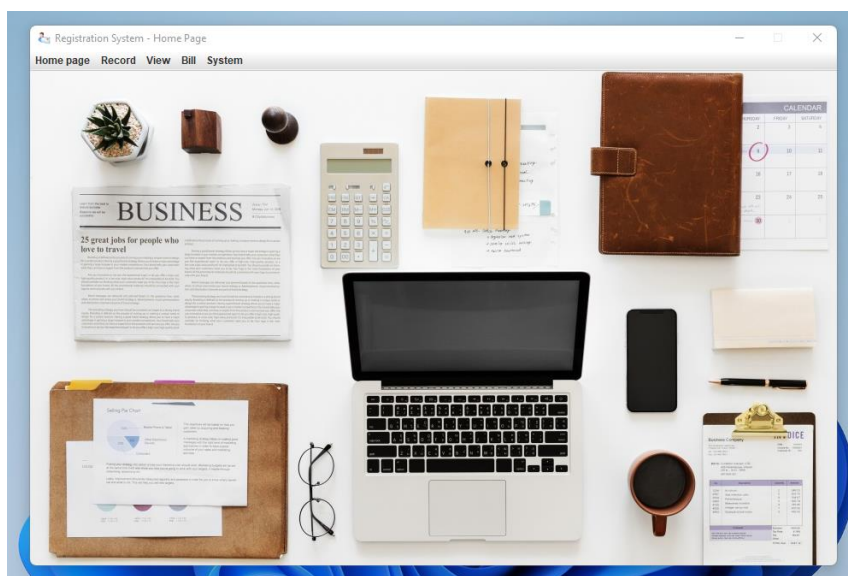
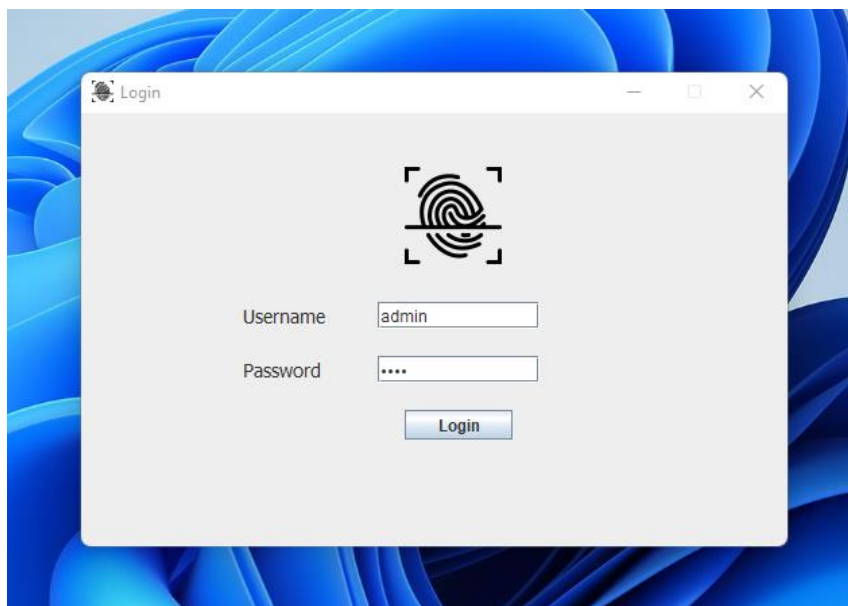


Taller 5:

El repositorio en GitHub que se selecciono para hacer el análisis de un patrón de diseño es un proyecto completamente hecho en Java, que contiene una interfaz implementada con javax.swing. El programa es descrito por su autor como un programa de contaduría que contiene la información de empleados y empleadores junto a registros de las relaciones entre estos. El programa tiene como funcionalidades un inicio de sesión que viene con un nombre de usuario y una contraseña, el registro de nuevos empleadores, solicitando nombre, apellido, su rama de negocio y contacto; el registro de empleados, incluyendo nombre, apellido y un número de teléfono, búsqueda de los empleados y trabajadores, y permite la visualización de los trabajadores que laboran para un empleador.





NEW EMPLOYER

Name

Surname

Business

Phone Number

SAVE



Name

Surname

Phone Number

SAVE

Registration System - Worker Payment

Home page Record View Bill System


SEARCH WORKER

JE

If to view all workers, leave the field blank and press the Search button.

