

# REPORTE

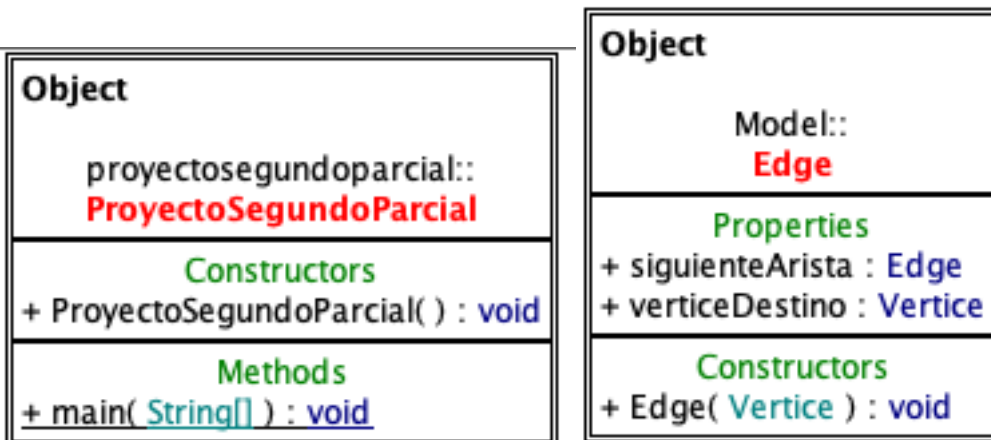
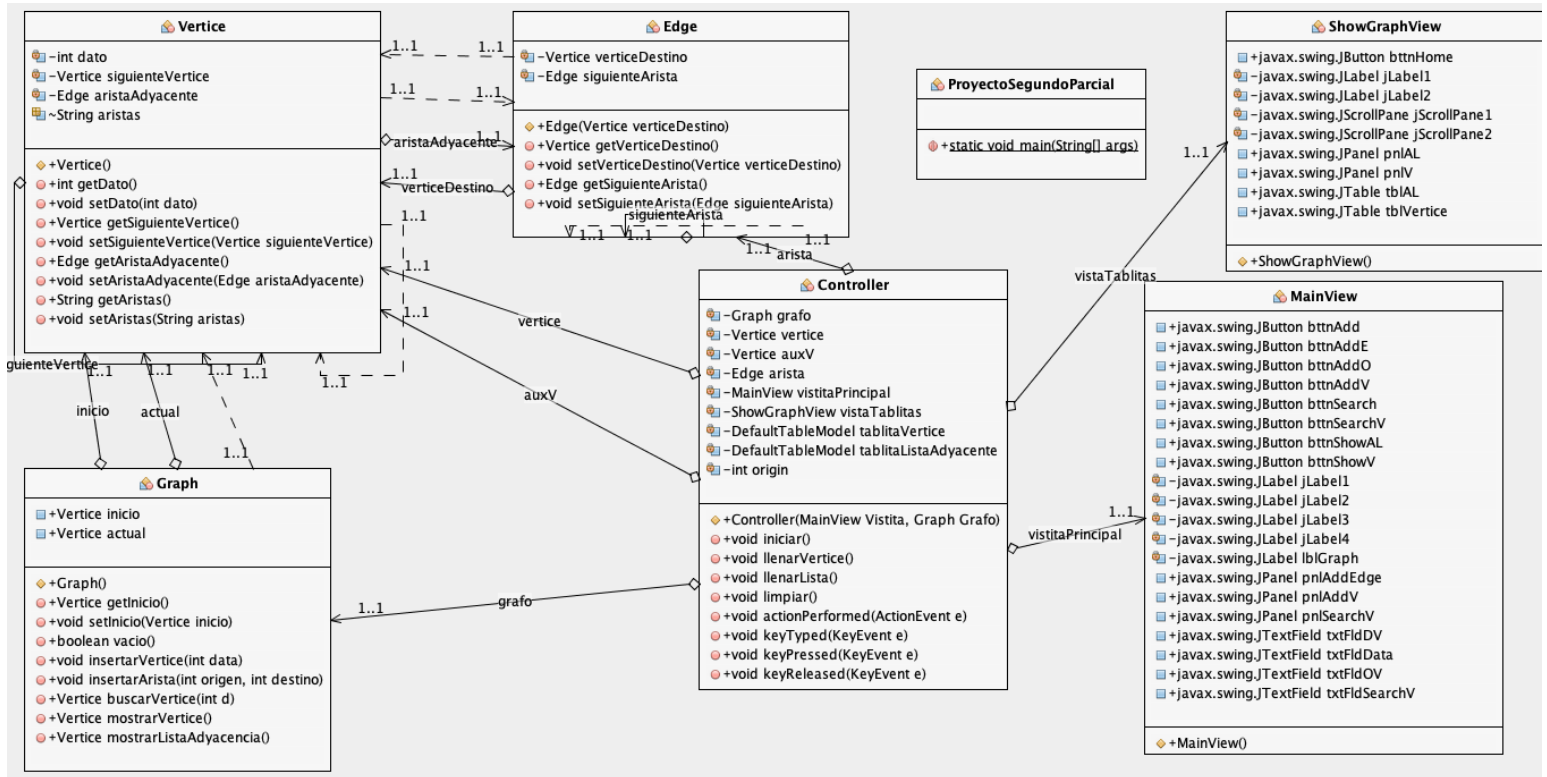
PROYECTO SEGUNDO PARCIAL

