Aplicación TPV

Autor: Miguel Aguirre Varela

Curso 2017/2018

Centro de estudios superiores

Contenido

[1. Requisitos iniciales2](#_Toc515380906)

[2. Fase de modelado de datos2](#_Toc515380907)

[3. Fase de modelado de negocio2](#_Toc515380908)

[4. Fase de desarrollo3](#_Toc515380909)

[5. Fase de modelado de negocio definitivo3](#_Toc515380910)

[6. Modelo de datos definitivo4](#_Toc515380911)

[6.1. Modelo relacional5](#_Toc515380912)

[7. Funciones no implementadas6](#_Toc515380913)

[8. Futuras revisiones7](#_Toc515380914)

[9. Instalación7](#_Toc515380915)

[9.1. Instalación del entorno y del modelo de datos8](#_Toc515380916)

[9.1.1. Entorno de desarrollo8](#_Toc515380917)

[9.1.2. Sistema gestor de bases de datos8](#_Toc515380918)

[9.1.3. Programa de administración de la base de datos9](#_Toc515380919)

[9.2. Instalación y compilación de los ficheros fuente9](#_Toc515380920)

[9.2.1. Copia y compilación de fuentes10](#_Toc515380921)

[9.2.2. Creación de la base de datos y ejecución de scripts10](#_Toc515380922)

[9.3. Puesta en marcha12](#_Toc515380923)

[9.3.1. Incluir las librerías externas13](#_Toc515380924)

[10. Glosario14](#_Toc515380925)

# Requisitos iniciales

Se requiere realizar una aplicación TPV con una funcionalidad base que permita persistir y recuperar valores en una base de datos, que calcule los impuestos, subtotales y totales con una interfaz de usuario sencilla, con botones para productos y cálculos, y una experiencia de usuario aceptable.

# Fase de modelado de datos

Serán necesarias, al menos, dos tablas, una tabla en la que se almacenarán los productos y otra en la que se almacenarán las comandas

# Fase de modelado de negocio

Se implementaran clases para mapear las tablas, clases que interactuarán con la base de datos mediante estos objetos y clases que llevarán el control de la vista y las acciones (controladoras)

# Fase de desarrollo

Una vez iniciado el desarrollo se observan necesidades adicionales, tales como la implementación de interfaces y clases abstractas para centralizar el modelo de negocio en la parte común y facilitar la implementación del desarrollo y el mantenimiento futuro. Se ha generado una interfaz común a todos los DAO y una clase abstracta que la implementa, añadiendo en esta última una serie de métodos comunes pero no obligatorios para todas las clases DAO.

Se añaden en la interfaz métodos comunes y obligatorios para acceder a las tablas mediante clave, para obtener tablas completas y para insertar en las mismas. Es necesario sobrecargar este método debido a que en unos casos serán necesarios unos parámetros y en otros casos otros parámetros.

Una vez desarrolladas las clases y la interfaz, se generan las pruebas unitarias para los métodos que implementan y, terminada esta fase, se procede a implementar la clase pantalla de forma paulatina y realizando pruebas funcionales según se va avanzando.

Tras la finalización del desarrollo de la clase pantalla se van mejorando paulatinamente las funcionalidades y modelado estético de la aplicación. Por ejemplo, la existencia de botones de cierre para evitar el marco del sistema operativo. Aunque la ventana se podría enviar a segundo plano y no se está evitando mediante la función setAlwaysOnTop(), al no tener el marco del sistema operativo, no se implementa ese botón lo cual lo hace más complicado.

