

# Informe de Laboratorio 23

## Tema: Proyecto Final

| Nota |
|------|
|      |

| Estudiante   | Escuela  | Asignatura  |
|--|--|---|
| Julio Rubén Chura Acabana<br>jchuraaca@unsa.edu.pe | Escuela Profesional de<br>Ingeniería de Sistemas | F. de Programación 2<br>Semestre: I<br>Código: 20230472 |

| Laboratorio | Tema           | Duración |
|-------------|----------------|----------|
| 23          | Proyecto Final | 04 horas |

| Semestre académico | Fecha de inicio      | Fecha de entrega    |
|--------------------|----------------------|---------------------|
| 2023 - B           | Del 22 de Enero 2023 | Al 29 de Enero 2023 |

### 1. Tarea

- Cree una versión del videojuego de estrategia usando componentes básicos GUI: Etiquetas, botones, cuadros de texto, JOptionPane, Color.

### 2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- Apache NetBeans IDE 19
- OpenJDK 64-Bits 20.0.2.
- Git 2.42.0.
- Xampp v3.3.0
- Cuenta en GitHub con el correo institucional.
- Base de Datos e interfaz gráfica de usuario

### 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/JulioChura/fp2-23b.git>
- URL para el laboratorio 01 en el Repositorio GitHub.
- [https://github.com/JulioChura/fp2-23b/tree/main/fase03/lab23\\_juegoFinal](https://github.com/JulioChura/fp2-23b/tree/main/fase03/lab23_juegoFinal)

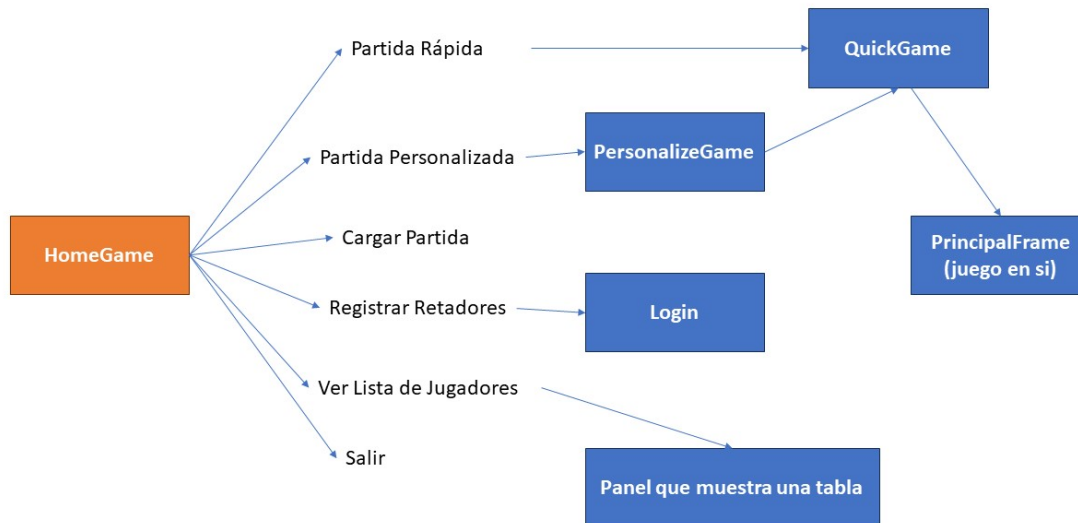
## 4. Actividades con el repositorio GitHub

### 4.1. Preparación del espacio de trabajo

Listing 1: Se crea un nuevo proyecto en Netbeans

```
mkdir lab23_juegoFinal
xcopy C:\Users\USUARIO\Desktop\fp2-23b\fase02\lab22
C:\Users\USUARIO\Desktop\fp2-23b\fase02\lab23_juegoFinal /s /e
cd ..\lab23
```

