

INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

INGENIERÍA EN SISTEMAS COMPUTACIONALES



TEMA 3. Programación concurrente

Proyecto Tema 3

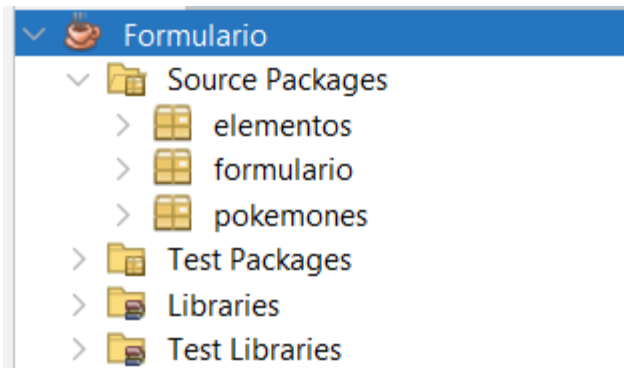
Nombre	Número de control
Sergio Gustavo Acosta Esquivel	23480879
Irma Yarith Alvarado Camargo	23480873
Angela Berenice Morales Martínez	23480853
César Emilio Oliva Vázquez	23480864

INTRODUCCIÓN	3
Estructura del Proyecto	4
Carpeta elementos.....	4
Link del proyecto.....	4
Package elementos	5
Clase BackgroundTextField	5
Clase CheckBoxSelected.....	6
Clase CustomCB.....	7
Clase CustomScroll.....	9
Clase RBSelected	11
Clase RoundedButton	12
Clase RoundedPanel	13
Clase SliderSize.....	14
Clase TransparentImage.....	16
Package formulario	17
Clase ejecuter	17
Clase formulario	18
Clase Pokemon.....	24
Clase Mostrar.....	28
Clase JuegoCartas.....	32
CONCLUSIÓN	35

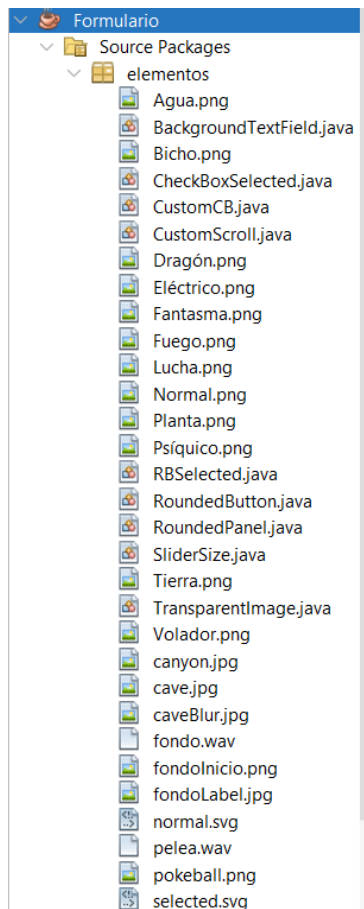
INTRODUCCIÓN

En este proyecto, desarrollamos una interfaz gráfica interactiva que permite a los usuarios buscar y filtrar Pokémon según diferentes criterios, como su nombre, tipo, tamaño, región, categoría y otras características relevantes. Para lograr esto, diseñamos y personalizamos distintos componentes gráficos en Java utilizando la biblioteca Swing, con el objetivo de mejorar la experiencia del usuario y proporcionar una navegación intuitiva. El sistema implementa una serie de controles visuales personalizados, incluyendo botones redondeados, campos de texto con estilos únicos, deslizadores configurables y elementos desplegables diseñados para una mejor accesibilidad. Además, se trabajó en la optimización de la visualización y organización de los elementos dentro de la interfaz para garantizar un diseño atractivo y funcional. Gracias a estas modificaciones, el usuario puede interactuar con el filtro de búsqueda de manera eficiente, obteniendo resultados precisos y rápidos sobre los Pokémon disponibles en la base de datos.

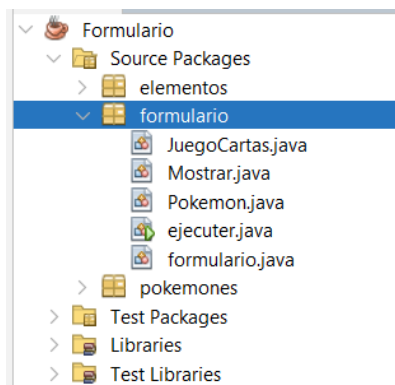
Estructura del Proyecto



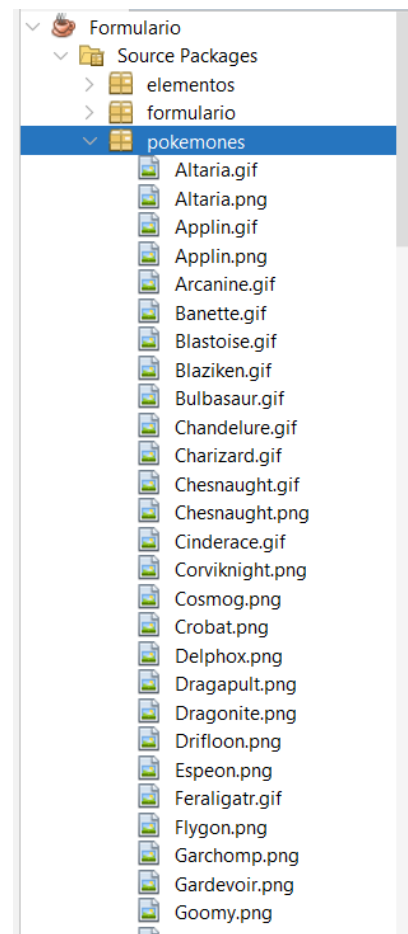
Carpeta elementos



Carpeta formulario



Carpeta pokemones

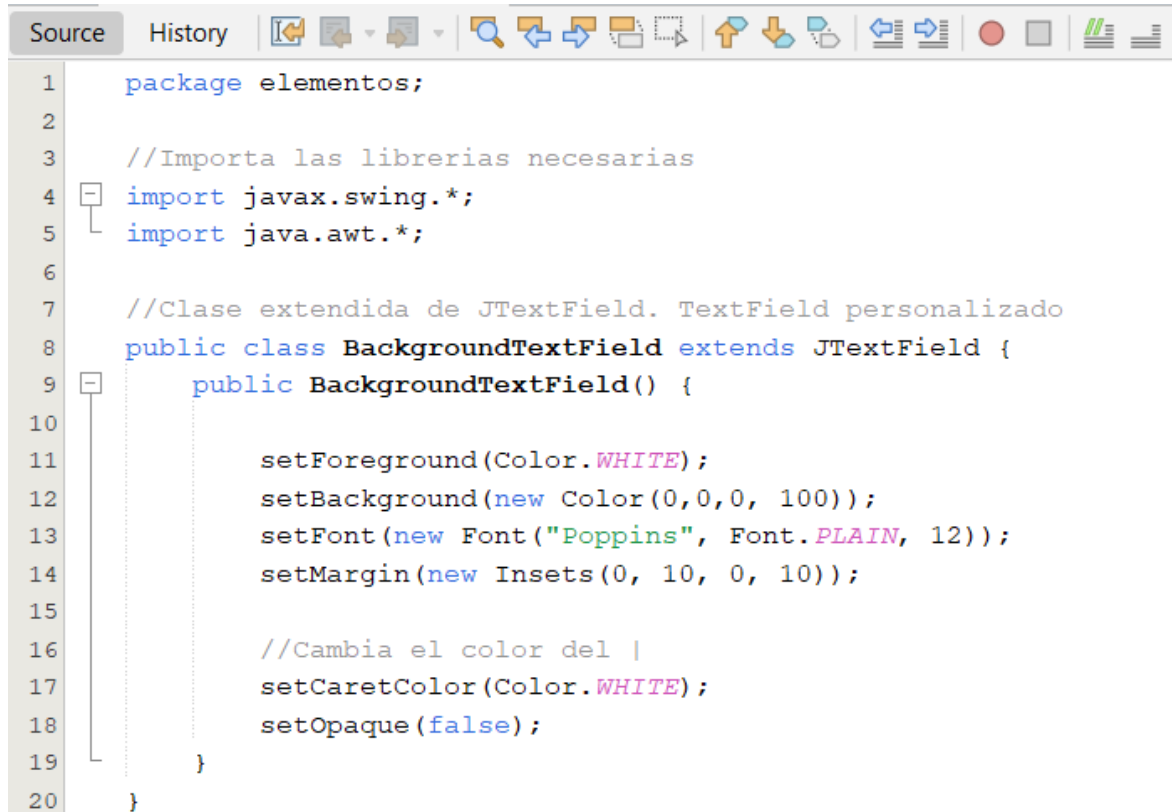


Link del proyecto.

<https://drive.google.com/file/d/1PKJfK4jb8czUiPx2uJm4wzEbVcnrRATa/view?usp=sharing>

Package elementos

Clase BackgroundTextField

The image shows a screenshot of an IDE's source code editor. The editor has a toolbar at the top with various icons for file operations, navigation, and editing. The code is written in Java and is for a class named BackgroundTextField. The code is as follows:

```
1 package elementos;
2
3 //Importa las librerias necesarias
4 import javax.swing.*;
5 import java.awt.*;
6
7 //Clase extendida de JTextField. TextField personalizado
8 public class BackgroundTextField extends JTextField {
9     public BackgroundTextField() {
10
11         setForeground(Color.WHITE);
12         setBackground(new Color(0,0,0, 100));
13         setFont(new Font("Poppins", Font.PLAIN, 12));
14         setMargin(new Insets(0, 10, 0, 10));
15
16         //Cambia el color del |
17         setCaretColor(Color.WHITE);
18         setOpaque(false);
19     }
20 }
```

En este código se define una clase personalizada llamada BackgroundTextField, que extiende la clase JTextField. El propósito de esta clase es crear un campo de texto con un estilo específico, usando algunas propiedades personalizadas para cambiar su apariencia.

