

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

12-1-2020

Tienda Online

Práctica de Acceso a Datos

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Pablo Oraá

2º DAM (2019-2020)

V1.3-1.3.1

Contenido

1. Introducción	1
2. Requisitos	1
3. Instalación	3
4. Características base de la Tienda Online.....	6
5. Características extra de la Tienda Online	6
6. Flujo detallado de la Tienda Online.....	8
7. Documentación	13

1. Introducción

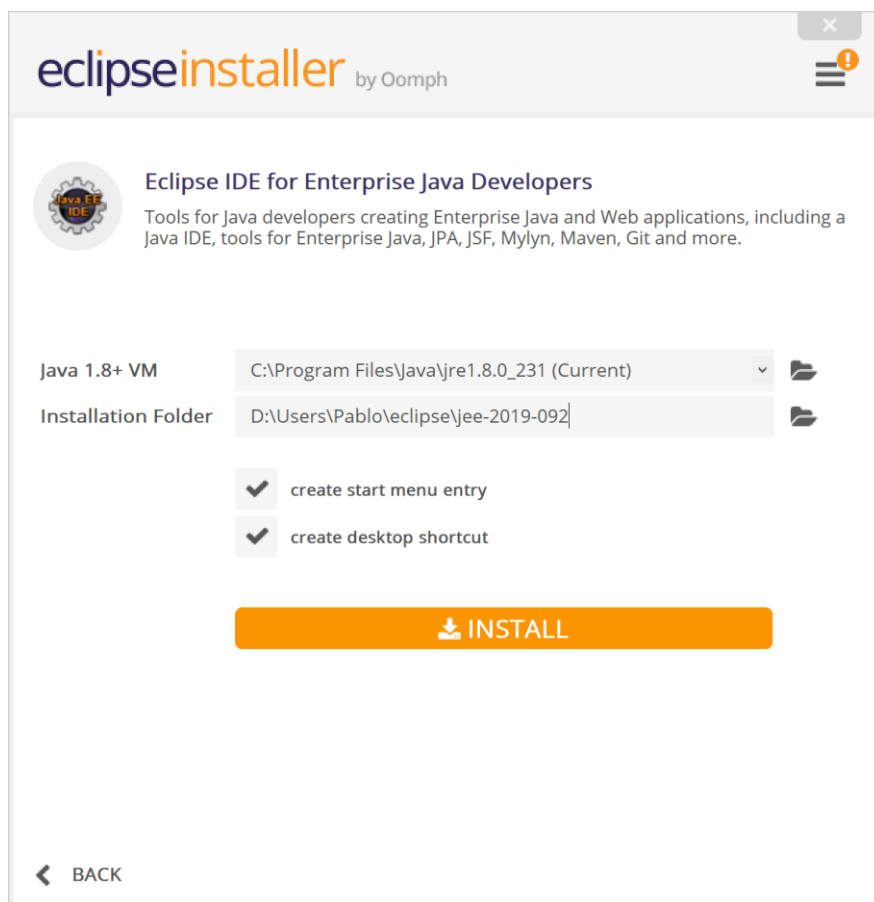
Tienda Online es un proyecto realizado para la clase Acceso de Datos de segundo de DAM por Pablo Oraá López durante la última semana de clase antes de las vacaciones de navidad para Victor del Pozo, profesor del Gregorio Fernández. Toda la información sobre los requisitos mínimos exigidos podrán encontrarse en el documento Practica Tienda Online MVC.

Con esta práctica se busca demostrar el uso de pool de conexiones, jsp, servlet y otro tipo de teoría y práctica aplicada al desarrollo web durante la primera evaluación del curso. Durante todo el documento se hará referencia a los productos introducidos por defecto, así como al usuario “admin/admin” a modo de ejemplo cuando sea necesario.

2. Requisitos

Para poder ejecutar esta Tienda Online, será necesario disponer de un IDE de programación JavaEE para poder crear un proyecto web dinámico. Durante el desarrollo se ha utilizado Eclipse (en su versión Photon 2018 y en la 2019-09) junto al servidor Tomcat 9.0.24 (en formato zip). Tanto [Eclipse](#) como [Tomcat](#) podrán descargarse de forma gratuita desde la página web correspondiente.