### 4.2. Estructura básica de las Ventanas



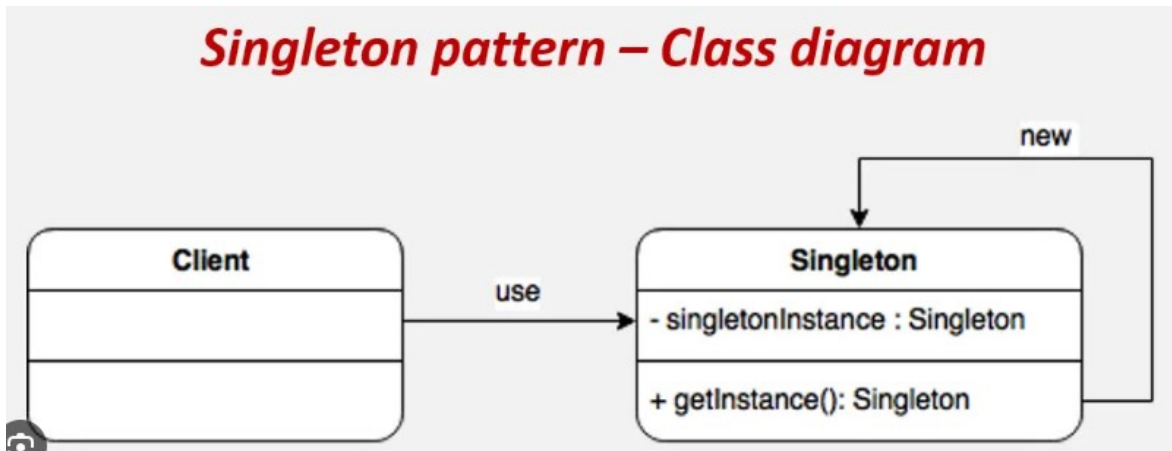
### 4.3. Evolución del código y test del mismo

Listing 2: Commit: de0f855ead9602d62a573015aa8941d54e373d59

```
git add Menu.java
git commit -m "Debido a la dificultad de trabajar, se decide mudar los archivos del
proyecto a netbeans, hasta este cambio se creo un metodo dentro de conectar para
crear un nuevo usuario"
git push -u origin main
```

- Debido a que este trabajo requería el uso de interfaz gráfica, se opta de mudar el proyecto de vs code a Netbeans debido a su herramienta que tiene para poder crear GUI de forma más sencilla y personalizables. También ya se estaba viendo para conectarse a una de datos, por lo que se se copia la clase Conectar.java de un trabajo que se dejó en los grupos de teoría. Esta clase que nos permite conectarnos con la bae de datos que tiene implementado el patrón de diseño Singleton.

Además en el proyecto se decide crear paquetes que permiten diferenciar las clases que serán GUI, las clases que manejan la lógica y la que nos permitirá usar base de datos (persistencia)