- Importación de librerías
- Definición de la clase BackgroundTextField
- Constructor de la clase BackgroundTextField
- Configuración del color del texto
- Configuración del color de fondo
- Configuración de la fuente
- Configuración de los márgenes
- Cambio del color del cursor (caret)

- Hacer el fondo transparente

Clase CheckBoxSelected

```
Output - Formulario (run) X BackgroundTextField.java X CheckBoxSelected.java X CustomCB.java X
1 package elementos;
2
3 // Se importan las librerías a usar
4 import com.formdev.flatlaf.extras.FlatSVGIcon;
5 import javax.swing.*;
6 import java.awt.*;
7
8 //Clase heredada de JCheckBox. CheckBox personalizado
9 public class CheckBoxSelected extends JCheckBox {
10     public CheckBoxSelected(String text) {
11         super(text);
12
13         // Cargar imágenes en FlatLaf
14         FlatSVGIcon normalsVG = new FlatSVGIcon("elementos/normal.svg", 20, 20);
15         FlatSVGIcon selectedSVG = new FlatSVGIcon("elementos/selected.svg", 20, 20);
16
17         // Aplicar iconos personalizados
18         setIcon(normalsVG);
19         setSelectedIcon(selectedSVG);
20
21         // Configuración del checkbox
22         setOpaque(false);
23         setBorderPainted(false);
24         setFocusPainted(false);
25         setContentAreaFilled(false);
26         setFont(new Font("Poppins", Font.PLAIN, 14));
27         setForeground(Color.WHITE);
28     }
29 }
```

Clase `CheckBoxSelected`, que extiende `JCheckBox` y personaliza su apariencia con iconos SVG para los estados normal y seleccionado, elimina los bordes y el fondo, y ajusta el texto a blanco con la fuente "Poppins".

Clase CustomCB

```
1 package elementos;
2
3 //Se importan las librerias a utilizar
4 import javax.swing.*;
5 import java.awt.*;
6 import javax.swing.plaf.basic.BasicComboBoxUI;
7 import javax.swing.plaf.basic.BasicComboPopup;
8 import javax.swing.plaf.basic.BasicScrollBarUI;
9 import javax.swing.plaf.basic.ComboPopup;
10
11 //Clase heredada de JCheckBox. CheckBox personalizado
12 public class CustomCB extends JComboBox {
13     public CustomCB(String[] lista) {
14         super(lista);
15
16         setBackground(Color.WHITE);
17         setForeground(Color.black);
18         setFont(new Font("Poppins", Font.PLAIN, 13));
19         setFocusable(false);
20
21         // Aplica diseño personalizado
22         setUI(new personalizar());
23     }
24
25     // Personalizar el ComboBox editando los elementos defecto del comboBox
26     private static class personalizar extends BasicComboBoxUI {
27         @Override
28         protected ComboPopup createPopup() {
29             BasicComboPopup popup = new BasicComboPopup(comboBox) {
30                 @Override
31                 protected JScrollPane createScroller() {
32                     JScrollPane scrollPane = super.createScroller();
33                     // Modificar el scroll
34                     scrollPane.getVerticalScrollBar().setUI(new CustomScrollBar());
35                     scrollPane.setBackground(Color.GRAY);
36
37                     return scrollPane;
38                 }
39             };
40             return popup;
41         }
42
43         // Clase para personalizar el scroll del ComboBox
44         private static class CustomScrollBar extends BasicScrollBarUI {
45             @Override
46             //Color de los elementos del scroll
47             protected void configureScrollBarColors() {
48                 this.thumbColor = new Color(164, 169, 171);
49                 this.trackColor = new Color(204, 210, 213);
50             }
51
52             // Oculta el botón superior
53             @Override
54             protected JButton createDecreaseButton(int orientation) {
55                 return new JButton() {
56                     {
57                         setPreferredSize(new Dimension(0, 0));
58                     }
59                 };
60             }
61
62             // Oculta el botón superior
63             @Override
64             protected JButton createIncreaseButton(int orientation) {
65                 return new JButton() {
66                     {
67                         setPreferredSize(new Dimension(0, 0));
68                     }
69                 };
70             }
71         }
72     }
73 }
```

El código crea una clase personalizada CustomCB, que extiende JComboBox. Personaliza su apariencia con:

- Fondo blanco, texto negro, y fuente "Poppins" de 13 puntos.
- Diseño personalizado del desplegable y barras de desplazamiento.
- Las flechas de aumento y disminución del scrollbar se ocultan.
- El color del scrollbar es ajustado a tonos grisáceos.

Clase CustomScroll

```
1 package elementos;
2
3 //Se importan las librerias a utilizar
4 import java.awt.*;
5 import javax.swing.*;
6 import javax.swing.plaf.basic.BasicScrollBarUI;
7
8 //Clase CustomScroll extendida de JScrollPane. ScrollPane personalizado.
9 public class CustomScroll extends JScrollPane {
10     public CustomScroll(JPanel panel) {
11         super(panel);
12
13         // Aplica diseño personalizado
14         getVerticalScrollBar().setUI(new CustomScrollBar());
15         getHorizontalScrollBar().setUI(new CustomScrollBar());
16
17         setBorder(null);
18         setOpaque(false);
19         getViewport().setOpaque(false);
20     }
21
22     //Personalizar el ScrollBar
23     private static class CustomScrollBar extends BasicScrollBarUI {
24         @Override
25         protected void configureScrollBarColors() {
26             this.thumbColor = new Color(164, 169, 171);
27             this.trackColor = new Color(204, 210, 213);
28         }
29         @Override
30         protected JButton createDecreaseButton(int orientation) {
31             return new JButton() {
32                 {
33                     setPreferredSize(new Dimension(0, 0));
34                 }
35             };
36         }
37     }
```

```

36
37         return scrollPane;
38     }
39 };
40     return popup;
41 }
42
43 // Clase para personalizar el scroll del ComboBox
44 private static class CustomScrollBar extends BasicScrollBarUI {
45     @Override
46     //Color de los elementos del scroll
47     protected void configureScrollBarColors() {
48         this.thumbColor = new Color(164, 169, 171);
49         this.trackColor = new Color(204, 210, 213);
50     }
51
52     // Oculta el botón superior
53     @Override
54     protected JButton createDecreaseButton(int orientation) {
55         return new JButton() {
56             {
57                 setPreferredSize(new Dimension(0, 0));
58             }
59         };
60     }
61
62     // Oculta el botón superior
63     @Override
64     protected JButton createIncreaseButton(int orientation) {
65         return new JButton() {
66             {
67                 setPreferredSize(new Dimension(0, 0));
68             }
69         };
70     }
71 }
72

```

El código crea un CustomScroll, una clase personalizada de JScrollPane con barras de desplazamiento (scrollbars) estilizadas. Las características son:

- Personaliza las barras de desplazamiento vertical y horizontal con colores específicos.
- Elimina los botones de aumento y disminución del scrollbar.
- Hace que el JScrollPane y su área de vista sean transparentes, sin bordes ni opacidad.

Clase RBSelected

```
Source History
1 package elementos;
2
3 // Se importan las librerías a usar
4 import com.formdev.flatlaf.extras.FlatSVGIcon;
5 import javax.swing.*;
6 import java.awt.*;
7
8 //Clase heredada de JRadioButton. RadioButton personalizado
9 public class RBSelected extends JRadioButton {
10     public RBSelected(String text) {
11         super(text);
12
13         // Cargar imágenes en FlatLaf
14         FlatSVGIcon normalSVG = new FlatSVGIcon("elementos/normal.svg", 20, 20);
15         FlatSVGIcon selectedSVG = new FlatSVGIcon("elementos/selected.svg", 20, 20);
16
17         // Aplicar iconos personalizados
18         setIcon(normalSVG);
19         setSelectedIcon(selectedSVG);
20
21         // Configuración del checkbox
22         setOpaque(false);
23         setBorderPainted(false);
24         setFocusPainted(false);
25         setContentAreaFilled(false);
26         setFont(new Font("Poppins", Font.PLAIN, 14));
27         setForeground(Color.WHITE);
28     }
29 }
```

El código crea una clase personalizada RBSelected, que extiende JRadioButton. Personaliza su apariencia con:

- Iconos SVG para los estados normal y seleccionado del botón.
- Elimina los bordes, fondo y efectos de foco.
- Usa la fuente "Poppins" de 14 puntos y el texto en blanco.

Clase RoundedButton

```
Source History [Icons]
1 package elementos;
2
3 //Importa las librerías necesarias
4 import javax.swing.JButton;
5 import java.awt.*;
6
7 //Clase extendida de JButton. Botón personalizado
8 public class RoundedButton extends JButton {
9     public RoundedButton(String text) {
10         super(text);
11         // Evita el fondo por defecto
12         setContentAreaFilled(false);
13         // Quita el borde de selección
14         setFocusPainted(false);
15         //Quita el borde
16         setBorder(null);
17
18         setFont(new Font("Poppins", Font.PLAIN, 14));
19     }
20
21
22     //Esquinas redondeadas
23     @Override
24     protected void paintComponent(Graphics g) {
25         Graphics2D g2 = (Graphics2D) g;
26         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
27
28         // Dibujar fondo redondeado
29         g2.setColor(getBackground());
30         g2.fillRoundRect(0, 0, getWidth(), getHeight(), 10, 10); // Esquinas redondeadas
31         super.paintComponent(g);
32     }
33 }
```

El código crea una clase personalizada `RoundedButton`, que extiende `JButton`. Personaliza su apariencia con:

- Esquinas redondeadas en el botón.
- Elimina el fondo y el borde de selección predeterminado.
- Usa la fuente "Poppins" de 14 puntos.