# Fase de modelado de negocio definitivo

Al término del desarrollo se tiene una aplicación capaz de obtener la lista de productos para generar la botonera desde la cual se podrán ir añadiendo elementos a la comanda mediante pulsaciones de ratón (o táctiles si fuera necesario), capaz de calcular las tasas sobre el sumatorio de la comanda, que es el subtotal, y calcular la suma de ambos resultando así el total. Con respecto a la comanda, se permite añadir productos, tal como ya se ha comentado, y borrarlos de uno en uno o borrar la lista completa. En ambos casos la aplicación de encarga de recalcular el subtotal, las tasas y el total. Una vez se ha logrado la comanda deseada y se desea totalizar, mediante el botón “aceptar” se procede a cerrar la misma informando al usuario de este hecho mediante la ventana de alerta correspondiente, que dispone de botón “sí” para continuar y “no” para volver a la comanda sin realizar ninguna acción. Si la respuesta del usuario es “sí” y la lista de comanda no está vacía, se almacenará en base de datos todo lo relativo a la comanda (véase [Modelo de datos definitivo](#_Modelo_de_datos)) y se procederá a imprimir el ticket de la factura (no implementado al no disponer de impresora pero simulado mediante ventana emergente).

Se implementa un botón que cierra la aplicación (y el proceso del sistema, para ello se utiliza la función close() de la clase System en lugar de setVisible(false) para dejar de consumir recursos una vez finalizado su uso) con lo que se puede evitar el uso del marco que implementa el propio sistema operativo. También se han observado ciertos comportamientos, tales como no poder acceder a la base de datos, los cuales se han controlado mediante una ventana de alerta informando al usuario del motivo del cierre, controlando así algunos de los errores en tiempo de ejecución que se pueden dar.

Mediante el botón “?” se obtiene la información relativa a la versión y al equipo de desarrollo.

# Modelo de datos definitivo

Se establecen las tablas necesarias para almacenar los productos, las comandas (tablas cruzadas mediante clave primaria-foránea) y los usuarios (aunque esta última no se está utilizando - véase [Funciones no implementadas](#_Funciones_no_implementadas)).

La tabla de productos (productos) se compone de los campos:

id: numérico autoincremental - clave

nombre: cadena de texto de 15 caracteres donde se almacena el nombre del producto

precio: valor de doble precisión donde se almacena el precio del producto

La tabla de identificador de la comanda (identifcomanda) se compone de los campos:

ID: numérico autoincremental - clave primaria

fecha: campo tipo fecha donde se almacenará la fecha del sistema en la inserción

usuario: cadena de texto de 20 caracteres donde se almacenará el usuario que hace la comanda

subtotal: valor de doble precisión donde se almacenará el subtotal de la comanda sin impuestos

impuestos: valor de doble precisión donde se almacenará el valor de impuestos de la comanda

total: valor de doble precisión donde se almacenará el total de la comanda completa

tasa: valor de doble precisión donde se almacenará el porcentaje calculado de impuestos

La tabla de productos de la comanda (productoscomanda) se compone de los campos:

comanda\_id: numérico - clave foránea relacionada con la clave primaria de la tabla identifcomanda

producto\_nombre: cadena de texto de 15 caracteres donde se almacenará el nombre del producto

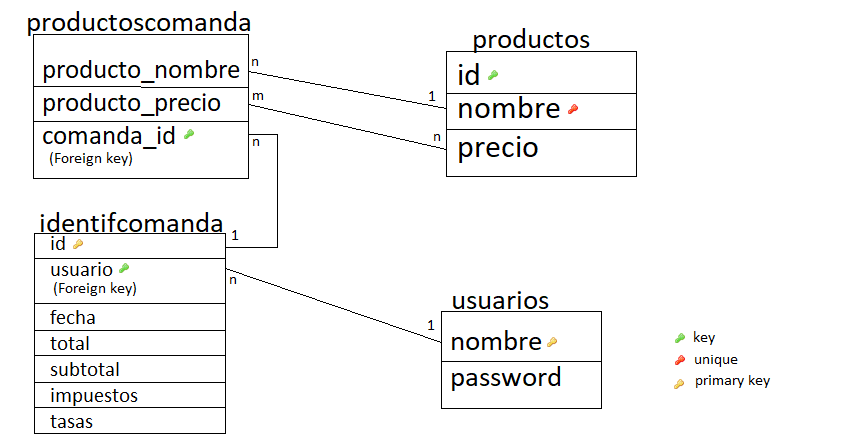
producto\_precio: valor de doble precisión donde se almacenará el precio del producto