- Este es el diagrama del patrón de diseño en mención

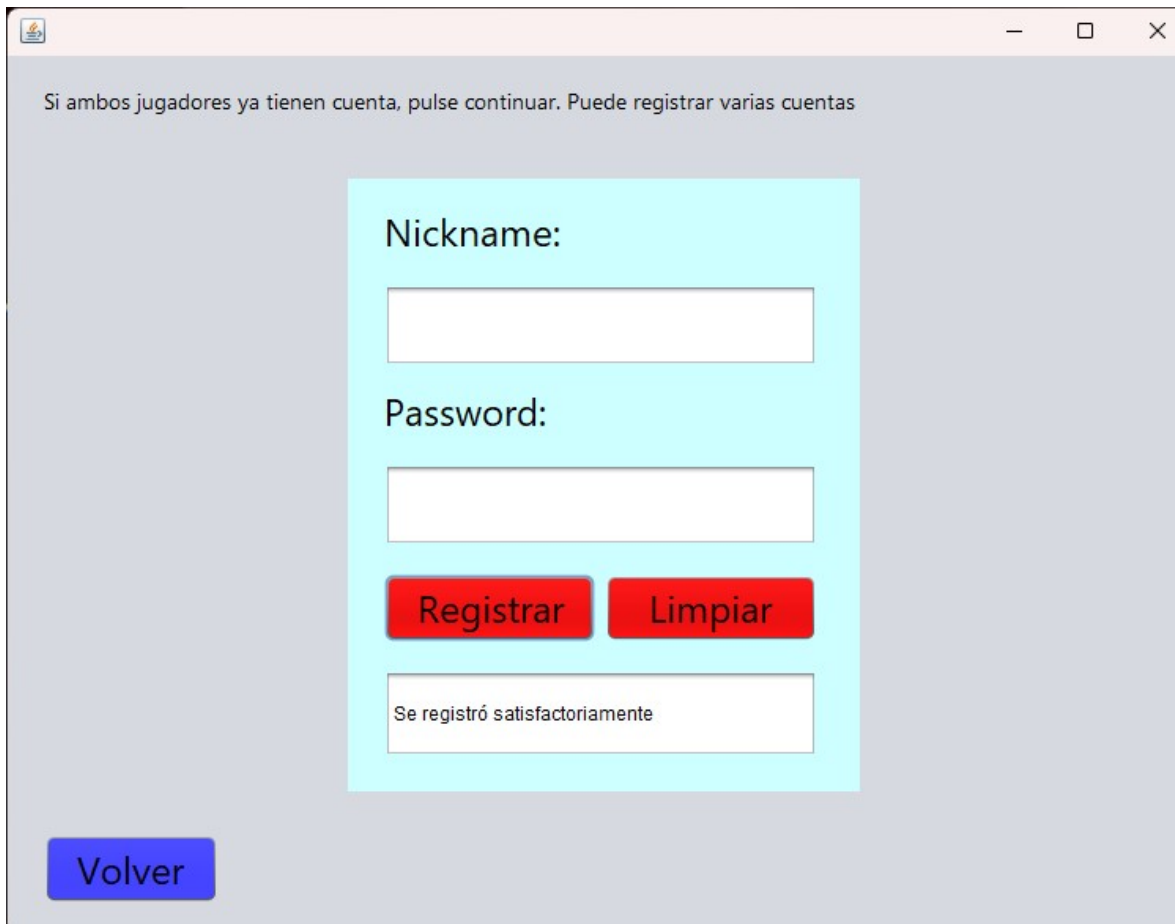


- Para el proyecto se trató de aplicar este modelo

Listing 3: Commit: 8b752b0993f8facea77c5b3a97a7856e0e6e73ed

```
git add .
git commit -m "Se completa la logica para poder registrar nuevos jugadores"
git push -u origin main
```

- En esta parte se crea la ventana para que el usuario pueda logearse. Tiene un diseño sencillo.



- Este vendría a ser el diseño final. El usuario podrá ver si se registró en el área de texto. Como se puede apreciar se acaba de registrar un nuevo usuario

Listing 4: Commit: ad82e8abf5a8c58897d2ac6b10c4959f81e56965

```
git add .
git commit -m "En login ya se crean las demas tablas y para ello se modifica la clase
Conectarr"
git push -u origin main
```

- Ya que se ha implementado en la interfaz gráfica del logeo, ahora toca crear en la clase Conectar un método que nos permita realizar una consulta a la base y verificar si el usuario en mención ya tiene una cuenta o no, como este es el primer commit para esta versión del método, adelanto que también se incorporará la función de poder crear las tablas partidas y ejércitos

