCE_LOCAL">
4     <class>model.Comentario</class>
5     <class>model.Receta</class>
6     <class>model.Usuario</class>
7
8     <!-- Esto se configura en Connection con la base de datos -->
9     <properties>
10
11       <!-- Los parametros a la conexion de la base de datos -->
12
13         <property name="javax.persistence.jdbc.url" value="jdbc:mysql://alumnos.iesvp.es:16133/BDBlog"/>
14         <property name="javax.persistence.jdbc.user" value="root"/>
15         <property name="javax.persistence.jdbc.password" value="admin"/>
16         <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
17
18     </properties>
19
20
21
22   </persistence-unit>
23 </persistence>

```

General | Connection | Options | Schema Generation | Properties | Source

2) Creación GenericDao y GenericDaolmpl

GenericDao es una interfaz con genéricos, este tipo de clase nos permitirá definir una clase de un tipo concreto.

GenericDaolmpl esta clase implementa la interface GenericDao.



Java Resources

- src
 - config
 - controller
 - AdminController.java
 - UsuarioController.java
 - dao
 - ComentarioDao.java
 - ComentarioDaolmpl.java
 - GenericDao.java
 - GenericDaolmpl.java
 - RecetaDao.java
 - RecetaDaolmpl.java
 - UsuarioDao.java
 - UsuarioDaolmpl.java
 - exception
 - BlogException.java
 - model
 - service
 - UsuarioService.java
 - UsuarioServiceImpl.java
 - META-INF
 - Libraries

BLOG

```

1 package dao;
2
3 import java.util.List;
4
5 //Es una interfaz con genericos
6 //para que luego se pueda implementar
7
8 public interface GenericDao<T, K> {
9   public void agregar(T obj);
10  public void modificar(T obj);
11  public void eliminar(K id);
12  public T obtener(K id);
13  public List<T> obtenerTodos();
14
15 }
16

```

```

public class GenericDaoImpl<T, K> implements GenericDao<T, K> {
    @PersistenceContext // Spring manejará las conexiones
    protected EntityManager entityManager;
    private Class<T> clase;

    @SuppressWarnings("unchecked")
    // see: http://stackoverflow.com/questions/182636/how-to-determine

    public GenericDaoImpl() {
        this.clase = (Class<T>) ((ParameterizedType)
            getClass()
                .getGenericSuperclass()
                .getActualTypeArguments()[0]);
    }

    @Override
    public void agregar(T obj) {
        entityManager.persist(obj);
    }

    @Override
    public void modificar(T obj) {
        entityManager.merge(obj);
    }

    @Override
    public void eliminar(K id) {
        T o = obtener(id);
        entityManager.remove(o);
    }

    @Override
    public T obtener(K id) {
        return entityManager.find(clase, id);
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<T> obtenerTodos() {
        return entityManager
            .createNamedQuery(clase.getSimpleName() + ".findAll")
            .getResultList();
    }
}

```

@Autowired. Es una anotación de Spring. Sirve para indicarle a Spring que cuando va a crear la instancia de una clase que debe pasarle.

@Component indica al contenedor de Spring que podemos usar esta clase a través de Spring Inyección de Dependencia. **@Service** y **@Controller** extienden de esta componente.

@PersistenceContext que indica la inyección de dependencia a una instancia adecuada de EntityManager en la configuración de ProyectoBlog-Servlet.xml

@Controller donde su solicitud de mapeo desde la página de presentación hecha, es decir, la capa de presentación no va a ningún otro archivo que va directamente a la clase **@Controller** y comprobar la ruta solicitada en **@RequestMapping** anotación que escribió antes del método, llama si es necesario.

@Service indica que las clases marcadas con esta anotación está en una capa de servicios o de lógica de negocios. Los datos relacionados con los cálculos.

@Transactional es que hay dos conceptos diferentes a considerar, el contexto de persistencia, la transacción a la de base de datos. La propia anotación transaccional define el ámbito de una única transacción de base de datos.

3) Clase UsuarioController

```

 15
 16  @Controller
 17  @RequestMapping("/usuario/*")
 18  public class UsuarioController {
 19      final static Logger milogger = Logger.getLogger(UsuarioController .class);
 20  @Autowired // esto es necesario para crear el constructor
 21  private UsuarioService usuarioService;
 22
 23  @RequestMapping("autenticar") //este es el nombre que tiene y se llama desde el formulario
 24  public String autenticar(@RequestParam String nomusuario, @RequestParam String clave, HttpSession session, Model model) {
 25      milogger.info("entro en autenticar");
 26      // la consulta de todas las recetas de primero
 27      List<Receta> listaRecetasTodas = new ArrayList<Receta>();
 28      listaRecetasTodas =usuarioService.obtenerTodosReceta();
 29      model.addAttribute("recetas", listaRecetasTodas);
 30
 31      // todas las recetas de Postre
 32      List<Receta> listaRecetasPostres = new ArrayList<Receta>();
 33      listaRecetasPostres =usuarioService.obtenerTodosRecetaPC();
 34      model.addAttribute("recetasP", listaRecetasPostres);
 35
 36      //Ingresar usuario
 37      List<String> errores = new ArrayList<String>();
 38      Usuario u = null;
 39      try {
 40          milogger.info("al llamar al usuario service autenticar");
 41          u = usuarioService.autenticar(nomusuario, clave);
 42
 43      } catch (Exception e) {
 44
 45          milogger.error("Error autenticando el usuario ",e);
 46          errores.add("Error autenticando el usuario");
 47      }
 48
 49      if (u == null)
 50          errores.add("Usuario o contraseña invalido ");
 51      else { session.setAttribute("usuario", u); }
 52
 53      if (errores.size() > 0) {
 54          model.addAttribute("errores", errores);
 55          return "/WEB-INF/jsp/usuario/index.jsp"; //hay que crear estas carpetas como indica
 56      } else
 57          if(nomusuario.equals("anonimo")){
 58              return "/WEB-INF/jsp/usuario/index.jsp";
 59          } else{
 60              return "redirect:/action/admin/";
 61      }
 62  }

```

```

 61  @RequestMapping("")
 62  public String index(Model model) { //esto es un inicio, a mostrar
 63      milogger.info("entro en el inicio de usuario");
 64
 65      //la consulta de todas las recetas de primero
 66      List<Receta> listaRecetasTodas = new ArrayList<Receta>();
 67      milogger.info("al llamar al usuario service Obtener todos Receta de primero ");
 68      listaRecetasTodas =usuarioService.obtenerTodosReceta();
 69      model.addAttribute("recetas", listaRecetasTodas);
 70
 71      // todas las recetas de Postre
 72      List<Receta> listaRecetasPostres = new ArrayList<Receta>();
 73      milogger.info("al llamar al usuario service obtener Todos Recetas de postre");
 74      listaRecetasPostres =usuarioService.obtenerTodosRecetaPC();
 75      model.addAttribute("recetasP", listaRecetasPostres);
 76
 77
 78      return "/WEB-INF/jsp/usuario/index.jsp";
 79
 80  }
 81
 82  @RequestMapping("cerrar")
 83  public String sesionCerrar(Model model,HttpSession session) {
 84      milogger.info("entro en la sesion de cerrar para salir como administrador");
 85      //la consulta de todas las recetas de primero
 86      List<Receta> listaRecetasTodas = new ArrayList<Receta>();
 87      milogger.info("al llamar al usuario service obtener todos las recetas de primero");
 88      listaRecetasTodas =usuarioService.obtenerTodosReceta();
 89      model.addAttribute("recetas", listaRecetasTodas);
 90
 91
 92      // todas las recetas de Postre
 93      List<Receta> listaRecetasPostres = new ArrayList<Receta>();
 94      milogger.info("al llamar al usuario service obtener todas la receta de Postre");
 95      listaRecetasPostres =usuarioService.obtenerTodosRecetaPC();
 96      model.addAttribute("recetasP", listaRecetasPostres);
 97
 98      session.invalidate();
 99
100     return "/WEB-INF/jsp/usuario/index.jsp";
101
102  }
103
104
105

```

```
AdminController.java  UsuarioController.java  UsuarioServiceImpl.java  UsuarioDaoImpl.java

@RequestMapping("iriniciarsesion")
public String iriniciarsesion(Model model) {
    milogger.info("entro para enviar al index para autenticar usuario");
    //la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas =usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service obtener todas la receta de Postre");
    listaRecetasPostres =usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    return "/WEB-INF/jsp/InicioSesion/index.jsp";
}

@RequestMapping("irRegistrate")
public String irRegistrate(Model model) {
    milogger.info("Me envia al index para registrarte");
    //la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas =usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service obtener todas la receta de Postre");
    listaRecetasPostres =usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    return "/WEB-INF/jsp/InicioRegistro/index.jsp";
}
```

```
AdminController.java UsuarioController.java UsuarioServiceImpl.java UsuarioDaoImpl.java GenericDaoImpl.java

@RequestMapping("agregar") //este es el nombre que tiene y se llama desde el formulario de dar de alta
public String registrar(@RequestParam String nomusuario, @RequestParam String clave, HttpSession session, Model model) {
    logger.info("Entro en agregar para registrar un nuevo usuario ");

    //la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    logger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas = usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    logger.info("al llamar al usuario service de todas recetas de postre");
    listaRecetasPostres = usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    List<String> errores = new ArrayList<String>();
    Usuario u = new Usuario();
    if(usuarioService.usuarioExiste(nomusuario)==false){
        u.setNombreUsuario(nomusuario);
        u.setClave(clave);
        logger.info("al llamar al usuario service de agregar usuario");
        usuarioService.agregarUsuario(u);
        session.setAttribute("usuario", u);
    }
    else{
        logger.error("Usuario ya existe en el sistema");
        errores.add("Usuario ya existe en el sistema");
    }

    if (errores.size() > 0) {

        model.addAttribute("errores", errores);
        return "/WEB-INF/jsp/usuario/index.jsp"; //hay que crear estas carpetas como indica
    } else
        return "redirect:/action/admin/";
}
```