ESTRUCTURA DE DATOS AVANZADAS



Universidad Politécnica de Pachuca  
Ingeniería en Software  
94533  
Giselle Medina Maximiano  
1931117595

# DIAGRAMA DE CLASES



<b>Object</b>
Model:: <b>Vertice</b>
<b>Properties</b> + aristaAdyacente : <b>Edge</b> + aristas : <b>String</b> + dato : <b>int</b> + siguienteVertice : <b>Vertice</b>
<b>Constructors</b> + Vertice( ) : <b>void</b>

<b>Object</b>
Model:: <b>Graph</b>
<b>Fields</b> + actual : <b>Vertice</b>
<b>Properties</b> + inicio : <b>Vertice</b>
<b>Constructors</b> + Graph( ) : <b>void</b>
<b>Methods</b> + buscarVertice( <b>int</b> ) : <b>Vertice</b> + insertarArista( <b>int</b> , <b>int</b> ) : <b>void</b> + insertarVertice( <b>int</b> ) : <b>void</b> + mostrarListaAdyacencia( ) : <b>Vertice</b> + mostrarVertice( ) : <b>Vertice</b> + vacio( ) : <b>boolean</b>

<b>Object</b>
Controller:: <b>Controller</b>
<b>Fields</b> - arista : <b>Edge</b> - auxV : <b>Vertice</b> - destiny : <b>int</b> - grafo : <b>Graph</b> - origin : <b>int</b> - tablitaListaAdyacente : <b>DefaultTableModel</b> - tablitaVertice : <b>DefaultTableModel</b> - vertice : <b>Vertice</b> - vistaTablitas : <b>ShowGraphView</b> - visitaPrincipal : <b>MainView</b>
<b>Constructors</b> + Controller( <b>MainView</b> , <b>Graph</b> ) : <b>void</b>
<b>Methods</b> + actionPerformed( <b>ActionEvent</b> ) : <b>void</b> + iniciar( ) : <b>void</b> + keyPressed( <b>KeyEvent</b> ) : <b>void</b> + keyReleased( <b>KeyEvent</b> ) : <b>void</b> + keyTyped( <b>KeyEvent</b> ) : <b>void</b> + limpiar( ) : <b>void</b> + llenarLista( ) : <b>void</b> + llenarVertice( ) : <b>void</b>

<b>JFrame</b>
View:: <b>ShowGraphView</b>
<b>Fields</b> + btnHome : <b> JButton</b> - jLabel1 : <b> JLabel</b> - jLabel2 : <b> JLabel</b> - jScrollPane1 : <b> JScrollPane</b> - jScrollPane2 : <b> JScrollPane</b> + pnlAL : <b> JPanel</b> + pnlV : <b> JPanel</b> + tblAL : <b> JTable</b> + tblVertice : <b> JTable</b>
<b>Constructors</b> + ShowGraphView( ) : <b>void</b>
<b>Methods</b> - initComponents( ) : <b>void</b> + main( <b>String[]</b> ) : <b>void</b>

## JFrame

View::

**MainView**

### Fields

- + btnAdd : JButton
- + btnAddE : JButton
- + btnAddO : JButton
- + btnAddV : JButton
- + btnSearch : JButton
- + btnSearchV : JButton
- + btnShowAL : JButton
- + btnShowV : JButton
- jLabel1 : JLabel
- jLabel2 : JLabel
- jLabel3 : JLabel
- jLabel4 : JLabel
- lblGraph : JLabel
- + pnlAddEdge : JPanel
- + pnlAddV : JPanel
- + pnlSearchV : JPanel
- + txtFldDV : JTextField
- + txtFldData : JTextField
- + txtFldOV : JTextField
- + txtFldSearchV : JTextField