Listing 5: Commit: 962b10dee42f6ec69b35431ea5b3b76094dbd215

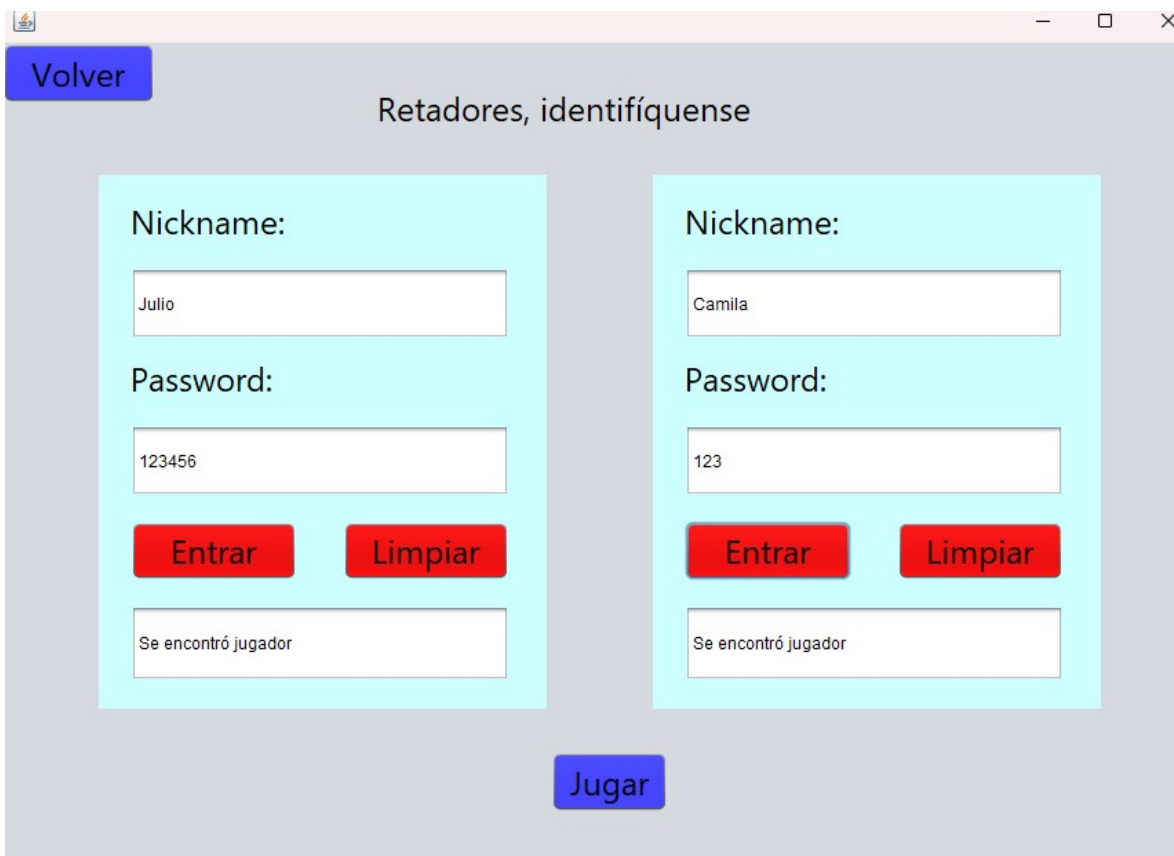
```
git add .
git commit -m "Se modifica la clase Conectar la cual ahora tiene un metodo para obtener
las victorias, ademas se hacen leves cambios en la clase para hacer las consultas"

git push -u origin main
```

- Ya en la parte cuando nos registramos para poder jugar, se tenía pensado crear el objeto Jugador para que guarde los atributos de la tabla jugadores, esto para más adelante incorporar funciones como cargar partida, sin embargo, por falta de tiempo la idea se descarta, por lo que la consulta para obtener la racha de victorias es algo innecesario.

Listing 6: Commit: fde297f82adb07850d0ebfb414669aee7afef58

```
git add .
git commit -m "Se crea ya la parte para poder entrar a la partida,es decir se edita la
               clase Personalize"
git push -u origin main
```