Clase RoundedPanel

```
1 package elementos;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.geom.RoundRectangle2D;
6
7 public class RoundedPanel extends JPanel {
8     private Image imagenFondo;
9     private int ancho, alto;
10
11     // Constructor para redondear la imagen
12     public RoundedPanel(String rutaImagen, int ancho, int alto) {
13         this.imagenFondo = new ImageIcon(getClass().getResource(rutaImagen)).getImage();
14         this.ancho = ancho;
15         this.alto = alto;
16         setOpaque(false);
17         setPreferredSize(new Dimension(ancho, alto)); // Establecer el tamaño del panel
18     }
19
20     @Override
21     protected void paintComponent(Graphics g) {
22         super.paintComponent(g);
23         // Crear un objeto Graphics2D para mayor control
24         Graphics2D g2d = (Graphics2D) g;
25         // Establecer suavizado de bordes
26         g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
27
28         // Redondear los bordes del panel
29         g2d.setClip(new RoundRectangle2D.Float(0, 0, getWidth(), getHeight(), 4, 4)); // 20 es el radio de redondeo
30
31         // Dibujar la imagen dentro del panel
32         g2d.drawImage(imagenFondo, 0, 0, ancho, alto, this); // Dibuja la imagen redimensionada
33     }
34 }
```

El código crea un RoundedPanel, un panel personalizado con bordes redondeados y una imagen de fondo. La imagen se ajusta al tamaño del panel y los bordes del panel son suavizados y redondeados. El panel no tiene opacidad (transparente) para permitir que se vea el fondo de la ventana.

Clase SliderSize

```
1 package elementos;
2
3 //Se importan las librerias a utilizar;
4 import javax.swing.*;
5 import java.util.Hashtable;
6 import javax.swing.plaf.basic.BasicSliderUI;
7 import java.awt.*;
8
9 //Clase SliderSize heredada de JSlider. Slider Personalizado
10 public class SliderSize extends JSlider {
11     public SliderSize() {
12         super(0, 100, 50);
13
14         // Hacer que avance de 25 en 25
15         setMajorTickSpacing(25);
16         setPaintTicks(true);
17         setPaintLabels(true);
18         setOpaque(false);
19         setFocusable(false);
20
21         // Cambiar etiquetas de los valores
22         Hashtable<Integer, JLabel> etiquetas = new Hashtable<>();
23
24         JLabel mchico = new JLabel("Muy Chico");
25         mchico.setFont(new Font("Poppins", Font.PLAIN, 14));
26         mchico.setForeground(Color.WHITE);
27
28         JLabel chico = new JLabel("Chico");
29         chico.setFont(new Font("Poppins", Font.PLAIN, 14));
30         chico.setForeground(Color.WHITE);
31
32         JLabel mediano = new JLabel("Mediano");
33         mediano.setFont(new Font("Poppins", Font.PLAIN, 14));
34         mediano.setForeground(Color.WHITE);
35
36         JLabel grande = new JLabel("Grande");
37         grande.setFont(new Font("Poppins", Font.PLAIN, 14));
38         grande.setForeground(Color.WHITE);
39
40         JLabel mgrande = new JLabel("Muy Grande");
41         mgrande.setFont(new Font("Poppins", Font.PLAIN, 14));
42         mgrande.setForeground(Color.WHITE);
43
44         //Reemplaza el numero por el label
45         etiquetas.put(0, mchico);
46         etiquetas.put(25, chico);
47         etiquetas.put(50, mediano);
48         etiquetas.put(75, grande);
49         etiquetas.put(100, mgrande);
50         setLabelTable(etiquetas);
51
52         // Aplica diseño personalizado
53         setUI(new personalizar(this));
54         //Cambia el color del tick
55         UIManager.put("Slider.tickColor", Color.WHITE);
56     }
57
58     // Personalizar el Slider editando los elementos defecto del Slider
59     private static class personalizar extends BasicSliderUI {
60         public personalizar(JSlider slider) {
61             super(slider);
62         }
63
64         //Cambia el diseño del thumb
65         @Override
66         public void paintThumb(Graphics g) {
67             Graphics2D g2 = (Graphics2D) g;
68             g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
69         }
70     }
71 }
```

```

70         // Cambia el color y tamaño del thumb
71         int thumbSize = 18;
72         g2.setColor(Color.red);
73         g2.fillOval(thumbRect.x, thumbRect.y, thumbSize, thumbSize);
74     }
75
76     //Cambia el diseño de la barra
77     @Override
78     public void paintTrack(Graphics g) {
79         Graphics2D g2 = (Graphics2D) g;
80         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
81
82         int trackHeight = 8;
83         int x = trackRect.x;
84         int y = trackRect.y + trackRect.height / 2 - trackHeight / 2;
85         int width = trackRect.width;
86
87         // Pinta la parte seleccionada de la barra
88         int selectedWidth = thumbRect.x - (x-9);
89         g2.setColor(Color.red);
90         g2.fillRoundRect(x, y, selectedWidth, trackHeight, trackHeight, trackHeight);
91
92         // Pinta el borde en toda la barra
93         g2.setColor(Color.red);
94         g2.setStroke(new BasicStroke(1));
95         g2.drawRoundRect(x, y, width, trackHeight, trackHeight, trackHeight);
96     }
97 }
98

```

El código crea una clase personalizada SliderSize, que extiende JSlider. Personaliza su apariencia con:

- Avance del slider de 25 en 25 unidades.
- Etiquetas personalizadas en valores 0, 25, 50, 75 y 100.
- Color rojo para el "thumb" (el control deslizante) y la barra del slider.
- Elimina el fondo y el enfoque predeterminado del slider.
- Aplica un diseño personalizado para el "thumb" y la barra.

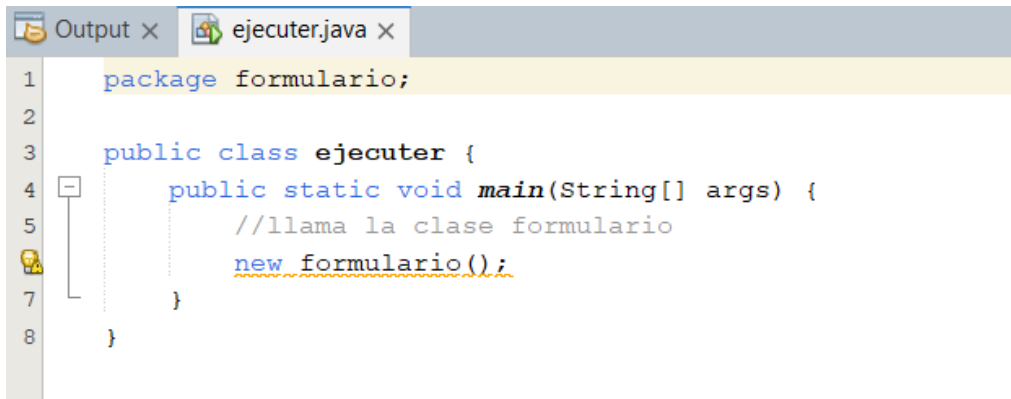
Clase TransparentImage

```
Source History
1 package elementos;
2
3 //Se importan las librerias a usar
4 import java.awt.*;
5 import java.awt.image.BufferedImage;
6 import javax.swing.*;
7
8 public class TransparentImage extends ImageIcon {
9
10     public TransparentImage(String path, float opacity) {
11         super(TransparentImage.makeTransparent(new ImageIcon(TransparentImage.class.getResource(path)).getImage(), opacity));
12     }
13
14     private static Image makeTransparent(Image img, float opacity) {
15         BufferedImage bufferedImage = new BufferedImage(img.getWidth(null), img.getHeight(null), BufferedImage.TRANSLUCENT);
16         Graphics2D g2d = bufferedImage.createGraphics();
17         g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, opacity));
18         g2d.drawImage(img, 0, 0, null);
19         g2d.dispose();
20         return bufferedImage;
21     }
22 }
```

El código crea una clase `TransparentImage` que extiende `ImageIcon`. Permite cargar una imagen desde una ruta y ajustar su opacidad. La imagen es convertida a un formato con transparencia usando un valor de opacidad especificado.