```

DG AdminController.java UsuarioController.java UsuarioServiceImpl.java UsuarioDaoImpl.java GenericDaoImpl.java
  @RequestMapping("mostrar")
  public String iriniciarsesion(@RequestParam Integer id, Model model, HttpSession session) {
    List<String> errores = new ArrayList<String>();
    milogger.info("Entra en mostrar todos los datos de la receta ");

    //La consulta para devolver solo una receta de la receta
    Receta listaRecetas = new Receta();
    milogger.info("al llamar al usuario service para obtener por id la receta que le corresponde");
    listaRecetas = usuarioService.obtenerPorId(id);
    model.addAttribute("recetasID", listaRecetas);

    //la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas = usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de postre");
    listaRecetasPostres = usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    //la consulta de todos los comentarios de la receta
    int idReceta=listaRecetas.getId();
    List<Comentario> listaComentario=new ArrayList<Comentario>();

    try{
      milogger.info("al llamar al usuario service de obtener por nombre comentario su lista de comentario");
      listaComentario=usuarioService.obtenerPorNombreComentario(idReceta);
      model.addAttribute("comentarioT", listaComentario);
    } catch(Exception e){
      errores.add("Error, no hay comentario");
      milogger.error("Error, no hay comentario", e);
    }

    return "/WEB-INF/jsp/adminRecetas/index.jsp";
}

```

```

DG AdminController.java UsuarioController.java UsuarioServiceImpl.java UsuarioDaoImpl.java GenericDaoImpl.java
  @RequestMapping("agregarComentario") //tener cuidado para refreshar esta pagina
  public String agregarComentario(@RequestParam int personaid,@RequestParam int recetaid,@RequestParam String comentarios, Model model) {
    milogger.info("Es para agregar un comentario nuevo");

    List<String> errores = new ArrayList<String>();

    //la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas = usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de postre");
    listaRecetasPostres = usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    //Para agregar comentario
    Comentario c=new Comentario();
    c.setDescripcoment(comentarios);

    Receta r=new Receta();
    milogger.info("al llamar al usuario service para obtener por id de la recetaid ");
    r=usuarioService.obtenerPorId(recetaid);
    c.setReceta(r);

    Usuario u = new Usuario();
    milogger.info("al llamar al usuario service para obtener por el usuario su id");
    u=usuarioService.obtenerPorIdusuario(personaid);
    c.setUsuario(u);
    milogger.info("al llamar al usuario service para agrega comentario");
    usuarioService.agregarComentario(c);

    //la consulta para devolver solo una receta de la receta
    Receta listaRecetas = new Receta();
    milogger.info("al llamar al usuario service para obtener por id la receta que le corresponde");
    listaRecetas = usuarioService.obtenerPorId(recetaid);
    model.addAttribute("recetasID", listaRecetas);

    //la consulta de todos los comentarios de la receta
    int idReceta=listaRecetas.getId();
    List<Comentario> listaComentario=new ArrayList<Comentario>();

```

4) Clase AdminController

BLOG AdminController.java UsuarioController.java UsuarioServiceImpl.java UsuarioDaoImpl.java Generic

```

27 @Controller
28 @RequestMapping("/admin/*")
29 public class AdminController {
30     final static Logger milogger = Logger.getLogger(AdminController .class);
31     @Autowired
32     private UsuarioService usuarioService;
33
34     //Cuando uso el cuestionario que hay en InicioSesion viene aqui
35     @RequestMapping("")
36     public String admin(Model model, HttpSession session) {
37         milogger.info("Entro en la parte del administrador, que solo entrar el usuario");
38
39         // la consulta de todas las recetas de primero
40         List<Receta> listaRecetasTodas = new ArrayList<Receta>();
41         milogger.info("al llamar al usuario service de todas recetas de primero");
42         listaRecetasTodas =usuarioService.obtenerTodosReceta();
43         model.addAttribute("recetas", listaRecetasTodas);
44
45         // todas las recetas de Postre
46         List<Receta> listaRecetasPostres = new ArrayList<Receta>();
47         milogger.info("al llamar al usuario service de todas recetas de postre");
48         listaRecetasPostres =usuarioService.obtenerTodosRecetaP();
49         model.addAttribute("recetasP", listaRecetasPostres);
50
51         // Todas las recetas que tiene el usuario
52         Usuario u = new Usuario();
53
54         u = (Usuario) session.getAttribute("usuario");
55
56         try {
57             Integer id = u.getIdusu();
58
59             List<Receta> listaRecetas = new ArrayList<Receta>();
60             milogger.info("al llamar al usuario service para obtener la recetas que tiene el usuario");
61             listaRecetas =usuarioService.obtenerPorNombreReceta(id);
62             model.addAttribute("recetasU", listaRecetas);
63
64             return "/WEB-INF/jsp/admin/index.jsp";    //es donde se redirige para que se muestre
65         } catch (NullPointerException e) {
66             return "redirect:/action/admin/";
67         }
68     }
69
70
71
  
```

DG AdminController.java UsuarioController.java UsuarioServiceImpl.java UsuarioDaoImpl.java GenericDaoImpl.java index.jsp

```

@RequestMapping("ingresarRecetas")
public String ingresarRecetas (@RequestParam Integer personaid,Model model, HttpSession session ){
    // la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas =usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de postre");
    listaRecetasPostres =usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    return "/WEB-INF/jsp/adminIngresar/index.jsp";
}

public void cargarFoto(Receta RI, String ruta, int n, int numeroAleatorio){
    String base64Image = ruta.split(",")[1];

    // Convierte la image code to bytes.
    byte[] imageBytes = javax.xml.bind.DatatypeConverter.parseBase64Binary(base64Image);

    BufferedImage bufferedImage = null;
    try {
        bufferedImage = ImageIO.read(new ByteArrayInputStream(imageBytes)); } catch (IOException e2) {e2.printStackTrace(); }

    // /opt/tomcat8/webapps/Imagenes/      //esta ruta se usa en cuando se va usar el servidor
    // /Users/laurairinamendoza/Desktop/    //esta ruta se usa en el ordenador

    String rutaFinal="/Users/laurairinamendoza/Desktop/"+RI.getNombrerect()+n+numeroAleatorio+".png";

    // "/Users/laurairinamendoza/Desktop/ImagePrueba.png" //en esta ruta esta la imagen guardada en el ordenador
    // "http://alumnos.iesvjp.es:16181/Imagenes/ImagePrueba.png" //esta ruta es en la que esta imagen el servidor

    File imageFile = new File(rutaFinal);
    try {
        ImageIO.write(bufferedImage, "png", imageFile);
    } catch (IOException e1) { e1.printStackTrace(); }
    RI.setImagen(imageFile.getName());
}
  
```



```

@RequestMapping("agregarReceta")
public String agregarReceta (@RequestParam Integer personaid,@RequestParam String nomReceta,@RequestParam String nomResumen,@RequestParam String nomCategoria,
@RequestParam String nomIngredientes,@RequestParam String nomDescripcion,@RequestParam String ruta,Model model, HttpSession session){
    Usuario u = new Usuario();
    Receta RI= new Receta();

    if(((nomResumen.equals("")==false) && (nomReceta.equals("")==false) ) && (nomIngredientes.equals("")==false) && (nomDescripcion.equals("")==false) ){
        try {
            Date fechaCorrecta = new Date();
            SimpleDateFormat formatoDelTexto = new SimpleDateFormat("yyyy-MM-dd");

            String strFecha=formatoDelTexto.format(fechaCorrecta);
            fechaCorrecta = formatoDelTexto.parse(strFecha);
            milogger.info("llama usuario service para obtener un usuario en concreto");
            u =usuarioService.obtenerPorIdUsuario(personaid);
            RI.setNombreRecet(nomReceta);
            RI.setResumen(nomResumen);
            RI.setCategoria(nomCategoria);
            RI.setIngredientes(nomIngredientes);
            RI.setDescripcion(nomDescripcion);
            RI.setFechaPubli(fechaCorrecta);

            int numeroAleatorio = (int) (Math.random()*4000+1);
            cargarFoto(RI ,ruta,personaid,numeroAleatorio);

            RI.setUsuario(u);
            milogger.info("llama usuario service para agregar una nueva receta");
            usuarioService.agregarReceta(RI);

        } catch (ParseException e) { milogger.equals(e); }
    }

    List<Receta> listaRecetas = new ArrayList<Receta>();
    milogger.info(" llamar al usuario service para obtener la recetas que tiene el usuario");
    listaRecetas =usuarioService.obtenerPorNombreReceta(personaid);
    model.addAttribute("recetasU", listaRecetas);

    // la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas =usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);
}

```

```

// todas las recetas de Postre
List<Receta> listaRecetasPostres = new ArrayList<Receta>();
milogger.info("al llamar al usuario service de todas recetas de postre");
listaRecetasPostres =usuarioService.obtenerTodosRecetaP();
model.addAttribute("recetasP", listaRecetasPostres);

return "/WEB-INF/jsp/admin/index.jsp";

}

@RequestMapping("eliminarReceta")
public String eliminarReceta(@RequestParam Integer idreceta ,Model model, HttpSession session){

    Usuario u = new Usuario();
    u = (Usuario) session.getAttribute("usuario");

    Integer id = u.getIdusu();
    milogger.info("llamar al usuario service para eliminar una receta en concreto");
    usuarioService.EliminarReceta(idreceta);

    //recetas del usuario
    List<Receta> listaRecetas = new ArrayList<Receta>();
    milogger.info(" llamar al usuario service para obtener la recetas que tiene el usuario");
    listaRecetas =usuarioService.obtenerPorNombreReceta(id);
    model.addAttribute("recetasU", listaRecetas);

    // la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas =usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de postre");
    listaRecetasPostres =usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    return "/WEB-INF/jsp/admin/index.jsp";
}

```

```

 ① @RequestMapping("agregarComentario") //tener cuidado para refrescar esta pagina
public String agregarComentario(@RequestParam int personaid,@RequestParam int recetaid,@RequestParam String comentarios, Model model, HttpSession session){

    List<String> errores = new ArrayList<String>();
    // la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas = usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de postre");
    listaRecetasPostres = usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    //Para agregar comentario
    Comentario c=new Comentario();
    c.setDescripcoment(comentarios);