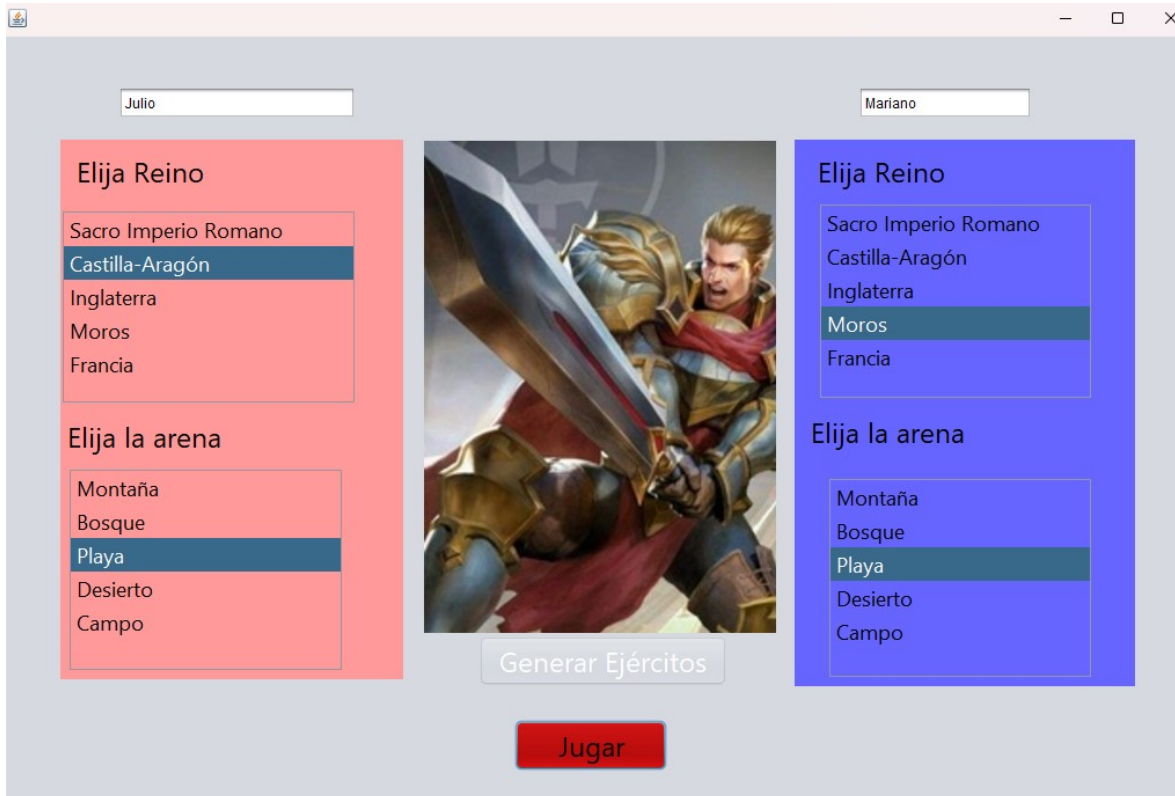
- Esta vendría a ser la ventana para poder ingresar a una partida con su usuario y contraseña, el botón jugar nos dirige al menú para seleccionar el reino y la arena

Listing 7: Commit: 2f725690a798849fc409e205c4096bf66af3cc1d

```
git add .
git commit -m "Se implementa las funcionalidades de crear ejercito y seleccionar arena
               en la clase QuickGameWindows"
git push -u origin main
```

- Se opta por generar los ejércitos en esta ventana aprovechando que también se seleccionará el reino y la arena. Para evitar que el usuario apriete el botón "Generar Ejército" cada rato, se

decide que cuando el botón sea clickeado, se verifiquen una serie de cosas, como que se hayan seleccionado las arenas y los reinos, cuando ya se tiene esa información el botón cambie de valor a false (esto se logra con un método de la clase JButton) por lo que ahora ya no podrá ser clickeado. También se hace una serie de filtros en el Botón jugar que verifica que el valor del botón sea false. Para ambos casos, si no se logran cumplir las condiciones se lanza un mensaje de advertencia.



- Como podemos ver, se han seleccionado los reinos y las arenas, aparte de que se pulsó el botón "Generar Ejércitos", por lo que cambio el color.

