

## **Prog. III – TP grupal**

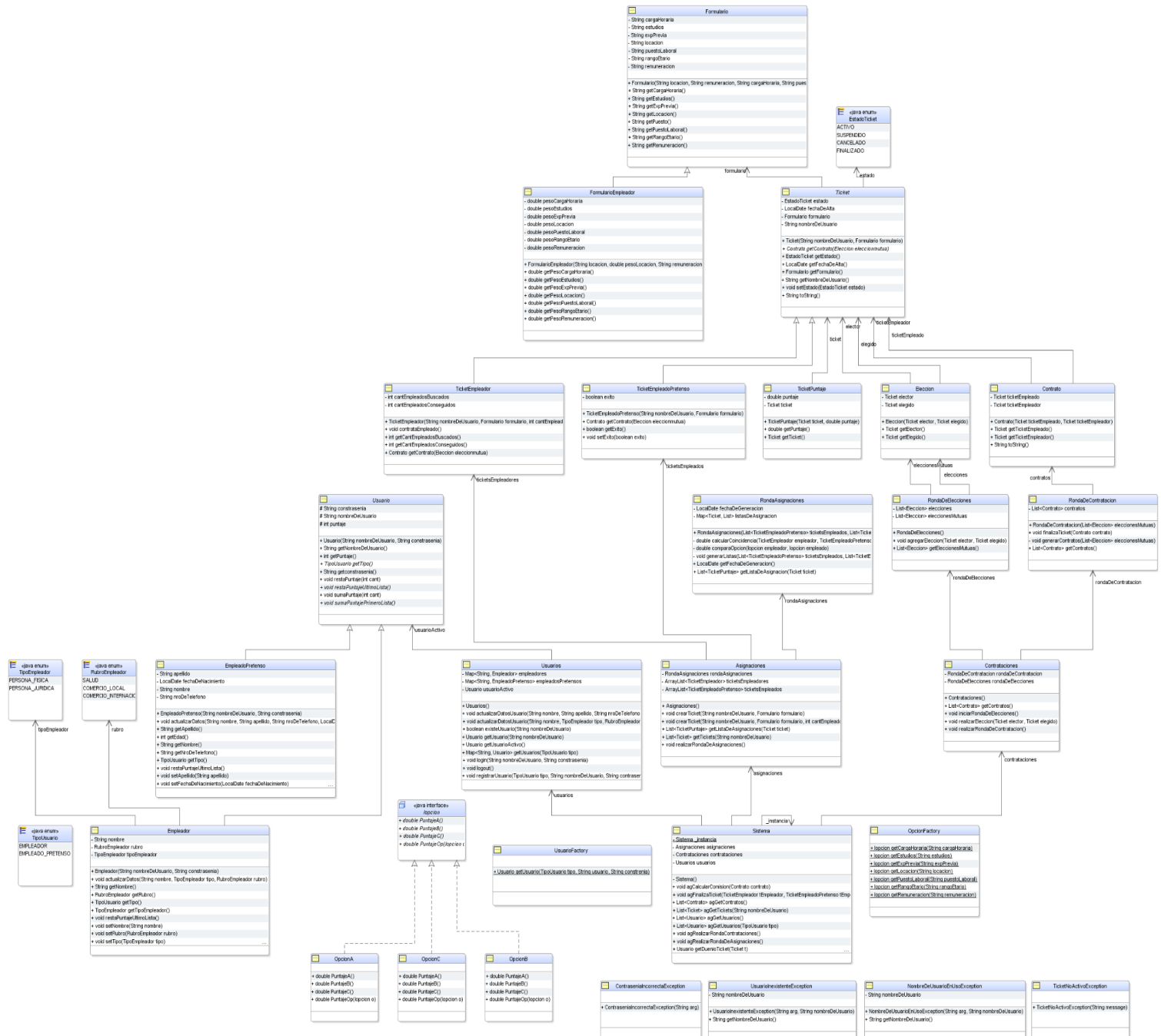
### **Parte 1**

- Programación III, UNMDP 2022.
- TPG - Primera Parte . V1 - Sistema de Gestión de Búsquedas Laborales .
- Grupo 11.
- Integrantes: Alvarez Juan, Gomez Tomas.

