

Trabajo Interfaz Gráfico para Aplicaciones de Bioinformática

****REPOSITORIO GIT:** <https://github.com/Fiorellaps/practica1.git>

1. Introducción.

En esta práctica vamos a crear en Java Swing una GUI (Interfaz Gráfica de Usuario) que tenga un funcionamiento y apariencia similar al del programa Blast, que se encarga de buscar coincidencias entre secuencias de proteínas o de nucleótidos introducidas por el usuario.

Para ello voy utilizar los distintos elementos de Java Swing aprendidos en clase, con el objetivo de que mi programa sea lo más parecido posible al programa Blast ya mencionado. Además, empleo los documentos proporcionados en el campus, que contiene una base de datos de la secuencia de proteína de la Levadura y otro fichero con los índices de las proteínas.

2. Clase miPanelBlast.

```
public class miPanelBlast extends JPanel {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JRadioButton rp;
    private JRadioButton rn;
    private JComboBox<String> combo;
    private JTextField tPorc;
    private JButton botonB;
    private JLabel intro;
    private JLabel Lporcentaje;
    private JLabel Lsec;
    private JTextArea Tresultado;

    public miPanelBlast() {
        // añadimos el primer JPanel en el que se selecciona el tipo de secuencia a comparar
        JPanel p1 = new JPanel(new BorderLayout());
        rp = new JRadioButton("Proteína");
        rn = new JRadioButton("Nucleotido");
        //añadimos un grupo de botones
        ButtonGroup group = new ButtonGroup();
        group.add(rp);
        group.add(rn);
        //como predeterminado aparece señalado el boton de 'Proteina'
        rp.setSelected(true);
        rn.setSelected(false);
        intro = new JLabel();
        intro.setText("Elige el tipo de secuencia");
        //añadimos los componentes al panel, cada uno en su localización determinada
        p1.add(rp, BorderLayout.WEST);
        p1.add(rn, BorderLayout.EAST);
        p1.add(intro, BorderLayout.NORTH);

        // añadimos el segundo JPanel en el que se va a introducir la secuencia; el JPanel posee un GridLayout
        //para poder colocar los elementos en una celda específica
        JPanel p2= new JPanel(new GridLayout(2,1));
        // creamos un comboBox que se puede editar
        combo = new JComboBox<>();
        combo.setEditable(true);
        Lsec=new JLabel();
        Lsec.setText("Introduce la secuencia  ");
        // añadimos los componentes al panel por orden de colocación
        p2.add(Lsec);
        p2.add(combo);

        // añadimos el tercer JPanel en el que habrá un campo de texto para introducir el porcentaje de coincidencia entre las secuencias.
        JPanel p3= new JPanel(new GridLayout(2,1));
        Lporcentaje = new JLabel();
        Lporcentaje.setText("Introduce el porcentaje");
        tPorc = new JTextField(30);
        //Introducimos el rótulo y el campo de texto al tercer panel
        p3.add(Lporcentaje);
        p3.add(tPorc);
    }
}
```

```
//añadimos área de texto en el que se mostrará el resultado de la búsqueda
Tresultado= new JTextArea(50,70);
//añadimos este área a un panel que se puede deslizar
JScrollPane scroll= new JScrollPane(Tresultado);
Tresultado.setEditable(false);
//añadimos el botón de inicio de búsqueda
 botonB = new JButton("BLAST");

//añadimos todos los paneles y demás componentes al panel principal, se colocarán por orden.
add(p1);
add(p2);
add(p3);
add(botonB);
add(scroll);

}

//Creamos los metos get o set, que sean necesarios para desarrollar el resto de clases
public JTextField getTextPorc() {
    return this.tPorc;
}
public JButton getBoton() {
    return this.botonB;
}
public JComboBox<String> getCombo (){
    return this.combo;
}
public JRadioButton getRp() {
    return this.rp;
}

public JTextArea getResultado() {
    return this.Tresultado;
}
public JLabel getSecuencia(){
    return this.lsec;
}
}
```

En esta clase que hereda de JPanel, voy a implementar la ventana principal del programa. Para ello, creo los paneles que sean necesarios para después añadirlos al panel final y así, puedo colocar todos los elementos en la posición que yo quiera, cosa que sería prácticamente imposible si solo empleáramos un panel.