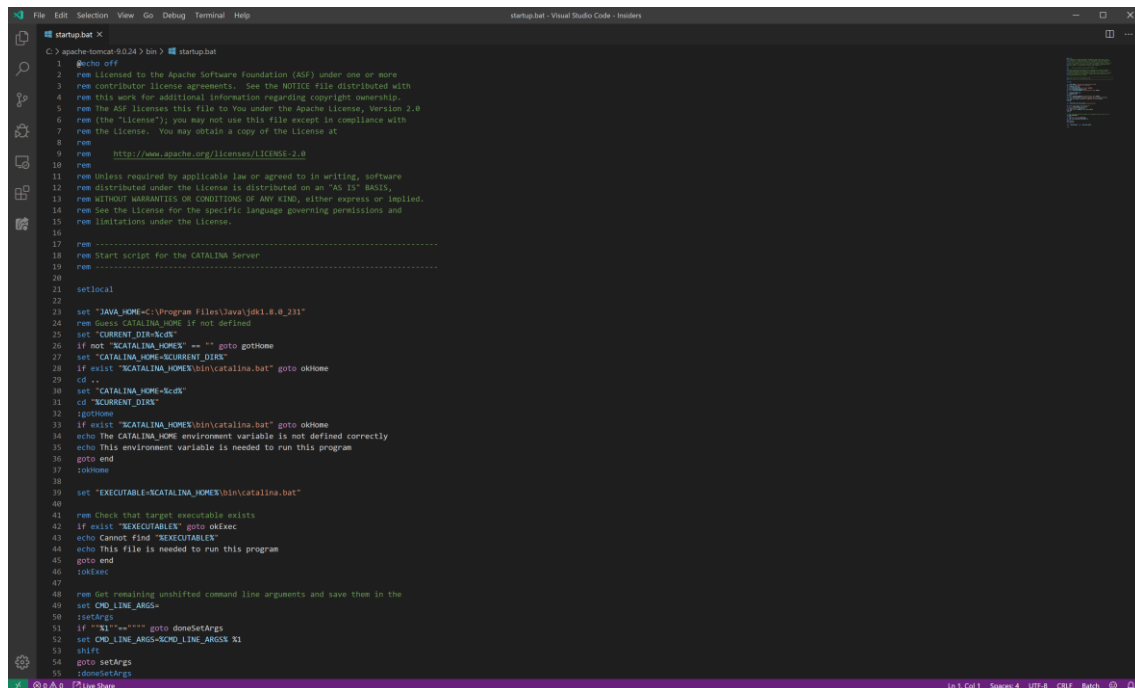
Además, será necesario disponer del JDK y JRE de Java, que podremos descargar desde la página de [Oracle](#). Una vez tengamos todo descargado, podremos proceder a la instalación de Java directamente desde el ejecutable. A continuación, continuaremos con la instalación de Eclipse, siempre marcando la opción JEE.



Muestra de la instalación de Eclipse

A continuación, podemos proceder con la configuración inicial de Tomcat. Tendremos que descomprimir el fichero .zip dentro de la unidad C de nuestro equipo. Una vez esto haya tenido lugar, tendremos que acceder a la carpeta en cuestión, a continuación, ir a bin y editar el fichero startup.bat (Este es el encargado de iniciar el servidor en el puerto indicado (por defecto 8080)).

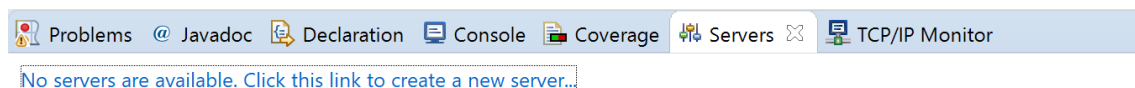
En este, tendremos que añadir al principio la ruta hacia el JDK que tengamos instalado en el equipo. Si dejamos todo por defecto, este se encontrará en C:\Program Files\Java\jdkx.x.x_xxx, siendo las x sustituidas por la versión del JDK que tengamos.



```
1 @echo off
2 rem Licensed to the Apache Software Foundation (ASF) under one or more
3 rem contributor license agreements. See the NOTICE file distributed with
4 rem this work for additional information regarding copyright ownership.
5 rem The ASF licenses this file to You under the Apache License, version 2.0
6 rem (the "License"); you may not use this file except in compliance with
7 rem the License. You may obtain a copy of the License at
8 rem
9 rem http://www.apache.org/licenses/LICENSE-2.0
10 rem
11 rem Unless required by applicable law or agreed to in writing, software
12 rem distributed under the License is distributed on an "AS IS" BASIS,
13 rem WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 rem See the License for the specific language governing permissions and
15 rem limitations under the License.
16
17 rem -----
18 rem Start script for the CATALINA Server
19 rem -----
20
21 setlocal
22
23 set "JAVA_HOME=C:\Program Files\Java\jdk1.8.0_231"
24 rem Guess CATALINA_HOME if not defined
25 set "CURRENT_DIR=%%~dp0"
26 if not "%CATALINA_HOME%" == "" goto gotHome
27 set "CATALINA_HOME=%%~dp0"
28 if exist "%CATALINA_HOME%\bin\catalina.bat" goto okHome
29 cd .
30 set "CATALINA_HOME=%%~dp0"
31 cd "%CURRENT_DIR%"
32 gotoHome
33 if exist "%CATALINA_HOME%\bin\catalina.bat" goto okHome
34 echo The CATALINA_HOME environment variable is not defined correctly
35 echo This environment variable is needed to run this program
36 goto end
37 :okHome
38
39 set "EXECUTABLE=%CATALINA_HOME%\bin\catalina.bat"
40
41 rem Check that target executable exists
42 if exist "%EXECUTABLE%" goto okExec
43 echo Cannot find "%EXECUTABLE%"
44 echo This file is needed to run this program
45 goto end
46 :okExec
47
48 rem Get remaining unshifted command line arguments and save them in the
49 set CMD_LINE_ARGS=
50 :setArgs
51 if "%1" == "" goto doneSetArgs
52 set CMD_LINE_ARGS=%CMD_LINE_ARGS% %1
53 shift
54 goto setArgs
55 :doneSetArgs
```