Package formulario

Clase ejecuter



```
1 package formulario;
2
3 public class ejecuter {
4     public static void main(String[] args) {
5         //llama la clase formulario
6         new formulario();
7     }
8 }
```

The screenshot shows a code editor with two tabs: 'Output' and 'ejecuter.java'. The code in 'ejecuter.java' is as follows:

Clase que ejecuta el método main encargado de llamar a la clase formulario

Clase formulario

```
Output X  ejecutar.java X  formulario.java X
1 package formulario;
2
3 //Se importan las librerias a usar
4 import javax.swing.*;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.util.ArrayList;
8 import java.util.List;
9 import java.util.Arrays;
10 import java.io.File;
11 import javax.sound.sampled.AudioInputStream;
12 import javax.sound.sampled.AudioSystem;
13 import javax.sound.sampled.Clip;
14
15 //Importar las "librerias" personalizadas a utilizar
16 import elementos.CheckBoxSelected;
17 import elementos.BackgroundTextField;
18 import elementos.CustomCB;
19 import elementos.RoundedButton;
20 import elementos.RBSelected;
21 import elementos.SliderSize;
22
23
24 //Clase principal que muestra los elementos del formulario
25 public class formulario extends JFrame{
26     //Filtros
27     private JTextField nombreTF;
28     private JSlider slider;
29     private JComboBox categoriasComboBox;
30     //Listas de elementos de tipo especifico
31     private JCheckBox [] tiposCB;
32     private JRadioButton [] regionRB;
33     private ButtonGroup regiones;
34
35     private Clip clip;
36     private static formulario instancia;
37
38     //Arreglo de opciones para el comboBox
39
40     //Arreglo de opciones para el comboBox
41     String categoriasLista [] = {"Pokémon Ratón", "Pokémon Pez", "Pokémon Ave", "Pokémon Bicho",
42     "Pokémon Dragón", "Pokémon Perro/Lobo", "Pokémon Gato/Felino", "Pokémon Planta/Flor",
43     "Pokémon Humanoide", "Pokémon Fantasma/Espectral"};
44
45     //Tipos de pokemones para iterar
46     String tiposLista [] = {"Eléctrico", "Fuego", "Lucha", "Agua", "Normal", "Fantasma",
47     "Planta", "Bicho", "Psíquico", "Volador", "Tierra", "Dragón"};
48
49     //Nombre de regiones para iterar
50     String regionLista [] = {"Kanto", "Johto", "Hoenn", "Sinnoh", "Unova", "Kalos",
51     "Alola", "Galar", "Paldea"};
52
53     //Constructor clase formulario
54     public formulario(){
55         //Configuración de la ventana
56         setTitle("Selección de personaje");
57         setSize(500, 700);
58         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
59         setResizable(false);
60         setLayout(new BorderLayout());
61         setLocationRelativeTo(null);
62
63         //Comienza la musica de fondo
64         startMusic();
65
66         //Creación del panel
67         JPanel panel = new JPanel();
68         panel.setLayout(null);
69
70         instancia = this;
71
72         //Lista de checkBox
73         tiposCB = new JCheckBox[tiposLista.length];
74         //Lista de regiones
75         regionRB = new JRadioButton[regionLista.length];
76         //Grupo de los RadioButton
```

```

Output x ejecutar.java x formulario.java x
75 //Grupo de los RadioButton
76 regiones = new ButtonGroup();
77
78 //Imagen del fondo panel.
79 JLabel fondo = new JLabel(new ImageIcon(getClass().getResource("/elementos/fondoInicio.png")))
80 fondo.setBounds(0, 0, 500, 700); // Ajusta al tamaño de la ventana
81
82 //Elementos del panel
83 JLabel nombre = new JLabel("Busqueda por nombre:");
84 nombre.setFont(new Font("Poppins", Font.PLAIN, 16));
85 nombre.setForeground(Color.WHITE);
86 nombre.setBounds(30, 30, 200, 30);
87
88 nombreTF = new BackgroundTextField();
89 nombreTF.setBounds(230, 30, 130, 30);
90
91 JButton select = new RoundedButton("Seleccionar");
92 select.setBackground(new Color(125, 120, 163));
93 select.setForeground(Color.WHITE);
94 select.setBounds(370, 30, 100, 30);
95
96 JLabel tipo = new JLabel("Tipo de Pokémon: ");
97 tipo.setFont(new Font("Poppins", Font.PLAIN, 16));
98 tipo.setForeground(Color.WHITE);
99 tipo.setBounds(30, 70, 200, 30);
100
101 //Crea checkbox y los agrega al panel
102 int pos1 = 100;
103 int pos2 = 100;
104 int pos5 = 100;
105 for(int i=0;i<tiposLista.length;i++){
106     tiposCB[i] = new CheckBoxSelected(tiposLista[i]);
107     if(i<5){
108         //Cambio de posicion en el eje y con un x fijo
109         tiposCB[i].setBounds(30, pos1, 120, 30);
110         pos1+=30;
111     }else if(i<10){
112         //Cambio de posicion en el eje y con un x fijo

```

```

150         //Cambio de posicion en el eje y con un x fijo
151         regionRB[i].setBounds(200, pos4, 100, 30);
152         pos4+=30;
153     }
154     //Se agregan los radioButton al grupo de radioButton y al panel
155     regiones.add(regionRB[i]);
156     panel.add(regionRB[i]);
157 }
158
159 JLabel categorias = new JLabel("Categoria:");
160 categorias.setFont(new Font("Poppins", Font.PLAIN, 16));
161 categorias.setForeground(Color.WHITE);
162 categorias.setBounds(30, 550, 100, 30);
163
164 categoriasComboBox = new CustomCB(categoriasLista);
165 categoriasComboBox.setMaximumRowCount(3);
166 categoriasComboBox.setBounds(130, 550, 250, 30);
167
168 JButton reset = new RoundedButton("Limpiar");
169 reset.setBounds(125, 600, 100, 30);
170 reset.setBackground(new Color(193,191,203));
171 reset.setForeground(Color.BLACK);
172
173 JButton search = new RoundedButton("Buscar");
174 search.setBounds(275, 600, 100, 30);
175 search.setBackground(Color.RED);
176 search.setForeground(Color.white);
177
178
179 //Funcionalidad del botón de selección
180 select.addActionListener((ActionEvent e)->{
181     //Llama al metodo de busqueda por nombre utilizando el texto del textField
182     Pokemon.busquedaNombre(getNombre());
183     limpiarFiltros();
184 });
185
186 //Funcionalidad del botón de reset
187 reset.addActionListener((ActionEvent e)->{

```

```

187     reset.addActionListener((ActionEvent e)->{
188         limpiarFiltros();
189     });
190
191     //Funcionalidad del botón de búsqueda
192     search.addActionListener((ActionEvent e)->{
193         //Abre el frame de selección de personaje
194         new Mostrar(getFiltros());
195         limpiarFiltros();
196     });
197
198     //Agrega los elementos al panel
199     panel.add(nombre);
200     panel.add(nombreTF);
201     panel.add(select);
202     panel.add(tipo);
203     panel.add(tamaño);
204     panel.add(slider);
205     panel.add(region);
206     panel.add(categorias);
207     panel.add(categoriasComboBox);
208     panel.add(reset);
209     panel.add(search);
210     panel.add(fondo);
211
212     //Agrega el panel al frame
213     add(panel);
214
215     //Quita el focus del textField al abrir la interfáz
216     SwingUtilities.invokeLater(() -> {
217         nombreTF.setEnabled(false);
218         nombreTF.setEnabled(true);
219         nombre.requestFocusInWindow();
220     });
221
222     setVisible(true);
223 }
224

```

```

225 //Metodo de iniciar musica
226 public void startMusic(){
227     try{
228         File musica = new File("src/elementos/fondo.wav");
229         if(musica.exists()){
230             AudioInputStream audio = AudioSystem.getAudioInputStream(musica);
231             clip = AudioSystem.getClip();
232             clip.open(audio);
233             clip.loop(Clip.LOOP_CONTINUOUSLY);
234             clip.start();
235         }
236     }catch(Exception e){
237     }
238 }
239
240 public void VerificarCB(){
241     int CBSelected = 0;
242     for(int i=0;i<tiposLista.length;i++){
243         if(tiposCB[i].isSelected()){
244             CBSelected++;
245         }
246         if(CBSelected>=3){
247             tiposCB[i].setSelected(false);
248         }
249     }
250 }
251
252 //Obtener el nombre
253 public String getNombre(){
254     //Devuelve el nombre
255     return nombreTF.getText();
256 }
257
258 //Obtener tipo
259 public List<String> getTipo(){
260     //Lista que guarda los elementos seleccionados
261     List<String> tipoString = new ArrayList<>();
262     for(int i=0;i<tiposLista.length;i++){

```

```

261 List<String> tipoString = new ArrayList<>();
262 for(int i=0;i<tiposLista.length;i++){
263     if(tiposCB[i].isSelected()){
264         //Agrega a la lista el checkbox seleccionado
265         tipoString.add(tiposCB[i].getText());
266     }
267 }
268 //Devuelve la lista con los checkbox seleccionados
269 return tipoString;
270 }
271
272 //Obtener tamaño
273 public String getTamaño(){
274     int valor = slider.getValue();
275     //Devuelve el tamaño en base a los valores del slider
276     if(valor<33){
277         return "Chico";
278     }else if(valor<66){
279         return "Mediano";
280     }else{
281         return "Grande";
282     }
283 }
284
285 //Obtener región
286 public String getRegion(){
287     for(int i=0;i<regionRB.length;i++){
288         if(regionRB[i].isSelected()){
289             //Devuelve la región
290             return regionLista[i];
291         }
292     }
293     //Devuelve vacío si no está nada seleccionado
294     return "";
295 }
296
297 //Obtener categoría
298 public String getCategoria(){