### Constructors

- + MainView( ) : void

### Methods

- initComponents( ) : void
- + main( String[] ) : void

# CÓDIGO DE LAS CLASES

**En el paquete de "MODELO" (Model):**

**Clase Vertice:**

```
package Model;
```

```
/**
```

```
 *ProyectoSegundoParcial
```

```
 * Programa que hace un grafo
```

```
 * 29 de julio de 2020
```

```
 * @author Giselle Medina
```

```
 * Versión 1.0
```

```
 */
```

```
public class Vertice {  
    private int dato;  
    private Vertice siguienteVertice;  
    private Edge aristaAdyacente;  
    String aristas="";
```

```
    public Vertice() {  
        this.dato = 0;  
        this.siguienteVertice = null;  
        this.aristaAdyacente = null;  
    }
```

```
    public int getDato() {  
        return dato;  
    }
```

```
    public void setDato(int dato) {  
        this.dato = dato;  
    }
```

```
    public Vertice getSiguienteVertice() {  
        return siguienteVertice;  
    }
```

```
    public void setSiguienteVertice(Vertice siguienteVertice) {  
        this.siguienteVertice = siguienteVertice;  
    }
```

```
    public Edge getAristaAdyacente() {
```

```

        return aristaAdyacente;
    }

    public void setAristaAdyacente(Edge aristaAdyacente) {
        this.aristaAdyacente = aristaAdyacente;
    }

    public String getAristas(){
        return aristas;
    }

    public void setAristas(String aristas){
        this.aristas=aristas;
    }
}

```

### **Clase Graph:**

```

package Model;

import javax.swing.JOptionPane;

/**
 *ProyectoSegundoParcial
 * Programa que hace un grafo
 * 29 de julio de 2020
 * @author Giselle Medina
 * Versión 1.0
 */

public class Graph {
    public Vertice inicio;
    public Vertice actual;

    //Constructor
    public Graph() {
        this.inicio = null;
    }

    public Vertice getInicio() {
        return inicio;
    }

    public void setInicio(Vertice inicio) {

```

```

        this.inicio = inicio;
    }

    /**
     * Nombre de la función: vacio
     * Descripción: Comprueba si hay
     * algún grafo ingresado
     * @return boolean
     */

    public boolean vacio(){
        if(this.inicio==null){
            return true;
        }else{
            return false;
        }
    }

    /**
     * Nombre de la función: insertarVertice
     * Descripción: función que inserta
     * vértices al grafo
     * @param data
     */

    public void insertarVertice(int data){
        if(vacio()==true){
            Vertice nuevo=new Vertice();
            nuevo.setDato(data);
            setInicio(nuevo);
        }else{
            Vertice nuevo=new Vertice();
            nuevo.setDato(data);
            Vertice aux=getInicio();
            while(aux.getSiguienteVertice()!=null){
                aux=aux.getSiguienteVertice();
            }
            aux.setSiguienteVertice(nuevo);
        }
    }

    /**
     * Nombre de la función: insertarArista
     * Descripción: función que inserta aristas

```

```

* al grafo
* @param origen
* @param destino
*/

public void insertarArista(int origen, int destino){
    Vertice origin, destiny;
    origin=buscarVertice(origen);
    destiny=buscarVertice(destino);
    if(origin==null || destiny==null){
        JOptionPane.showMessageDialog(null,"No se puede insertar la arista");
    } else{
        //origin.setAristaAdyacente marca el inicio de la lista de aristas
        Edge nuevaArista = new Edge(destiny);
        if(origin.getAristaAdyacente()==null){
            origin.setAristaAdyacente(nuevaArista);
        } else{
            Edge aux=origin.getAristaAdyacente();
            while(aux.getSiguienteArista()!=null){
                aux=aux.getSiguienteArista();
            }
            aux.setSiguienteArista(nuevaArista);
        }
        JOptionPane.showMessageDialog(null,"¡La arista ha sido añadida con éxito!");
    }
}

/**
* Nombre de la función: buscarVertice
* Descripción: función que recibe un
* valor (d) y busca ese valor en el grafo
* @param d
* @return vertice
*/
public Vertice buscarVertice(int d){
    Vertice actual = this.inicio;
    if(vacio()==true){
        return null;
    } else{
        while(actual!=null && actual.getDato()!=d){
            actual=actual.getSiguienteVertice();
        }
    }
}

```



```

        return actual;
    }
}

/**
 * Nombre de la función: mostrarVertice
 * Descripción: función que muestra los
 * vértices en el grafo
 * @return vertice
 */

public Vertice mostrarVertice(){
    Vertice auxiliar=new Vertice();
    auxiliar=actual;
    if(vacio()){
        JOptionPane.showMessageDialog(null,"No hay vertices");
    }else{
        if(auxiliar!=null){
            auxiliar.getDato();
        }
    }
    this.actual=auxiliar.getSiguienteVertice();
    return auxiliar;
}

/**
 * Nombre de la función: mostrarListaAdyacencia
 * Descripción: función que muestra la relación
 * entre los vértices del grafo
 * @return vertice
 */

public Vertice mostrarListaAdyacencia(){
    Vertice verticeActual=new Vertice();
    verticeActual=actual;
    if(vacio()){
        JOptionPane.showMessageDialog(null,"No hay ningun vertice");
    }else{
        if(verticeActual!=null){
            verticeActual.getDato();

            //Inicia recorrido de aristas
            Edge aristaActual=verticeActual.getAristaAdyacente());