**Repositorio GitHub:**

**<https://github.com/juancruzalvarez/TP-ProgIII.git>**

(Imagen completa en la carpeta)



## JavaDoc:

Se encuentra en la carpeta.

## Patrones de diseño utilizados:

-Inicialmente se utilizó un Patrón Singleton en la clase Sistema para crear una única instancia, la cual cumple con todas las funcionalidades de la AGENCIA.

```
public class Sistema {
    private static Sistema _instancia;
    private Asignaciones asignaciones;
    private Usuarios usuarios;
    private Contrataciones contrataciones;

    private Sistema(){
        usuarios = new Usuarios();
        asignaciones = new Asignaciones();
        contrataciones = new Contrataciones();
    }

    public static Sistema getInstancia(){
        if(_instancia == null)
            _instancia = new Sistema();

        return _instancia;
    }
}
```

También, se utilizó el patrón Factory el cual crea un objeto de clase Usuario sin exponer la lógica de la creación al cliente y retorna una referencia al nuevo objeto creado ya sea un Usuario Empleado o Empleado Pretense.

```

public class UsuarioFactory {
    public static Usuario getUsuario(TipoUsuario tipo, String usuario, String constrenia){
        switch (tipo){
            case EMPLEADOR          -> { return new Empleador(usuario, constrenia); }
            case EMPLEADO_PRETENSO  -> { return new EmpleadoPretenso(usuario, constrenia); }
            default                  -> { return null; }
        }
    }
}

```

Además, se usó este patrón para resolver el cálculo de la matriz de puntajes, debido a que los datos ingresados por el usuario en el Formulario son de tipo String y utilizando el Factory se crea un Objeto de la clase correspondiente a la respuesta seleccionada por el Usuario. Una vez obtenida la respuesta del Usuario en formato de tipo interfaz "Iopcion" se usa el patrón Double Dispatch, obteniendo el puntaje resultante entre la elección del Empleador y el Empleado Pretenso, creando una relación entre una Opción específica con las restantes.

```

public interface Iopcion {
    public double PuntajeOp(Iopcion o);
    public double PuntajeA();
    public double PuntajeB();
    public double PuntajeC();
}

```

```

    public static Iopcion getLocacion(String locacion) {
        Iopcion rta=null;
        switch(locacion) {
            case "Home Office"-> {rta= new OpcionA();}
            case "Presencial"  -> {rta= new OpcionB();}
            case "Indistinto"->{rta= new OpcionC();}
        }

        return rta;
    }

```

No es el único caso en donde se aplica el patrón Double Dispatch se usa repetidas veces, por ejemplo al momento de sumar o restar puntaje dependiendo si el Usuario sale primero o último en la Lista de

Asignación, la cantidad de puntaje a modificar depende del tipo de Usuario ya sea Empleado Pretenso o Empleador, por lo tanto se definen los métodos en sus respectivas clases agregando un método abstracto en la clase padre, haciendo esto se evita preguntar el tipo de clase. Otro caso importante donde se usa este patrón es al momento de generar un contrato nuevo, no se sabe exactamente cómo se guardarán los tickets dependiendo su tipo, por eso en la clase Ticket se crea un método abstracto “getContrato” el cual devuelve un Contrato nuevo y ordenado correctamente dependiendo si el Elector es un Empleador o Empleado Pretenso.

```
private void generarContratos(List<Eleccion> eleccionesMutuas){
    eleccionesMutuas.forEach(e -> {
        Contrato aux=e.getElector().getContrato(e);
        contratos.add(aux);
        finalizaTicket(aux);
    });
}
```

```
public Contrato getContrato(Eleccion eleccionmutua) {
    return new Contrato(eleccionmutua.getElegido(),eleccionmutua.getElector());
}
```

Si se ejecuta este método el Elector es de tipo Empleador.

```
public Contrato getContrato(Eleccion eleccionmutua) {
    return new Contrato(eleccionmutua.getElector(),eleccionmutua.getElegido());
}
```

Si se ejecuta este método el Elector es de tipo Empleado Pretenso.