Listing 8: Commit: b0ebb059321682a27d2f21c912ae36be7d544d08

```
git add .
git commit -m "Se crea la clase jugar y se copia los metodos de la clase Aplicacion
               modificandolo en lago, pero el juego se congela"
git push -u origin main
```

- Ahora como ya se tenía la relación entre las ventanas, falta lo más importante, que era poder mover las piezas, o en otras palabras, poder empezar a jugar, para ello se copió los métodos que generaban el juego de la clase Aplicación a una nueva clase la cual se llama Juego, cuando se hizo la prueba, fue grande la sorpresa al ver que la ventana de había congelado

Listing 9: Commit: 1026560a06a7659c2d6af35188a74a8c46d75c44

```
git add .
git commit -m "Se soluciona el problema del tablero congelado"
git push -u origin main
```

- Debido a una sobrecarga de la ventana, ya que se ejecutaban bucles y demás procesos, se decide hacer uso de hilos, para ser más específicos SwingWorker. Para evitar que el programa se congele, se crea una instancia de PrincipalFrame que representa la ventana principal del juego. Esta instancia se crea en el hilo de fondo (doInBackground) para evitar bloquear la interfaz de usuario durante la creación.

Listing 10: Commit: 3e51aab60d8dfbb00b2b5d27c9e9d890677624a5

```
git add .
git commit -m "Se implementa la funcion de registro de victorias"
git push -u origin main
```

- En el método game de la clase Jugar, después de que el método jugarTurnos culminé dictaminando un ganador, se implementa una serie de condiciones para poder determinar al ganador para que su victoria se sume a historial, esto a maneja de darle uso a la base de datos, .

Listing 11: Commit: c3f9cabe03aa0fe30794dc4f15b4fe910aa38ab2

```
git add .
git commit -m "Se borra una clase y se termina de dar diminutas funcionalidades a las clases"
git push -u origin main
```

- Más antes se habían realizado cambios, este commit más que nada evidencia los cambios que se hicieron, entre lo más rescatable se tiene la implementación de la función para mostrar jugadores, además de agregar una barra de menú con las opciones de poder guardar, ir a home y salir de la aplicación





- Como podemos ver, se han seleccionado los reinos y las arenas, aparte de que se pulsó el botón "Generar Ejércitos", por lo que cambio el color.

Listing 12: Commit: 89ff6b1edf4542deb3ff42f005f490882c63ed89

```
git add .  
git commit -m "Se recupera la version anterior y se hace los cambios en las clases para  
poder implementar guardar, ahora falta hacer la consulta a la base de datos"  
git push -u origin main
```

- Como nos falta crear la opción de cargar partida se decide crear la clase Partida la cual tendrá los atributos necesarios para reanudar el juego.

Listing 13: Commit: e24b6eb460d4e868fad82818e2b735efea714c23

```
git add .  
git commit -m "Se crea el metodo guardarPartida"  
git push -u origin main
```

- En la clase Conectar se implementa la lógica y la clase PrincipalFrame la llama cuando se da click al botón guardar.

Listing 14: Commit: c15b7d87562a4a723111d504c4ae5b4a27526f91

```
git add .  
git commit -m "Se crea la opcion de cargar partida"  
git push -u origin main
```

- En la clase HomeGame se implementa la acción de poder cargar la última partida.

Listing 15: Interfaz del juego

```
javac HomeGame.java  
java HomeGame
```



- Se puede apreciar la ejecución del juego, además de que en la parte superior de la izquierda se puede apreciar un pequeño menú, debido al tiempo la función de guardar no se encuentra disponible, pero si las demás