La tabla de usuarios (usuarios) se compone de los campos:

nombre: cadena de texto de 20 caracteres donde se almacenan los usuarios de la aplicación

password: cadena de texto de 20 caracteres donde se almacenan las contraseñas de la aplicación

## Modelo relacional



# Funciones no implementadas

No se ha terminado de implementar el sistema de login mediante el cual se podría identificar qué usuario es el que genera la comanda y emite el ticket para realizar auditorías futuras de uso. Este sistema, según se pensó inicialmente, debería permitir el login de un usuario para la manipulación del terminal. Esta idea se descartó por el hecho de que sería poco funcional cambiar de usuario en un entorno de producción. Bajo esta conclusión se pensó en realizar un tablero de botones con una serie de usuarios y que se pudiesen autenticar mediante un PIN de seguridad, pero también se descartó tras evaluar las necesidades de uso, en ocasiones puntuales, de forma frenética viendo que esto generaría retrasos para el cliente y su negocio. Tras esto se evaluó la posibilidad de establecer un simple sistema de identificación no securizado solo para tener un control del usuario que introduce la comanda pero, al ver que no tiene sentido que se guarde registro de este tipo si nada asegura la identidad de quien introduce los datos, se descarta y se guarda siempre como usuario “admin” postergando el desarrollo a una futura revisión del producto evaluando pros y contras del uso de diferentes sistemas según su fiabilidad y rendimiento.

Por esta misma razón no se ha implementado una función mediante la cual se puedan manipular los datos de las tablas, tales como insertar productos nuevos, alterar los existentes y obtener los datos de una fecha, rango de fechas, número o rango de números id de comanda. El sistema sería, tras un login de usuario admin, mediante una ventana con cuatro campos (dos de fecha - desde/hasta - y dos de id de comanda - desde/hasta) obtener, en pantalla o fichero de texto, una relación de comandas para una fecha (introduciendo la fecha desde únicamente), para un rango de fechas (introduciendo ambas), un ID de comanda conocido (introduciendo un id desde únicamente), un rango de comandas (introduciendo ambos) o una combinación de estos. Para la escritura en pantalla se usarían los métodos y clases ya utilizados en esta aplicación, para la escritura sobre fichero de texto, se usarían las clases File, FileWriter y BufferedWriter y sus métodos write() y close(), fundamentalmente. Para la inserción y edición de la tabla de productos, bastaría con implementar una pantalla en la que existan tantos cuadros de texto como productos y precios haya, en los que poder editar dichos campos y, tras unas validaciones de formatos y longitudes, realizar una actualización de la base de datos. También serían necesarios los botones que borren los registros y unos campos vacíos en los que realizar una inserción de un nuevo registro. Como las tasas también deben ser un valor editable, sería apropiado establecerlo en una tabla donde se establezcan ciertos parámetros de configuración y permitir la edición en la misma pantalla en la que se hace el CRUD (Create-Read-Update-Delete) de productos.

Una vez establecidos estos dos sistemas de control de usuarios, el siguiente sistema a implementar sería el de encriptación de las contraseñas de los usuarios mediante codificación de la librería commons-codec-1.5, la cual nos permite codificar mediante SHA y MD5 con la clase DigestUtils y los métodos md5Hex, shaHex, sha256 y sha512.

Para un control más estricto de la base de datos, además de las constraints establecidas mediante las cuales se trata de evitar el borrado de ciertos datos si estos existen en otra tabla, sería prudente preparar disparadores que guarden un registro de los elementos añadidos, borrados y alterados de las tablas junto con el usuario de base de datos que realiza esta acción y la fecha del sistema (funciones sql user() y now(), respectivamente).

