

Gestión de trenes Swing

Índice

1. Condiciones generales	1
2. Introducción	1
3. Modelo de datos	1
3.1. Descripción del modelo de la base de datos relacional	2
3.1.1. Estaciones	2
3.1.2. Líneas	2
3.1.3. Vagones	3
3.1.4. VagonesEnLinea	3
3.1.5. Billetes	3
3.2. Formato de las tablas en MySQL	3
3.3. Creación de tablas y datos de ejemplo	4
4. Funcionalidad mínima de la aplicación	5
5. Funcionalidad opcional de la aplicación	5
6. Recomendaciones de diseño	6
7. Formato de entrega	6
8. Criterios de evaluación	7
9. Apéndice: conexión con la base de datos	7
10. Referencias y material complementario	8
10.1. Swing	8
10.2. MySQL	9

1. Condiciones generales

- Es necesario aprobar esta práctica para aprobar la asignatura.
- La práctica debe ser entregada estrictamente en el plazo estipulado.
- La práctica debe realizarse y entregarse por parejas. Aquellos alumnos que no tengan pareja, o cuya pareja renuncie a trabajar en la práctica, deben hablar con los profesores de la asignatura. Sólo se admitirán entregas individuales en aquellos casos que hayan sido autorizados previamente por los profesores.
- A cada pareja se le proporcionará una base de datos en un servidor MySQL para almacenar los datos a manejar por su aplicación.
- Las entregas deben funcionar en los ordenadores del Depto. de Ingeniería Telemática (sistema operativo Linux) de los laboratorios en que se imparten las clases prácticas, usando las bases de datos proporcionadas.
- Para que la práctica sea evaluada, es imprescindible que sus autores se presenten a la entrevista de evaluación de la misma en la fecha y la hora convenidas. En la medida de lo posible, los profesores intentarán ser flexibles en el establecimiento de citas, aunque el horario preferente será en las clases de prácticas posteriores a la entrega.
- La calificación de esta práctica se conserva, si el alumno lo desea, para la convocatoria de septiembre. Se entenderá que el alumno no desea conservarla si entrega una nueva versión de la práctica en la convocatoria de septiembre, en cuyo caso se evaluará la nueva versión.
- Las prácticas a entregar en la convocatoria de septiembre serán las mismas que en la convocatoria de febrero.

2. Introducción

Se desea programar un sistema de gestión líneas y billetes de tren. Los usuarios de la aplicación serán operarios de la compañía de tren operando en los despachos de billetes (aplicación Swing) y clientes finales (aplicación Web).

La aplicación estará compuesta por tres sistemas:

- Una base de datos relacional, que almacena los datos de líneas de tren y su composición prevista, disponibilidad de plazas, billetes expedidos, etc.
- Un cliente gráfico programado con Swing, que permite a los operarios consultar la disponibilidad de plazas, expedir billetes y obtener listados de billetes vendidos.
- Un cliente Web, que permite a los clientes realizar reservas directamente desde un navegador Web.

