Juan Borrás Sada

Memoria Gestor de Trabajadores

Para empezar, voy a explicar la estructura del proyecto:

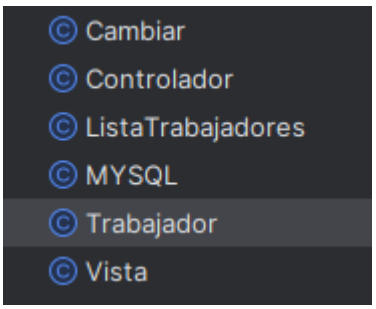
Como bien hemos visto en clase, los proyectos con JavaFX siguen la estructura de "Vista-Controlador", así que vamos a tener estas dos clases.

Usaremos una clase llamada MYSQL en la que vamos a realizar tanto la conexión como los cambios en la base de datos.

Luego, para crear objetos de clase Trabajador, tenemos la propia clase Trabajador.

También tenemos una clase ListaTrabajadores, para poder acceder a una lista de objetos Trabajador, de manera más cómoda y más ordenada.

Por último, para la pestaña de realizar cambios en la base de datos, tenemos la clase "Cambiar".



Después, lo primero que voy a hacer es crear las dos pantallas de la interfaz de la aplicación. Las dos están guardadas en la carpeta "resources". También creé la clase trabajador, para empezar a trabajar después en la lógica del controlador.

Pestaña "Añadir Empleado":

Después de diseñar y asignar todos los ID de cada etiqueta para tener organizado todo, paso a empezar la pantalla de añadir empleado.

Toda la lógica de la aplicación que no tenga que ver con el acceso a la base de datos, está creada en la clase Controlador.

Primero hice el método *"añadirEmpleado*" que se encarga de crear un trabajador con los datos que el usuario haya introducido en la interfaz anteriormente. simplemente linkeando las el id de las etiquetas como atributos de nuestra clase Controlador, podemos acceder a los datos que se han introducido y usarlos para crear los datos del trabajador a introducir.

Para que al hacer clic en el botón añadir, se añade un trabajador nuevo al Arraylist de trabajadores y a la base de datos, lo conseguimos con la etiqueta"ONACTION" de nuestro FXML que hace referencia al botón.

Cabe decir también que es importante a la hora de crear los métodos, fijarse bien en poner @FXML y poner la misma etiqueta que en el fxml. Me pasé bastante tiempo atascado por no seguir esta norma.

Pestaña “Consultar”:

public void LeerTrabajadores()
public Trabajador parsearLinea(String trabajadorLinea)

Primero creé dos métodos que se encargan de leer el archivo “Trabajadores.txt” (con el método *“LeerTrabajadores”*) y transforme cada línea del archivo en un objeto trabajador (con el método *“parsearLinea”*). No tenía muy claro como atacar este problema así que me serví bastante del proyecto que nos mandasteis de los festivales.

public void clickrefrescar()

Luego me dediqué a crear el método para que salgan todos los trabajadores en la tabla de la izquierda de la interfaz. Lo hice iterando mi ArrayList de trabajadores, y añadiendo estos a un ListView. nos sirve para poder mostrar en la interfaz la lista de trabajadores (Desconozco si se puede hacer directamente con el propio ArrayList).

Clase MYSQL:

Para hacer la conexión, probé de varias maneras diferentes. Al final, entre varios compañeros que teníamos el mismo problema, tanto para este trabajo como para el trabajo grupal, encontramos varios tutoriales que hacían referencia a un mismo tipo de estructura bastante flexible y que parecía dar pocos problemas para implementar en los proyectos.

Para usar esta estructura, solo había que implementar en el proyecto la librería “mysql-connector” e implementar una clase con la conexión.

La verdad es que fué un quebradero de cabeza aprender a usar bien esta clase (Ni siquiera después de acabar el trabajo sé corregir todos los errores que tiene)

Esta clase la uso para importar, exportar, cambiar y borrar datos de mi base de datos en base a lo que hagamos en nuestra aplicación.

Después de todo lo aprendido en este año, me es mucho más fácil apoyarse en los objetos creados en mi propia aplicación, que en los datos de mi base de datos. por eso decidí crear una clase para la propia conexión y trabajar con los propios objetos

y estructuras de java. Al hacer los cambios y cerrar la aplicación, esta clase hará todos los cambios oportunos en la base de datos.

Clase Cambiar:

Esta clase la uso para que, cuando se selecciona un Trabajador y se le da al botón Editar, se acceda a una nueva pestaña la cual te pide unos datos. Estos datos se modifican en la lista de trabajadores y se sobrescriben.

El cambio de escena para llegar a la pantalla de Cambiar, se hace en la clase controlador.

también se hace la función SQL para modificar los datos en la base de datos.

Nuevo Nombre:

Nuevo Puesto:

Nuevo Salario:

Cancelar

Editar

Botón Eliminar:

La acción de eliminar la podemos separar en dos partes.

public void eliminarTrabajador()

Primero, hacemos un método que borre el trabajador que se ha seleccionado con el cursor, del objeto LISTVIEW. también, si no hay nada seleccionado, lanzamos un error.

Si el usuario confirma la acción, eliminamos al trabajador del listview de trabajadores.

private void confirmarEliminacion()

Luego, creamos otro método que lance la escena de la pantalla de confirmación.

Para terminar:

He hecho las conexiones entre el controlador y la clase mysql, de tal forma que al leer el archivo trabajadores.txt también se hagan los imports a la base de datos.

He dejado un script en el proyecto *(“BBDDtrabajadores.sql”)* que contiene la estructura de las tablas, para luego hacer los inserts con las sentencias que se hacen en la clase MYSQL de la aplicación.