PRÁCTICA 4 de JAVA (17/10/2022):

Bases de datos

Fecha tope entrega	Miércoles, 26 de octubre.	**Fecha tope defensa**	Miércoles, 2 de noviembre.
Tipo	Individual	**Entrega**	Colaborador en Github

1.- Desarrolla una base datos libre sobre Apache Derby integrado en el IDE Netbeans 8.2.

1.1.- Debe ser **diferente a la del ejemplo y tener el visto bueno del profesor.**

1.2- Crea dos tablas con una relación 1→N, como en el ejemplo:

1.3.- Tipos de datos obligatorios: int, float, GregorianCalendar, String.

1.4.- Cliente tendrá usuario y contraseña.

1.5.- Habrá una imagen la tabla cliente.

1.6.- Existirán PK y FK con el formato “cliNumero”: “cli” son las 3 primeras letras de la tabla donde es PK (de “ cliente”) y “número” de el nombre del campo de dicha tabla (“Producto”).

1.7.- En la clase madre (Cliente) habrá un **campo calculado** obtenido de operaciones matemáticas sobre la clase hija (cuenta). Por ejemplo el campo cliente.saldoTotal equivale a la suma de todos los campos cuenta.saldo. También puede ser la media, máximo, mínimo, etc.

2.- Basándote en MVC desarrolla una aplicación en Java. Cada capa tendrá su propio paquete en el proyecto.

Vista

3.- Se basará en clases con un JFrame y varios JPanel.

4.- Menú contendrá las opciones: Conexión (abrir y cerrar), visualizar (detalle y resumen) y Acerca de.

4.1.- Los elementos estarán activados o desactivados según la lógica de la aplicación.

5.- Funcionamiento: tras elegir en el menú “Conexión / abrir” se pedirá un nombre de usuario (Cliente) y su contraseña.

5.1.- Si la conexión ha tenido éxito, activará el resto de menús.

5.2.- La conexión a la base de datos es transparente al usuario y será automática.

6.- En el panel **visualizar detalle:**

6.1.- Mostrará uno a uno los registros de la segunda tabla (cuenta) asociados al elemento de la primera tabla (cliente) validado.

6.2.- Se podrá **modificar** el campo tipo **fecha** usando un **DatePicker.**

6.3.- También se podrá modificar el campo para los cálculos (cuenta.saldo).

6.3.- Tendrá un botón para **guardar** datos, que actualizará solo la cuenta del cliente visualizado, en la base de datos.

6.4.- Se podrá **avanzar** y **retroceder** entre los clientes directamente en la BDs.

7.- Panel visualizar resumen:

7.1.- Se mostrarán todos los datos de la primera tabla (cliente), incluida la **imagen**.

7.2.- En un JTable o JList se mostraran los campos más descriptivos de la segunda tabla y el campo base para los cálculos (cuenta.numero y cuenta.saldo)

7.3.- Tendrá un botón “Calcular”: tras pulsarse se mostrará en una ventana emergente y después actualizará el valor **real del campo calculado** según la BDs. En el ejemplo: el saldoTotal del cliente visualizado, inicialmente tendrá el valor guardado en la BDs., que no tiene por que ser el correcto.

8.- **Acerca de**: Ventana informativa personalizada de la aplicación (autores, fechas, versión, etc.). Debe realizarse con JDialog **modal** (solo se permitirá pulsar el “botón de aceptar”).

9.- Aparecerán **ventanas informativas prediseñadas** (JoptionPane).

10.- **Modelo**: se deben crear **una clase por tabla** de la BDs. con sus métodos get y set, pero **no** los métodos para guardar y modificar en la BDs. Estos deben estar en las clases del nivel controlador.

11.- **Controlador** (lo más importante a resolver en esta práctica): se deben crear 4 clases que gestionen las operaciones sobre la base de datos:

11.1.- Una orientada a la parte común: abrir, cerrar y gestionar la conexión. Es la única que puede ser estática. Sus métodos deberán devolver, entre otros datos, dos Statement, uno para cada uno de las siguientes clases:

11.2.- Una orientada a ver al JPanel “visualizar detalle”. El Statement será sensible a los cambios de la Bds. Y sus métodos permitirán:

11.2.1.- Controlar el avance, retroceso, inicio, fin, etc. sobre el ResultSet mediante modos. Así como guardar y/o modificar un registro cliente.

11.2.2.- Devolverán objetos de la clase de la segunda tabla (cliente) directamente.

11.3.- Y otra orientada al JPanel “visualizar resumen”, para rellenar el Jtable o JList: Statement y ResultSet simples. Sus métodos se encargarán de:

11.3.1.- Intercambiar una **collection** con objetos de Cuenta.

11.4.- Y otra orientada a la primera tabla (cliente): Statement y ResultSet simples. Sus métodos se encargarán:

11.4.1.- Intercambiar y actualizar objetos de Cliente.

12.- Mejoras: Utilizar Dockers con mysql. Sistema de errores centralizado.