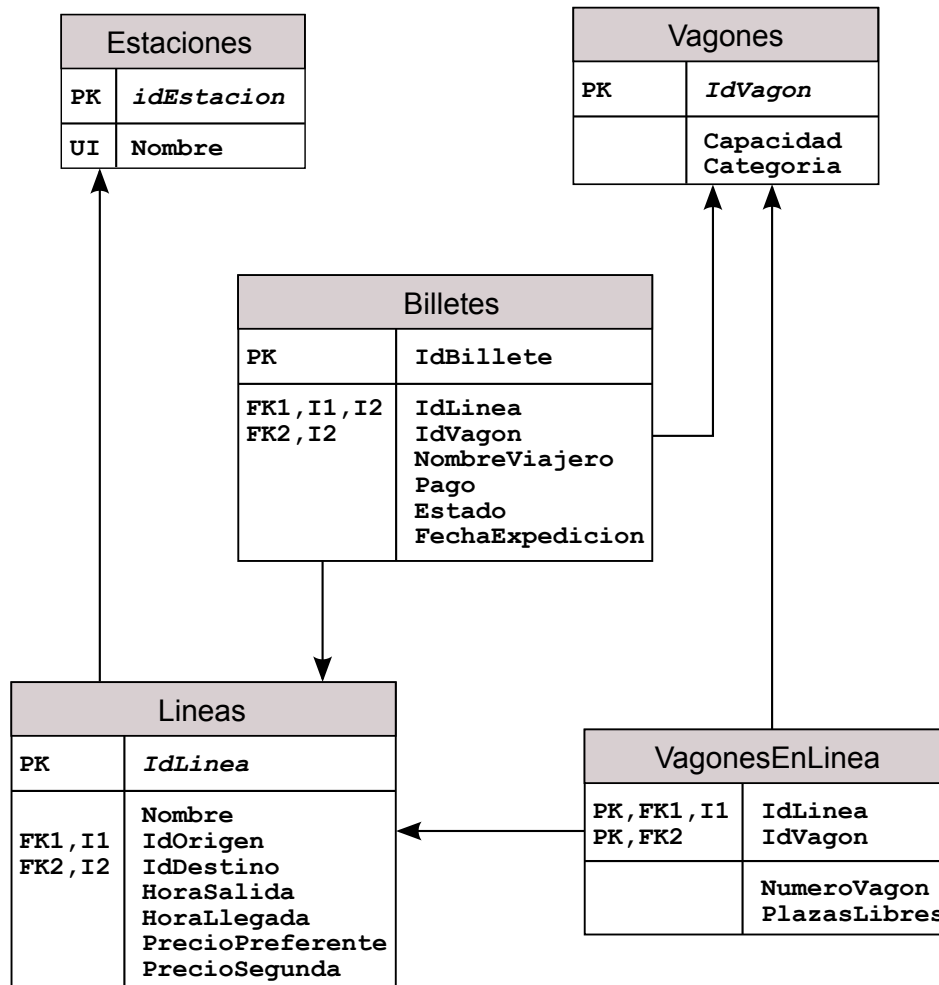
En esta práctica se programará el cliente gráfico con Swing. Al abrir la aplicación, el operario de un despacho de billetes podrá escoger una opción de un menú que ofrece consulta de disponibilidad y reservas, o listados de billetes (véase el apartado de **Funcionalidad mínima**). La aplicación podrá incorporar otras opciones adicionales en el menú, como se explica en el apartado de **Funcionalidad opcional**.

3. Modelo de datos

En este apartado se describe el modelo de datos de la aplicación, y se proporciona el código MySQL que permite crear las tablas para almacenar dichos datos. La aplicación debe respetar estrictamente este modelo de datos.

3.1. Descripción del modelo de la base de datos relacional

La siguiente figura muestra el modelo relacional de la base de datos, cuyas tablas y campos se describen a continuación:



3.1.1. Estaciones

La aplicación utiliza una tabla con todas las estaciones de la compañía. Para cada una se almacena:

- *IdEstacion*: identificador único de la estación, de uso interno en el sistema.
- *Nombre*: nombre de la estación.

3.1.2. Líneas

La aplicación debe ser capaz de gestionar la información referente a los distintos trenes y líneas de la compañía. El concepto de *línea* se refiere a un servicio entre una estación origen y una estación destino en una fecha y hora dadas. Para simplificar el modelo de datos, no se consideran paradas intermedias entre origen y destino. La tabla de líneas incluye los siguientes campos:

- *IdLinea*: identificador único de la línea, de uso interno en el sistema.
- *Nombre*: nombre de la línea, del estilo "Estrella del Norte" o "Rayo del Mediodía".
- *IdOrigen* e *IdDestino*: identificadores de las estaciones origen y destino.
- *HoraSalida* y *HoraLlegada*: fecha y hora previstas de salida y llegada de la línea.
- *PrecioPreferente* y *PrecioSegunda*: precio de los billetes en vagones de categoría preferente y segunda.

3.1.3. Vagones

Esta tabla contiene datos de los vagones de tren de que dispone la compañía:

- **IdVagon:** identificador único del vagón, de uso interno en el sistema.
- **Capacidad:** número de plazas totales del vagón.
- **Categoría:** categoría del vagón, Preferente o Segunda.

3.1.4. VagonesEnLinea

Los vagones que forman parte de la composición de cada línea se almacenan como filas de esta tabla. Consta de los siguientes campos:

- **IdLinea:** identificador de la línea en la que dará servicio el vagón (referencia a la tabla de líneas).
- **IdVagon:** identificador del vagón (referencia a la tabla de vagones).
- **NumeroVagon:** número con que se etiquetará este vagón, para facilitar que los viajeros lo localicen en el tren. Este número aparecerá impreso en el billete del viajero y en las puertas del vagón.
- **PlazasLibres:** número de plazas que todavía están disponibles en este vagón. Se decrementa cada vez que se expide un billete para este vagón y línea.