Antes de crear el constructor de la clase, inicializo los objetos que van a formar parte de la clase.

Como se puede observar, en el constructor principal de la clase se crea un panel que contiene todos los elementos necesarios para que el programa funcione de forma adecuada.

Posteriormente, se creará una instancia de esta clase en programa principal y en el Action Listener se emplearán los métodos para tener accesos a alguno de los componentes del panel.

3. Clase BlastActionListener.

Esta clase se encarga del funcionamiento del Action Listener que salta en respuesta a un evento, que en este caso sería el haber pulsado el botón de inicio de búsqueda. Por tanto, esta clase implementa Action Listener y el constructor principal va a tener como atributos un objeto de la clase miPanleBlast y otro de la clase BlastController que permitirá la búsqueda en la base de datos.

El código que he escrito es el siguiente:

```
public class BlastActionListener implements ActionListener {
    //objeto panel de la clase miPanleBlast
    private miPanelBlast panel;
    private static final String DataBaseFile = new String("yeast.aa");
    private static final String DataBaseIndexes = new String("yeast.aa.indexs");
    //objeto de la clase BlastController que va a permitir la búsqueda en la base de datos
    private BlastController blastC;

    public BlastActionListener(miPanelBlast e, BlastController bCnt) {
        //Incializamos componentes
        this.panel=e;
        this.blastC= bCnt;
    }
    //método para que se lleve a cabo la respuesta al evento
    public void actionPerformed(ActionEvent e) {
        //creo un JDialog para avisar a usuario cualquier error o problema que pueda surgir
        JDialog d = new JDialog();
        String seq= null;
        try {
            //guardo la secuencia introducida en una variable
            seq=panel.getCombo().getSelectedItem().toString().toLowerCase();
            //compruebo si la secuencia ya se había introducido antes
            // si no se había introducido, la añado al comboBox, en minúscula para que no haya conflicto
            if(encontrada(seq)==false) panel.getCombo().addItem(seq);

        }
        catch(NullPointerException n) {
            //error
            System.out.println("Error al introducir la secuencia");
            d.add(new JLabel("Error al introducir la secuencia"));
            d.setVisible(true);
        }

        float perc= -1;

        try {
            // guardo en una variable el porcentaje introducido
            perc= Float.parseFloat(panel.getTextPorc().getText());
            //pporcentaje introducido fuera de rango
            if(perc<0 || perc>1) {
                d.add(new JLabel("Porcentaje:"+ perc+ " introducido fuera de rango"));
                d.setVisible(true);
            }
        }
    }
}
```

```

catch(NumberFormatException nf) {
    //error en el formato
    System.out.println("Porcentaje:" + perc + " introducido con formato incorrecto");
    d.add(new JLabel("Porcentaje:" + perc + " introducido con formato incorrecto"));
    d.setVisible(true);
}

// Recojo en una variable booleana si el botón está seleccionado o no
//solo lo hago con un botón pues como solo hay dos botenes, si uno no está seleccionado, lo tiene que estar el otro]
Boolean pSelec= panel.getRp().isSelected();
if (pSelec==true) {
    //si está seleccionada la opción de 'Proteína', muestro el resultado en el área de texto
    String result = null;

    try{
        result = blastC.blastQuery('p', DataBaseFile,
                                   DataBaseIndexes, (float) perc, seq);
    } catch(Exception exc){
        //error en la búsqueda
        System.out.println("Error en la llamada: " + exc.toString());
        d.add(new JLabel("Error en la llamada"));
        d.setVisible(true);
    }
    panel.getResultado().setText(result);
    //añado el resultado de la búsqueda
}
else {
    // si la opción seleccionada es secuencia de nucleótidos, como en nuestro caso no tenemos información suficiente,
    // mostramos un comentario
    System.out.println("No se pueden buscar secuencias de nucleótidos");
    d.add(new JLabel("No se pueden buscar secuencias de nucleótidos"));
    d.setVisible(true);
}
}

//método para averiguar si una secuencia ya ha sido introducida
public boolean encontrada( String seq) {
    boolean encontrada = false;
    // recorro los elementos del combo box, si alguno coincide, sale del bucle
    // utilizo un bucle for porque hay que recorrer todos los elementos del comboBox
    for(int i = 0; i<panel.getCombo().getItemCount();i++) {
        if(seq.equals(panel.getCombo().getItemAt(i).toString())){
            encontrada=true;
        }
    }
    return encontrada;
}