```

```

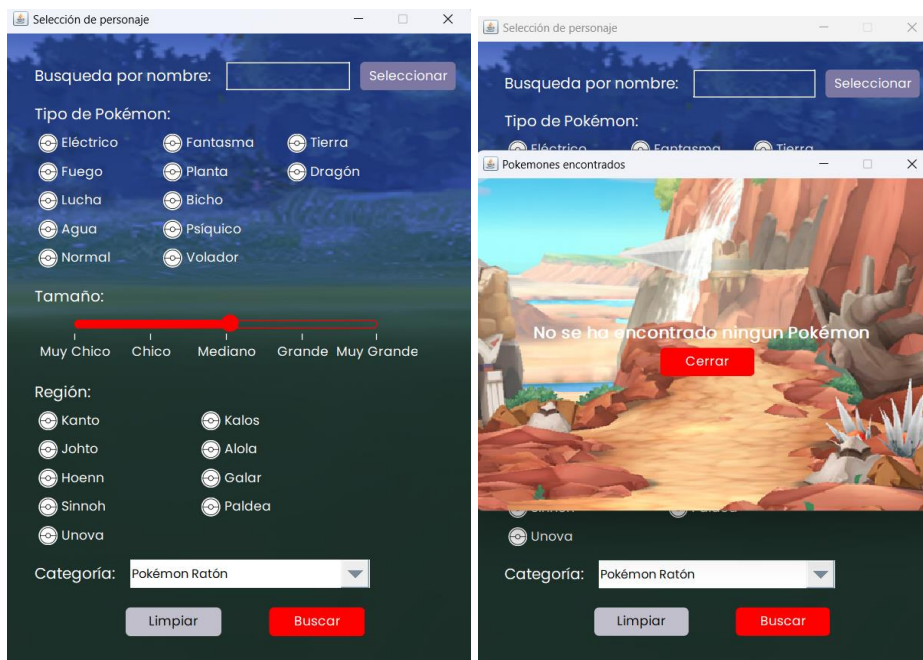
299 //Obtener categoría
300 public String getCategoria(){
301     //Obtiene el texto del comboBox
302     String categoria = (String) categoriasComboBox.getSelectedItemAt();
303     //Separa el texto en cada espacio para separar la categoría de la palabra "Pokémon"
304     String [] categoriaSplit = categoria.split(" ");
305     //Devuelve la categoría
306     return categoriaSplit[1];
307 }
308
309 //Obtiene todos los filtros juntos
310 public List <Pokemon> getFiltros(){
311     List<String> tipo = getTipo();
312     String tamaño = getTamaño();
313     String categoria = getCategoria();
314     String region = getRegion();
315
316     //Dependiendo de los filtros llama un metodo u otro (Sobrecarga)
317     if(regiones.getSelection() != null && Arrays.stream(tiposCB).anyMatch(JCheckBox::isSelected)){
318         return Pokemon.pokemonesFiltrados(tipo, tamaño, region, categoria);
319     }else if(Arrays.stream(tiposCB).anyMatch(JCheckBox::isSelected)){
320         return Pokemon.pokemonesFiltrados(tipo, tamaño, categoria);
321     }else if(regiones.getSelection() != null ){
322         return Pokemon.pokemonesFiltrados(tamaño, region, categoria);
323     }else{
324         return Pokemon.pokemonesFiltrados(tamaño, categoria);
325     }
326 }
327
328 //Limpia los filtros
329 public void limpiarFiltros(){
330     nombreTF.setText("");
331
332     for(JCheckBox cb : tiposCB){
333         cb.setSelected(false);
334     }

```

```

329 //Limpia los filtros
330 public void limpiarFiltros() {
331     nombreTF.setText("");
332
333     for(JCheckBox cb : tiposCB) {
334         cb.setSelected(false);
335     }
336
337     slider.setValue(50);
338     regiones.clearSelection();
339     categoriasComboBox.setSelectedIndex(0);
340 }
341
342 //Obtiene una instancia de la clase formulario
343 public static formulario getInstancia() {
344     return instancia;
345 }
346
347 //Cierra la ventana
348 public void cerrar() {
349     dispose();
350     clip.stop();
351     clip.close();
352 }
353 }

```



Clase formulario extendida de Frame que permite al usuario filtrar Pokémon según diversas características como nombre, tipo, región, categoría y tamaño. Incluye funcionalidades para buscar, seleccionar y limpiar filtros.

Hace la configuración del frame agregando características como el tamaño de la ventana, el título, opciones de cierre, opciones de layout, etc, aparte hace uso de las “librerías” personalizadas del package elementos.

Hace uso de los componentes básicos de un formulario como:

- Campos de texto para búsqueda por nombre
- Listas de checkboxes para tipos de Pokémon
- Radio buttons para selección de región
- Slider para filtrar por tamaño
- Combo box para categorías
- Botones ("Seleccionar", "Limpiar", "Buscar")

Contiene los métodos de obtención de nombre, tipo, categoría, tamaño, región y la obtención de estos filtros juntos

Limpiar los filtros para regresar a las opciones default e iniciar una música de fondo.

Al hacer uso de la búsqueda por nombre si encuentra al pokemon manda directamente al juego de cartas y si no muestra ventana de no encontrado.

Al hacer uso de la búsqueda por filtros, si encuentra pokemones coincidentes con los filtros inicia la ventana de mostrar, de lo contrario muestra que no existen pokemones de tales características

Clase Pokemon

```
Output - Formulario (run) X ejecutar.java X formulario.java X Mostrar.java X Pokemon.java X
1 package formulario;
2
3 //Se importan las librerias a utilizar
4 import elementos.RoundedButton;
5 import java.awt.Color;
6 import java.awt.Font;
7 import java.awt.event.ActionEvent;
8 import java.util.*;
9 import javax.swing.*;
10
11 //Clase pokemon. Estructura/Formato de pokemones
12 public class Pokemon {
13     //Atributos del tipo Pokemon
14     private String nombre;
15     private List<String> tipos;
16     private String tamaño;
17     private String region;
18     private String categoria;
19     private String url;
20
21     //Lista de tipo Pokemon
22     private static List<Pokemon> pokemones = new ArrayList<>();
23
24
25     //Constructor del pokémon
26     public Pokemon(String nombre, List<String> tipos, String tamaño, String region, String categoria, String url) {
27         //Atributos
28         this.nombre = nombre;
29         this.tipos = tipos;
30         this.tamaño = tamaño;
31         this.region = region;
32         this.categoria = categoria;
33         this.url = url;
34     }
35
36     //Metodos para obtener los datos del pokemon
37     protected String getNombre() {return nombre;}
38     protected List<String> getTipo() {return tipos;}
```

```
39     protected String getTamaño() {return tamaño;}
40     protected String getRegion() {return region;}
41     protected String getCategoria() {return categoria;}
42     protected String getURL() {return url;}
43
44     //Clase interna Pokedex que crea los pokemones y los guarda en una lista
45     static class Pokedex {
46         //Obtiene la lista de los pokemones
47         private List<Pokemon> getListPokemon() {
48             //Se crean los pokemones
49             pokemones.add(new Pokemon("Pikachu", Arrays.asList("Eléctrico"), "Chico", "Kanto", "Ratón",
50                 "/pokemones/Pikachu.png"));
51             pokemones.add(new Pokemon("Raichu", Arrays.asList("Eléctrico"), "Mediano", "Kanto", "Ratón",
52                 "/pokemones/Raichu.png"));
53             pokemones.add(new Pokemon("Pawmot", Arrays.asList("Eléctrico", "Lucha"), "Chico", "Paldea", "Ratón",
54                 "/pokemones/Pawmot.png"));
55             pokemones.add(new Pokemon("Jolteon", Arrays.asList("Eléctrico"), "Chico", "Kanto", "Gato/Felino",
56                 "/pokemones/Jolteon.png"));
57             pokemones.add(new Pokemon("Luxray", Arrays.asList("Eléctrico"), "Mediano", "Sinnoh", "Gato/Felino",
58                 "/pokemones/Luxray.png"));
59             pokemones.add(new Pokemon("Manectric", Arrays.asList("Eléctrico"), "Mediano", "Hoenn", "Perro/Lobo",
60                 "/pokemones/Manectric.png"));
61             pokemones.add(new Pokemon("Zapdos", Arrays.asList("Eléctrico", "Volador"), "Grande", "Kanto", "Ave",
62                 "/pokemones/zapdos.png"));
63             pokemones.add(new Pokemon("Elektross", Arrays.asList("Eléctrico"), "Grande", "Unova", "Pez",
64                 "/pokemones/elektross.png"));
65             pokemones.add(new Pokemon("Rotom-Mow", Arrays.asList("Eléctrico", "Planta"), "Mediano", "Sinnoh", "Planta/Flor",
66                 "/pokemones/rotom-mow.png"));
67
68             pokemones.add(new Pokemon("Arcanine", Arrays.asList("Fuego"), "Grande", "Kanto", "Perro/Lobo",
69                 "/pokemones/arcanine.png"));
70             pokemones.add(new Pokemon("Typhlosion", Arrays.asList("Fuego"), "Mediano", "Johto", "Perro/Lobo",
71                 "/pokemones/typhlosion.png"));
72             pokemones.add(new Pokemon("Houndoom", Arrays.asList("Fuego"), "Mediano", "Johto", "Perro/Lobo",
73                 "/pokemones/houndoom.png"));
74             pokemones.add(new Pokemon("Blaziken", Arrays.asList("Fuego", "Lucha"), "Mediano", "Hoenn", "Humanoide",
75                 "/pokemones/blaziken.png"));
76             pokemones.add(new Pokemon("Infernape", Arrays.asList("Fuego", "Lucha"), "Mediano", "Sinnoh", "Humanoide",
```



```

225         "/pokemones/Goomy.png"));
226
227         //Devuelve la lista de pokemones
228         return pokemones;
229     }
230 }
231
232
233 //Busqueda de pokemones por nombre
234 public static void busquedaNombre(String nombre) {
235     if(pokemones == null || pokemones.isEmpty()){
236         pokemones = new Pokedex().getListaPokemon();
237     }
238
239     //Guarda el valor para verificar si existe el pokémon
240     Boolean encontrado = false;
241
242     //Recorre toda la lista de pokemones
243     for (Pokemon pokemon : pokemones) {
244         //Compara el nombre con los de los pokemones
245         if(pokemon.nombre.equalsIgnoreCase(nombre)){
246             encontrado = true;
247             new JuegoCartas(pokemon.getNombre());
248             formulario.getInstancia().cerrar();
249         }
250     }
251     //Si no lo encuentra muestra panel de no encontrado
252     if(!encontrado){
253         //Creación de la ventana
254         JFrame frame = new JFrame("Pokemones encontrados");
255         frame.setSize(500,390);
256         frame.setResizable(false);
257         frame.setLocationRelativeTo(null);
258
259         //Creación del panel
260         JPanel panel = new JPanel(null);
261
262         //Creación de los elementos
263
264         //Creación de los elementos
265         JLabel imagen = new JLabel(new ImageIcon(ClassLoader.getResource("elementos/canyon.jpg")
266         imagen.setBounds(0, 0, 500, 360);
267
268         JLabel text = new JLabel("Pokémon no encontrado");
269         text.setFont(new Font("Poppins", Font.BOLD, 18));
270         text.setForeground(Color.white);
271         text.setBounds(130, 150, 400, 30);
272
273         JButton cerrar = new RoundedButton("Cerrar");
274         cerrar.setBackground(Color.red);
275         cerrar.setFont(new Font("Poppins", Font.PLAIN, 14));
276         cerrar.setForeground(Color.white);
277         cerrar.setBounds(210, 180, 80, 30);
278
279         //funcionalidad del boton cerrar
280         cerrar.addActionListener((ActionEvent e)->{
281             frame.dispose();
282         });
283
284         //Agrega el panel
285         panel.add(text);
286         panel.add(cerrar);
287         panel.add(imagen);
288         frame.add(panel);
289
290         frame.setVisible(true);
291     }
292 }
293
294 //Sobrecarga de metodos
295 //Metodo que devuelve los pokemones que cumplen con los filtros establecidos
296 protected static List<Pokemon> pokemonesFiltrados(String tamaño, String categoria){
297     //Crea una lista para guardar los pokemones que coincidan con los filtros
298     ArrayList<Pokemon> filtroPokemon = new ArrayList<>();