3.1.5. Billetes

Los billetes expedidos por el sistema se almacenan en esta tabla. Consta de los siguientes campos:

- **IdBillete:** identificador único del billete.
- **IdLinea e IdVagon:** identificadores de línea y vagón asociados a este billete. Hacen referencia a las tablas de líneas y vagones.
- **NombreViajero:** nombre y apellidos del viajero titular del billete.
- **Pago:** modo de pago, efectivo o tarjeta.
- **Estado:** estado del billete, activo o anulado. Siempre se expide con estado activo.
- **FechaExpedicion:** fecha y hora en que se ha expedido el billete.

3.2. Formato de las tablas en MySQL

Las siguientes sentencias definen el formato de las tablas de la base de datos en sintaxis MySQL. Es necesario respetar estrictamente este formato.

```
CREATE TABLE Estaciones (  
    IdEstacion INT(5) NOT NULL auto_increment,  
    Nombre VARCHAR(256),  
    PRIMARY KEY (IdEstacion),  
    UNIQUE INDEX (nombre)  
) ENGINE = INNODB;
```

```
CREATE TABLE Lineas (  
    IdLinea INT(11) NOT NULL auto_increment,  
    Nombre VARCHAR(128) NOT NULL default '',  
    IdOrigen INT(5) NOT NULL,  
    IdDestino INT(5) NOT NULL,
```

```

    HoraSalida DATETIME NOT NULL,
    HoraLlegada DATETIME NOT NULL,
    PrecioPreferente DECIMAL(4,2) NOT NULL,
    PrecioSegunda DECIMAL(4,2) NOT NULL,
    PRIMARY KEY (IdLinea),
    INDEX (IdOrigen),
    INDEX (IdDestino),
    INDEX (HoraSalida),
    FOREIGN KEY (IdOrigen) REFERENCES Estaciones (IdEstacion)
        ON UPDATE CASCADE,
    FOREIGN KEY (IdDestino) REFERENCES Estaciones (IdEstacion)
        ON UPDATE CASCADE
) ENGINE = INNODB;

CREATE TABLE Vagones (
    IdVagon INT(5) NOT NULL auto_increment,
    Capacidad INT(2) NOT NULL,
    Categoria enum('Preferente','Segunda') NOT NULL,
    PRIMARY KEY (IdVagon)
) ENGINE = INNODB;

CREATE TABLE VagonesEnLinea (
    IdLinea INT(11) NOT NULL,
    IdVagon INT(5) NOT NULL,
    NumeroVagon INT(2) NOT NULL,
    PlazasLibres INT(2) NOT NULL,
    PRIMARY KEY (IdVagon, IdLinea),
    INDEX (IdLinea),
    FOREIGN KEY (IdVagon) REFERENCES Vagones (IdVagon)
        ON UPDATE CASCADE,
    FOREIGN KEY (IdLinea) REFERENCES Lineas (IdLinea)
        ON UPDATE CASCADE
) ENGINE = INNODB;

CREATE TABLE Billetes (
    IdBillete INT(11) NOT NULL auto_increment,
    IdLinea int(11) NOT NULL,
    IdVagon int(5) NOT NULL,
    NombreViajero VARCHAR(256) NOT NULL,
    Pago enum('efectivo', 'tarjeta') NOT NULL default 'efectivo',
    Estado enum('activo', 'anulado') NOT NULL default 'activo',
    FechaExpedición DATETIME NOT NULL,
    PRIMARY KEY (IdBillete),
    INDEX (IdLinea),
    INDEX (IdLinea, IdVagon),
    FOREIGN KEY (IdVagon) REFERENCES Vagones (IdVagon)
        ON UPDATE CASCADE,
    FOREIGN KEY (IdLinea) REFERENCES Lineas (IdLinea)
        ON UPDATE CASCADE
) ENGINE = INNODB;

```

3.3. Creación de tablas y datos de ejemplo

En este apartado se proporciona un script MySQL que elimina las tablas (si existen previamente), las crea de nuevo, e inserta algunos datos de ejemplo. Puede usarse para crear las tablas por primera vez, así como para restaurar las tablas a un estado inicial.

Es recomendable añadir a este script sentencias que carguen otros datos iniciales que puedan ser útiles para depurar la aplicación. Descarga [el script MySQL que elimina y crea las tablas](#) de la base de datos. Se puede utilizar este fichero desde el cliente de `mysql` (véase [Conexión con la base de datos](#)) para ejecutar las sentencias SQL almacenadas:

```
mysql> source elimina_y_crea_tablas.sql
```

Esta definición de las tablas permite mantener la integridad referencial de la base de datos, ya que contiene las sentencias SQL requeridas para establecer las relaciones entre ellas. Para más información, consúltese el [Manual de Referencia de MySQL: Capítulo 15: El motor de almacenamiento InnoDB](#).