```

4. Clase BlastPrincipal.

```
package miBlastGUI;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import blast.BlastController;

public class BlastPrincipal {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame frame = new JFrame("miBlast");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                // creamos el objeto panel
                miPanelBlast panel= new miPanelBlast();
                // creamos el objeto que va a permitir la búsqueda en la base de datos
                BlastController bCnt = new BlastController();
                // creamos el action listener
                BlastActionListener listener= new BlastActionListener(panel, bCnt);
                //añadimos el panel
                frame.getContentPane().add(panel);
                // añadimos el evento al boton
                panel.getBoton().addActionListener(listener);
                frame.pack();
                frame.setVisible(true);
            }
        });
    }
}
```

Esta es la clase más sencilla, pues solo se encarga de generar la ventana sobre la que se van a añadir los objetos ya creados en las otras clases. Sin embargo, es la más importante pues sin ella no podríamos interactuar con el programa.

Aunque tengo algunas dudas, creo que en este programa la clase BlastPrincipal sería el modelo de nuestro patrón MVC, pues representa los datos y reglas que permiten el acceso y las actualizaciones de los datos.

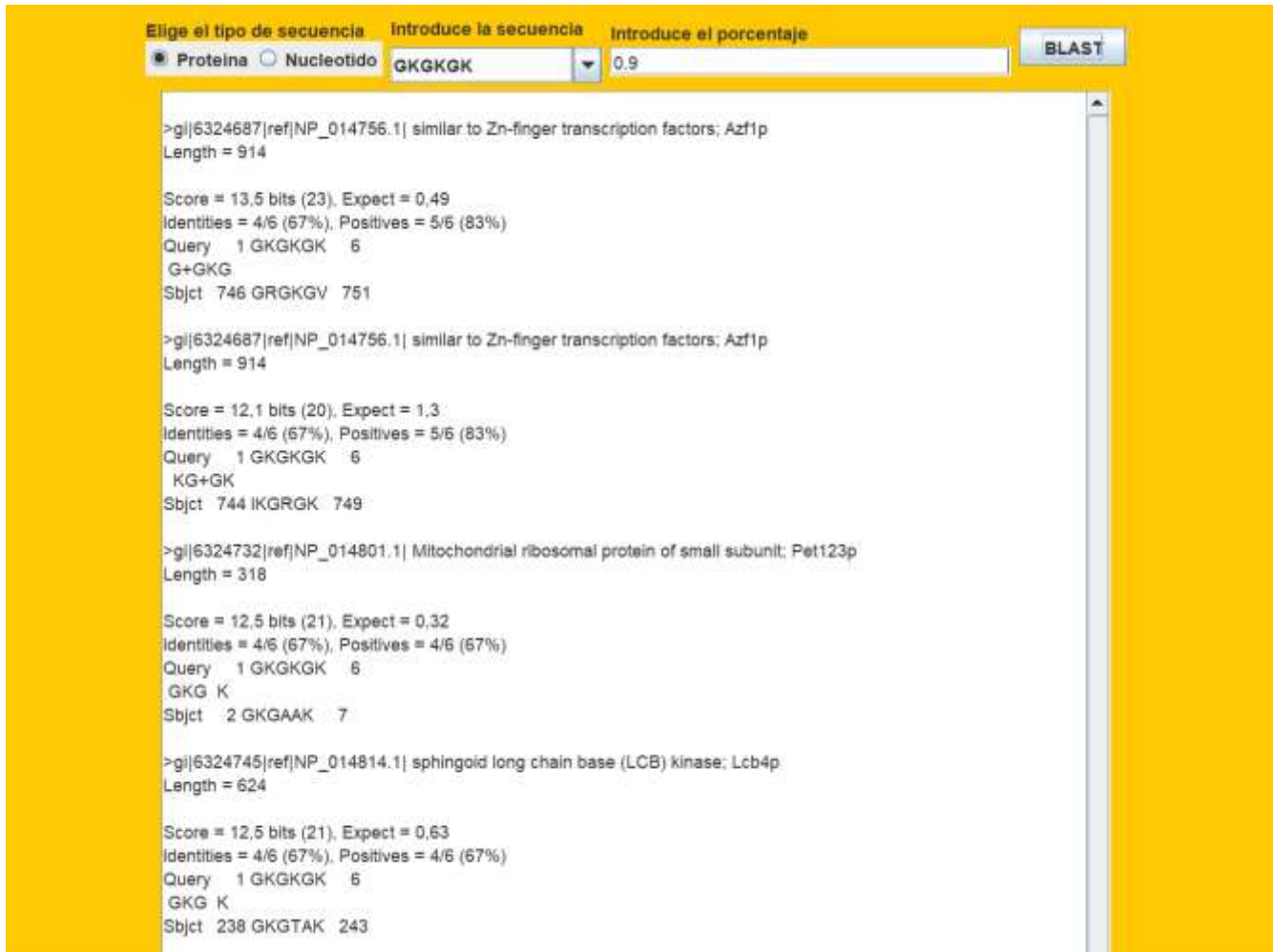
Por otra parte, la vista sería la clase MiPanelBlast, ya que representa el contenido del modelo y especifica la representación de los elementos del modelo.