```

```

        while(aristaActual!=null){

verticeActual.setAristas(verticeActual.aristas+aristaActual.getVerticeDestino().getDato()+"
->");
            aristaActual=aristaActual.getSiguienteArista();
        }
    }
    }
    this.actual=verticeActual.getSiguienteVertice();
    return verticeActual;
}
}

```

### **CLASE Edge:**

```
package Model;
```

```
/**
```

```

 *ProyectoSegundoParcial
 * Programa que hace un grafo
 * 29 de julio de 2020
 * @author Giselle Medina
 * Versión 1.0
 */

```

```

public class Edge {
    private Vertice verticeDestino;
    private Edge siguienteArista;

    public Edge(Vertice verticeDestino) {
        this.verticeDestino = verticeDestino;
        this.siguienteArista = null;
    }

    public Vertice getVerticeDestino() {
        return verticeDestino;
    }

    public void setVerticeDestino(Vertice verticeDestino) {
        this.verticeDestino = verticeDestino;
    }

    public Edge getSiguienteArista() {
        return siguienteArista;
    }
}

```

```

    public void setSiguienteArista(Edge siguienteArista) {
        this.siguienteArista = siguienteArista;
    }
}

```

## En el paquete de “Vista” (view):

### Clase Mainview

```

package View;

/**
 * ProyectoSegundoParcial
 * Programa que hace un grafo
 * 29 de julio de 2020
 * @author Giselle Medina
 * Versión 1.0
 */
public class ShowGraphView extends javax.swing.JFrame {

    /**
     * Creates new form ShowGraphView
     */
    public ShowGraphView() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        pnlV = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        tblVertice = new javax.swing.JTable();
        btnHome = new javax.swing.JButton();
        pnlAL = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();

    }
}

```

```

tblAL = new javax.swing.JTable();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 15)); // NOI18N
jLabel1.setText("Vértices");

tblVertice.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null},
        {null},
        {null},
        {null}
    },
    new String [] {
        "Vertice"
    }
));
jScrollPane1.setViewportView(tblVertice);

javax.swing.GroupLayout pnlVLayout = new javax.swing.GroupLayout(pnlV);
pnlV.setLayout(pnlVLayout);
pnlVLayout.setHorizontalGroup(
    pnlVLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(pnlVLayout.createSequentialGroup()
            .add(pnlVLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(jLabel1)
                .add(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 170,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(0, 11, Short.MAX_VALUE))
        );
pnlVLayout.setVerticalGroup(
    pnlVLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .add(pnlVLayout.createSequentialGroup()
            .add(jLabel1)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .add(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 240,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

```

```

        btnHome.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Icons/btnnHome.png"))); // NOI18N
        btnHome.setToolTipText("Inicio");
        btnHome.setActionCommand("Home");

jLabel2.setFont(new java.awt.Font("Lucida Grande", 0, 15)); // NOI18N
jLabel2.setText("Lista de Adyacencia");

tblAL.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null},
        {null, null},
        {null, null},
        {null, null}
    },
    new String [] {
        "Vertice Origen", "Vertice Destino"
    }
));
jScrollPane2.setViewportView(tblAL);

javax.swing.GroupLayout pnlALLayout = new javax.swing.GroupLayout(pnlAL);
pnlAL.setLayout(pnlALLayout);
pnlALLayout.setHorizontalGroup(
    pnlALLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(pnlALLayout.createSequentialGroup()
            .add(pnlALLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(jLabel2)
                .add(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 180,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap())
);
pnlALLayout.setVerticalGroup(
    pnlALLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(pnlALLayout.createSequentialGroup()
            .add(jLabel2)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .add(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 240,
javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(pnlV, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(pnlAL, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createSequentialGroup()
                .addGap(146, 146, 146)
                .addComponent(btnHome))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
                    false)
                    .addComponent(pnlAL, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(pnlV, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(btnHome)
                .addContainerGap(14, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">

```

```
/* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
```

```
 * For details see
```

```
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
```

```
 */
```

```
try {
    for (javax.swing.UIManager.LookAndFeelInfo info :
        javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
```

```
    java.util.logging.Logger.getLogger(ShowGraphView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
```

```
    java.util.logging.Logger.getLogger(ShowGraphView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
```

```
    java.util.logging.Logger.getLogger(ShowGraphView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
    java.util.logging.Logger.getLogger(ShowGraphView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>
```

```
/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new ShowGraphView().setVisible(true);
    }
});
}
```

```
// Variables declaration - do not modify
public javax.swing.JButton btnHome;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
```

```

private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
public javax.swing.JPanel pnlAL;
public javax.swing.JPanel pnlV;
public javax.swing.JTable tblAL;
public javax.swing.JTable tblVertice;
// End of variables declaration
}

```

## Clase ShowGraphView

```
package View;
```

```

/**
 * ProyectoSegundoParcial
 * Programa que hace un grafo
 * 29 de julio de 2020
 * @author Giselle Medina
 * Versión 1.0
 */
public class ShowGraphView extends javax.swing.JFrame {

    /**
     * Creates new form ShowGraphView
     */
    public ShowGraphView() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        pnlV = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        tblVertice = new javax.swing.JTable();
        btnHome = new javax.swing.JButton();
        pnlAL = new javax.swing.JPanel();
    }
}