# Futuras revisiones

Quedan pendientes de implementación los puntos mencionados en [el apartado anterior](#_Funciones_no_implementadas) y se añaden, como ideas a medio-largo plazo, las siguientes mejoras:

* Implementación del sistema en dispositivos táctiles, tanto fijos como móviles (Tablet, Smartphone, etc.).
* Dar apariencia, imágenes y color a los botones y fondo del terminal, tales como imágenes en miniatura de los productos y un fondo de marca de agua con el logotipo del terminal en la lista de comanda y colores llamativos a los botones principales de acción del TPV (verde para el botón aceptar y rojo para eliminar, por ejemplo).
* Implementar un sistema de ayuda al usuario de ventana emergente para las dudas más frecuentes, como el sistema de eliminación de elementos de la comanda, con un sistema de ayuda para inserción, actualización y borrado de registros de la tabla “productos” que guie al usuario administrador para mantener la coherencia de datos.
* Creación de un sistema de instalación de la aplicación bajo entorno Windows que se encargue de compilar, asociar las librerías y establecer el archivo ejecutable, instalar un SGDB y ejecutar el script sql para definir la base de datos, las tablas y los datos iniciales necesarios.

# Instalación

El proceso de instalación consta de tres fases:

* Instalación del entorno y del modelo de datos
* Instalación y compilación de los ficheros fuente
* Puesta en marcha

## Instalación del entorno y del modelo de datos

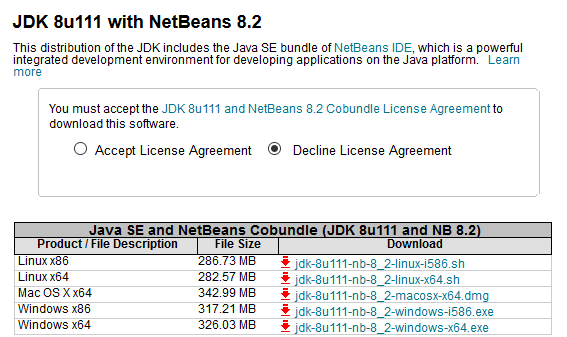
Se deberán instalar las siguientes aplicaciones:

* Un entorno de desarrollo, se recomiendan NetBeans o Eclipse
* Un paquete con un sistema gestor de base de datos MySQL, se recomienda WAMP
* Un programa de administración de bases de datos, se recomienda HeidiSQL

### Entorno de desarrollo

Para la instalación de NetBeans se recomienda utilizar el paquete distribuido por Oracle, el cual es gratuito y que ya contiene el JDK de Java, disponible en su página web:

<http://www.oracle.com/technetwork/es/java/javase/downloads/jdk-netbeans-jsp-3413139-esa.html>



En caso de elegir una instalación del IDE Eclipse, el proceso es similar con la salvedad de que puede ser necesario instalar previamente el JDK de Java (si no está ya instalado). Está disponible en su página web:

<https://www.eclipse.org/downloads/eclipse-packages/>



Se deberá descargar el que corresponda a su sistema operativo y proceder con la instalación que viene por defecto en los asistentes, sea cual sea la instalación elegida.

### Sistema gestor de bases de datos

Se deberá descargar e instalar el paquete WAMP de la página web de la compañía:

<http://www.wampserver.es/>



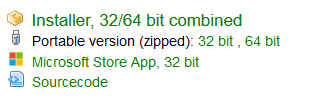
De la misma forma que con el IDE, la instalación se dejará con los parámetros por defecto del instalador.

La particularidad que tiene esta instalación es que se requiere que el puerto 3306 esté libre, pero si no se dispone ya de un sistema gestor de bases de datos MySQL, este problema no existirá. En caso contrario, quizá este paso no sea necesario y sea posible utilizar el ya existente. Tambien se deberá tener en cuenta que es posible que los sistemas FireWall y otros sistemas de protección pidan permisos para este paquete. Será necesario concedérselos para el correcto funcionamiento de la aplicación una vez finalizada la instalación e incluso durante algunos pasos de la propia instalación.

### Programa de administración de la base de datos

El programa recomendado a continuación es útil aunque no estrictamente necesario, ya que el paquete anterior dispone de una aplicación llamada phpMyAdmin al que se puede acceder para gestionar la base de datos mediante interfaz gráfica (aplicación web en entorno local) y con comandos SQL (también dispone de editores donde ejecutar dichas sentencias). No obstante, en este manual se explicará la instalación de la aplicación HeidiSQL ya que es el sistema que se utilizará en posteriores explicaciones. El programa está disponible en la página web:

<https://www.heidisql.com/download.php>



Se recomienda usar el instalador (en lugar de la versión portable) y mantener la configuración que ofrece por defecto.