Tomcat configurado con JAVA_HOME en Startup.bat

Una vez hecho esto, podremos abrir Eclipse y configurar este servidor para que lo reconozca el IDE. Una vez abierto, tendremos que ir a la pestaña de servers en la parte inferior de la ventana. Si no aparece, podremos acceder a este desde Windows > Show Perspective > Other y en el cuadro de búsqueda poner "Servers". Aquí se nos mostrará un enlace para configurar rápidamente el servidor.

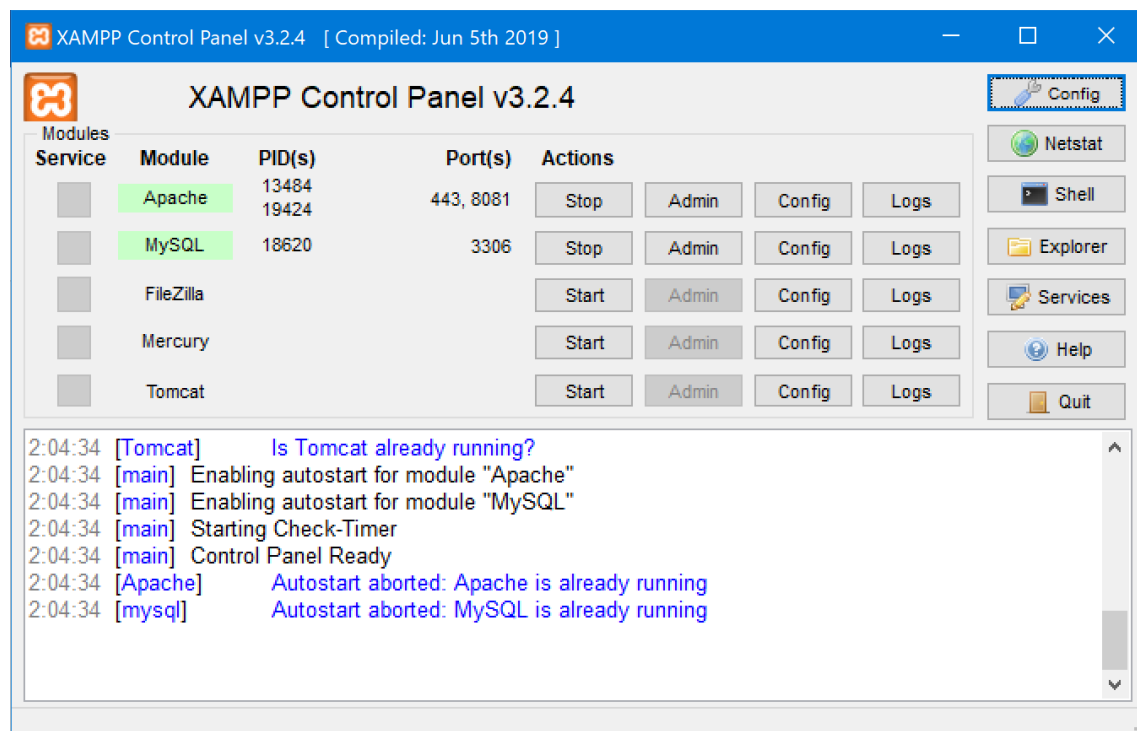


Pestaña servers vacía

Aquí tendremos que seleccionar el servidor, en nuestro caso la versión 9 con el nombre Localhost. Con esto hecho, ya estará configurado para su utilización en el desarrollo de proyectos. Podemos probar a iniciarlo nosotros mismos. Si ya hay una instancia de Tomcat abierta, esta de Eclipse dará error. Por lo tanto, no podremos contar con el startup.bat abierto, a la vez que desarrollamos en Eclipse (no sin pasos previos que no hacen falta aquí).