```



```

jLabel2 = new javax.swing.JLabel();
jScrollPane2 = new javax.swing.JScrollPane();
tblAL = new javax.swing.JTable();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 15)); // NOI18N
jLabel1.setText("Vértices");

tblVertice.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null},
        {null},
        {null},
        {null}
    },
    new String [] {
        "Vertice"
    }
));
jScrollPane1.setViewportView(tblVertice);

javax.swing.GroupLayout pnlVLayout = new javax.swing.GroupLayout(pnlV);
pnlV.setLayout(pnlVLayout);
pnlVLayout.setHorizontalGroup(
    pnlVLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(pnlVLayout.createSequentialGroup()
            .add(pnlVLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 170,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .add(jLabel1))
            .addContainerGap(11, Short.MAX_VALUE))
        .addGroup(pnlVLayout.createSequentialGroup()
            .add(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 240,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .add(jLabel2)
            .addContainerGap(11, Short.MAX_VALUE))
);

```

```

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    btnHome.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/icons/btnHome.png"))); // NOI18N
    btnHome.setToolTipText("Inicio");
    btnHome.setActionCommand("Home");

    jLabel2.setFont(new java.awt.Font("Lucida Grande", 0, 15)); // NOI18N
    jLabel2.setText("Lista de Adyacencia");

    tblAL.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null},
            {null, null},
            {null, null},
            {null, null}
        },
        new String [] {
            "Vertice Origen", "Vertice Destino"
        }
    ));
    jScrollPane2.setViewportViewView(tblAL);

    javax.swing.GroupLayout pnlALLayout = new javax.swing.GroupLayout(pnlAL);
    pnlAL.setLayout(pnlALLayout);
    pnlALLayout.setHorizontalGroup(
        pnlALLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(pnlALLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(pnlALLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    G)
            .addComponent(jLabel2)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 180,
    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(14, Short.MAX_VALUE))
    );
    pnlALLayout.setVerticalGroup(
        pnlALLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(pnlALLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 240,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(pnlV, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(pnlAL, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup()
            .addGap(146, 146, 146)
            .addComponent(btnHome))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(pnlAL, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(pnlV, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(btnHome)
            .addContainerGap(14, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

```

```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */

```

```

    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger>ShowGraphView.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger>ShowGraphView.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger>ShowGraphView.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger>ShowGraphView.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ShowGraphView().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
public javax.swing.JButton btnHome;

```

```

private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
public javax.swing.JPanel pnlAL;
public javax.swing.JPanel pnlV;
public javax.swing.JTable tblAL;
public javax.swing.JTable tblVertice;
// End of variables declaration
}

```

## **En el paquete de “Controlador” (Controller): Clase Controller**

```

package Controller;

/**
 * ProyectoSegundoParcial
 * Programa que hace un grafo
 * 29 de julio de 2020
 * @author Giselle Medina
 * Versión 1.0
 */

import Model.Edge;
import Model.Graph;
import Model.Vertice;
import View.MainView;
import View.ShowGraphView;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class Controller implements ActionListener, KeyListener{
    private Graph grafo = new Graph();
    private Vertice vertice=new Vertice();
    private Vertice auxV;
    private Edge arista = new Edge(vertice);
    private MainView visitaPrincipal=new MainView();
    private ShowGraphView vistaTablitas;