NOTA:

Si se decidiese, por cualquier circunstancia, utilizar alguna alternativa a las descritas en este manual, es seguro que se pueden replicar los pasos y obtener los mismos resultados consultando la documentación y/o foros de ayuda en línea.

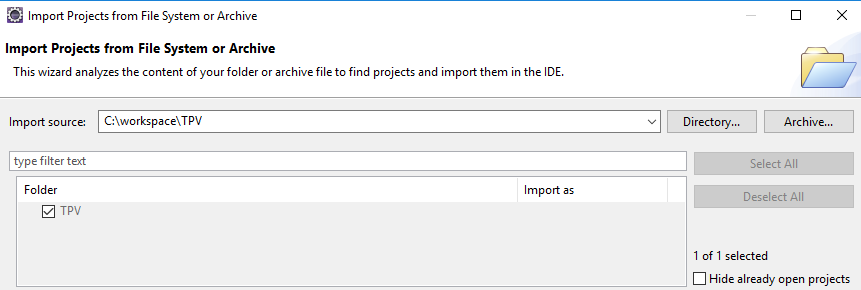
## Instalación y compilación de los ficheros fuente

Para poder ejecutar la aplicación en nuestro sistema, se deberán realizar dos pasos:

* Copia y compilación de los ficheros fuentes (no se dispone de instalador)
* Creación de la base de datos, las tablas que la componen y los datos que deberán contener inicialmente, mediante sentencias SQL.

### Copia y compilación de fuentes

Se deberá establecer un entorno de trabajo (workspace) en el IDE que se haya instalado. Una vez establecido se deberá copiar (manteniendo siempre los ficheros originales libres de modificaciones en la medida de lo posible, por si fuese necesario realizar el proceso de nuevo o instalarlo en otro entorno) la carpeta TPV que se encuentra dentro del paquete de instalación) con los ficheros tal como está. Después habrá que importar el proyecto a partir de ficheros fuente (opción del IDE, en el ejemplo Eclipse).

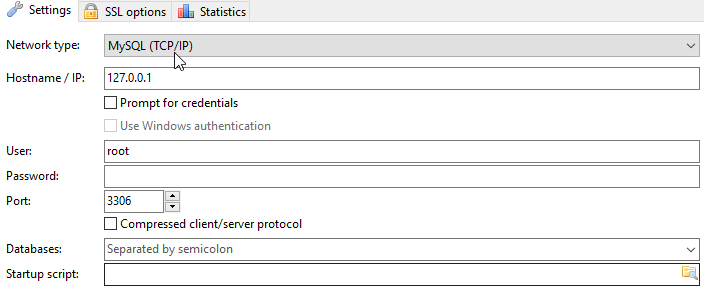


Esto nos creará una librería con los ficheros .java que deberemos compilar. En este momento, aunque tratásemos de ejecutarlo, no nos resultaría posible ya que nos falta la base de datos y algunos pasos de configuración más.

### Creación de la base de datos y ejecución de scripts

En este punto, se deberá proceder con la instalación de la base de datos que será necesaria para nuestra instalación. Será necesario tener en ejecución el sistema WAMP (se ejecuta en segundo plano y es necesario arrancarlo cada vez que se inicie el sistema o, incluso, ordenar una tarea al sistema operativo para que lo ejecute al iniciar el sistema). Una vez arrancado el SGBD, se deberá ejecutar el programa HeidiSQL.