4. Funcionalidad mínima de la aplicación

El objetivo de esta práctica es el desarrollo de una aplicación gráfica con Swing que permita realizar varias operaciones en el sistema de gestión de reservas de tren. La aplicación debe proporcionar, como mínimo, la siguiente funcionalidad:

1. **Consulta de disponibilidad / expedición de billetes:** la aplicación debe mostrar información acerca de la disponibilidad de plazas en cada línea, vagón por vagón. En aquellos vagones en que haya plazas libres, debe dar la oportunidad de expedir un billete.

La interfaz gráfica ofrecerá distintos componentes gráficos para obtener los datos de estación origen, estación destino, fecha, preferente o segunda, etc. Obtendrá la información correspondiente de la base de datos y se la presentará al usuario para que éste pueda escoger entre las opciones de acuerdo con su criterio de búsqueda. En caso de que el usuario quiera expedir billetes en una línea/vagón concretos, la aplicación debe recoger la información adicional necesaria para el billete (nombre del viajero y modo de pago) y realizar la reserva.

2. **Listado de billetes:** la aplicación debe proporcionar al usuario un listado de todos los billetes vendidos de un tren o de un vagón. Se mostrará este listado en una tabla.

La interfaz gráfica ofrecerá distintos componentes gráficos para obtener los datos de estación origen, estación destino, fecha, etc. Una vez que el usuario selecciona una línea o un vagón concreto, en una determinada fecha, el sistema debe ofrecer un listado de los billetes reservados hasta el momento para ese tren.

La aplicación desarrollada, además de proporcionar la funcionalidad mínima que se pide, debe cumplir los siguientes requisitos:

1. La aplicación debe tener, en su ventana principal, una barra de menú con las entradas necesarias para llevar a cabo las operaciones anteriores, así como cerrar la aplicación. Adicionalmente, este menú puede contener cualquier otra opción que se estime oportuna.
2. El listado de billetes debe ser ordenable por cualquiera de sus columnas.

5. Funcionalidad opcional de la aplicación

Adicionalmente, se propone la realización de tareas opcionales, que serán tenidas en cuenta en la evaluación, pero no son necesarias para obtener el aprobado en la práctica.

- **Mostrar gráficamente el estado de un tren:** dada una línea, se mostrará un dibujo en pantalla que represente el tren con sus vagones, en que se debe ver de forma clara el número de plazas libres disponibles en cada uno. Por ejemplo, se pueden representar en color rojo los vagones sin plazas libres y en color verde el resto. Aunque no es imprescindible, se valorará positivamente que pinchando sobre un vagón con plazas libres se active un diálogo para expedir un billete para el mismo. Para realizar esta función, pueden ser útiles el [apartado "Performing Custom Painting" del manual de Swing](#) y el [manual "2D Graphics"](#)
- **Filtrado en la tabla de billetes:** esta función debe permitir filtrar los billetes mostrados en la tabla que proporciona el listado de billetes, esto es, que sólo se muestren aquellos que cumplan un determinado requisito. El filtrado debe poder hacerse, como mínimo, por vagón (mostrar sólo los billetes de un vagón dado) y por medio de pago (mostrar sólo los billetes correspondientes a un medio de pago concreto). La solución debe implementarse utilizando las funciones que Swing proporciona para ello (véase el apartado relativo a `JTable` en el manual de Swing).

- **Exportar billetes a formato Excel:** esta opción permite exportar los billetes expedidos para una línea dada en formato compatible con hojas de cálculo como Excel, Gnumeric u OpenOffice.org. Para ello, se exportarán en un fichero en que cada línea contiene los campos de una solicitud separados por comas.
- **Representación gráfica de estadísticas:** se mostrarán de forma gráfica estadísticas acerca de los billetes emitidos. Por ejemplo, se puede mostrar una gráfica tipo *tarta* en que se indique el porcentaje de billetes se han pagado con tarjeta y en efectivo, o la ocupación media de los trenes, etc. Para implementar esta función se recomienda usar la biblioteca de código **JFreeChart**.

El objetivo de estas funciones adicionales es que el alumno explote su capacidad para resolver los problemas que se le planteen de forma autónoma. Por ello, la ayuda de los profesores con respecto a estas funciones será más limitada que con respecto a las funciones mínimas.