Finalmente, sin duda alguna, el controlador sería la clase BlastActionListener, debido a que gracias a esta se pueden traducir las interacciones entre el usuario y la vista, las cuales se registran en el modelo. En este caso, como ya se ha comentado, la interacción sería el pulsar el botón.

Es importante que los programas tengan una estructura similar a esta (MVC) pues esto permite el desacoplamiento entre el acceso a los datos y la forma en que se muestran los datos. Además, permite que se separe claramente dónde va cada tipo de lógica, permitiendo el mantenimiento y la escalabilidad de nuestra aplicación.

1. Programa

Ahora vamos a ejecutar el programa y vamos a ver su funcionamiento principal.



The screenshot shows the BLAST web interface with the following fields and results:

- Elige el tipo de secuencia:** Proteína (selected), Nucleótido
- Introduce la secuencia:** GKGK GK
- Introduce el porcentaje:** 0.9
- BLAST button:** Present

The results area displays four matches:

- >gi|6324687|ref|NP_014756.1| similar to Zn-finger transcription factors; Azf1p
Length = 914

Score = 13,5 bits (23), Expect = 0,49
Identities = 4/6 (67%), Positives = 5/6 (83%)
Query 1 GKGK GK 6
G+GKG
Sbjct 746 GRGKGV 751
- >gi|6324687|ref|NP_014756.1| similar to Zn-finger transcription factors; Azf1p
Length = 914

Score = 12,1 bits (20), Expect = 1,3
Identities = 4/6 (67%), Positives = 5/6 (83%)
Query 1 GKGK GK 6
KG+GK
Sbjct 744 IKGRGK 749
- >gi|6324732|ref|NP_014801.1| Mitochondrial ribosomal protein of small subunit; Pet123p
Length = 318

Score = 12,5 bits (21), Expect = 0,32
Identities = 4/6 (67%), Positives = 4/6 (67%)
Query 1 GKGK GK 6
GKG K
Sbjct 2 GKGA AK 7
- >gi|6324745|ref|NP_014814.1| sphingoid long chain base (LCB) kinase; Lcb4p
Length = 624

Score = 12,5 bits (21), Expect = 0,63
Identities = 4/6 (67%), Positives = 4/6 (67%)
Query 1 GKGK GK 6
GKG K
Sbjct 238 KGKTAK 243

Entramos en el programa, introducimos la secuencia, el porcentaje y seleccionamos el tipo de secuencia. Al pulsar el botón Blast, el área de texto vacío inicialmente, se rellena con la respuesta.

Si introdujéramos un número erróneo o seleccionáramos el tipo de secuencia nucleótido, ocurriría lo siguiente: Se abre una ventana indicando el error



The screenshot shows the BLAST web interface with an error message displayed in a small window:

- Elige el tipo de secuencia:** Proteína (selected), Nucleótido
- Introduce la secuencia:** GKGK GK
- Introduce el porcentaje:** 1.1
- BLAST button:** Present

The error message window displays:

Porcentaje: 1.1 fuera de rango.
Error en la llamada.