    Receta r=new Receta();
    milogger.info(" llamar al usuario service para obtener una receta en concreto");
    r=usuarioService.obtenerPorId(recetaid);
    c.setReceta(r);

    Usuario u = new Usuario();
    milogger.info(" llamar al usuario service para obtener un usuario por su id");
    u =usuarioService.obtenerPorIdusuario(personaid);
    c.setUsuario(u);

    milogger.info(" llamar al usuario service para agregar comentario");
    usuarioService.agregarComentario(c);

    //La consulta para devolver solo una receta de la receta
    Receta listaRecetas = new Receta();
    milogger.info("llamar al usuario service para obtener por id la receta que le corresponde");
    listaRecetas = usuarioService.obtenerPorId(recetaid);
    model.addAttribute("recetasID", listaRecetas);
}
  
```

```

 ① @RequestMapping("agregarComentario") //tener cuidado para refrescar esta pagina
public String agregarComentario(@RequestParam int personaid,@RequestParam int recetaid,@RequestParam String comentarios, Model model, HttpSession session){

    List<String> errores = new ArrayList<String>();
    // la consulta de todas las recetas de primero
    List<Receta> listaRecetasTodas = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de primero");
    listaRecetasTodas = usuarioService.obtenerTodosReceta();
    model.addAttribute("recetas", listaRecetasTodas);

    // todas las recetas de Postre
    List<Receta> listaRecetasPostres = new ArrayList<Receta>();
    milogger.info("al llamar al usuario service de todas recetas de postre");
    listaRecetasPostres = usuarioService.obtenerTodosRecetaP();
    model.addAttribute("recetasP", listaRecetasPostres);

    //Para agregar comentario
    Comentario c=new Comentario();
    c.setDescripcoment(comentarios);

    Receta r=new Receta();
    milogger.info(" llamar al usuario service para obtener una receta en concreto");
    r=usuarioService.obtenerPorId(recetaid);
    c.setReceta(r);

    Usuario u = new Usuario();
    milogger.info(" llamar al usuario service para obtener un usuario por su id");
    u =usuarioService.obtenerPorIdusuario(personaid);
    c.setUsuario(u);

    milogger.info(" llamar al usuario service para agregar comentario");
    usuarioService.agregarComentario(c);

    //La consulta para devolver solo una receta de la receta
    Receta listaRecetas = new Receta();
    milogger.info("llamar al usuario service para obtener por id la receta que le corresponde");
    listaRecetas = usuarioService.obtenerPorId(recetaid);
    model.addAttribute("recetasID", listaRecetas);
}
  
```

G AdminController.java ✘ UsuarioController.java UsuarioServiceImpl.java UsuarioDaoImpl.java GenericDaoImpl.java

```
//la consulta de todos los comentarios de la receta

int idReceta=listaRecetas.getIdrect();
List<Comentario> listaComentario=new ArrayList<Comentario>();
try{
    milogger.info("al llamar al usuario service de obtener por nombre comentario su lista de comentario");
    listaComentario=usuarioService.obtenerPorNombreComentario(idReceta);
    model.addAttribute("comentarioT",listaComentario);

}
catch(Exception e){
    milogger.error("Error, no hay comentario",e);
    errores.add("Error, no hay comentario");
}
return "/WEB-INF/jsp/adminRecetas/index.jsp";
}

@RequestMapping("eliminarComent")
public String eliminarComent(@RequestParam Integer id,Model model, HttpSession session){
List<String> errores = new ArrayList<String>();

// la consulta de todas las recetas de primero
List<Receta> listaRecetasTodas = new ArrayList<Receta>();
milogger.info("al llamar al usuario service de todas recetas de primero");
listaRecetasTodas =usuarioService.obtenerTodosReceta();
model.addAttribute("recetas", listaRecetasTodas);

// todas las recetas de Postre
List<Receta> listaRecetasPostres = new ArrayList<Receta>();
milogger.info("al llamar al usuario service de todas recetas de postre");
listaRecetasPostres =usuarioService.obtenerTodosRecetaP();
model.addAttribute("recetasP", listaRecetasPostres);
```

OG AdminController.java ✘ UsuarioController.java UsuarioServiceImpl.java UsuarioDaoImpl.java GenericDaoImpl.java

```
//La consulta para devolver solo una receta de la receta
Receta listaRecetas = new Receta();
Comentario c=new Comentario();
try{
    milogger.info("llamar al usuario service para obtener la receta del comentario");
    c=usuarioService.recetaComentario(id);
    int recetaid=c.getReceta().getIdrect();

    milogger.info("llamar al usuario service para obtener por id la receta que le corresponde");
    listaRecetas = usuarioService.obtenerPorId(recetaid);
    model.addAttribute("recetasID", listaRecetas);

    //Para eliminar comentario
    milogger.info("llama usuario service para eliminar un comentario");
    usuarioService.eliminarComentario(id);

}
catch(Exception e){
    milogger.error("Error, no hay comentario",e);
    errores.add("Error, no hay comentario");

}
//la consulta de todos los comentarios de la receta

int idReceta=listaRecetas.getIdrect();
List<Comentario> listaComentario=new ArrayList<Comentario>();
try{
    milogger.info("al llamar al usuario service de obtener por nombre comentario su lista de comentario");
    listaComentario=usuarioService.obtenerPorNombreComentario(idReceta);

    model.addAttribute("comentarioT",listaComentario);

}
catch(Exception e){
    //e.printStackTrace();
    errores.add("Error, no hay comentario");
    milogger.error("Error, no hay comentario", e);
}
return "/WEB-INF/jsp/adminRecetas/index.jsp";
```

5) Clase UsuarioService

En esta clase sera donde se gestione las clases dao y donde se llamaran para los respectivos casos que se necesiten en el proyecto.

En todas las clases tienen su interface, en la cual es donde ponemos todos los nombres de los métodos que necesitaremos que luego implementaremos.

```
LOG Usuarioservice.java ✘ UsuarioDaoImpl.java GenericDaoImpl.java
package service;

import java.util.List;

public interface UsuarioService {

    public void agregarUsuario(Usuario u);
    public Usuario autenticar(String nomusuario, String clave);
    public Usuario obtenerPorIdusuario(int id);
    public boolean usuarioExiste(String nomusuario);

    public Receta obtenerPorId(int id);
    public List<Receta> obtenerTodosRecetaP();
    public void agregarReceta(Receta RI);
    public void EliminarReceta(Integer idreceta);
    public List<Receta> obtenerTodosReceta();
    public List<Receta> obtenerPorNombreReceta(int id);

    void agregarComentario(Comentario c);
    public List<Comentario> obtenerPorNombreComentario(int idReceta);
    public void eliminarComentario(int idcoment);
    public Comentario recetaComentario(int id);

}
```

En clase implements se usa el componente `@Transactional` que sera la transacción a la base de datos. Con esto aseguramos que en esta clase es donde se llamará a los métodos que usará el `entityManager`.

Para los métodos agregar, eliminar, obtener id, serán los métodos de la clase generic que se creó. Y para los demás métodos serán de la clase dao, porque se necesita hacer una query para la consulta que se necesite.

```

  G UsuarioServiceImpl.java UsuarioService.java UsuarioDaoImpl.java GenericDaoImpl.java index.jsp index.html
  @Component //esto gestiona las instancias en general, y spring lo crea
  @Transactional //spring
  @Service //nuevo
  public class UsuarioServiceImpl implements UsuarioService {
    @Autowired //spring va inventar
    private UsuarioDao usuarioDao; //se usa la interfaz

    @Autowired
    private RecetaDao recetaDao; //se usa la interfaz

    @Autowired
    private ComentarioDao comentarioDao; //se usa la interfaz

    /**--UsuarioDAO--*/
    @Override
    public void agregarUsuario(Usuario u) {
        //Con esto se obtiene el usuario
        Usuario uBD = usuarioDao.obtenerPorNombreUsuario(u.getNombreusuario());

        if (uBD != null) //si este usuario ya existe en la base de datos
            throw new BlogException("Ya existe un usuario con ese nombre de usuario");

        usuarioDao.agregar(u); //sino se agrega, es el metodo generico
    }
    @Override
    public Usuario autenticar(String nomusuario, String clave) {
        Usuario u = usuarioDao.obtenerPorNombreUsuario(nomusuario); //este metodo esta primero en la interfaz
        if (u == null || !u.getClave().equals(clave))
            return null;
        else
            return u;
    }
    @Override
    public boolean usuarioExiste(String nomusuario) {
        Usuario u = usuarioDao.obtenerPorNombreUsuario(nomusuario); //este metodo esta primero en la interfaz
        if (u == null)
            return false; //el usuario no existe
        else
            return true; //el usuario existe
    }
}

  G UsuarioServiceImpl.java UsuarioService.java UsuarioDaoImpl.java GenericDaoImpl.java index.jsp index.html
  }

  public Usuario obtenerPorIdusuario(int id) { return usuarioDao.obtener(id); }

  /**--RecetasDAO--*/
  @Override
  public void agregarReceta(Receta RI) { recetaDao.agregar(RI); }
  @Override
  public void EliminarReceta(Integer idreceta) {
      comentarioDao.EliminarListaComentario(idreceta);
      recetaDao.eliminar(idreceta);
  }

  public Receta obtenerPorId(int id) { return recetaDao.obtener(id); }
  public List<Receta> obtenerTodosRecetaPC(){ return recetaDao.obtenerTodosRecetaPC(); }
  public List<Receta> obtenerTodosReceta() { return recetaDao.obtenerTodosReceta(); }

  public List<Receta> obtenerPorNombreReceta(int id){
      return recetaDao.obtenerPorNombreReceta(id);
  }

  /**--ComentarioDAO--*/
  public List<Comentario> obtenerPorNombreComentario(int idReceta) {
      return comentarioDao.obtenerPorNombreComentario(idReceta);
  }

  @Override
  public void agregarComentario(Comentario c) {
      comentarioDao.agregar(c);
  }