JED NELSON	<input type="button" value="Choose"/>
JEFFREY HALL	<input type="button" value="Choose"/>
JENSON GREER	<input type="button" value="Choose"/>
JERRY WINTRINGHAM	<input type="button" value="Choose"/>
JESSE HOLMES	<input type="button" value="Choose"/>

DOCUMENT

 **Employer**

JESSE WIZARD

Workers

3 - JACKSON POWER

4 - DARLENE SHARP

5 - RITA BOND

6 - NAOMI HANSON

7 - MELISSA TERRY

8 - HADWIN COHEN

9 - EGBERT GILL

10 - JADE DENNIS

11 - HELEN DICKINSON

12 - DANIELLE GUZMAN

13 - RUFUS REED

Date

2022-03-11

Selected workers

13 Workers

Note

NOTES

La URL del proyecto es la siguiente:

<https://github.com/cbozan/employer-worker-registration-system>

La clase en específico que se desea o estructura del fragmento que se quiere revisar gira en torno a la clase DataBase. Esta clase se conecta, o es capaz de conectar con una base de datos SQL cada vez que se hace llamado a esta. Cabe recalcar que, esta clase no posee un constructor o no aparenta ser capaz

de generar instancias de si misma. Adicionalmente, no parece que ninguna otra clase del proyecto cree una instancia de esta.

Según lo que se logro entender de la implementación del programa, la mayoría de las clases del proyecto son elementos en swing, los cuales se encargan de generarse a sí mismos, como JPanels o JDialogs, etc. En donde poseen su lógica de sus widgets y parte de lógica de presentación dentro de su implementación. Para añadir o consultar elementos que se guardan en la base de datos. Se hace el llamado publico de una manera similar a la siguiente: (el siguiente texto es pseudocódigo)

```
Info=getInfoEmpleado;  
DataBase.añadirEmpleado(Info);
```

La segunda línea lo que hace es, primero en el llamado de DataBase, se asegura que la conexión con la base de datos este funcionando, de lo contrario devuelve un error, y después añade al empleado a la base de Datos, de la cual no se tiene mucho conocimiento de su funcionamiento, pero para propósitos del análisis de patrón de diseño no importa.

Teniendo en cuenta que múltiples clases del proyecto llaman a DataBase para hacer búsquedas o agregar información a la base de datos, como una interfaz unificada capaz de mediar el flujo de datos. Podemos identificar que la clase DataBase funciona como una Fachada.

El libro “Design Pattern Elements of Reusable Object-Oriented Software” define la intención de la fachada como una forma de proveer una interfaz unificada a una serie de interfaces en un subsistemas. Se utiliza para poder reducir la complejidad de los subsistemas de todo el programa. Si no se utilizara, cada interfaz que se encarga de mediar los datos de trabajadores y empleados tendrían que interactuar o implementar su propia versión de conectar con la base de datos SQL.

Dentro del proyecto, no se implementan conexiones directas de la fachada (DataBase.Java) a las interfaces que la utilizan. Pero dado a que su llamado es público, se puede realizar sin problemas.

Tiene sentido haber utilizado este patrón de diseño en el proyecto. Ya que permite que cada una de las interfaces del proyecto pueda realizar subida de datos o búsquedas de manera independiente y sencilla. Adicionalmente, Visto desde el punto de vista de la base de datos SQL, para que esta funcione o que sea compatible con el proyecto solo es necesario realizar cambios sobre la fachada, y esta base de datos es completamente ajena a los subsistemas del proyecto.

Las desventajas de hacer uso de este patrón de diseño es que dicha Clase DataBase, pasa a cargar con toda la carga o responsabilidad de lo que implica realizar la mediación de los subsistemas con el cliente de la Base de datos SQL, además de que agrega un alto grado de dependencia del proyecto en la fachada. Si esta termina fallando, el resto del programa no se puede ejecutar.

Con respecto a diferentes alternativas para la solución del problema, específicamente en el caso en el que se utiliza un SQL para mantener los datos, no parece haber muchas otras maneras de cómo resolverlo. Sin embargo, seria posible haber hecho uso de ServiceProviders que se conectan con la base de datos y realizan tareas similares a la fachada. Específicamente, proveer los servicios de subida de datos y de búsqueda en la base de datos. Sin embargo, en este caso también habría que añadir una tercera clase que sea la que medie las interacciones de estos proveedores de servicios con la base de datos.