```

```

296     protected static List<Pokemon> pokemonesFiltrados(String tamaño, String categoria){
297         //Crea una lista para guardar los pokemones que coincidan con los filtros
298         ArrayList<Pokemon> filtroPokemon = new ArrayList<>();
299
300         if(pokemones == null || pokemones.isEmpty()){
301             pokemones = new Pokedex().getListaPokemon();
302         }
303
304         //Recorre toda la lista de pokemones
305         for(Pokemon pokemon : pokemones){
306             if(pokemon.tamaño.equals(tamaño) && pokemon.categoria.equals(categoria)){
307                 filtroPokemon.add(pokemon);
308             }
309         }
310         return filtroPokemon;
311     }
312
313     protected static List<Pokemon> pokemonesFiltrados(List<String> tipo, String tamaño, String categoria){
314         //Crea una lista para guardar los pokemones que coincidan con los filtros
315         ArrayList<Pokemon> filtroPokemon = new ArrayList<>();
316
317         if(pokemones == null || pokemones.isEmpty()){
318             pokemones = new Pokedex().getListaPokemon();
319         }
320
321         String lado1 = tipo.toString();
322         String [] split = lado1.replace("[", "").replace("]", "").split(", ");
323         String lado2 = "";
324         if(tipo.size()==2){
325             lado2 = "["+split[1]+", "+split[0]+"]";
326         }else{
327             lado2 = split[0];
328         }
329
330         //Recorre toda la lista de pokemones
331         for (Pokemon pokemon : pokemones) {
332             if (pokemon.tamaño.equals(tamaño) && pokemon.categoria.equals(categoria) &&
333                 (pokemon.tipos.toString().equals(lado1) || pokemon.tipos.toString().equals(lado2))){

```

```

331             for (Pokemon pokemon : pokemones) {
332                 if (pokemon.tamaño.equals(tamaño) && pokemon.categoria.equals(categoria) &&
333                     (pokemon.tipos.toString().equals(lado1) || pokemon.tipos.toString().equals(lado2))){
334                     filtroPokemon.add(pokemon);
335                 }else if (pokemon.tamaño.equals(tamaño) && pokemon.categoria.equals(categoria) &&
336                     (pokemon.tipos.toString().contains(lado1) || pokemon.tipos.toString().contains(lado2))){
337                     filtroPokemon.add(pokemon);
338                 }
339             }
340
341             return filtroPokemon;
342         }
343
344     protected static List<Pokemon> pokemonesFiltrados(String tamaño, String region ,String categoria){
345         //Crea una lista para guardar los pokemones que coincidan con los filtros
346         ArrayList<Pokemon> filtroPokemon = new ArrayList<>();
347
348         if(pokemones == null || pokemones.isEmpty()){
349             pokemones = new Pokedex().getListaPokemon();
350         }
351
352         //Recorre toda la lista de pokemones
353         for(Pokemon pokemon : pokemones){
354             if(pokemon.tamaño.equals(tamaño) && pokemon.region.equals(region) && pokemon.categoria.equals(categoria)){
355                 filtroPokemon.add(pokemon);
356             }
357         }
358         return filtroPokemon;
359     }
360
361     protected static List<Pokemon> pokemonesFiltrados(List<String> tipo, String tamaño, String region ,String categoria){
362         //Crea una lista para guardar los pokemones que coincidan con los filtros
363         ArrayList<Pokemon> filtroPokemon = new ArrayList<>();
364
365         if(pokemones == null || pokemones.isEmpty()){
366             pokemones = new Pokedex().getListaPokemon();
367         }
368
369         String lado1 = tipo.toString();

```

```

364
365     if (pokemones == null || pokemones.isEmpty()) {
366         pokemones = new Pokedex().getListaPokemon();
367     }
368
369     String lado1 = tipo.toString();
370     String [] split = lado1.replace("[", "").replace("]", "").split(", ");
371     String lado2 = "";
372     if (tipo.size() == 2) {
373         lado2 = "[" + split[1] + ", " + split[0] + "]";
374     } else {
375         lado2 = split[0];
376     }
377
378     //Recorre toda la lista de pokemones
379     for (Pokemon pokemon : pokemones) {
380         if (pokemon.tamaño.equals(tamaño) && pokemon.categoria.equals(categoria) && pokemon.region.equals(region) &&
381             (pokemon.tipos.toString().equals(lado1) || pokemon.tipos.toString().equals(lado2))) {
382             filtroPokemon.add(pokemon);
383         } else if (pokemon.tamaño.equals(tamaño) && pokemon.categoria.equals(categoria) && pokemon.region.equals(region) &&
384             (pokemon.tipos.toString().contains(lado1) || pokemon.tipos.toString().contains(lado2))) {
385             filtroPokemon.add(pokemon);
386         }
387     }
388     return filtroPokemon;
389 }
390

```

La clase Pokemon representa la estructura básica de un Pokémon y contiene métodos para gestionar una lista de Pokémon, realizar búsquedas y aplicar filtros según diferentes criterios.

Cada objeto pokemon tendrá los siguientes atributos

- Nombre: Nombre del Pokémon
- Tipos: Lista de tipos (puede tener más de uno)
- Tamaño: Tamaño del Pokémon
- Región: Región de origen
- Categoría: Categoría específica
- URL: Ruta de la imagen del Pokémon

Esta clase contiene una clase interna Pokedex, que se encarga de crear y almacenar los objetos de tipo pokemon.

Esta clase contiene el método de búsqueda de pokemon por nombre, que se encarga de comparar el nombre ingresado con el de los pokemones existentes.

Así mismo, contiene el método de comparación de pokemones que regresa una lista de los pokemones encontrados coincidentes con los filtros establecidos.

Al compararlos llama a la clase Mostrar, encargada de mostrar la lista de los pokemones encontrados.