  @Override
  public void eliminarComentario(int idcoment) {
      comentarioDao.EliminarComentario(idcoment);
  }

  public Comentario recetaComentario(int id){
      return comentarioDao.idRecetaComentario(id);
  }
}

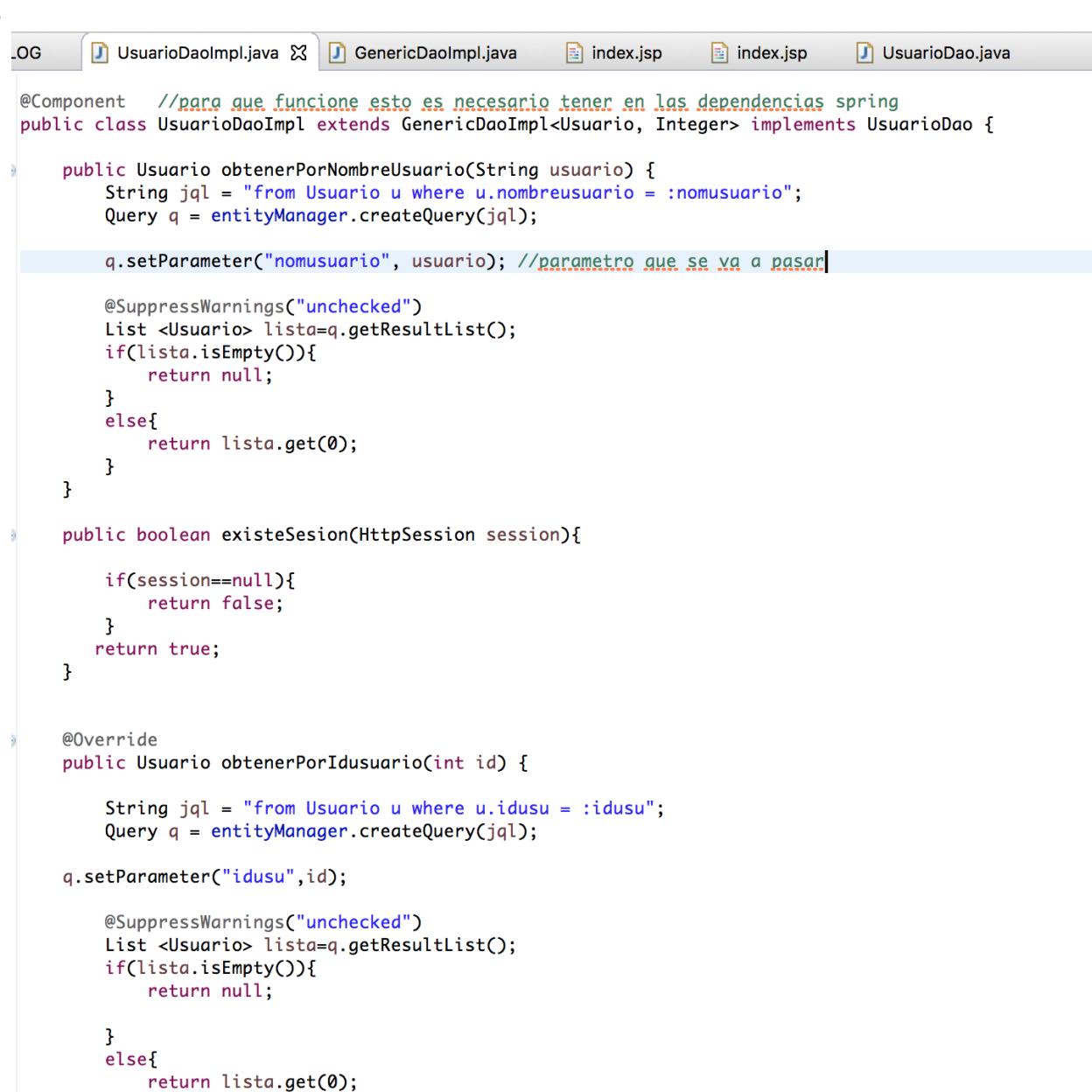
```

6) Clase UsuarioDao

En esta clase crearemos los métodos de consulta query con respecto a la clase usuario.

```
package dao;
import model.Usuario;
```

```
public interface UsuarioDao extends GenericDao<Usuario, Integer> {
    public Usuario obtenerPorNombreUsuario(String usuario);
    public Usuario obtenerPorIdusuario(int id);
}
```



```

@Component //para que funcione esto es necesario tener en las dependencias spring
public class UsuarioDaoImpl extends GenericDaoImpl<Usuario, Integer> implements UsuarioDao {

    public Usuario obtenerPorNombreUsuario(String usuario) {
        String jql = "from Usuario u where u.nombreusuario = :nomusuario";
        Query q = entityManager.createQuery(jql);

        q.setParameter("nomusuario", usuario); //parametro que se va a pasar

        @SuppressWarnings("unchecked")
        List <Usuario> lista=q.getResultList();
        if(lista.isEmpty()){
            return null;
        }
        else{
            return lista.get(0);
        }
    }

    public boolean existeSesion(HttpSession session){

        if(session==null){
            return false;
        }
        return true;
    }

    @Override
    public Usuario obtenerPorIdusuario(int id) {

        String jql = "from Usuario u where u.idusu = :idusu";
        Query q = entityManager.createQuery(jql);

        q.setParameter("idusu",id);

        @SuppressWarnings("unchecked")
        List <Usuario> lista=q.getResultList();
        if(lista.isEmpty()){
            return null;

        }
        else{
            return lista.get(0);
        }
    }
}

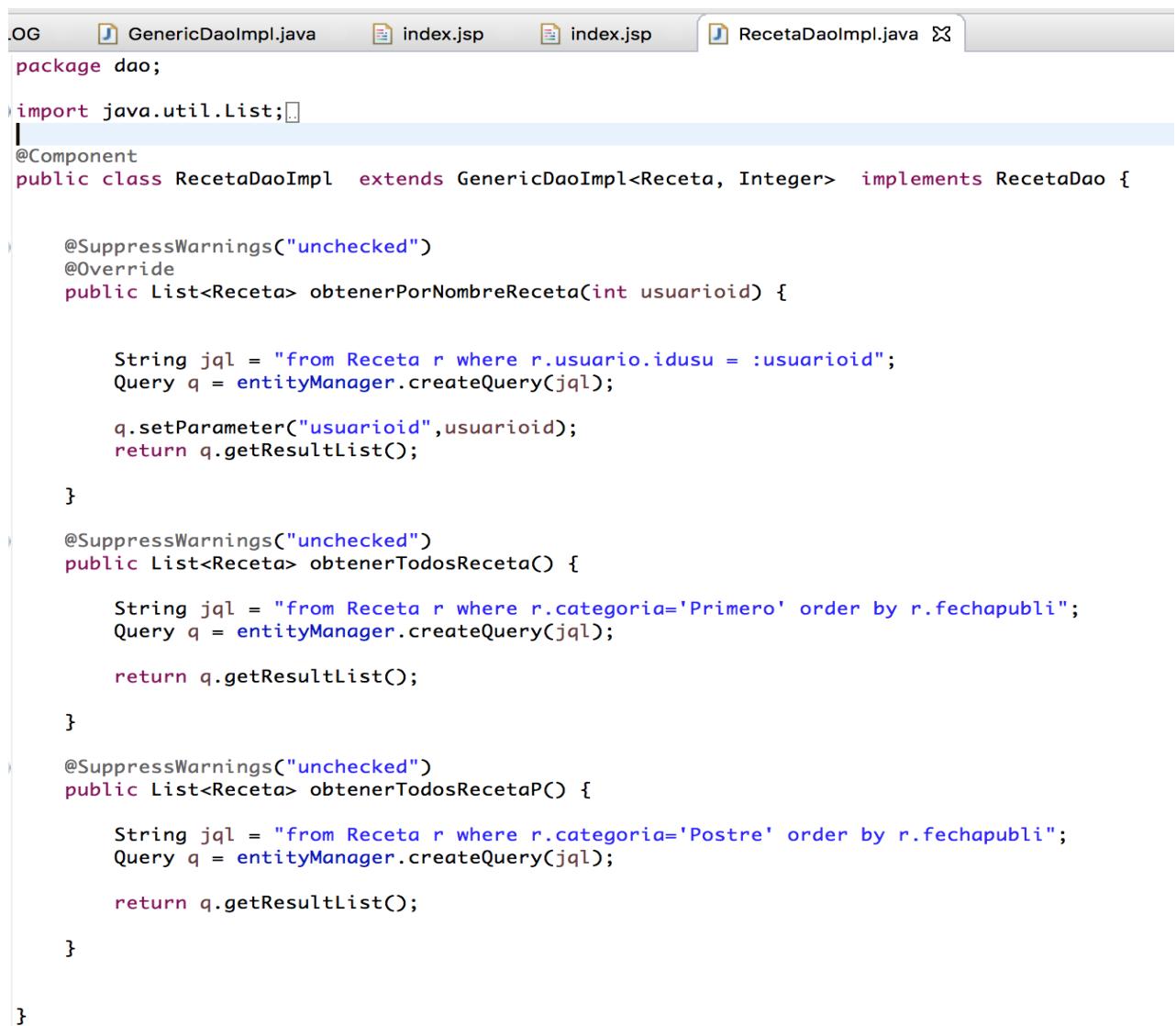
```

7) Clase RecetaDao

En esta clase crearemos los métodos de consulta query con respecto a la clase recetas.