Los alumnos pueden proponer funciones adicionales que no se detallan en este enunciado, pero es recomendable consultar a los profesores acerca de la posible valoración de las mismas antes de implementarlas.

6. Recomendaciones de diseño

Dado que el modelo de datos de las dos prácticas globales es el mismo, si se diseña adecuadamente la aplicación, una parte del código de esta práctica puede ser reutilizada para la siguiente práctica global. Por ello, se proponen las siguientes recomendaciones:

1. En general, conviene mantener lo más desacoplado posible el código que realice la presentación en la interfaz gráfica del código que represente los datos, los procese y se comunique con la base de datos. El código para estas últimas tareas podrá ser reutilizado en la siguiente práctica global con mayor facilidad.
2. Es recomendable programar una clase bean para cada entidad lógica cuyos datos se almacenen en la base de datos. Cada campo de la base de datos será, en principio, un atributo de esta clase, y se proporcionarán métodos `get` y `set` para acceder a cada atributo. Por ejemplo, en una aplicación que almacene en la base de datos clientes y facturas, los clientes se representarían en una clase `Cliente` (con atributos `identificador`, `NIF`, `nombre`, `ciudad`, etc.) y las facturas en una clase `Factura` (con atributos `identificador`, `cliente`, `fecha`, `importe`, etc.)
3. Debes crear una o más clases que se encarguen de gestionar la conexión con la base de datos y realizar consultas. Estas clases pueden proporcionar una API de alto nivel al resto de la aplicación, con métodos adecuados para las consultas habituales. Por ejemplo, en una aplicación que maneje clientes y facturas, se podría programar un método que devuelva una factura dado su `identificador`, otro que devuelva todas las facturas asociadas a un cliente concreto, etc. Los parámetros y valores de retorno de estos métodos pueden ser, en la medida de lo posible, los objetos tipo *bean* propuestos en el punto anterior. Por ejemplo: `Factura[] obtenerFacturas(Cliente cliente)`.
4. Dado que las clases de una aplicación Web deben estar organizadas forzosamente en paquetes, es recomendable que en esta práctica hagas lo mismo, y de esta forma no será necesario que renombres ninguna clase posteriormente para poder reutilizarla. Se explicará en clases prácticas cómo organizar las clases en paquetes. También dispones de información acerca de paquetes en Java en el apartado sobre paquetes en la documentación de Java.

7. Formato de entrega

La práctica se entregará a través del sistema Web que se habilitará al efecto, cuya URL se publicará en la página principal de la Web de la asignatura.

Se entregará **un único fichero, comprimido con JAR**, que contenga todo aquello que sea necesario para compilar y ejecutar la práctica:

- Código fuente de la aplicación.
- Imágenes y ficheros adicionales que necesite la aplicación.

Para reducir el tamaño de la entrega, este fichero no debe contener datos innecesarios como los siguientes:

- Ficheros compilados .class.
- El fichero JAR del conector de JDBC, si se usa el recomendado en este enunciado.
- Ficheros de copia de seguridad generados por algunos editores (por ejemplo, *.bak, *~, #*, etc.).

Independientemente de que se haya desarrollado la aplicación en un entorno integrado como Eclipse o NetBeans, los autores de la práctica deben ser capaces, el día de la entrevista de evaluación, de compilar y ejecutar su aplicación desde línea de comandos. Aquellos que no sepan hacerlo pueden preguntar a los profesores en las prácticas o en tutorías, antes de vencer el plazo de entrega.

8. Criterios de evaluación

La calificación obtenida en esta práctica estará entre 0 y 10, y supondrá un 25% de la calificación final en la asignatura. Es necesario aprobar esta práctica (obtener al menos un 5) para aprobar la asignatura.

Parte de la evaluación se basará en una entrevista personal de los profesores con los autores de la práctica, que se realizará con cita previa en los horarios de clases prácticas. Los dos miembros de la pareja de prácticas deben demostrar conocimientos suficientes de la práctica que hayan entregado.

Además de que se implemente correctamente la funcionalidad obligatoria, se tendrán en cuenta los siguientes aspectos en la evaluación:

- Funcionalidad opcional implementada, tanto la propuesta en este enunciado como la desarrollada a iniciativa de los alumnos.
- Calidad y elegancia del diseño y código de la aplicación. Uso adecuado de la orientación a objetos. Gestión de errores adecuada.
- Usabilidad de la aplicación.