Clase Mostrar

```
Output - Formulario (run) x ejecutar.java x formulario.java x Mostrar.java x Pokemon.java x
1 package formulario;
2
3 //Se importan las librerias a usar
4 import javax.swing.*;
5 import java.awt.*;
6 import java.util.List;
7 import java.awt.event.ActionEvent;
8
9 //Importar las "librerias" personalizadas a utilizar
10 import elementos.RoundedButton;
11 import elementos.CustomScroll;
12 import elementos.RoundedPanel;
13 import elementos.TransparentImage;
14
15 //Clase Mostrar heredada de JFrame. Frame personalizado
16 public class Mostrar extends JFrame {
17     JPanel fila;
18
19     public Mostrar(List<Pokemon> filtrados) {
20         //Configuración de la ventana
21         setTitle("Pokemones encontrados");
22         setSize(700, 480);
23         setDefaultCloseOperation(DISPOSE_ON_CLOSE);
24         setLocationRelativeTo(null);
25         setResizable(false);
26         setLayout(new BorderLayout());
27
28         //Si hay al menos un pokemon encontrado
29         if (filtrados.size() > 0) {
30
31             int filas = (int) Math.ceil((double) filtrados.size() / 3);
32
33             //Creación del panel
34             //Grid layout de n filas y 1 columna
35             JPanel panel = new JPanel(new GridLayout(filas, 1));
36             panel.setBackground(new Color(0, 0, 0, 0));
37
38             //Imagen del fondo panel.
```

```
Output - Formulario (run) x ejecutar.java x formulario.java x Mostrar.java x Pokemon.java x
37
38 //Imagen del fondo panel.
39 JLabel fondo = new JLabel(new ImageIcon(getClass().getResource("/elementos/cave.jpg")));
40 fondo.setBounds(0, 0, 700, 480);
41
42 //Guarda el numero de pokemon recorrido
43 int pokemonRecorrido = 0;
44
45 //Crea n filas de paneles segun la cantidad de pokemones encontrados
46 for (int i = 0; i < filas; i++) {
47     //Crea el panel de las filas
48     fila = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));
49     fila.setBackground(new Color(0, 0, 0, 0));
50
51     //Recorre la lista de pokemones encontrados
52     for (int j = 0; j < filtrados.size(); j++) {
53         if (j < 3) {
54             if (pokemonRecorrido < filtrados.size()) {
55                 Pokemon pokemon = filtrados.get(pokemonRecorrido);
56
57                 //Creación de cards de cada pokemon encontrado
58                 JPanel card = new JPanel(null);
59                 card.setPreferredSize(new Dimension(180, 380));
60                 card.setBackground(new Color(0, 0, 0, 0));
61
62                 JLabel fondoCard = new JLabel(new TransparentImage("/elementos/caveBlur.jpg", .8f));
63                 fondoCard.setBounds(0, 0, 180, 380);
64
65                 //Creación de elementos
66                 ImageIcon icon = new ImageIcon(getClass().getResource(pokemon.getURL()));
67                 Image imagenIcon = icon.getImage().getScaledInstance(120, 120, Image.SCALE_SMOOTH);
68
69                 JLabel imagen = new JLabel(new ImageIcon(imagenIcon));
70                 imagen.setBounds(30, 10, 120, 120);
71
72                 JLabel nombre = new JLabel(pokemon.getNombre());
73                 nombre.setBounds(0, 130, 180, 30);
74                 nombre.setHorizontalAlignment(SwingConstants.CENTER);
75             }
76             pokemonRecorrido++;
77         }
78     }
79 }
```

```

74 nombre.setHorizontalAlignment(SwingConstants.CENTER);
75 nombre.setForeground(Color.white);
76 nombre.setFont(new Font("Poppins", Font.PLAIN, 14));
77
78 RoundedPanel underName = new RoundedPanel("/elementos/FondoLabel.jpg", 140, 30);
79 underName.setBounds(20, 130, 140, 30);
80
81 //Crea un arreglo que contenga los tipos de pokemones
82 String tipo = pokemon.getTipo().toString().replace("[", "").replace("]", "");
83 String [] tipoSplit = tipo.split(", ");
84
85 int x = 60;
86 for(int k=0;k<pokemon.getTipo().size();k++){
87     String path = "/elementos/"+tipoSplit[k]+".png";
88     ImageIcon iconImage = new ImageIcon(getClass().getResource(path));
89     Image imagenTipo = iconImage.getImage().getScaledInstance(25,25,Image.SCALE_SMOOTH);
90     JLabel labelTipo = new JLabel(new ImageIcon(imagenTipo));
91     if(tipoSplit.length==2){
92         labelTipo.setBounds(x, 172, 25, 25);
93         x+=40;
94     }else{
95         labelTipo.setBounds(0, 172, 180, 25);
96     }
97
98     card.add(labelTipo);
99 }
100
101 RoundedPanel underType = new RoundedPanel("/elementos/FondoLabel.jpg", 140, 30);
102 underType.setBounds(20, 170, 140, 30);
103
104 JLabel tamaño = new JLabel(pokemon.getTamaño());
105 tamaño.setBounds(0, 210, 180, 30);
106 tamaño.setHorizontalAlignment(SwingConstants.CENTER);
107 tamaño.setForeground(Color.white);
108 tamaño.setFont(new Font("Poppins", Font.PLAIN, 14));
109
110 RoundedPanel underSize = new RoundedPanel("/elementos/FondoLabel.jpg", 140, 30);
111 underSize.setBounds(20, 210, 140, 30);

```

```

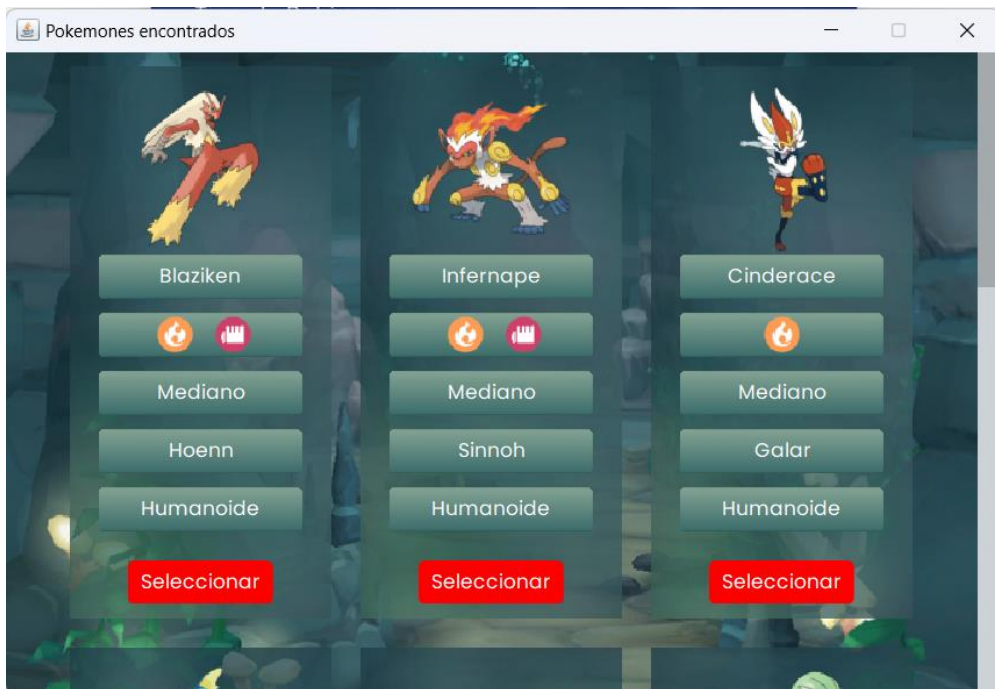
111 underSize.setBounds(20, 210, 140, 30);
112
113 JLabel region = new JLabel(pokemon.getRegion());
114 region.setBounds(0, 250, 180, 30);
115 region.setHorizontalAlignment(SwingConstants.CENTER);
116 region.setForeground(Color.white);
117 region.setFont(new Font("Poppins", Font.PLAIN, 14));
118
119 RoundedPanel underRegion = new RoundedPanel("/elementos/FondoLabel.jpg", 140, 30);
120 underRegion.setBounds(20, 250, 140, 30);
121
122 JLabel categoria = new JLabel(pokemon.getCategoria());
123 categoria.setBounds(0, 290, 180, 30);
124 categoria.setHorizontalAlignment(SwingConstants.CENTER);
125 categoria.setForeground(Color.white);
126 categoria.setFont(new Font("Poppins", Font.PLAIN, 14));
127
128 RoundedPanel underCat = new RoundedPanel("/elementos/FondoLabel.jpg", 140, 30);
129 underCat.setBounds(20, 290, 140, 30);
130
131 JButton select = new RoundedButton("Seleccionar");
132 select.setBackground(Color.red);
133 select.setForeground(Color.WHITE);
134 select.setBounds(40, 340, 100, 30);
135
136 //Funcionalidad del botón de selección
137 select.addActionListener((ActionEvent e) -> {
138     new JuegoCartas(pokemon.getNombre());
139     formulario.getInstancia().cerrar();
140     dispose();
141 });
142
143 //Agrega los elementos al card
144 card.add(imagen);
145 card.add(nombre);
146 card.add(underName);
147 card.add(underType);
148 card.add(tamaño);

```

```

149         card.add(underSize);
150         card.add(region);
151         card.add(underRegion);
152         card.add(categoria);
153         card.add(underCat);
154         card.add(select);
155         card.add(fondoCard);
156         //Agrega la card a la fila
157         fila.add(card);
158     }
159     pokemonRecorrido++;
160 }
161 }
162
163 //Agrega la fila al panel.
164 panel.add(fila);
165 }
166
167 //ScrollPane por si la lista de pokemones no entra en el panel
168 JScrollPane scroll = new CustomScroll(panel);
169
170 //Se agrega la imagen del fondo
171 scroll.add(fondo);
172 //Agrega el ScrollPane
173 add(scroll);
174 }else{
175     setSize(500,390);
176     setLocationRelativeTo(null);
177
178     //Creación del panel
179     JPanel panel = new JPanel(null);
180
181     //Creación de los elementos
182     JLabel imagen = new JLabel(new ImageIcon(getClass().getResource("/elementos/canyon.jpg")));
183     imagen.setBounds(0, 0, 500, 360);
184
185     JLabel text = new JLabel("No se ha encontrado ningun Pokémon");
186     text.setFont(new Font("Poppins", Font.BOLD, 18));
187
188     JLabel text = new JLabel("No se ha encontrado ningun Pokémon");
189     text.setFont(new Font("Poppins", Font.BOLD, 18));
190     text.setForeground(Color.white);
191     text.setBounds(60, 150, 400, 30);
192
193     JButton aceptar = new RoundedButton("Cerrar");
194     aceptar.setBackground(Color.red);
195     aceptar.setFont(new Font("Poppins", Font.PLAIN, 14));
196     aceptar.setForeground(Color.white);
197     aceptar.setBounds(195, 180, 100, 30);
198
199     aceptar.addActionListener((ActionEvent e)->{
200         dispose();
201     });
202
203     //Agrega el panel
204     panel.add(text);
205     panel.add(aceptar);
206     panel.add(imagen);
207     add(panel);
208 }
209 setVisible(true);
210 }

```



Clase Mostrar extendida de JFrame que presenta los resultados de búsqueda de Pokémon en formato de "cards". Muestra los Pokémon que coinciden con los criterios de búsqueda o un mensaje cuando no se encuentran resultados.