```
package dao;
import java.util.List;
import model.Receta;

public interface RecetaDao extends GenericDao<Receta, Integer> {
    public List<Receta> obtenerPorNombreReceta(int id);
    public List<Receta> obtenerTodosReceta();
    public List<Receta> obtenerTodosRecetaP();
}
```



```
.OG  GenericDaoImpl.java  index.jsp  index.jsp  RecetaDaoImpl.java X
package dao;

import java.util.List;
|
@Component
public class RecetaDaoImpl extends GenericDaoImpl<Receta, Integer> implements RecetaDao {

    @SuppressWarnings("unchecked")
    @Override
    public List<Receta> obtenerPorNombreReceta(int usuarioid) {

        String jql = "from Receta r where r.usuario.idusu = :usuarioid";
        Query q = entityManager.createQuery(jql);

        q.setParameter("usuarioid", usuarioid);
        return q.getResultList();
    }

    @SuppressWarnings("unchecked")
    public List<Receta> obtenerTodosReceta() {

        String jql = "from Receta r where r.categoría='Primero' order by r.fechapubli";
        Query q = entityManager.createQuery(jql);

        return q.getResultList();
    }

    @SuppressWarnings("unchecked")
    public List<Receta> obtenerTodosRecetaP() {

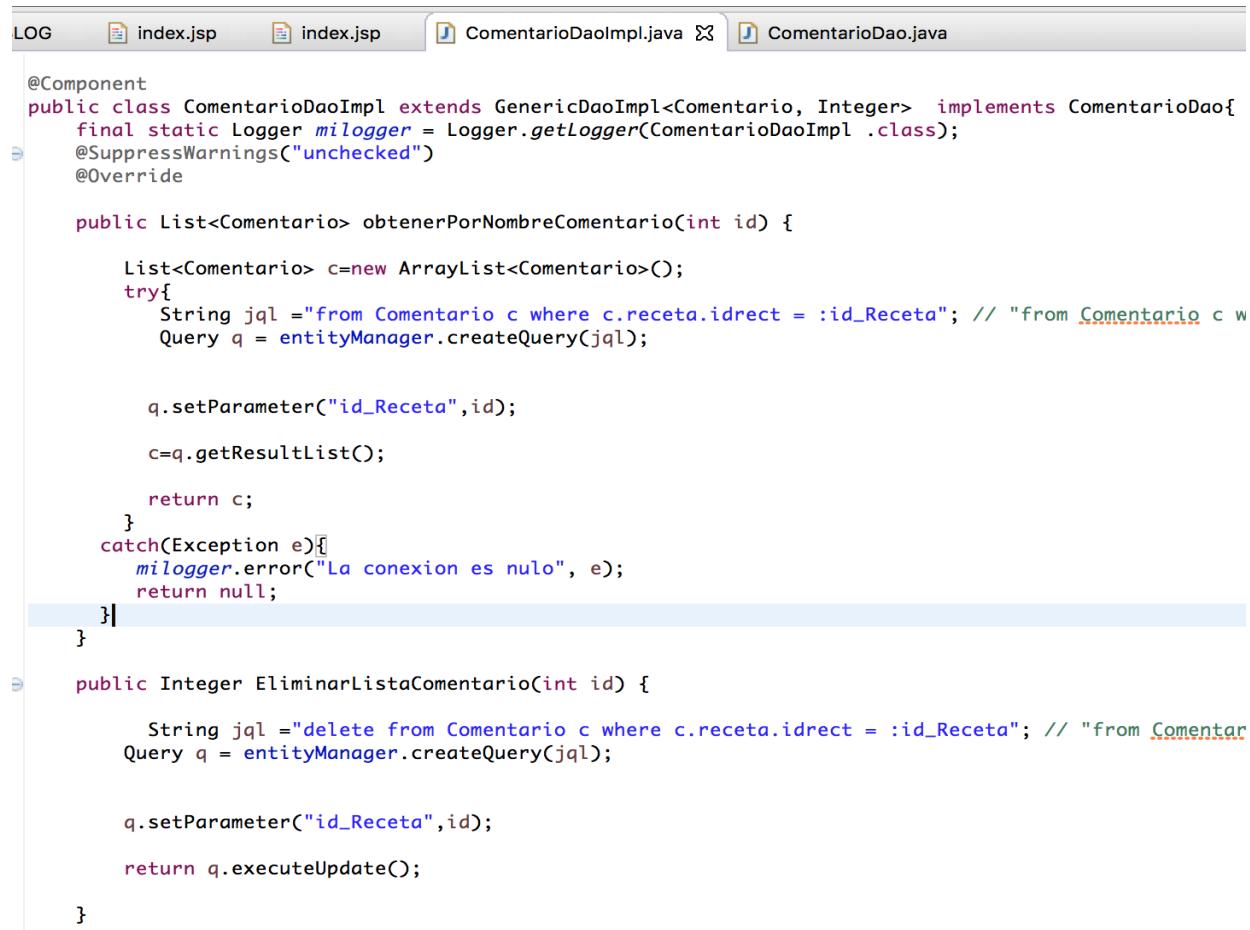
        String jql = "from Receta r where r.categoría='Postre' order by r.fechapubli";
        Query q = entityManager.createQuery(jql);

        return q.getResultList();
    }
}
```

8) Clase ComentarioDao

En esta clase crearemos los métodos de consulta query con respecto a la clase comentario.

```
package dao;
import model.Comentario;
import java.util.List;
public interface ComentarioDao extends GenericDao<Comentario, Integer> {
    public List<Comentario> obtenerPorNombreComentario(int id);
    public Comentario obtenerPorIdComentario(int idReceta);
    public Integer EliminarListaComentario(int id);
    public Integer EliminarComentario(int id);
    public Comentario idRecetaComentario(int id);
}
```



```
@Component
public class ComentarioDaoImpl extends GenericDaoImpl<Comentario, Integer> implements ComentarioDao{
    final static Logger milogger = Logger.getLogger(ComentarioDaoImpl .class);
    @SuppressWarnings("unchecked")
    @Override

    public List<Comentario> obtenerPorNombreComentario(int id) {
        List<Comentario> c=new ArrayList<Comentario>();
        try{
            String jql ="from Comentario c where c.receta.idrect = :id_Receta"; // "from Comentario c w
            Query q = entityManager.createQuery(jql);

            q.setParameter("id_Receta",id);
            c=q.getResultList();

            return c;
        }
        catch(Exception e){
            milogger.error("La conexión es nulo", e);
            return null;
        }
    }

    public Integer EliminarListaComentario(int id) {

        String jql ="delete from Comentario c where c.receta.idrect = :id_Receta"; // "from Comentario c
        Query q = entityManager.createQuery(jql);

        q.setParameter("id_Receta",id);
        return q.executeUpdate();
    }
}
```

```

OG index.jsp index.jsp ComentarioDaoImpl.java ComentarioDao.java

@Override
public Comentario obtenerPorIdComentario(int idReceta) {
    String jql = "from Comentario r where r.receta.idrect = :id_Receta";
    Query q = entityManager.createQuery(jql);
    q.setParameter("id_Receta", idReceta);

    @SuppressWarnings("unchecked")
    List <Comentario> lista=q.getResultList();
    if(lista.isEmpty()){
        return null;
    }
    else{
        return lista.get(0);
    }
}

public Integer EliminarComentario(int id) {
    String jql ="delete from Comentario c where c.idcoment = :id_comentario"; // "from Comentario c where c.Receta.id = :idReceta";
    Query q = entityManager.createQuery(jql);
    q.setParameter("id_comentario",id);
    return q.executeUpdate();
}

public Comentario idRecetaComentario(int id) {
    String jql = "from Comentario r where r.idcoment = :id_Coment";
    Query q = entityManager.createQuery(jql);
    q.setParameter("id_Coment",id);

    @SuppressWarnings("unchecked")
    List <Comentario> lista=q.getResultList();
    if(lista.isEmpty()){
        return null;
    }
    else{ return lista.get(0); }
}
}

```

9) Clase BlogException

Esta clase se usa en la clase UsuarioDao para las excepciones que se cree cuando no exista el usuario.

package exception;

//Aqui iran todas las excepciones que puedan haber

public class BlogException extends RuntimeException {

private static final long serialVersionUID = -2950205365998448571L; //id autogenerado

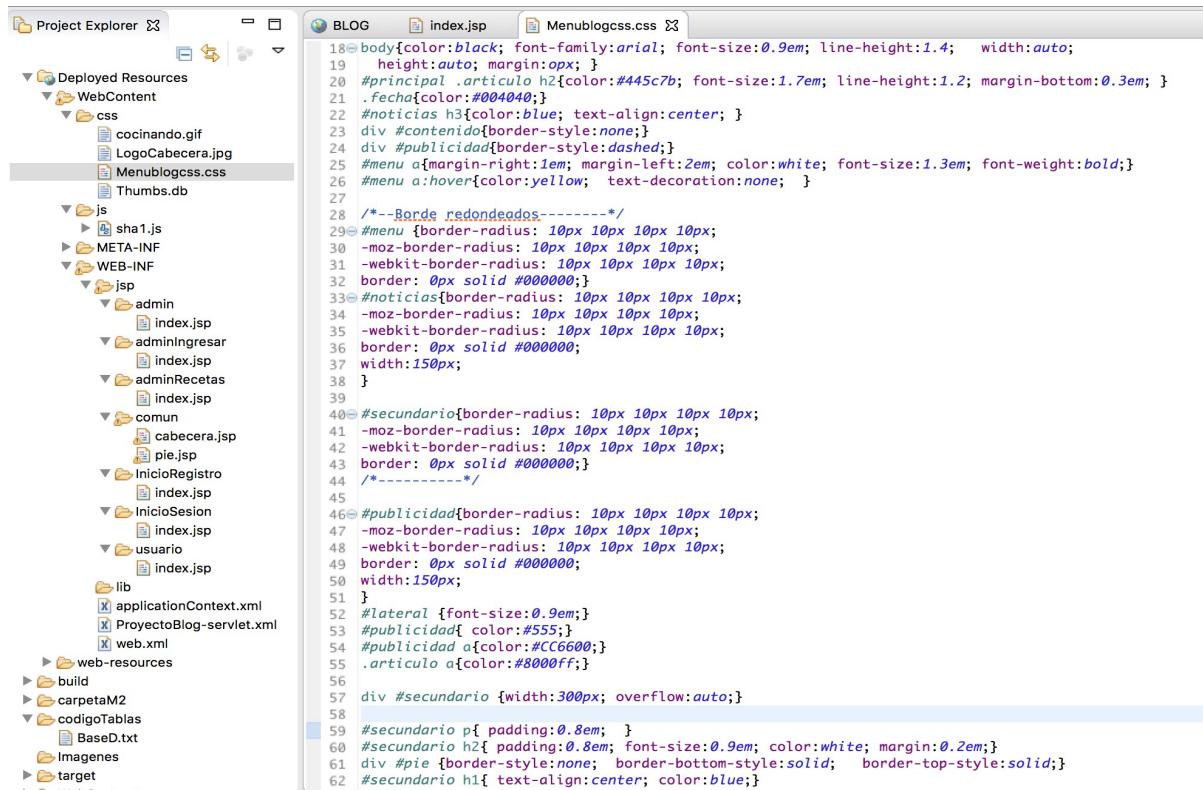
```

public BlogException() { //esto es llamado por el constructor
    super();
}
public BlogException(String message) { //una exception con parametro
    super(message);
}
public BlogException(Throwable e) {
    super(e);
}
}

```

4.6 Contenido WebContent

En esta carpeta es donde se encontraran los jsp, css, imagenes y sha1.js.