Listing 16: Jugar.java

```

1 package logica;
2
3 import gui.HomeGame;
4 import gui.PrincipalFrame;
5 import gui.Tablero;
6 import java.util.Random;
7 import javax.swing.JOptionPane;
8 import javax.swing.SwingUtilities;
9 import javax.swing.SwingWorker;
10 import persistencia.Conectar;
11
12 public class Jugar {
13
14     protected Tablero tablero;
15     protected Army red;
16     protected Army blue;
17     protected String arena;
18     protected Player playerRed;
19     protected Player playerBlue;
20
21     public Jugar(Tablero tab, Army blue, Army red, String arena, Player playerRed, Player
        playerBlue) {
22         this.tablero = tab;
23         this.red = red;
24         this.blue = blue;

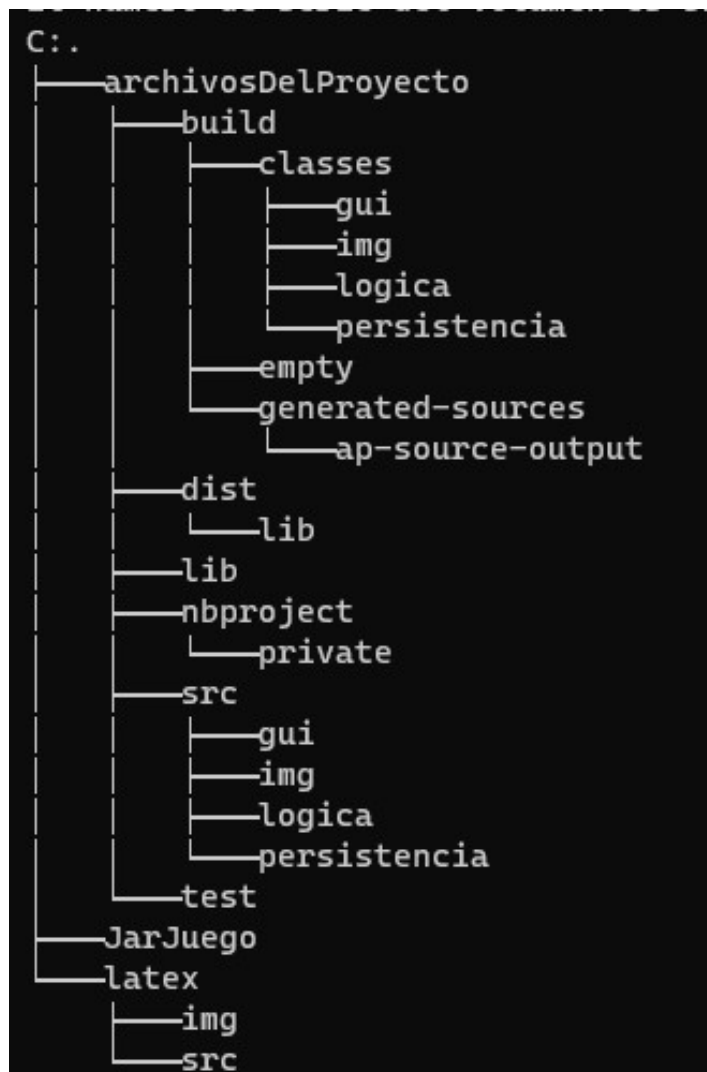
```

```
25     this.arena = arena;
26     this.playerRed = playerRed;
27     this.playerBlue = playerBlue;
28 }
29
30 public void game() {
31
32     //Para evitar el congelamiento del programa se haec uso de hilos
33     SwingWorker<Void, Void> gameWorker = new SwingWorker<Void, Void>() {
34         @Override
35         protected Void doInBackground() throws Exception {
36             PrincipalFrame principal = new PrincipalFrame(blue, red, arena, tablero,
37                 playerBlue, playerRed);
38             jugarTurnos(tablero, principal);
39
40             if (playerRed == null || playerBlue == null) {
41                 //
42             } else {
43                 Conectar conectar = Conectar.obtenerInstancia();
44                 if (blue.converterToArrayUni() == null) {
45                     conectar.registrarVictoria(playerRed.getNombre(),
46                         playerRed.getPassword());
47                     JOptionPane.showMessageDialog(principal, "Se registro la victoria "+
48                         playerRed.getNombre());
49                 } else {
50                     conectar.registrarVictoria(playerBlue.getNombre(),
51                         playerBlue.getPassword());
52                     JOptionPane.showMessageDialog(principal, "Se registro la victoria de "+
53                         playerBlue.getNombre());
54                 }
55             }
56
57             principal.dispose();
58             HomeGame home = new HomeGame();
59             home.setVisible(true);
60             return null;
61         }
62     };
63
64     gameWorker.execute(); // Ejecuta el SwingWorker en un hilo separado
65 }
66
67 public static void jugarTurnos(Tablero tabla, PrincipalFrame pane) {
68     Army e1 = tabla.getEjercito1();
69     Army e2 = tabla.getEjercito2();
70     int turno = 0;
71     JOptionPane.showMessageDialog(null, "Bienvenidos a mi juego de guerra");
72     do {
73         if (turno % 2 == 0) {
74             int x = 0, y = 0, Dx = 0, Dy = 0;
75
76             do {
77                 JOptionPane.showMessageDialog(null, "Turno del reino de " + e2.getName() +
78                     "(Azul)");
79                 int[] arr = tabla.getCoordinates();
80                 x = arr[0];
```

```
75         y = arr[1];
76         int[] toarr = tabla.getCoordinates();
77         Dx = toarr[0];
78         Dy = toarr[1];
79         } while (Army.validatePosition(e2, x, y, Dx, y));
80         Army.moveSoldier(e1, e2, x, y, Dx, Dy);
81     } else {
82         int x = 0, y = 0, Dx = 0, Dy = 0;
83         do {
84             JOptionPane.showMessageDialog(null, "Turno del reino de " + e1.getName() +
85                 "(Rojo)");
86             int[] arr = tabla.getCoordinates();
87             x = arr[0];
88             y = arr[1];
89             int[] toarr = tabla.getCoordinates();
90             Dx = toarr[0];
91             Dy = toarr[1];
92             } while (Army.validatePosition(e1, x, y, Dx, Dy));
93             Army.moveSoldier(e2, e1, x, y, Dx, Dy);
94         }
95         SwingUtilities.invokeLater(() -> pane.repaintTablero()); // Actualiza el GUI en
96         el EDT
97         turno++;
98     } while (Army.winnerDefinitive(e2, e1));
99 }
```

