FUNCIONAMIENTO DE APLICACIÓN CON MVC

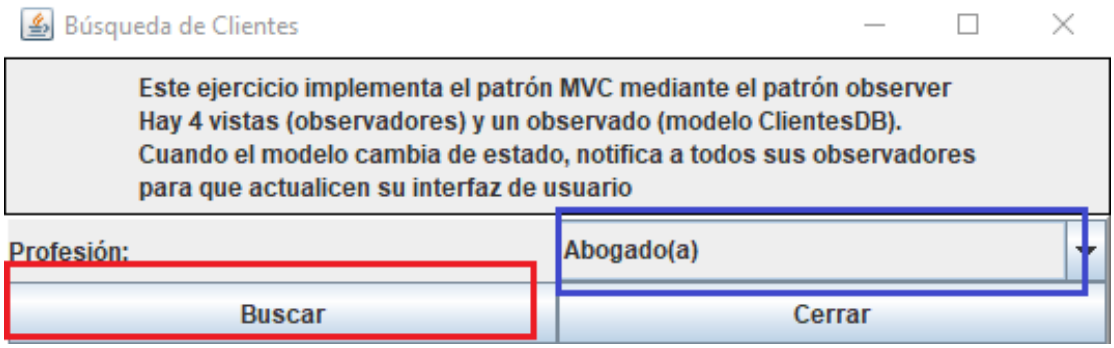
Realizado por:

Diana Marcela Pino

Fernanda Jaramillo Trullo

El funcionamiento de la aplicación de software que simula a una agencia de viajes que se basa en el modelo vista-controlador con el patrón observador podría explicarse con los siguientes pasos:

1. El usuario interactúa con la interfaz llamada en el proyecto GUIBusquedaClientes, seleccionando de la lista de profesiones la profesión de la que le interesa obtener la información y dando clic en buscar.



2. El controlador GUIBusquedaClientesController recibe por parte del observador GUIBusquedaClientes la notificación de la búsqueda solicitada por el usuario. El controlador gestiona el evento que llega.

3. El controlador accede al modelo y lo actualiza, en este caso, a través de la instancia "modelo" de la clase ClientesDB, llama al método buscarClientesPorProfesion cuya función es actualizar la lista de clientes por profesión y obtener el total de hombres y mujeres que pertenecen a ella.

```
private ClientesDB modelo;
private GUIBusquedaClientes vista;

@Override
public void actionPerformed(ActionEvent e) {
    modelo.buscarClientesPorProfesion(vista.getProfesion());
}
```

Cabe aclarar que ClientesDB es el observable por ello, al realizar la búsqueda, éste debe notificar a todos los observadores (las vistas) que se ha realizado un cambio llamando a la función notifyObservers(), esto después de haber

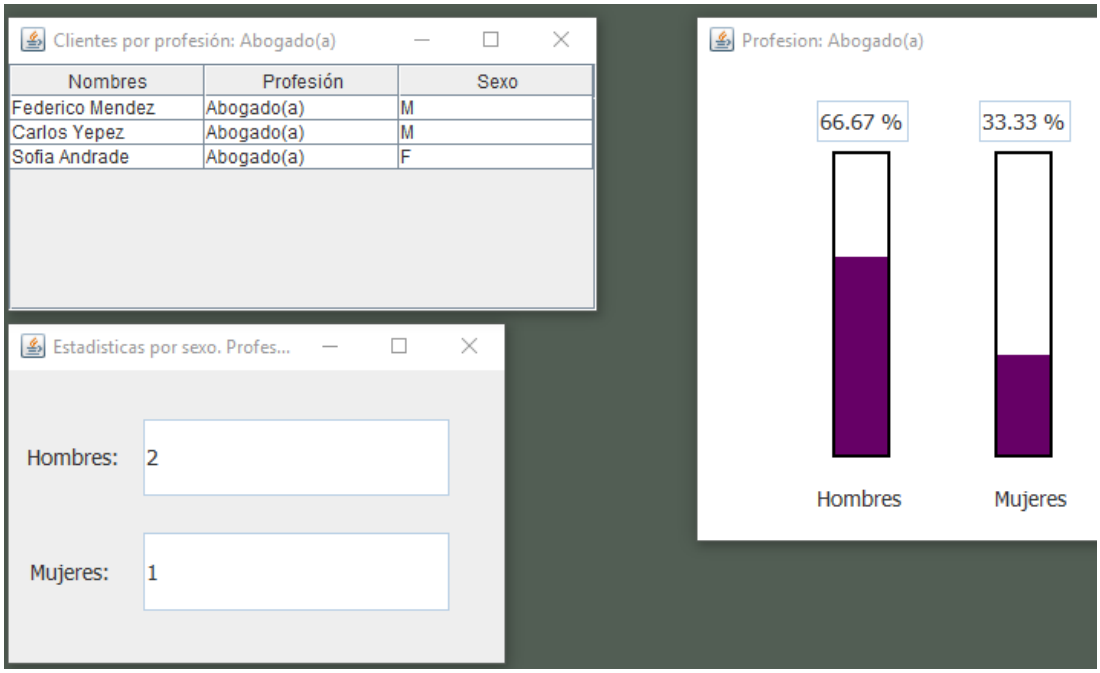
realizado los cambios pertinentes. La notificación a los observadores también se hace desde el método anteriormente mencionado.

```
public void buscarClientesPorProfesion(String profesion) {
    this.profesion = profesion;
    clientesPorProfesion = new ArrayList();
    this.totalHombres = 0;
    this.totalMujeres = 0;

    for (int i = 0; i < clientes.size(); i++) {
        Cliente cli = clientes.get(i);
        if (cli.getProfesion().equalsIgnoreCase(profesion)) {
            clientesPorProfesion.add(cli);
            if (cli.getSexo().equalsIgnoreCase("m")) {
                this.totalHombres++;
            } else {
                this.totalMujeres++;
            }
        }
    }
    setChanged();
    notifyObservers();
}
```

4. Las vistas obtienen sus datos del modelo para generar las interfaces apropiadas para el usuario donde se reflejan los cambios en el modelo (la lista de clientes por profesión, el número total de mujeres y hombres). Ya que éstas son observadores deben implementar el método update(), el cual les permite actualizarse para mostrar los cambios notificados.

   En nuestro caso tenemos una vista, GUIClientesPorProfesion, que como su nombre lo indica nos muestra el listado de clientes por profesión. Tenemos una segunda vista GUIEstadisticaPorSexo la cual nos enseña el total de mujeres y hombres que ejercen la profesión escogida por el usuario. Por último, GUIEstadisticaPorSexoGrafica que nos muestra una gráfica de Hombres vs. Mujeres con su respectivo porcentaje. Al realizar la búsqueda por “Abogado(a)” se obtiene el siguiente resultado por parte de las vistas anteriormente mencionadas:



La vista GUIBusquedaClientes es con la que interactúa el usuario y el controlador GUIBusquedaClientesController.

Cuando se utiliza el patrón Observador se provee cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Sin embargo, el modelo no está directamente relacionado con la vista, ya que un objeto vista puede esperar a los cambios del modelo, pero aun así el modelo en sí mismo sigue sin saber nada de la vista.

5. Por último, la interfaz de usuario GUIBusquedaClientes se queda esperando nuevas interacciones por parte del usuario, comenzando el ciclo nuevamente.