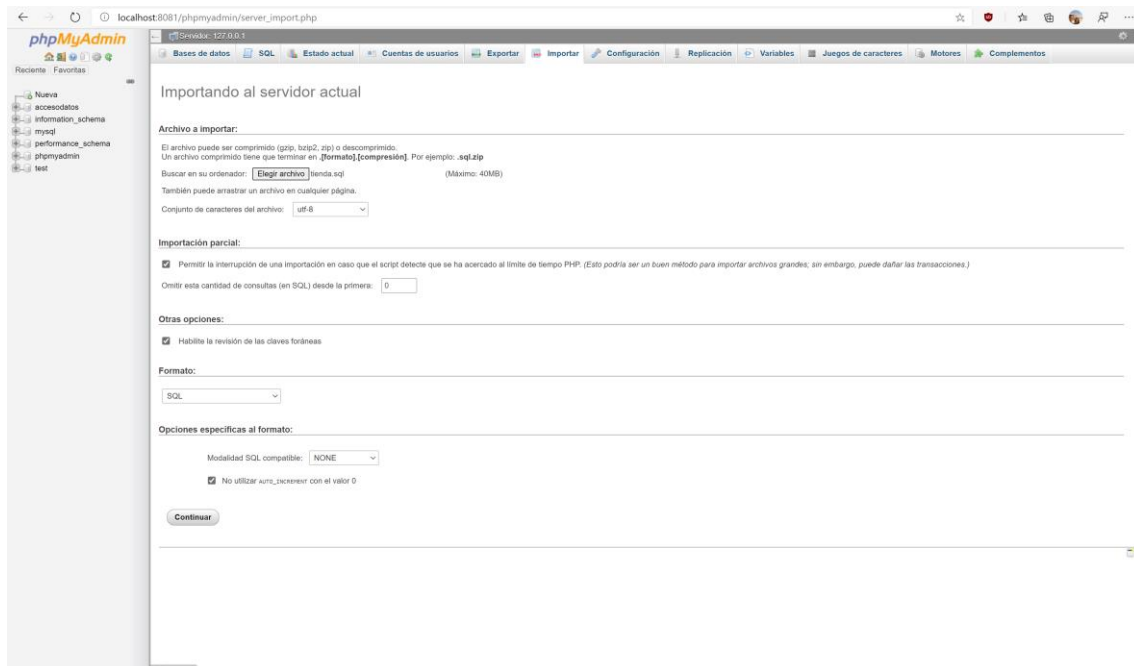
3. Instalación

Para utilizar este proyecto, es necesario que dispongamos de una base de datos MySQL donde se almacenará la información de usuarios, productos y compras realizadas. Para ello, utilizaremos XAMPP que descargaremos desde la página [Apachefriends](http://apachefriends.org). Una vez lo instalemos en la ruta que deseemos, podremos utilizar el panel de control para configurar y conectar Apache (PhpMyAdmin) y MySQL.