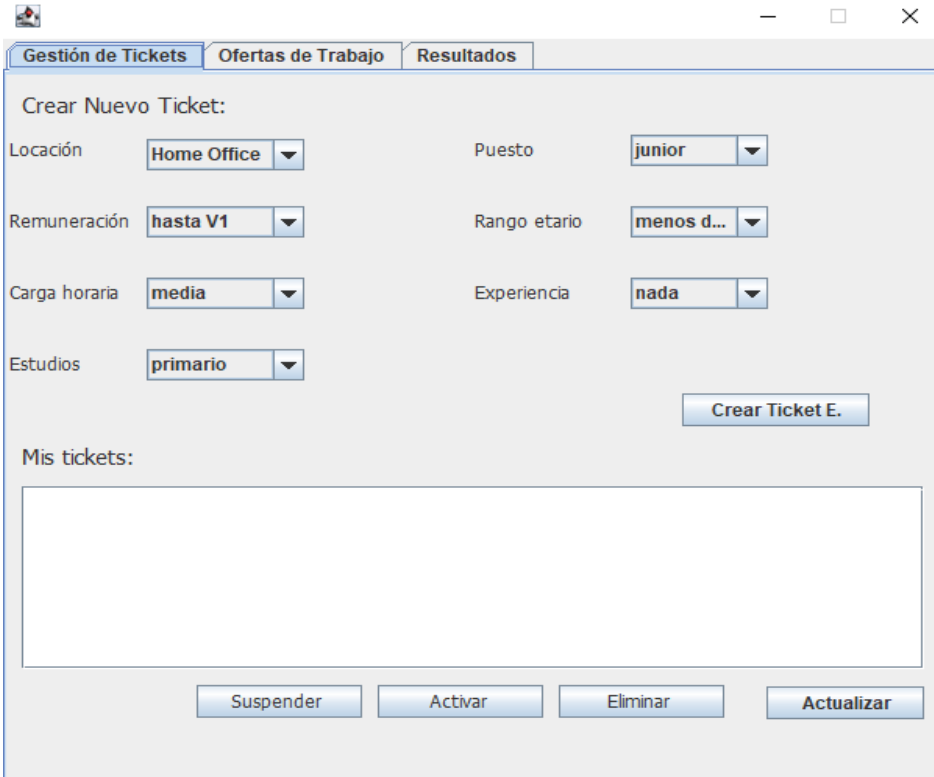
## **Prog. III – TP grupal**

### **Parte 2**

- Programación III, UNMDP 2022.
- TPG - Primera Parte . V2 - Sistema de Gestión de Búsquedas Laborales .
- Grupo 11.
- Integrantes: Alvarez Juan, Gomez Tomas.

## Patrones de diseño utilizados:

-En la segunda parte del trabajo se utilizo el Patron Modelo-vista-controlador (MVC) para hacer la interfaz de usuario, En este caso al haber 3 tipos de usuarios distintos se crearon 3 vistas diferentes “VentanaEmpleadoPretenso.java” , “VentanaEmpleador.java” y “VentanaAgencia.java” el cual estan unidas por un solo controlador “Controlador.java”, el objeto que se encarga de dirigir el flujo del control de la aplicación.



The screenshot shows a Java Swing window titled "VentanaEmpleadoPretenso.java". It has three tabs: "Gestión de Tickets" (selected), "Ofertas de Trabajo", and "Resultados". The "Gestión de Tickets" tab contains a section titled "Crear Nuevo Ticket:" with a form. The form has four rows of dropdown menus: "Locación" (Home Office), "Puesto" (junior), "Remuneración" (hasta V1), and "Carga horaria" (media). Below these are "Estudios" (primario) and "Experiencia" (nada). A "Rango etario" dropdown is set to "menos d...". A "Crear Ticket E." button is at the bottom right of the form. Below the form is a section titled "Mis tickets:" with a large empty text area. At the bottom of the window are four buttons: "Suspender", "Activar", "Eliminar", and "Actualizar".