```

body{color:black; font-family:arial; font-size:0.9em; line-height:1.4; width:auto;
height:auto; margin:0px; }
#principal .articulo h2{color:#445c7b; font-size:1.7em; line-height:1.2; margin-bottom:0.3em; }
.noticias h3{color:blue; text-align:center; }
div #contenido{border-style:none; }
div #publicidad{border-style:dashed; }
#menu a{margin-right:1em; margin-left:2em; color:white; font-size:1.3em; font-weight:bold; }
#menu a:hover{color:yellow; text-decoration:none; }

/*--Borde redondeados-----*/
#menu {border-radius: 10px 10px 10px 10px;
-moz-border-radius: 10px 10px 10px 10px;
-webkit-border-radius: 10px 10px 10px 10px;
border: 0px solid #000000; }
#noticias{border-radius: 10px 10px 10px 10px;
-moz-border-radius: 10px 10px 10px 10px;
-webkit-border-radius: 10px 10px 10px 10px;
border: 0px solid #000000;
width:150px; }
#secundario{border-radius: 10px 10px 10px 10px;
-moz-border-radius: 10px 10px 10px 10px;
-webkit-border-radius: 10px 10px 10px 10px;
border: 0px solid #000000;
border: 0px solid #000000;
width:150px; }
#publicidad{border-radius: 10px 10px 10px 10px;
-moz-border-radius: 10px 10px 10px 10px;
-webkit-border-radius: 10px 10px 10px 10px;
border: 0px solid #000000;
width:150px; }
.lateral {font-size:0.9em; }
#publicidad {color:#555; }
#publicidad a{color:#CC6600; }
.articulo a{color:#8000ff; }

div #secundario {width:300px; overflow:auto; }

#secundario p{ padding:0.8em; }
#secundario h2{ padding:0.8em; font-size:0.9em; color:white; margin:0.2em; }
div #pie {border-style:none; border-bottom-style:solid; border-top-style:solid; }
#secundario h1{ text-align:center; color:blue; }

```

Esquema de como sera la pagina web, el contenido en articulo sera lo que cambiara, lo demás iremos reciclando para usarlo en todas las paginas.



En estas jsp serán donde creemos los formularios en formato HMTL.

Y en algunas jsp usaremos javascript para enviar los datos recibidos o para redirigir a la pagina deseada que se enviara a la carpeta controller y con el mapping que se indique sera la clase indicada y metodo.

1) Jsp cabecera

jsp/comun/cabecera.jsp : Esta cabecera contendrá el logo, menu, lateral y parte del contenido hasta es donde comience el div articulo.

```

LOG cabecera.jsp
<%@page import="model.Receta"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/> <!-- Nombre o contexto de la aplicación -->
<!DOCTYPE html >
<html>
<head>
<meta charset=UTF-8>
<title>BLOG</title>
<link rel="stylesheet" href='<%= getServletContext().getContextPath() %>/css/MenuBlog.css' media="all"/>
</head>
<body>
<!-- Contenedor -->
<div id="contenedor">
  <!-- Cabecera -->
  <div id="cabecera">
    <div id="logo">
      <img src='<%= getServletContext().getContextPath() %>/css/LogoCabecera.jpg' height=100px alt="" />
    </div>
    <div class="clear"></div>
  </div>
  <!-- /Cabecera -->

  <!-- Menu principal -->
  <div id="menu">
    <ul id="menu_principal">
      <li><a href='${contextPath}/action/usuario/'>Inicio</a></li>
      <c:if test='${o!=null && o!="anónimo"}'>
        <li><a href='${contextPath}/action/usuario/irIniciarSesion'>Iniciar Sesión</a></li>
        <li><a href='${contextPath}/action/usuario/irRegistrar'>Registrarte</a></li>
        <li><a href='><${sessionScope.usuario.nombreusuario}</a></li>
      </c:if>
      <c:if var="o" value="${usuario.nombreusuario}">
        <c:if test='${o!=null && o!="anónimo"}'>
          <li><a href='${pageContext.request.contextPath}/action/admin/'>Página de Inicio usuario</a></li>
          <li><a href='${pageContext.request.contextPath}/action/admin/ingresarRecetas?personaid=${usuario.idusu}'>adir Recet
          <li><a href='${pageContext.request.contextPath}/action/usuario/cerrar'>Salir</a></li>
          <li><a href='><${sessionScope.usuario.nombreusuario}</a></li>
        </c:if>
      </c:if>
    </ul>
    <div class="clear"></div>
  </div>

  <!-- Lateral -->
  <div id="lateral">
    <!-- Noticias -->
    <div id="noticias">
      <h3>Recetas subidas</h3> <br/>
      <c:forEach var="b" items="${recetas}">
        <p><span class="fecha"> ${b.getFechaPubli()} </span><br/> <a href='mostrar?id=${b.getIdrect()}'> ${b.getNombrerect()} </a></p> <br/>
      </c:forEach>
    </div>
    <!-- /Noticias -->

    <!-- Publicidad -->
    <div id="publicidad">
      <img src='<%= getServletContext().getContextPath() %>/css/cocinando.gif' width=150px height=200px border=0 usemap="#deinteres">
    </div>
    <!-- /Publicidad -->
  </div>
  <!-- /Lateral -->

  <div id="contenido">
    <!-- Principal -->
    <div id="principal">
    <div class="articulo">

```

2) Jsp pie

jsp/comun/pie.jsp Este pie contendrá el final de cierre del div principal, secundario, pie y el cierre del div contenedor.

```

LOG pie.jsp ✎
<%@page import="model.Usuario"%>
<%@page import="model.Receta"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

</div>
</div>
<!-- /Principal -->

<!-- Secundario-->
<div id="secundario">
    <h1>Postres</h1>
    <c:forEach var="b" items="${recetasP}" >
        <h2> ${b.getNombrerect()}</h2>

        <p> ${b.getResumen()} </p>
        <p><a href='mostrar?id=${b.getIdrect()}'>Seguir leyendo...</a></p>
    </c:forEach>

</div>
<!-- /Secundario -->

</div>
<!-- /Contenido -->

<!-- Pie -->
<div id="pie">
    <span class="enlaces">
        <a href="http://eladerezo.hola.com/">El aderezo</a> |
        <a href="http://www.hogarutil.com/cocina/">Hogar util</a> |
        <a href="http://www.saludalacarta.com/">Salud</a> |
    </span>

    <span class="copyright">
        © Copyright LauraIMendoza
    </span>
    <div class="clear"></div>
</div>
<!-- /Pie -->

</div>
<!-- /Contenedor -->

</body>
</html>

```

Como estas partes serán los mismos en todas las páginas web, lo que haremos será incluirlo en las jsp `<%@ include file="/WEB-INF/jsp/comun/cabecera.jsp" %>` de esta manera tendremos todos los datos sin tener que estar repitiendo estas partes.

3) Jsp Inicio usuario

jsp/usuario/index.jsp :En esta jsp sera lo que verán como inicio de la pagina. El usuario en general podrá ver las recetas y comentarlas. Si desean subir recetas primero deberá estar registrado y a continuación podrá subir sus propias recetas.

```

index.jsp UserController.java AdminController.java

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="contextPath" value="${pageContext.request.contextPath}" />
<%@ include file="/WEB-INF/jsp/comun/cabecera.jsp" %>

<script src="${contextPath}/js/sha1.js"></script> <!-- -->

<!-- Comienzo de inicio, esta parte cambia -->
<div class="errores">
    <c:forEach var="e" items="${errores}">
        <h3>${e}</h3><br>
    </c:forEach>
</div>

<br/><br/>
<h1>BIENVENIDOS AL BLOG</h1> <br/>
<p id="parrafo">Este blog esta Pensado para las personas que quieran dar a conocer recetas nuevas y facil de preparar </p>

<c:forEach var="b" items="${recetas}" >
    <h2> ${b.getNombrerect()}</h2>
    <ul>
        <li>
            <h3>Resumen</h3>
            <p id="resumen"><c:out value=" ${b.getResumen()}" /></p>
            <p><a href='mostrar?id=${b.getIdrect()}'>Seguir leyendo....</a></p> <br/><br/>
        </li>
    </ul>
</c:forEach>
<!-- Fin de esta parte -->
<%@ include file="/WEB-INF/jsp/comun/pie.jsp" %>

```

Esta Jsp también se usara cuando se quieran registrar y el usuario ya existe o cuando se escriba mal la contraseña para cuando se inicie sesión, a continuación reenviaran a esta pagina con los errores que hayan pasado.

Hay varias maneras de llamar el mapping y enviar la información que se necesite

4) Jsp Mostrar receta

jsp/adminRecetas/index.jsp :En esta Jsp recibe la clase receta del id correspondiente que se ha seleccionado, para obtener toda la información que tiene la receta.