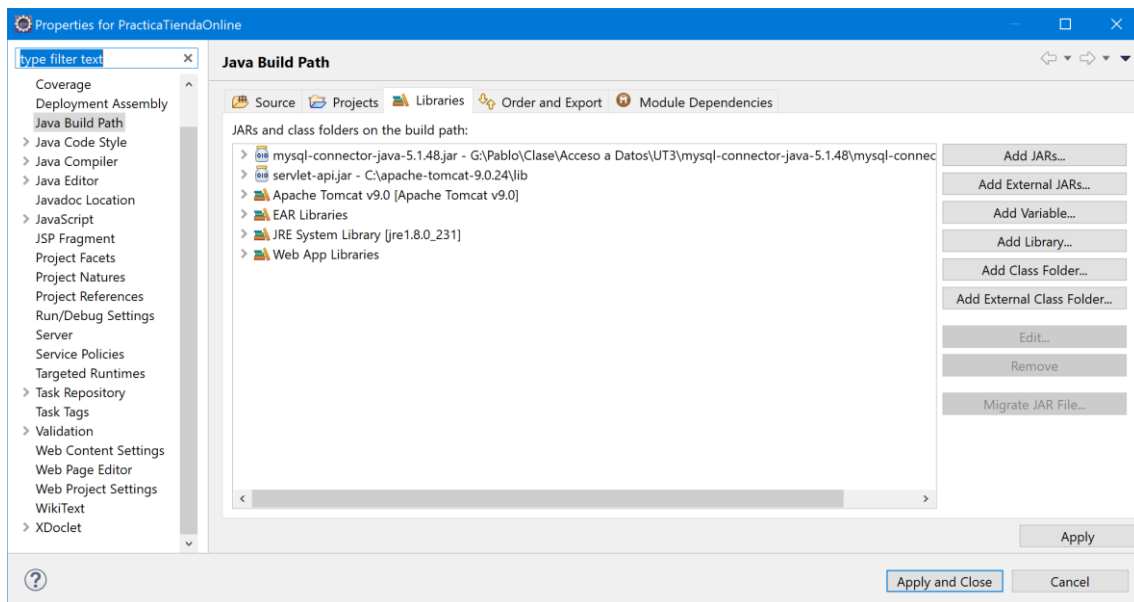


Inicialmente, Apache provocará conflicto con Tomcat ya que ambos intentarán utilizar el puerto 8080. Para ello, tendremos que cambiar uno de los dos a otro puerto (por comodidad se ha elegido el 8081). Esto se puede hacer desde el botón Config y seleccionando la primera opción (httpd.conf). Para acceder a PhpMyAdmin, tendremos que ir a un navegador web y poner en la barra de direcciones: localhost:8081/phpmyadmin.

Dentro del proyecto, podemos encontrar un fichero .sql que nos ofrecerá las sentencias para utilizar dentro de MySQL. Desde la creación de la base de datos hasta la inserción de los datos de prueba que se utilizarán en la aplicación. Para ello, tendremos que acceder a phpmyadmin, pulsar Importar en la barra superior, y seleccionar este fichero .sql.



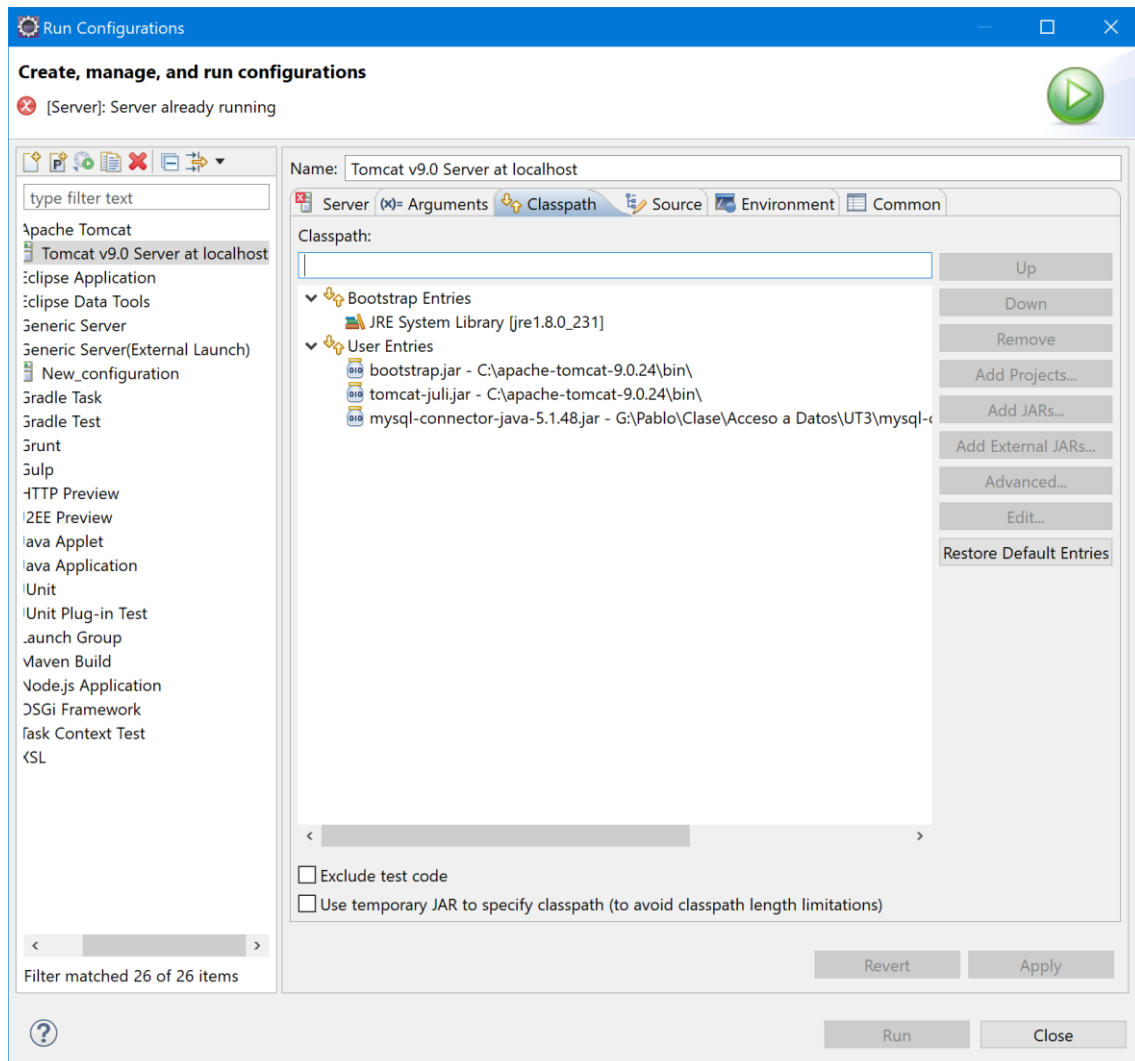
Ahora que tenemos la base de datos configurada y funcionando, solo nos quedará importar el proyecto java dentro de Eclipse. La opción Open Projects from File System nos permitirá seleccionar la ruta, y Eclipse se encargará de todo. Una vez se ha importado, tendremos que ir a las propiedades del proyecto > Java Build Path > Libraries y añadir un Jar externo. Aquí tendremos que seleccionar el archivo servlet-api.jar que encontraremos en la ruta de tomcat dentro de la carpeta lib.