VentanaEmpleadoPretenso.java

**Gestión de Tickets** | Contrataciones | Resultados

Crear Nuevo Ticket:

Locación: **Home ...** | Peso:  | Puesto: **junior** | Peso:

Remuneración: **hasta V1** | Peso:  | Rango etario: **menos ...** | Peso:

Carga horaria: **media** | Peso:  | Experiencia: **nada** | Peso:

Estudios: **primario** | Peso:  | Cantidad de empleados buscados:

**Crear Ticket**

Mis tickets:

**Suspender** | **Activar** | **Eliminar** | **Actualizar**

## VentanaEmpleador.java

**Panel de Control** | Empleadores | Empleados Pretensos

Panel de Control de Administrador

Ronda de Encuentros Laborales: **ACTIVAR Listas**

Ronda de Contratación: **ACTIVAR Ronda**

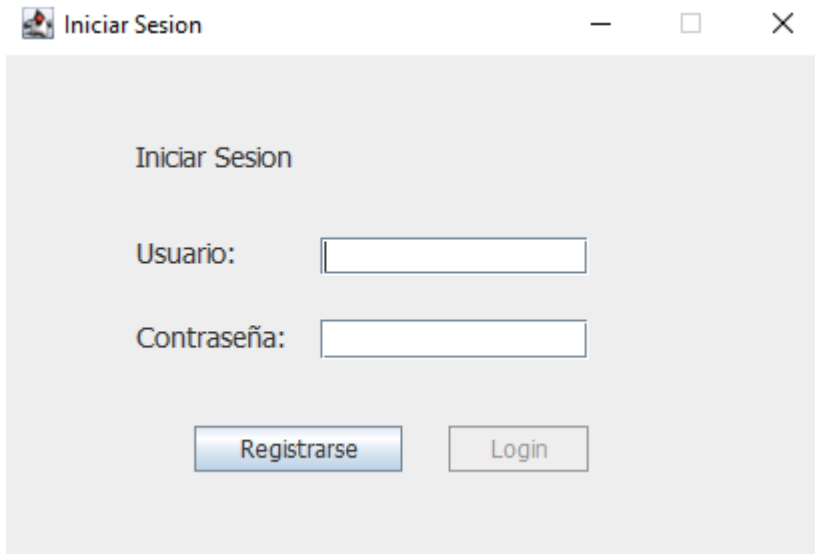
Detalles: **Ver Comision**

**Contratos Cerrados**

**Actualizar Contratos**

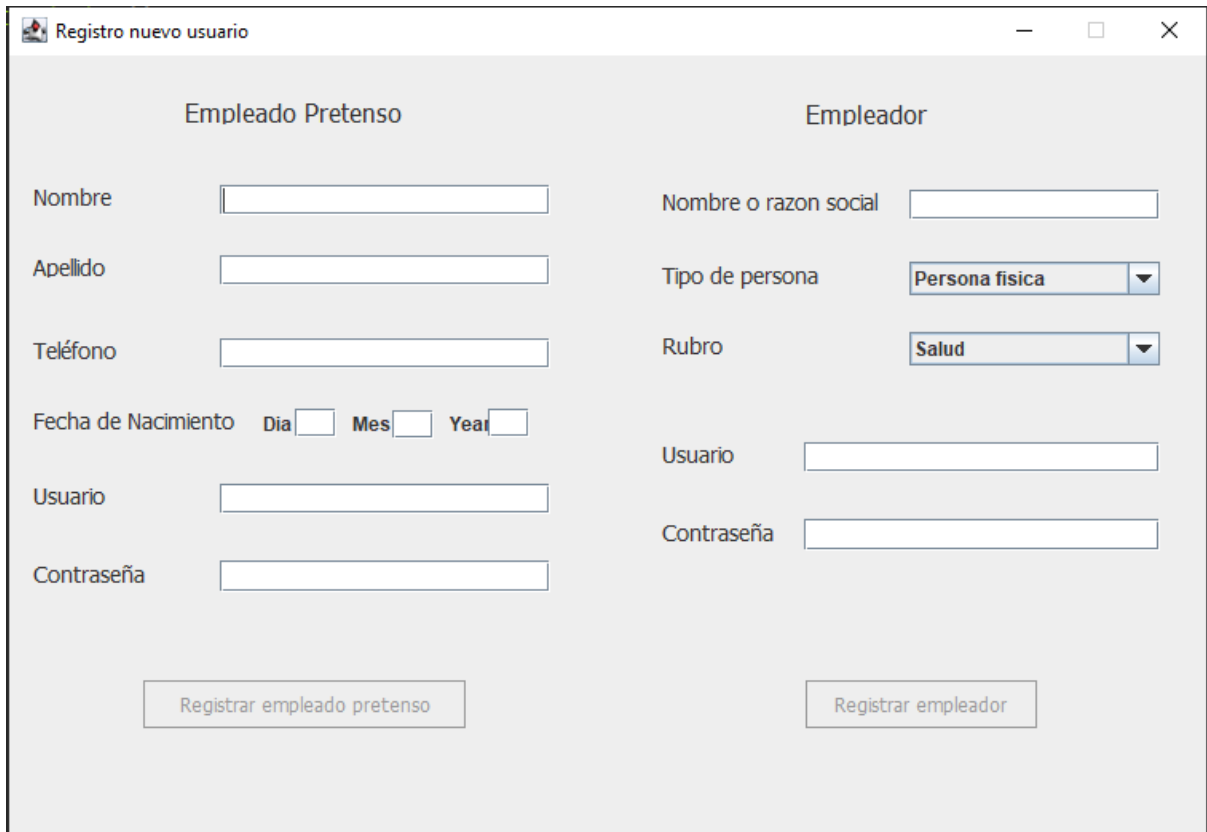
## VentanaAgencia.java

La Ventana “Login.java” es la vista “raíz” de donde se abren los distintos tipos de Ventanas dependiendo los datos de usuario cargados al entrar.



The screenshot shows a window titled "Iniciar Sesion" with standard Windows window controls (minimize, maximize, close). The window has a light gray background. At the top, the title "Iniciar Sesion" is displayed. Below the title, there are two text input fields: "Usuario:" and "Contraseña:". Below these fields, there are two buttons: "Registrarse" and "Login".

En cambio, si el usuario no esta registrado necesita completar distintos datos dependiendo del tipo de usuario que sea, dependiendo los datos completados, se habilita un botón o el otro.



The screenshot shows a window titled "Registro nuevo usuario" with standard Windows window controls (minimize, maximize, close). The window has a light gray background. It is divided into two columns: "Empleado Pretenso" and "Empleador".

**Empleado Pretenso:**

- Nombre:
- Apellido:
- Teléfono:
- Fecha de Nacimiento: Dia  Mes  Year
- Usuario:
- Contraseña:

**Empleador:**

- Nombre o razon social:
- Tipo de persona:
- Rubro:
- Usuario:
- Contraseña:

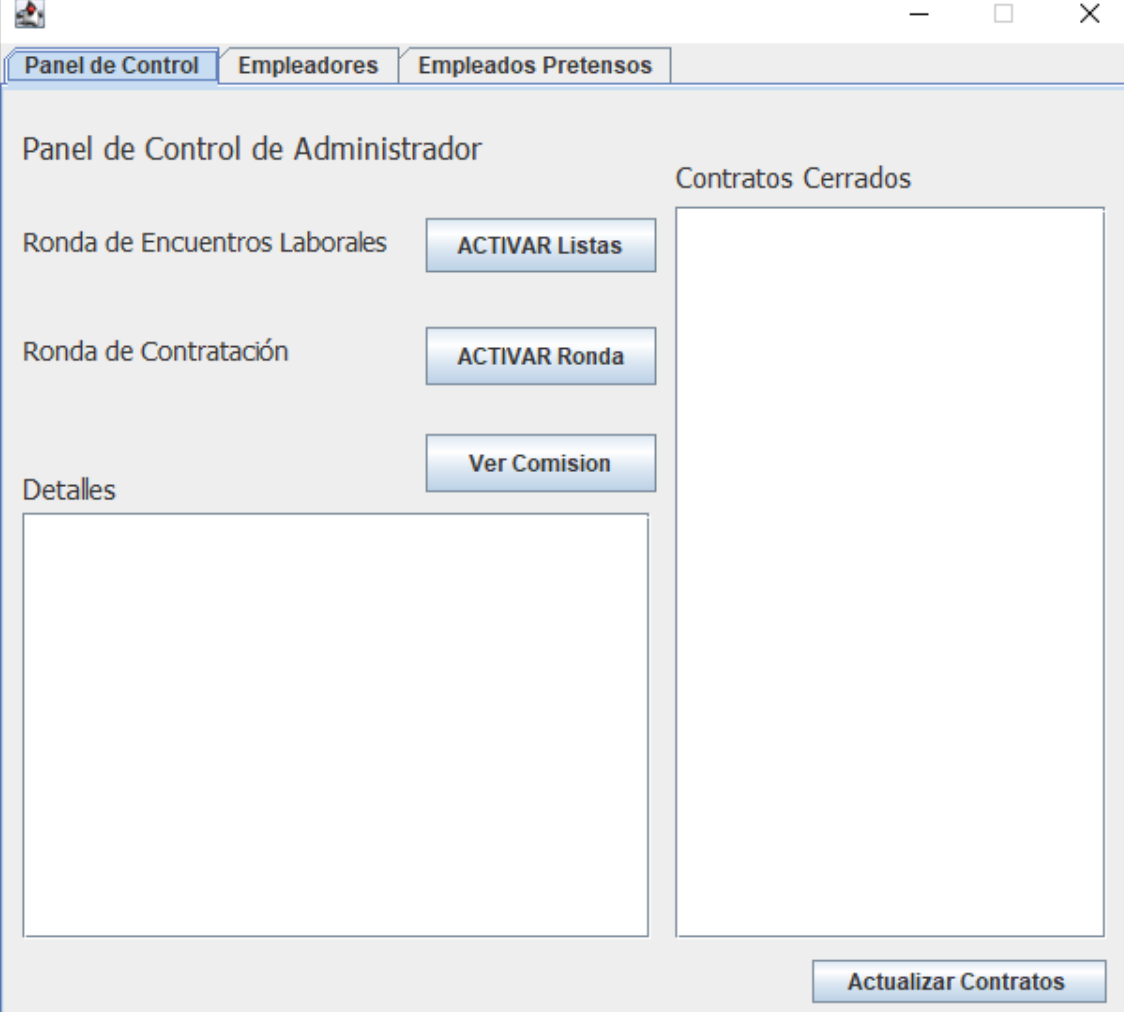
At the bottom of each column, there is a button: "Registrar empleado pretenso" and "Registrar empleador".



Por otro lado, si el Usuario es el administrador de la agencia, su cuenta ya viene previamente creada y esos datos son brindados únicamente al administrador. En este caso para acceder como Usuario Agencia los datos son:

User: admin

Password:admin



The screenshot shows a web application window titled 'Panel de Control de Administrador'. At the top, there are three tabs: 'Panel de Control' (selected), 'Empleadores', and 'Empleados Pretensos'. The main content area is divided into two columns. The left column contains three sections: 'Ronda de Encuentros Laborales' with an 'ACTIVAR Listas' button, 'Ronda de Contratación' with an 'ACTIVAR Ronda' button, and 'Detalles' with a 'Ver Comision' button. The right column is titled 'Contratos Cerrados' and contains a large empty rectangular box. At the bottom right of the window, there is an 'Actualizar Contratos' button.

- Patrón DAO – Patrón DTO