```

```
private DefaultTableModel tablitaVertice;  
private DefaultTableModel tablitaListaAdyacente;  
private int origin,destiny;
```

```
//Constructor
```

```
public Controller(MainView Vistita, Graph Grafo) {  
    this.vistaTablitas = new ShowGraphView();  
    this.vistitaPrincipal = Vistita;  
    this.vistitaPrincipal.setVisible(true);  
    this.grafo = Grafo;  
    this.vistitaPrincipal.btnAdd.addActionListener(this);  
    this.vistitaPrincipal.btnAddV.addActionListener(this);  
    this.vistitaPrincipal.btnAddO.addActionListener(this);  
    this.vistitaPrincipal.btnAddE.addActionListener(this);  
    this.vistitaPrincipal.btnAddSearch.addActionListener(this);  
    this.vistitaPrincipal.btnAddSearchV.addActionListener(this);  
    this.vistitaPrincipal.btnAddShowAL.addActionListener(this);  
    this.vistitaPrincipal.btnAddShowV.addActionListener(this);  
    this.vistaTablitas.btnAddHome.addActionListener(this);  
    this.vistitaPrincipal.txtFldData.addKeyListener(this);  
    this.vistitaPrincipal.txtFldOV.addKeyListener(this);  
    this.vistitaPrincipal.txtFldDV.addKeyListener(this);  
    this.vistitaPrincipal.txtFldSearchV.addKeyListener(this);  
}
```

```
/**
```

```
 * Nombre de la función: iniciar  
 * Descripción: función para inicializar  
 * el programa con ciertos ajustes.  
 */
```

```
public void iniciar(){  
    this.vistitaPrincipal.setVisible(true);  
    this.vistitaPrincipal.setTitle("Grafo");  
    this.vistitaPrincipal.setSize(290,300);  
    this.vistitaPrincipal.setResizable(false);  
    this.vistitaPrincipal.setLocationRelativeTo(null);  
    this.vistaTablitas.setLocationRelativeTo(null);  
    this.vistaTablitas.setVisible(false);  
    this.vistitaPrincipal.pnlAddV.setVisible(false);  
    this.vistitaPrincipal.pnlAddEdge.setVisible(false);  
    this.vistitaPrincipal.pnlSearchV.setVisible(false);  
    this.vistaTablitas.pnlV.setVisible(false);  
    this.vistaTablitas.pnlAL.setVisible(false);  
}
```

```
}
```

```
/**
```

```
* Nombre de la función: llenarVertice  
* Descripción: función que llena el  
* grafo con vértices y crea una tabla  
* para que esta muestre los vértices del grafo.  
*/
```

```
public void llenarVertice(){  
    String []data= new String[1];  
    tablitaVertice=new DefaultTableModel();  
    this.tablitaVertice.addColumn("Vertice");  
    while(this.grafo.actual!=null){  
        this.vertice=this.grafo.mostrarVertice();  
        data[0]=String.valueOf(this.vertice.getDato());  
        this.tablitaVertice.addRow(data);  
        vistaTablitas.tblVertice.setModel(tablitaVertice);  
    }  
}
```

```
/**
```

```
* Nombre de la función: llenarLista  
* Descripción: función que llena la  
* lista de adyacencia y crea una tabla  
* para que esta muestre la relación del grafo.  
*/
```

```
public void llenarLista(){  
    String []lista= new String [2];  
    tablitaListaAdyacente=new DefaultTableModel();  
    this.tablitaListaAdyacente.addColumn("Vertice origen");  
    this.tablitaListaAdyacente.addColumn("Vertice destino");  
    while(this.grafo.actual!=null){  
        vertice.setAristas("");  
        this.vertice=this.grafo.mostrarListaAdyacencia();  
        lista[0]=String.valueOf(vertice.getDato());  
        lista[1]=String.valueOf(vertice.getAristas());  
        this.tablitaListaAdyacente.addRow(lista);  
        vistaTablitas.tblAL.setModel(tablitaListaAdyacente);  
    }  
}
```

```
/**
```

```
* Nombre de la función: limpiar  
* Descripción: función que establece
```

```

* los text fields en limpio.
*/
public void limpiar(){
    this.vistitaPrincipal.txtFldData.setText("");
    this.vistitaPrincipal.txtFldOV.setText("");
    this.vistitaPrincipal.txtFldDV.setText("");
    this.vistitaPrincipal.txtFldSearchV.setText("");
    this.vistitaPrincipal.setTitle("Grafo");
    this.vistitaPrincipal.setSize(290,300);
    this.vistitaPrincipal.pnlAddV.setVisible(false);
    this.vistitaPrincipal.pnlSearchV.setVisible(false);
    this.vistitaPrincipal.pnlAddEdge.setVisible(false);
}

@Override
public void actionPerformed(ActionEvent e) {
    try{
        String botoncito=e.getActionCommand();
        switch(botoncito){
            case "AddVertice":
                this.vistitaPrincipal.setSize(455,300);
                this.vistitaPrincipal.setTitle("Añadir vértice");
                this.vistitaPrincipal.pnlAddV.setVisible(true);
                this.vistitaPrincipal.pnlSearchV.setVisible(false);
                this.vistitaPrincipal.pnlAddEdge.setVisible(false);
                break;
            case "AddV":
                this.grafo.inicio=this.grafo.getInicio();
                try{
                    int verticeAux=Integer.parseInt(this.vistitaPrincipal.txtFldData.getText());
                    if(!this.vistitaPrincipal.txtFldData.getText().isEmpty()){
                        try{
                            this.grafo.insertarVertice(verticeAux);

                            JOptionPane.showMessageDialog(null,"¡El vértice ha sido añadido con
éxito!");
                        }catch(Exception ex){
                            JOptionPane.showMessageDialog(null,"¡Error! No se ha podido realizar
la acción");
                        }
                    }else {
                        JOptionPane.showMessageDialog(null,"Ingresar información del vértice");
                    }
                }catch(Exception x){