Además, tendremos que añadir el jdbc para MySQL que podremos descargar desde la red, o utilizar el que se ha enviado dentro del propio proyecto. Es muy importante comprobar estas rutas, ya que a veces se desconfiguran y la aplicación dará problemas con los métodos de request y response.

El siguiente paso será configurar el servidor para que este se pueda conectar a la base de datos. Para ello, tendremos que acceder a Run > Run Configurations > Tomcat v9.0 Server (o el

nombre que tenga) y aquí acceder a Classpath. En User Entries veremos dos jar, y será necesario que añadamos el mysql-connector para poder funcionar correctamente.



Como siguiente paso, tendremos que ir al proyecto, acceder a la carpeta META-INF y crear un fichero que se llame context.xml. Este será el encargado de gestionar el pool de conexiones que vamos a crear dentro de la aplicación, así como de asociar el driver jdbc dentro de la aplicación utilizando el Contexto.

```
<Context path="/PracticaTiendaOnline" docBase =
"PracticaTiendaOnline" debug="5" reloadable="true"
crossContext="true">
  <Resource name="jdbc/miDataSource" auth="Container"
type="javax.sql.DataSource"
      driverClassName="com.mysql.jdbc.Driver"
maxActive="100" maxIdle="30" maxWait="10000" username="root"
password=""
      url= "jdbc:mysql://localhost/accesodatos" />
</Context>
```

4. Características base de la Tienda Online

Como parte del proyecto Tienda Online, se busca ofrecer una serie de características que pongan a prueba el uso de JSP, Servlet en Java y la conexión mediante un pool a una base de datos MySQL gestionada por XAMPP. Entre las características generales podemos encontrar:

- Página inicial con dos botones que nos lleven al Registro de un usuario o al Inicio de sesión.
- Registro mediante usuario y contraseña.
- Inicio de sesión con usuario único y contraseña previo registro.
- Selección entre una alta gama de productos al precio más bajo del mercado.
- Añadir productos a un carrito.
- Seleccionar un número concreto de unidades para el producto que deseamos comprar.
- Posibilidad de eliminar los productos que deseemos del carrito.
- Compra de los productos que hayamos seleccionado.

Se trata de una Tienda simple, pero con las características básicas para que cualquier usuario pueda encontrar el producto que desee. Esto además permite la futura adaptación a las tendencias y necesidades del mercado para hacer de nuestra tienda la mejor posible, con una evolución continua.

5. Características extra de la Tienda Online

- Mensaje de bienvenida dentro de Index.jsp agradeciendo la visita y confianza del usuario.
- Título identificativo en Alta/Validación indicando al usuario de donde se encuentra.
- Enlace desde Alta/Validación hacia Validación y Alta respectivamente.
 - Esto nos permite cambiar rápidamente entre ambas vistas sin tener que cambiar nuestra vista del formulario.
- Información sobre la seguridad actual de la contraseña en base a unos criterios sencillos que podremos encontrar a continuación de la lista.
- Centrado de los botones de Index.jsp, formulario de Inicio de sesión/Registro, Selección de producto y visualización de la tabla que representa el carrito del usuario.
- Enlace en la visualización de la tabla para poder volver atrás y seguir eligiendo nuevos productos.
- Añadido mensaje de error en Validación indicando que ha habido un problema con un usuario/contraseña incorrecta.
- Añadido un mensaje de error en Alta indicando que ya existe un usuario con el nombre que se desea registrar.
- Se ha añadido el nombre del usuario que ha iniciado sesión en la selección del producto.
- Si accedemos directamente al Servlet o a Tienda.jsp, se nos redireccionará al Inicio de sesión, ya que desde aquí podremos ir a Registro en caso de no contar con un usuario dentro de la tienda.
- Se ha añadido la tabla Tiquet que contiene el IdCompra, la fecha y hora en la que tiene lugar la compra, el usuario que la realiza y el total que haya pagado.