```

OG index.jsp
<%@page import="model.Usuario"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ include file="/WEB-INF/jsp/comun/cabecera.jsp" %>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>

<script>
var agregarComentario=function(){ document.forms[0].submit(); | } ;
</script>
<h2>${recetasID.getNombrerecto} </h2>
<br/>
<h3>Ingredientes</h3>
<c:out value="${recetasID.getIngredientes()}" />
<div id="imagenReceta">

</div>

<ul id="lista">
<c:forTokens items="${oracion}" delims='<%= "%>' var="token" varStatus="i" >
<li>
<c:out value="${token}" />
</li>
</c:forTokens>
</ul>
<br/><br/>
<h3>Preparaci&on</h3><br/>
<p><c:out value="${recetasID.getDescripcion()}" /></p> <br/>

<form action="agregarComentario" method="post" >
<c:set var="n" value="${sessionScope.usuario.nombreusuario} " />
<c:if test="${n==null || n=='anonomo'}">
<input type="hidden" name="personaid" value="2" > <!-- value='personaid?=>2'-->
</c:if>
<c:if test="${n!=null && n!='anonomo'}">
<input type="hidden" name="personaid" value="${sessionScope.usuario.idusu}" >
</c:if>
<input type="hidden" name="recetaid" value="${recetasID.getIdrecto()}" >
<table>
<tr>
<td><textarea name="comentarios" rows="5" cols="40" >Escribir Comentario</textarea></td>
<td><input type="button" value="Enviar" onclick="agregarComentario()"></td>
</tr>
</table>
</form>

<h2>Comentarios</h2>

<div class="comentario">
<c:forEach var="b" items="${comentarioT}" >
<p><span class="usuario">${b.getUsuario().nombreusuario} <br /> ${b.getDescripcomento()} &nbsp &nbsp &nbsp
<c:set var="u" value="${sessionScope.usuario.nombreusuario}" />
<c:if test="${u!=null && u!='anonomo'}">
<c:if test='${recetasID.getUsuario().getIdusu()==sessionScope.usuario.idusu}'>
<a href="${pageContext.request.contextPath}/action/admin/eliminarComment?id=${b.getIdcomento()}"> Eliminar Comentario</a>
</c:if>
</c:if>

</p>
<br /> <br />
</c:forEach>
<br/>

<%@ include file="/WEB-INF/jsp/comun/pie.jsp" %>

```

En el controller ya hemos almacenado la sesión del usuario si ha sido correctamente y esta guardado session.setAttribute("usuario", u). Solo para llamarlo y ver si hay algún usuario que ha iniciado sesión, para recuperar ese usuario seria \${sessionScope.usuario}

5) Jsp Registrarse

jsp/inicioRegistro/index.jsp :En el javascript encriptara la clave antes de enviarlo al controller, se usa el fichero sha1.js para usar el sha1 que nos devolverá la clave encriptada.

```
<!-- Comienzo de inicio, esta parte cambia -->
<br/><br/>
<h2>Darse de alta</h2>
<form action="agregar" method="post" name="agregar">
  <table>
    <tr>
      <td>Usuario</td>
      <td><input type="text" name="nomusuario"></td>
    </tr>
    <tr>
      <td>Contraseña</td>
      <td><input type="password" id="clave" name="clave">
          <input type="hidden" id="claveLimpia" >
        </td>
    </tr>
    <tr>
      <td>Repite contraseña</td>
      <td><input type="password" id="claveRepetida" name="claveRepetida">
        </td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="button" value="Dar de alta" onclick="darAlta()"> <!-- Envial javascript todos los datos -->
      </td>
    </tr>
  </table>
</form>

<!-- Fin de esta parte -->
<%@ include file="/WEB-INF/jsp/comun/pie.jsp"%> <!-- incluye el contenido del pie.jsp -->
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:if test="#{not empty param.error}">param.error

```

6) Jsp Iniciar sesión

jsp/inicioSesion/index.jsp : Esta jsp envía la clave encriptada y el nombre de usuario para comparar los datos. Si es correcto se abrirá el jsp con los datos de las recetas que tiene el usuario y sino no lo es te mostrar el jsp de inicio de sesión con el error que haya pasado.

```

LOG index.jsp <%
<%@ page language="java" contentType="text/html; charset=UTF-8"    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="contextPath" value="${pageContext.request.contextPath}" />

<%@ include file="/WEB-INF/jsp/comun/cabecera.jsp" %>
<script src="${contextPath}/js/sha1.js"></script>
<script>
  var ingresar = function() {
    // Calculando HASH de la clave
    var claveLimpia = document.getElementById("claveLimpia").value;
    var hash = CryptoJS.SHA1(claveLimpia);
    document.getElementById("clave").value = hash;

    document.forms[0].submit();
  };
</script>
<!-- Comienzo de inicio, esta parte cambia --&gt;
&lt;br/&gt;&lt;br/&gt;
&lt;h2>Iniciar Sesión</h2> <br/>
<form action="autenticar" method="post" > <!-- El formulario envia a autenticar -->
  <table>
    <tr>
      <td>Usuario</td>
      <td><input type="text" name="nomusuario"></td>
    </tr>
    <tr>
      <td>Contraseña</td>
      <td>
        <input type="hidden" id="clave" name="clave">
        <input type="password" id="claveLimpia" name="claveLimpia">
      </td>
    </tr>
    <tr>
      <td colspan="2" > <!-- El onclick su ingresar es una funcion que este arriba y la contraseña lo encripte -->
      <input type="button" value="Ingresar" onclick="ingresar()"/>
    </td>
  </table>
</form>
<!-- Fin de esta parte --&gt;
&lt;%@ include file="/WEB-INF/jsp/comun/pie.jsp" %&gt;
</pre>

```

7) Jsp Pagina usuario

jsp/admin/index.jsp :En esta jsp recibirá una lista de la clase receta de un usuario en concreto para poder mostrar toda la información de todas las recetas que tiene el usuario.

```

LOG index.jsp <%
<%@page import="model.Usuario"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ include file="/WEB-INF/jsp/comun/cabecera.jsp" %>
<!-- pagina principal del usuario ingresado -->

<c:set var="contextPath" value="${pageContext.request.contextPath}" />

<cc:forEach var="b" items="${recetasU}" >
  <h2> ${b.getNombrerect()} </h2> <br/>
  <ul>
    <li>
      <h3>Resumen</h3><br/>
      <cc:out value=" ${b.getResumen()}" /> <br/>
      <p><a href='mostrar?id=${b.getIdrect()}'>Seguir leyendo...</a> &ampnbsp &ampnbsp &ampnbsp <a href='eliminarReceta?idrect=${b.getIdrect()}'>Eliminar</a></p>
    </li>
  </ul>
</cc:forEach>
<br/>

<%@ include file="/WEB-INF/jsp/comun/pie.jsp" %>

```

8) Jsp ingresar receta

jsp/adminIngresar/index.jsp : En esta jsp se ingresara todos los datos de para una nueva receta.

```

.OG index.jsp index.jsp
<c:set var="contextPath" value="${pageContext.request.contextPath}" />
<%@ include file="/WEB-INF/jsp/comun/cabecera.jsp"%>
<%@ page import="java.util.*" %>
<script>
    var agregarReceta=function(){
        document.forms[0].submit();
    };
</script>

<!-- Comienzo de inicio, esta parte cambia -->
<br /> <br />
<h2>Ingresar Receta</h2>
<form action="agregarReceta" method="post" name="agregar" >
    <table>
        <tr>
            <td>Nombre de la receta</td>
            <td><input type="text" name="nomReceta"></td>
        </tr>
        <tr>
            <td>Escoger categoria</td>
            <td><select name="nomCategoria">
                <option>Primero</option>
                <option>Postre</option>
            </select></td>
        </tr>
        <tr>
            <td>Resumen</td>
            <td><textarea name="nomResumen" rows="5" cols="40"> </textarea></td>
        </tr>
        <tr>
            <td>Ingredientes</td>
            <td><textarea name="nomIngredientes" rows="5" cols="40"> </textarea></td>
        </tr>
        <tr>
            <td>Descripcion de la Preparacion</td>
            <td><textarea name="nomDescripcion" rows="5" cols="40"> </textarea></td>
        </tr>
        <tr>
            <td>Imagen</td>
            <td>
                <input type="file" id="files" name="files" value="" multiple />
                <output id="list"></output>
                <input type="hidden" id="ruta" name="ruta" type="text"/>
            </td>
        </tr>
    </table>
</form>

```

```

.OG index.jsp index.jsp
function archivo(evt) {
    var files = evt.target.files; // Lista de objetos

    // Se obtiene la imagen del campo file
    for (var i = 0, f; f = files[i]; i++) {
        //Solo admite imagenes
        if (!f.type.match('image.*')) {
            continue;
        }

        var reader = new FileReader();

        reader.onload = (function(theFile) {
            return function(e) {
                // Insertamos la imagen
                document.getElementById("ruta").value= e.target.result;
                document.getElementById("list").innerHTML = [''].join('');
            }
        })(f);