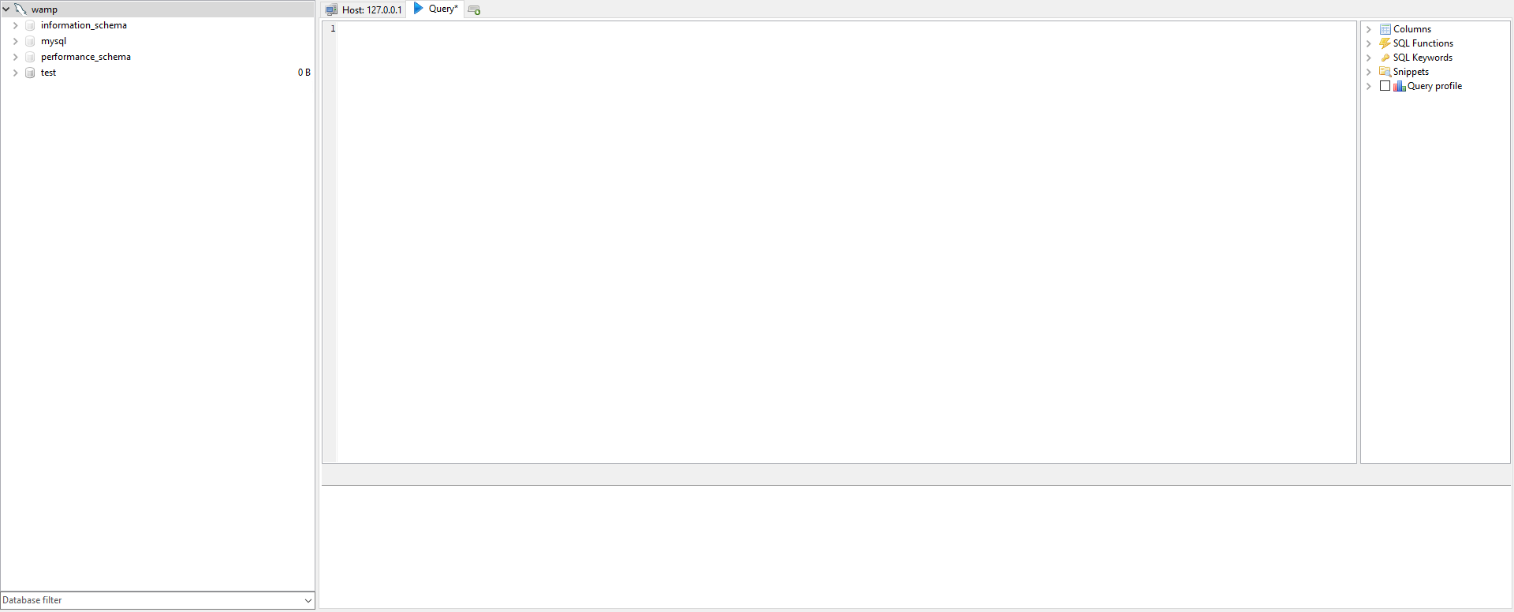
La primera pantalla que encontraremos al ejecutarlo, será similar (según la versión) a la siguiente:



La configuración por defecto del SGBD es la que aparece en la captura:

* Hostname: 127.0.0.1 (también vale “localhost”)
* User: root
* Password: (vacía)
* Port: 3306
* Databases: (vacío)

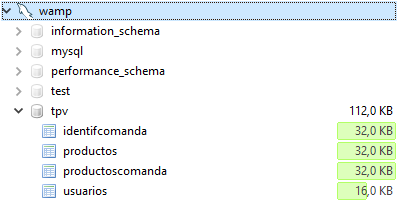
Al aceptar se accede a la pantalla donde se ven las bases de datos y las tablas a las que tiene permiso el usuario (root tiene todos los permisos de acceso y modificación, por lo que se recomienda seguir estrictamente estos pasos para evitar problemas en el futuro. En la pestaña “Query” de la imagen se deberá introducir el texto exacto que se encuentra en el fichero TPV.sql de la carpeta DataBase que se encuentra dentro del paquete de instalación:



Quedará de la siguiente manera:



Solo se deberá pulsar F9 para que se ejecuten los comandos en ese orden y se genere la base de datos con las tablas y los datos necesarios:



No es necesario realizar ninguna acción más en este programa (salvo una reinstalación en un futuro) por lo que ya no es necesario este programa (incluso se puede desinstalar del sistema) para la ejecución del TPV a nivel usuario.