- El enfoque de usar hilos hace posible que las fichas puedan moverse, ya que si no se hubiera usado `SwingWorker` la ventana se congelaría





- Debido a problemas en la codificación de latex con ciertos caracteres que genera el comando tree f, se opta por enviar la estructura de este laboratorio en formato de imagen

## 5. Rúbricas

### 5.1. Entregable Informe

Tabla 1: Tipo de Informe

| Informe |   |
|---------|---|
| Latex   | El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer. |

## 5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

|        | Nivel                |                 |                    |                     |
|--------|----------------------|-----------------|--------------------|---------------------|
| Puntos | Insatisfactorio 25 % | En Proceso 50 % | Satisfactorio 75 % | Sobresaliente 100 % |
| 2.0    | 0.5                  | 1.0             | 1.5                | 2.0                 |
| 4.0    | 1.0                  | 2.0             | 3.0                | 4.0                 |

Tabla 3: Rúbrica para contenido del Informe y demostración

| Contenido y demostración |  | Puntos | Checklist | Estudiante | Profesor |
|--------------------------|--|--------|-----------|------------|----------|
| <b>1. GitHub</b>         | Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.   | 2      | X         | 2          |          |
| <b>2. Commits</b>        | Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).   | 4      | X         | 4          |          |
| <b>3. Código fuente</b>  | Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.   | 2      | X         | 2          |          |
| <b>4. Ejecución</b>      | Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.   | 2      | X         | 2          |          |
| <b>5. Pregunta</b>       | Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).  | 2      | X         | 2          |          |
| <b>6. Fechas</b>         | Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.  | 2      | X         | 2          |          |
| <b>7. Ortografía</b>     | El documento no muestra errores ortográficos.  | 2      | X         | 2          |          |
| <b>8. Madurez</b>        | El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación). | 4      | X         | 2          |          |
| <b>Total</b>             |  | 20     |           | 18         |          |

## 6. Referencias

- <https://www.geeksforgeeks.org/insertion-sort/>