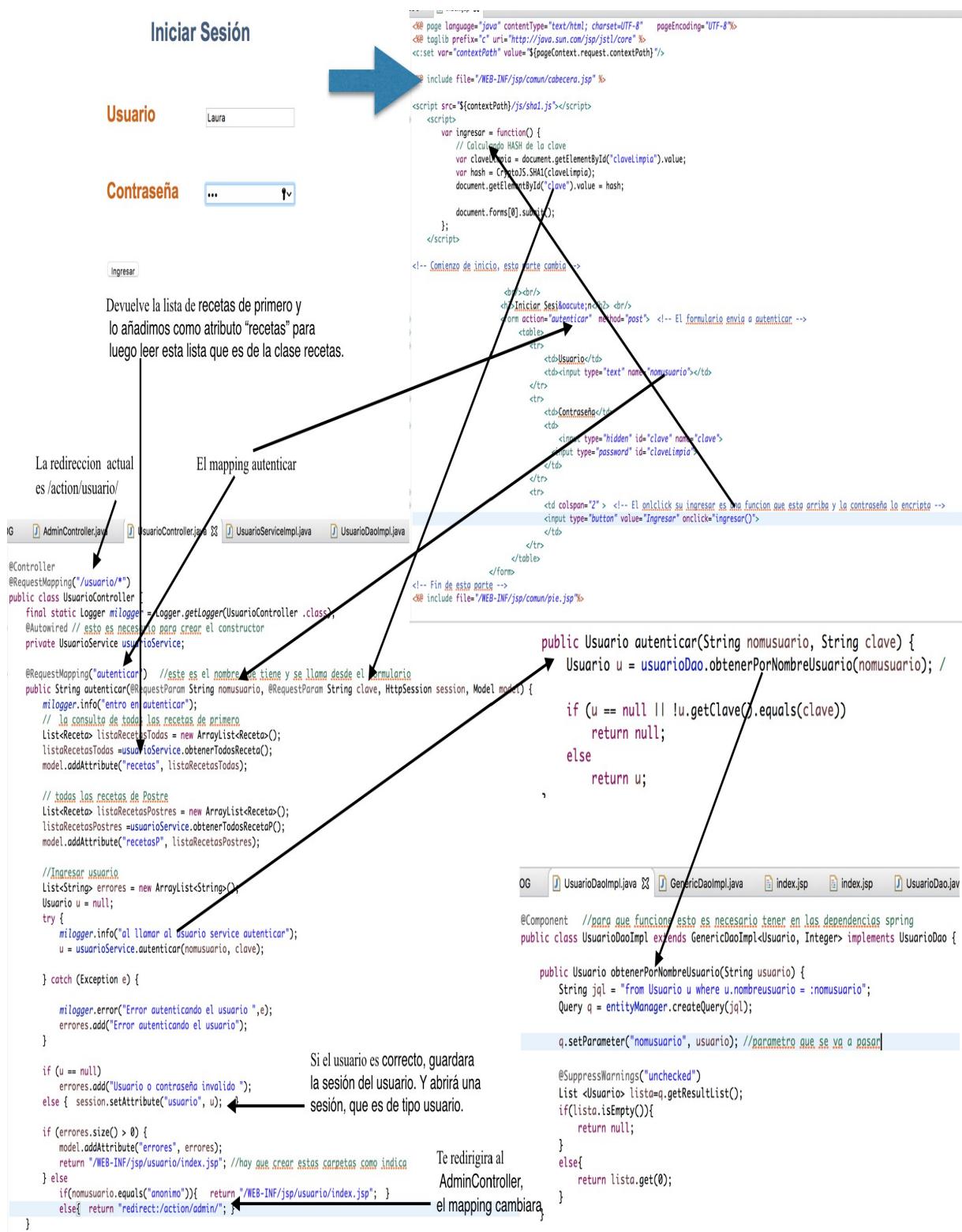
        reader.readAsDataURL(f);
    }
}

document.getElementById('files').addEventListener('change', archivo, false);

```

5. Explicación del funcionamiento del controller y el jsp

Para poder entender como funciona veremos como funciona el inicio de sesión de un usuario.



Al entrar leerá este mapping por defecto.

El Mapping ha cambiado

```

DG AdminController.java ✘ cabecera.jsp UsuarioDaoImpl.java ComentarioDaoImpl.java UsuarioControl
@Controller
@RequestMapping("/admin/*")
public class AdminController {
    final static Logger logger = Logger.getLogger(AdminController.class);
    @Autowired
    private UsuarioService userService;

    // Cuando uso el cuestionario que hay en InicioSesion viene aqui
    @RequestMapping("")
    public String admin(Model model, HttpSession session) {
        logger.info("Entró en la parte del administrador, que solo entró el usuario");

        // La consulta de todas las recetas de primero
        List<Receta> listaRecetasTodas = new ArrayList<Receta>();
        logger.info("al llamar al usuario service de todas recetas de primero");
        listaRecetasTodas = userService.obtenerTodosReceta();
        model.addAttribute("recetas", listaRecetasTodas);

        // Todas las recetas de Postre
        List<Receta> listaRecetasPostres = new ArrayList<Receta>();
        logger.info("al llamar al usuario service de todas recetas de postre");
        listaRecetasPostres = userService.obtenerTodosRecetaP();
        model.addAttribute("recetasP", listaRecetasPostres);

        // Todas las recetas que tiene el usuario
        Usuario u = new Usuario();
        u = (Usuario) session.getAttribute("usuario");

        try {
            Integer id = u.getIdusu();

            List<Receta> listaRecetas = new ArrayList<Receta>();
            logger.info("al llamar al usuario service para obtener la recetas que tiene el usuario");
            listaRecetas = userService.obtenerPorNombreReceta(id);
            model.addAttribute("recetasU", listaRecetas);

            return "/WEB-INF/jsp/admin/index.jsp"; // es donde se redirige para que se muestre
        } catch (NullPointerException e) {
            return "redirect:/action/usuario/";
        }
    }
}
  
```

Devuelve la lista de recetas de postre y lo añadimos como atributo "recetasP" para luego leer esta lista que es de la clase recetas.

La sesión que se había creado devolvemos su valor que tiene, que es de la clase usuario.

Devuelve la lista de recetas que tiene el usuario y lo añadimos como atributo "recetasU" para luego leer esta lista que es de la clase recetas.

Página de Inicio usuario Añadir Recetas Salir laura

Croqueta

Resumen

Las croquetas de pollo son una guarnición muy común como entrante a la hora de comer en un restaurante o una comida que organizamos nosotros. Ello se debe a que gracias a la bechamel y el pollo es un alimento muy ligero que se digiere bien en la boca, por lo que podemos disfrutar mejor de su sabor.

[Seguir leyendo...](#) [Eliminar](#)

Tarta de queso sin horno receta

Resumen

La tarta de queso sin horno también es uno de los postres más populares que con mucha frecuencia se suelen cocinar para disfrutar en familia o con los amigos. La principal diferencia con respecto a otras tartas de queso tradicionales es que aquí se utiliza el horno para su elaboración, lo que significa que prácticamente cualquiera puede hacer una tarta de queso.

[Seguir leyendo...](#) [Eliminar](#)

```

LOG index.jsp 23
page import="model.Usuario"
page language="java" contentType="text/html; charset=UTF-8"
pageEncoding=UTF-8
taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" 
include file="/WEB-INF/jsp/comun/cabecera.jsp"
<!-- página principal del usuario ingresado --&gt;

&lt;c:set var="contextPath" value="${pageContext.request.contextPath}" /&gt;

&lt;c:forEach var="b" items="${recetasU}"&gt;
    &lt;h2&gt; ${b.getNombreReceta()}&lt;/h2&gt; &lt;br/&gt;
    &lt;ul&gt;
        &lt;li&gt;
            &lt;h3&gt;Resumen&lt;/h3&gt;&lt;br/&gt;
            &lt;c:out value="${b.getResumen()}" /&gt; &lt;br/&gt;
            &lt;p&gt;&lt;a href="mostrar?id=${b.getIdrect()}"&gt;Seguir leyendo...&lt;/a&gt; &ampnbsp; &ampnbsp; &ampnbsp; &lt;a href="eliminarReceta?idreceta=${b.getIdrect()}"&gt;Eliminar&lt;/a&gt;&lt;/p&gt;
        &lt;/li&gt;
    &lt;/ul&gt;
&lt;/c:forEach&gt;
&lt;br/&gt;
  </pre>

```

5.1 Notas importantes

- Todas las clases UsuarioDao, RecetaDao, ComentarioDao extienden del GenericDao de la clase genérico que se creo. Al extender puede usar entityManager para poder consultar una query.
- Con el GenericDaoImpl como esta generalizado, el valor de T vendría hacer la clase Receta o Usuario o Comentario y el valor de K es de tipo Integer porque sera cuando se pase el id principal de la clase.
`GenericDaoImpl<T, K> implements GenericDao<T, K>`
`RecetaDaoImpl extends GenericDaoImpl<Receta, Integer>`
- Las clase RecetaDaoImpl, ComentarioDaoImpl, UsuarioDaoImpl tienen el `@Component` que indica al contenedor de Spring que podemos usar esta clase a través de Spring por la Inyección de Dependencia. Marca la clase java como un bean, por lo que el mecanismo de exploración de componentes de spring puede recogerlo y tirar de él en el contexto de la aplicación.
- Cuando se cree algún nuevo método en el Dao primero se pone el nombre del método en la interface del Dao y luego se implementa el método en su clase implements.
- En el MenuBlogcss es donde contiene todo los estilos que tiene el html. El tipo de fuente que tendrá, color de letra, tamaño, contornos de los bordes de los div, posicionamiento, margen, etc.

6. Medios utilizados

Los recursos que necesite son:

- Servidor Ubuntu 12.04.
- Apache tomcat 8.0.33
- Mysql 5.1
- Eclipse con el sistema operativo OS X El Capitan (MAC). Para java EE.
- Java Web Dynamic Project.
- Lenguajes de programación: Java, Maven, JPA, HTML, CSS, Hibernate, Javascript, Sql.

7. BIBLIOGRAFIA

<http://webresizer.com/resizer/>

<https://www.fotojet.com/es/app.html?category=poster&entry=design>

<https://help.ubuntu.com/12.04/serverguide/mysql.html>

<http://www.desarrolloweb.com/articulos/2408.php>

<http://stackoverflow.com/questions/30990488/how-do-i-install-command-line-mysql-client-on-mac>

<http://dbadiaries.com/how-to-install-mysql-5-5-on-ubuntu-server-12-04-lts>

<https://www.digitalocean.com/community/tutorials/how-to-install-apache-tomcat-8-on-ubuntu-14-04>

<https://www.liquidweb.com/kb/how-to-install-apache-tomcat-8-on-ubuntu-12-04/>

<https://parasitovirtual.wordpress.com/category/cursos-y-articulos/desarrollo-aplicaciones-software/java/j2ee/servlets/>

<http://www.sha1-online.com>

<http://stackoverflow.com/questions/5654819/how-can-i-decrypt-mysql-passwords>

<http://www.vicchiam.com/blog/?p=42>

<https://clouding.io/kb/conceder-y-quitar-privilegios-en-mysql/>

<https://mvnrepository.com>

<http://stackoverflow.com/questions/182636/how-to-determine-the-class-of-a-generic-type>

https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

<http://www.notodocodigo.com/spring/conexion-automatica-de-beans/>