Para obtener una calificación de 5 o superior es necesario implementar la funcionalidad mínima expuesta en este enunciado. Por otra parte, un programa que implemente sólo la funcionalidad mínima puede optar hasta una calificación máxima de en torno a 8 teniendo en cuenta el resto de aspectos a valorar. El resto de la calificación, hasta 10, puede obtenerse implementando funcionalidad opcional.

9. Apéndice: conexión con la base de datos

Para el desarrollo de esta práctica, debe utilizarse el gestor de bases de datos MySQL instalado en `mysql.lab.it.uc3m.es`.

1. Cada grupo dispondrá de un nombre de usuario y clave para acceder al gestor, desde cualquier máquina del dominio `lab.it.uc3m.es`.
2. Cada grupo dispondrá de una base de datos, inicialmente vacía, donde debe crear sus tablas e introducir datos.
3. Con el fin de facilitar la corrección de la práctica, en la fecha de vencimiento del plazo de entrega la base de datos de cada grupo debe contener suficientes datos para realizar pruebas.
4. Para conectarte al gestor y realizar tareas de administración sobre la base de datos (creación de tablas, introducción de datos, etc.) se puede utilizar el cliente MySQL. Debes invocarlo desde un terminal de línea de comandos con los siguientes parámetros:

```
mysql -h mysql.lab.it.uc3m.es -u usuario -D database -p
```

El cliente ofrece un entorno interactivo para introducir comandos. Es recomendable consultar el [manual de MySQL](#), especialmente [este apartado introductorio](#).

5. Para establecer una conexión con la base de datos desde una aplicación java, es necesario utilizar un conector JDBC para MySQL. Puedes descargar y guardar en tu cuenta [el conector MySQL Connector/J 5.1](#), descomprimirlo, tomar el fichero JAR que hay en su interior y añadirlo en la variable de entorno CLASSPATH. Para realizar la conexión, se puede reutilizar el código de la clase de prueba siguiente:

```
import java.sql.*;

public class PruebaJDBC {
    public static void main(String[] args) {
        Connection conn = null;

        try {
            conn =
                DriverManager.getConnection("jdbc:mysql://mysql.lab.it.uc3m.es/10_myblaptel_40?" +
                    "user=XXXXXXXX&password=YYYYYYY");
        } catch (SQLException ex) {
            // handle any errors
            System.out.println("SQLException: " + ex.getMessage());
            System.out.println("SQLState: " + ex.getSQLState());
            System.out.println("VendorError: " + ex.getErrorCode());
        }

        if (conn != null) {
            System.out.println("Conexión establecida");
            try {
                conn.close();
            } catch (SQLException ex) {
                // handle any errors
                System.out.println("SQLException: " + ex.getMessage());
                System.out.println("SQLState: " + ex.getSQLState());
                System.out.println("VendorError: " + ex.getErrorCode());
            }
        }
    }
}
```

En el código anterior hay que reemplazar XXXXXXXXX e YYYYYYYY por el nombre de usuario y contraseña respectivamente.

6. El gestor bloquea las conexiones que no procedan de este dominio, por lo que es recomendable ejecutar la aplicación Swing en máquinas del mismo. Si alguien desea probar la aplicación Swing desde un ordenador en otro entorno, se proponen las siguientes alternativas:
 - Instalar un gestor MySQL en dicho entorno.
 - Ejecutar la aplicación Swing en una máquina del laboratorio, a través de una conexión SSH exportando el display (requiere disponer de un servidor X en el ordenador remoto).
 - Ejecutar la aplicación Swing localmente en el ordenador, utilizando un túnel SSH a través de una máquina del laboratorio para la conexión con la base de datos.

Independientemente de lo anterior, la aplicación entregada por los alumnos debe funcionar necesariamente en las máquinas del laboratorio conectándose al gestor de bases de datos proporcionado.

10. Referencias y material complementario

10.1. Swing

- [Documentación de la API de Java \(JDK 1.5\)](#)
- [Documentación de la API de Java \(JDK 6\)](#)
- [The Swing Tutorial](#)

- [The Swing Tutorial](#) (traducción al castellano, posiblemente algo desactualizada)
- [User Interfaces that Swing: A Quick Start Guide](#)
- [Thinking in Java](#); primera y segunda ediciones disponibles [gratuitamente en formato digital](#).

10.2. MySQL

- [Manual de Referencia de MySQL](#)
 - [MySQL Connector/J 5.1](#)
 - [JDBC Database Access](#)
-