## Puesta en marcha

Como ya disponemos del código fuente y de la base de datos está operativa, se pueden realizar las configuraciones necesarias para la compilación y ejecución del TPV. Los pasos son los siguientes:

* Incluir las librerías externas necesarias para la ejecución
* Configurar los datos en la cadena de conexión que se utilizará desde la aplicación

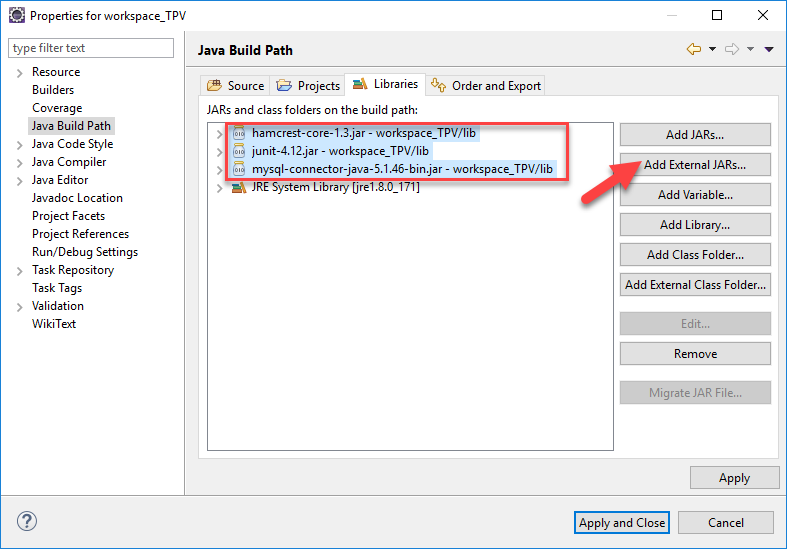
### Incluir las librerías externas

Se deben incluir las librerías desde las propiedades del proyecto, en la opción Java Build Path, pulsando en el botón Add External JARs… y buscando en el sistema de archivos los que vamos a necesitar. Para las pruebas unitarias JUnit se necesitan las librerías hamcrest y junit, pero para la ejecución del terminal solo será necesaria la librería mysql-connector-java. Todas ellas vienen incluidas en la carpeta LibreriasAdicionales, que se encuentra dentro del paquete de instalación, pero es posible conseguirlas de forma online en las direcciones:

Hamcrest: <http://www.java2s.com/Code/Jar/h/Downloadhamcrestcore13jar.htm>

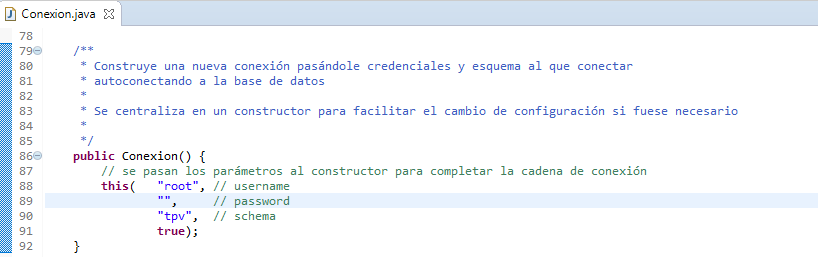
JUnit: <http://book2s.com/java/jar/j/junit/download-junit-4.12.jar.html> (es necesario tener hamcrest)

MySQL: <https://dev.mysql.com/downloads/file/?id=476198> (tras descomprimir, solo se necesita el fichero bin cuya extensión es jar)



Una vez incluidas las librerías (al menos mysql-connector-java-5.1.46-bin.jar) podemos proceder con la configuración de la cadena de conexión:

Si los datos de conexión son los que se asignan por defecto, no se deberá hacer nada. En caso contrario se deberán establecer en el siguiente fichero fuente (Conexión.java):



# Glosario

|  |  |  |  |
| --- | --- | --- | --- |
| Siglas | Significado | Siglas | Significado |
| TPV | Terminal Punto de Venta | DAO | Data Access Object |
| SHA | Secure Hash Algorithm | MD5 | Message-Digest algorithm 5 |
| WAMP | Windows / Apache / MySQL / PHP | IDE | Integrated Development Environment |
| JDK | Java Development Kit | SQL | Structured Query Language |
| SGBD | Sistema Gestor de Base de Datos | ID | IDentificador |