- Cuando seleccionamos el mismo producto dos veces, en lugar de añadir una nueva fila con el producto de nuevo, se actualizará la cantidad añadiendo los que ya tenía y los nuevos que ha seleccionado.
- Los botones de Alta y Validación estarán desactivados hasta que no se haya introducido al menos un valor tanto en el nombre de usuario como en la contraseña.
 - Será necesario pulsar fuera del campo (o pulsar Tab para que pierda el foco) para que se habilite/deshabilite correctamente el botón para iniciar.
- No se podrá pulsar el eliminar un producto hasta que no se haya seleccionado un elemento de la lista del producto.
- Acceso rápido desde la lista de productos para ver la compra que ya llevamos actualmente.
- La lista de productos que se visualiza por el usuario contiene un título indicando donde se encuentra el usuario.
- Los campos de Contraseña, Nombre de Usuario y Descripción de la tabla están marcados como no nulos.
- Se ha añadido la opción para ver las compras pasadas realizadas por un usuario una vez haya iniciado sesión.
- Se ha creado la clase Invoice para representar la fecha(día), productos, cantidad de cada producto y coste total de cada una de las compras que ha tenido un cliente. Esta clase se utilizará cuando veamos el historial de compras.

Cabe destacar que en Alta, al introducir la contraseña del usuario a registrar y pulsamos fuera del campo, o pierde el foco mediante la tecla tab, nos saldrá un pequeño mensaje informando sobre la seguridad de la contraseña en cuestión. Para utilizar un criterio sencillo, se ha dividido en Poco Segura, Segura y Muy Segura.

Poco Segura: La contraseña dispone de menos de 7 caracteres, independiente de los que sean estos.

Segura: La contraseña dispone de más de 7 caracteres, pero no cuenta con al menos una letra mayúscula, una letra minúscula, un símbolo (!"·\$%&/()=) y un número.

Muy Segura: La contraseña dispone de más de 7 caracteres y si dispone de los cuatro elementos indispensables, es decir, mayúscula, minúscula, símbolo y número.

6. Flujo detallado de la Tienda Online

Lo primer que encontraremos será un formulario que nos dará la bienvenida y nos dará dos opciones: Iniciar sesión dentro de la tienda o registrarnos si no contamos con una cuenta. Cada una de estas funciones nos llevará a un formulario diferente, guardado en el archivo Validación.jsp y Alta.jsp.



En ambos encontraremos la misma información, es decir, un título que nos indica donde estamos, el campo para introducir el usuario, otro igual para la contraseña, el botón de enviar y por último un enlace que nos permitirá saltar entre Validación y Alta.

Una vez hayamos introducido los datos, se nos redirigirá a ServletControlador.java. Este archivo gestiona todas las peticiones que hagamos dentro de los jsp. El método doPost está dedicado al Alta e Inicio de Sesión de los formularios anteriormente mencionado.

Si el usuario se quería registrar, se recogerá el nombre de usuario y la contraseña y se llamará a la clase TiendaDB donde se encuentra el INSERT INTO de la tabla USUARIO. Esta introducirá automáticamente el ID del usuario, por lo que luego (siempre que la consulta haya devuelto 1), se hará una consulta para recuperar el usuario y guardarlo dentro de la sesión de la aplicación.

Registro en la Tienda de DAM

Usuario:

Contraseña:

[Iniciar sesión](#)

*Si se han introducido los datos y sigue deshabilitado, pulsar en la página

Una vez hayamos completado el campo de contraseña, se utilizará el script de comprobarPass dentro de scripts.js que nos mostrará la seguridad asociada a la contraseña según las características indicada en el punto 5 de este documento. Esto se mostrará en cuanto el campo de contraseña pierda el foco, ya sea mediante la tecla tab o pulsando fuera de este cuadro.

Registro en la Tienda de DAM

Usuario:

Contraseña:

Contraseña Muy Segura

*Si se han introducido los datos y sigue deshabilitado, pulsar en la página

Registro en la Tienda de DAM

Usuario:

Contraseña:

Contraseña Segura

*Si se han introducido los datos y sigue deshabilitado, pulsar en la página

En caso de que el nombre de usuario ya exista dentro de la base de datos, es decir, la consulta al SQL dará una excepción, se volverá a Alta.jsp mostrando un mensaje de error indicando que el usuario ya existe. No hay ningún límite de intentos de inicio de sesión por lo que puede repetir cuantas veces quiera.

Registro en la Tienda de DAM

El usuario que se intenta crear ya existe

Usuario:

Contraseña:

[Iniciar sesión](#)

*Si se han introducido los datos y sigue deshabilitado, pulsar en la página

Si el usuario quería iniciar sesión, se recogerá el nombre de usuario y la contraseña y se llamará a la clase TiendaDB donde se encuentra el SELECT de USUARIO para recuperar todos los campos. El método devolverá un usuario que a posteriori se guardará en la sesión de la aplicación.