Contiene la configuración de la ventana como el tamaño, título, opciones de cierre, etc.

Muestra los pokemones que cumplen con los filtros y los muestra en cards separadas en tres columnas y las filas necesarias.

Cada card contiene al pokemon encontrado, mostrando tanto la imagen como las características del mismo.

Cada card contiene un botón que inicia el juego de cartas del pokemon correspondiente.

Clase JuegoCartas

```
Output - Formulario (run) x JuegoCartas.java x
1 package formulario;
2
3 //Se importan las librerías a utilizar
4 import com.google.gson.*;
5 import java.io.*;
6 import java.net.*;
7 import java.util.Random;
8 import java.util.List;
9 import java.util.ArrayList;
10 import java.awt.*;
11 import java.awt.event.ActionEvent;
12 import javax.swing.*;
13 import javax.sound.sampled.AudioInputStream;
14 import javax.sound.sampled.AudioSystem;
15 import javax.sound.sampled.Clip;
16
17 //Importar librerías personalizadas
18 import elementos.RoundedButton;
19
20 //Clase de juego de cartas
21 public class JuegoCartas extends JFrame{
22     private static final String API_URL = "https://api.pokemontcg.io/v2/cards";
23     private Clip clip;
24
25     public JuegoCartas(String nombre){
26         //Configuración de la ventana
27         setTitle(nombre + " - Cartas");
28         setSize(620,400);
29         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30         setResizable(false);
31         setLayout(new BorderLayout());
32         setLocationRelativeTo(null);
33
34         //Creación del panel
35         JPanel panel = new JPanel(null);
36         panel.setSize(620, 400);
37
38         //gif de fondo
39
40         ImageIcon iconGif = new ImageIcon(getClass().getResource("/pokemones/"+nombre+".gif"));
41         Image imageGIF = iconGif.getImage();
42
43         JLabel gif = new JLabel(new ImageIcon(imageGIF));
44         gif.setBounds(0,0,620,400);
45
46         JPanel fila = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));
47         fila.setBounds(0,20,610, 300);
48         fila.setBackground(new Color(0,0,0,0));
49
50         //Recorre la lista de cartas y las muestra
51         List<String> cartas = mostrarCartas(cardsProbability(nombre));
52
53         for(String carta : cartas){
54             String [] split = carta.split(" - ");
55             try {
56                 // URL de la imagen
57                 ImageIcon icon = new ImageIcon(new URL(split[1]));
58                 Image image = icon.getImage().getScaledInstance(180, 248, Image.SCALE_SMOOTH);
59                 icon = new ImageIcon(image);
60                 JLabel labelImg = new JLabel(icon);
61                 labelImg.setSize(180, 248);
62
63                 //Agregar la imagen al panel
64                 fila.add(labelImg);
65             } catch (Exception e) {
66             }
67         }
68
69         JButton volver = new RoundedButton("Regresar");
70         volver.setForeground(Color.white);
71         volver.setBackground(new Color(125, 120, 163));
72         volver.setBounds(250, 300, 100, 30);
73
74         volver.addActionListener((ActionEvent e)->{
75             dispose();
76             return;
77         });
78     }
79 }
```



```

78         new formulario();
79     });
80
81     //Inicia la musica
82     startMusic();
83
84     //agrega los elementos
85     panel.add(volver);
86     panel.add(fila);
87     panel.add(gif);
88
89     //Agrega el panel al frame y lo hace visible
90     add(panel);
91     setVisible(true);
92 }
93
94 //Obtiene las cartas
95 public static List<String> obtenerCartas(String nombre, String rareza) {
96     List<String> cartas = new ArrayList<>();
97     System.out.println(nombre+" "+rareza);
98     try {
99         //url
100         String urlString = API_URL + "?q=name:" + nombre;
101         if (!rareza.isEmpty()) {
102             urlString += " rarity:" + rareza;
103         }
104
105         //Conexión a la api
106         URL url = new URL(urlString);
107         HttpURLConnection conn = (HttpURLConnection) url.openConnection();
108         conn.setRequestMethod("GET");
109         conn.setRequestProperty("Accept", "application/json");
110
111         // Respuesta de petición
112         BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
113         StringBuilder response = new StringBuilder();
114         String line;
115         while ((line = br.readLine()) != null) {

```

```

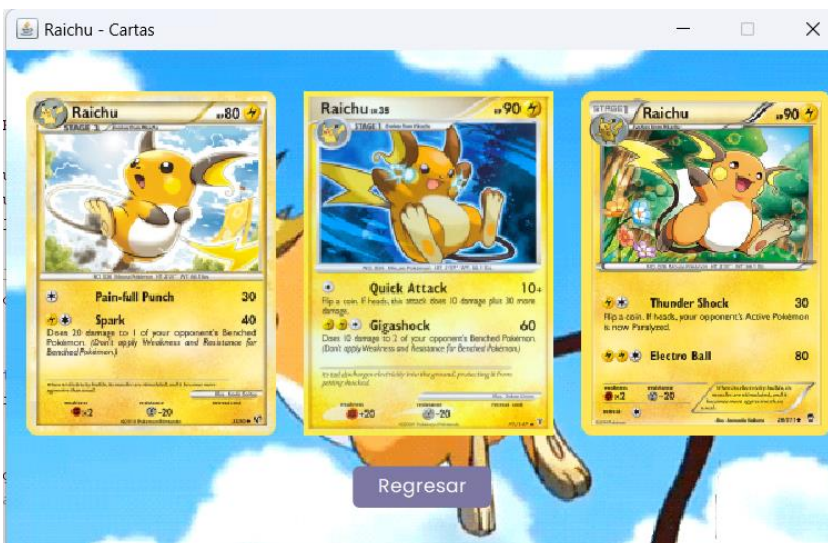
116             response.append(line);
117         }
118         br.close();
119
120         // creación de objeto JSON
121         JSONObject jsonObject = JsonParser.parseString(response.toString()).getAsJsonObject();
122         JSONArray cartasArray = jsonObject.getAsJSONArray("data");
123
124         // Obtener los valores
125         for (JsonElement carta : cartasArray) {
126             JSONObject obj = carta.getAsJsonObject();
127             String nombreCarta = obj.get("name").getAsString();
128             String imagenUrl = obj.getAsJsonObject("images").get("large").getAsString();
129             cartas.add(nombreCarta + " - " + imagenUrl);
130         }
131     } catch (Exception e) {
132         e.printStackTrace();
133     }
134     return cartas;
135 }
136
137 public List<String> cardsProbability(String nombre) {
138     int numero = new Random().nextInt(100);
139     if (numero < 40) {
140         return obtenerCartas(nombre, "Common");
141     } else if (numero < 65) {
142         return obtenerCartas(nombre, "Uncommon");
143     } else if (numero < 80) {
144         return obtenerCartas(nombre, "Rare");
145     } else if (numero < 90) {
146         return obtenerCartas(nombre, "%22Rare%20Holo%22");
147     } else if (numero < 96) {
148         return obtenerCartas(nombre, "%22Ultra%20Rare%22");
149     } else if (numero < 99) {
150         return obtenerCartas(nombre, "Rare");
151     } else {
152         return obtenerCartas(nombre, "Promo");
153     }
154 }

```

```

154
155 public List<String> mostrarCartas(List<String> cartas) {
156     List<String> cartasRandom = new ArrayList<>();
157
158     //Devuelve todas las cartas si son menores que 3
159     if(cartas.size() < 3) {
160         return cartas;
161     } else {
162
163         //Devuelve 3 cartas
164         int i = 0;
165         while(i < 3) {
166             int numero = new Random().nextInt(cartas.size());
167             cartasRandom.add(cartas.get(numero));
168             i++;
169         }
170         return cartasRandom;
171     }
172 }
173
174 //Metodo de iniciar musica
175 public void startMusic() {
176     try {
177         File musica = new File("src/elementos/pelea.wav");
178         if(musica.exists()) {
179             AudioInputStream audio = AudioSystem.getAudioInputStream(musica);
180             Clip clip = AudioSystem.getClip();
181             clip.open(audio);
182             clip.loop(Clip.LOOP_CONTINUOUSLY);
183             clip.start();
184         }
185     } catch (Exception e) {
186     }
187 }
188

```



Clase JuegoCartas heredada de JFrame que se encarga de mostrar de manera aleatoria en base a probabilidad ciertas tarjetas del Pokémon seleccionado.

Contiene la configuración de la ventana como el tamaño, titulo, opciones de cierre, etc.

Se encarga de mostrar cartas recuperadas de la API de Pokemon TCG, que realiza una consulta HTTP para obtener los datos que coinciden con el nombre y la rareza en base a la probabilidad.

Procesa los datos JSON recuperados de la petición y se encarga de mostrar hasta tres cartas aleatorias de las n encontradas.

CONCLUSIÓN

La implementación de este filtro de Pokémon demuestra la importancia de una interfaz bien estructurada y estilizada para mejorar la experiencia del usuario. A través de la personalización de los elementos de Swing, logramos un diseño atractivo y funcional que facilita la búsqueda y clasificación de Pokémon de manera intuitiva. Este proyecto nos permitió reforzar nuestros conocimientos, el manejo de eventos y la personalización de componentes gráficos. Además, nos enfrentamos a desafíos relacionados con la optimización del rendimiento y la integración de diferentes elementos visuales, lo que nos ayudó a mejorar nuestras habilidades en desarrollo de interfaces gráficas. En futuras versiones, podríamos ampliar las opciones de filtrado y mejorar aún más la interactividad del sistema para ofrecer una experiencia más completa y dinámica.