Inicio de sesión en la Tienda de DAM

Usuario:

Contraseña:

[Crear cuenta](#)

*Si se han introducido los datos y sigue deshabilitado, pulsar en la página

En caso de que el nombre de usuario o la contraseña sean incorrectas, es decir, la consulta al SQL no devolverá nada, se volverá a Validación.jsp mostrando un mensaje de error indicando que el usuario se ha equivocado al introducir los datos. No hay ningún limite de intentos de inicio de sesión por lo que puede repetir cuantas veces quiera.

Inicio de sesión en la Tienda de DAM

El usuario o la contraseña introducida no es correcto.

Usuario:

Contraseña:

[Crear cuenta](#)

*Si se han introducido los datos y sigue deshabilitado, pulsar en la página

Tras un inicio de sesión correcto, ServletControlador.java nos llevará a Tienda.jsp, donde se mostrará un mensaje de bienvenida al usuario que esté en la sesión, y se le mostrará una lista de productos con su precio, y un campo para marcar el número de unidades que desea adquirir. Este conjunto de productos se obtendrá mediante una consulta a la base de datos a través de ServletControlador que utilizará la base de datos.

Bienvenido a la Tienda de DAM **admin**.

Listado productos: Unidades:
 [Ver Carrito](#)

Como vemos, admin se encuentra en color azul, y es que si lo pulsamos podremos acceder al “perfil” del usuario donde podrá ver todas las compras que haya realizado desde que se registró en la tienda online. Se mostrará en un formato tabla donde veremos la fecha (año-mes-día) en la que se realizó la compra, una serie de filas con el producto y el número de unidades de cada producto y el total de la compra en cuestión.

Una vez hayamos visto las compras, contaremos con la posibilidad de volver atrás y seguir comprando nuevos productos, o cerrar sesión, volviendo a Index.jsp.

Historial de compras de admin:

Compra realizada el 2020-01-12	
PRODUCTO	UNIDADES
Producto3	12
Producto0	5
TOTAL COMPRA: 1193.0 euros	

[Comprar productos](#) [Cerrar sesión](#)

Una vez hayamos seleccionado el número de productos que deseemos, podremos pulsar en Cestar para que se guarden los productos. Este botón nos mandará al propio Tienda.jsp con los campos Producto y Unidades en la URL, que se recogerán y se almacenarán dentro de una lista que posteriormente se utilizará para guardar el producto en la base de datos.

Carrito de la compra de admin

PRODUCTOS COMPRADOS			
<input type="checkbox"/>	PRODUCTO	PRECIO (euros)	UNIDADES
<input type="radio"/>	Producto0	305.0euros	5
TOTAL COMPRA: 610.0 euros			

[Seguir comprando](#)

Si por el contrario no nos interesa comprar nada nuevo, si no solo ver el carrito en su estado actual, podremos pulsar en Ver Carrito, que nos permitirá las mismas opciones que al dar a Cestar, pero sin la necesidad de ver un campo vacío.

Aquí tendremos cuatro opciones. Si pulsamos en Seguir comprando, se nos redirigirá a Tienda.jsp sin ningún tipo de parámetro, por lo que nos mostrará la lista de productos como si accediésemos por primera vez para elegir un nuevo producto, o comprar más unidades del propio producto que ya habíamos elegido.

Si hemos comprado un producto que no deseamos, podremos marcarlo desde el radio button a la izquierda de la descripción del producto y se activará el botón Eliminar Producto, que nos recargará Tienda.jsp ya sin este producto en las listas, algo que podremos visualizar pulsando en Ver Carrito.

Carrito de la compra de admin

PRODUCTOS COMPRADOS			
	PRODUCTO	PRECIO (euros)	UNIDADES
<input checked="" type="radio"/>	Producto0	305.0euros	5
TOTAL COMPRA: 610.0 euros			

[Seguir comprando](#)

Si estamos conformes con todos los productos que hay dentro de nuestro carrito, podremos dar a Comprar, que solicitará a ServletControlador que utilice el método addCompra para añadir todos los productos de la lista dentro de la base de datos.

Por último, contamos con la opción de Salir, que anulará la sesión actualmente activa y comenzará el proceso de nuevo.

7. Documentación

Todas las funciones que podemos encontrar dentro del proyecto se encuentran documentadas explicando su funcionamiento, parámetros y retorno en caso de tenerlo. El conjunto de archivos HTML que configuran la documentación de las clases JAVA se encuentra dentro de Utilidades > Documentación.

En el caso de las funciones Javascript, estas se encuentran también documentadas según el estándar de JSDoc, aunque para poder visualizarlo es necesario acceder al fichero scripts.js. Las soluciones encontradas para la formación automática del documento son para Linux y no se ha podido encontrar una adecuada que funcione para Windows, por lo que se ha preferido dejar dentro del fichero.