```



```

        JOptionPane.showMessageDialog(null, "El vértice no ha sido añadido.
\nIntente de nuevo más tarde.");
    }
    limpiar();
    break;
case "AddEdge":
    this.vistitaPrincipal.setSize(290,470);
    this.vistitaPrincipal.setTitle("Añadir arista");
    this.vistitaPrincipal.pnlAddV.setVisible(false);
    this.vistitaPrincipal.pnlAddEdge.setVisible(true);
    this.vistitaPrincipal.pnlSearchV.setVisible(false);
    break;
case "AddE":
    if(!this.vistitaPrincipal.txtFldOV.getText().isEmpty() &&
        !this.vistitaPrincipal.txtFldDV.getText().isEmpty()){
        int dest=Integer.parseInt(this.vistitaPrincipal.txtFldDV.getText());
        int ori=Integer.parseInt(this.vistitaPrincipal.txtFldOV.getText());
        try{
            this.grafo.insertarArista(ori,dest);

        }catch(Exception x){
            JOptionPane.showMessageDialog(null, "La arista no ha sido añadida.
\nIntente de nuevo más tarde.");
        }
    }else {
        JOptionPane.showMessageDialog(null,"Ingresar información de la arista");
    }
    limpiar();
    break;
case "SearchVertice":
    this.vistitaPrincipal.setSize(475,450);
    this.vistitaPrincipal.setTitle("Buscar vértice");
    this.vistitaPrincipal.pnlSearchV.setVisible(true);
    this.vistitaPrincipal.pnlAddV.setVisible(false);
    this.vistitaPrincipal.pnlAddEdge.setVisible(false);
    break;
case "Search":
    try{
        if(!this.vistitaPrincipal.txtFldSearchV.getText().isEmpty()){
            int buscarG=Integer.parseInt(this.vistitaPrincipal.txtFldSearchV.getText());
            if(this.grafo.buscarVertice(buscarG)==null){
                JOptionPane.showMessageDialog(null,"No se encontró el vertice");
            }else{

```

```

        JOptionPane.showMessageDialog(null,"El vertice buscado es:
"+grafo.buscarVertice(buscarG).getDato());
    }
    }else {
        JOptionPane.showMessageDialog(null,"Favor de ingresar vertice");
    }
    }catch(Exception x){
        JOptionPane.showMessageDialog(null, "No se ha podido buscar el vértice.
\nIntente de nuevo más tarde.");
    }
    limpiar();
    break;
case "ShowVertice":
    this.vistitaPrincipal.setVisible(false);
    this.vistaTablitas.setVisible(true);
    this.vistitaPrincipal.setTitle("Vértices");
    this.vistitaPrincipal.pnlAddEdge.setVisible(false);
    this.vistitaPrincipal.pnlAddV.setVisible(false);
    this.vistitaPrincipal.pnlAddEdge.setVisible(false);
    this.vistaTablitas.pnlV.setVisible(true);
    this.vistaTablitas.pnlAL.setVisible(false);
    this.grafo.actual=this.grafo.getInicio();
    llenarVertice();
    break;
case "ShowAdjacencyList":
    this.vistitaPrincipal.setVisible(false);
    this.vistaTablitas.setVisible(true);
    this.vistitaPrincipal.setTitle("Lista de adyacencia");
    this.vistitaPrincipal.pnlAddEdge.setVisible(false);
    this.vistitaPrincipal.pnlAddV.setVisible(false);
    this.vistitaPrincipal.pnlAddEdge.setVisible(false);
    this.vistaTablitas.pnlV.setVisible(false);
    this.vistaTablitas.pnlAL.setVisible(true);
    this.grafo.actual=this.grafo.getInicio();
    llenarLista();
    break;
case "Home":
    iniciar();
    break;

    }
    }catch(Exception x){
        JOptionPane.showMessageDialog(null, "Error, hay un problema. \nIntente de nuevo
más tarde");
    }

```

```

    }
}
@Override
public void keyTyped(KeyEvent e) {
    char letra = e.getKeyChar();
    if(Character.isAlphabetic(letra)){
        e.consume();
        this.vistitaPrincipal.getToolkit().beep();
        JOptionPane.showMessageDialog(null, "Caracter inválido");
    }
}

@Override
public void keyPressed(KeyEvent e) {
    //throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
}

@Override
public void keyReleased(KeyEvent e) {
    //throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
}

}

```

## En el paquete de “ProyectoSegundoParcial”: Clase ProyectoSegundoParcial (Main)

```

package proyectoSegundoParcial;

/**
 * ProyectoSegundoParcial
 * Programa que hace un grafo
 * 29 de julio de 2020
 * @author Giselle Medina
 * Versión 1.0
 */
import View.MainView;
import Controller.Controller;
import Model.Graph;

public class ProyectoSegundoParcial {
    public static void main(String[] args) {

```

```
Graph theModel=new Graph();
MainView theView = new MainView();
Controller theController = new Controller(theView, theModel);
theView.setVisible(true);
theController.iniciar();
}

}
```




# CAPTURAS DE PANTALLA

## VISTA PRINCIPAL

♥ GRAFO ♥




Dato:



Vértice origen:


Vértice destino:

Vértice a buscar:

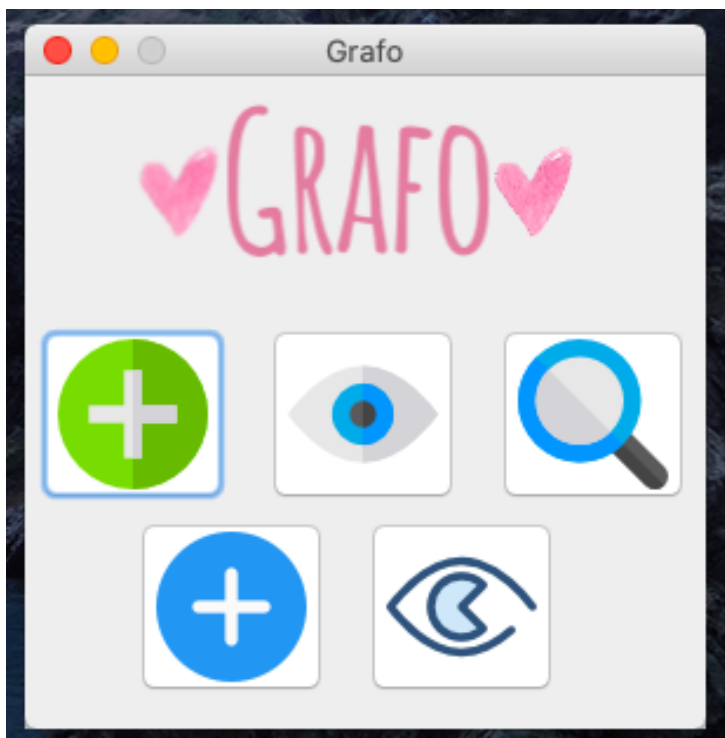


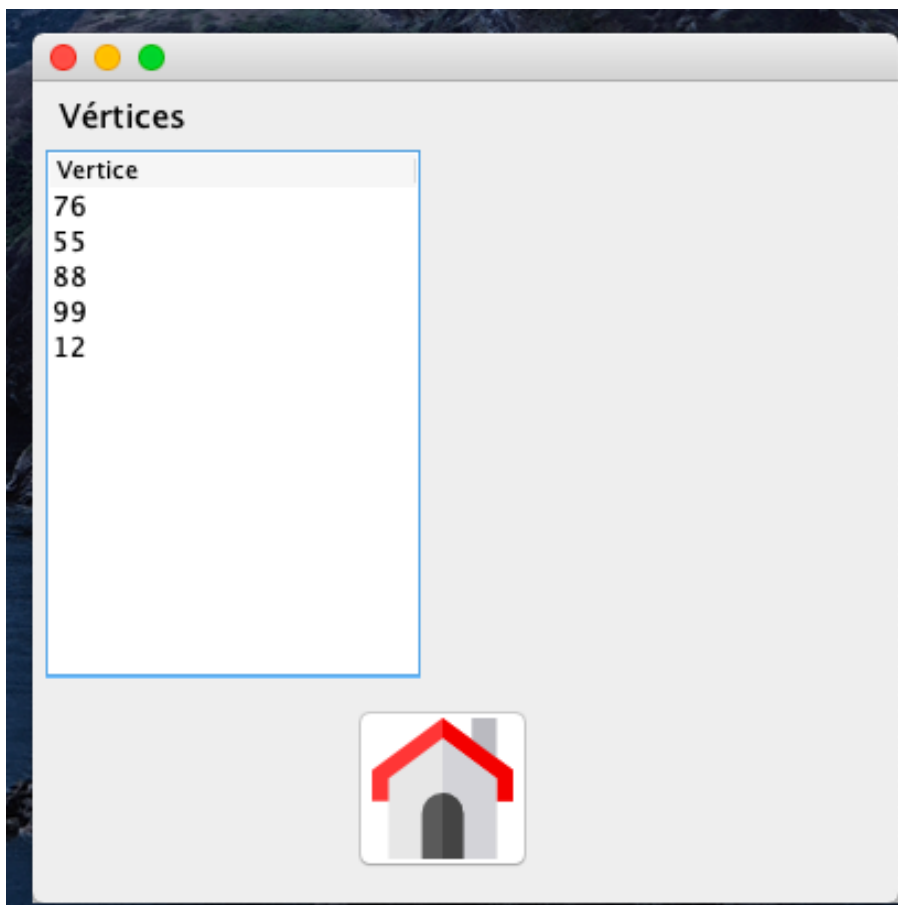
## VISTA SECUNDARIA

Vértices	Lista de Adyacencia			
<table><thead><tr><th>Vertice</th></tr></thead><tbody></tbody></table>	Vertice	<table><thead><tr><th>Vertice Origen</th><th>Vertice Destino</th></tr></thead><tbody></tbody></table>	Vertice Origen	Vertice Destino
Vertice				
Vertice Origen	Vertice Destino			

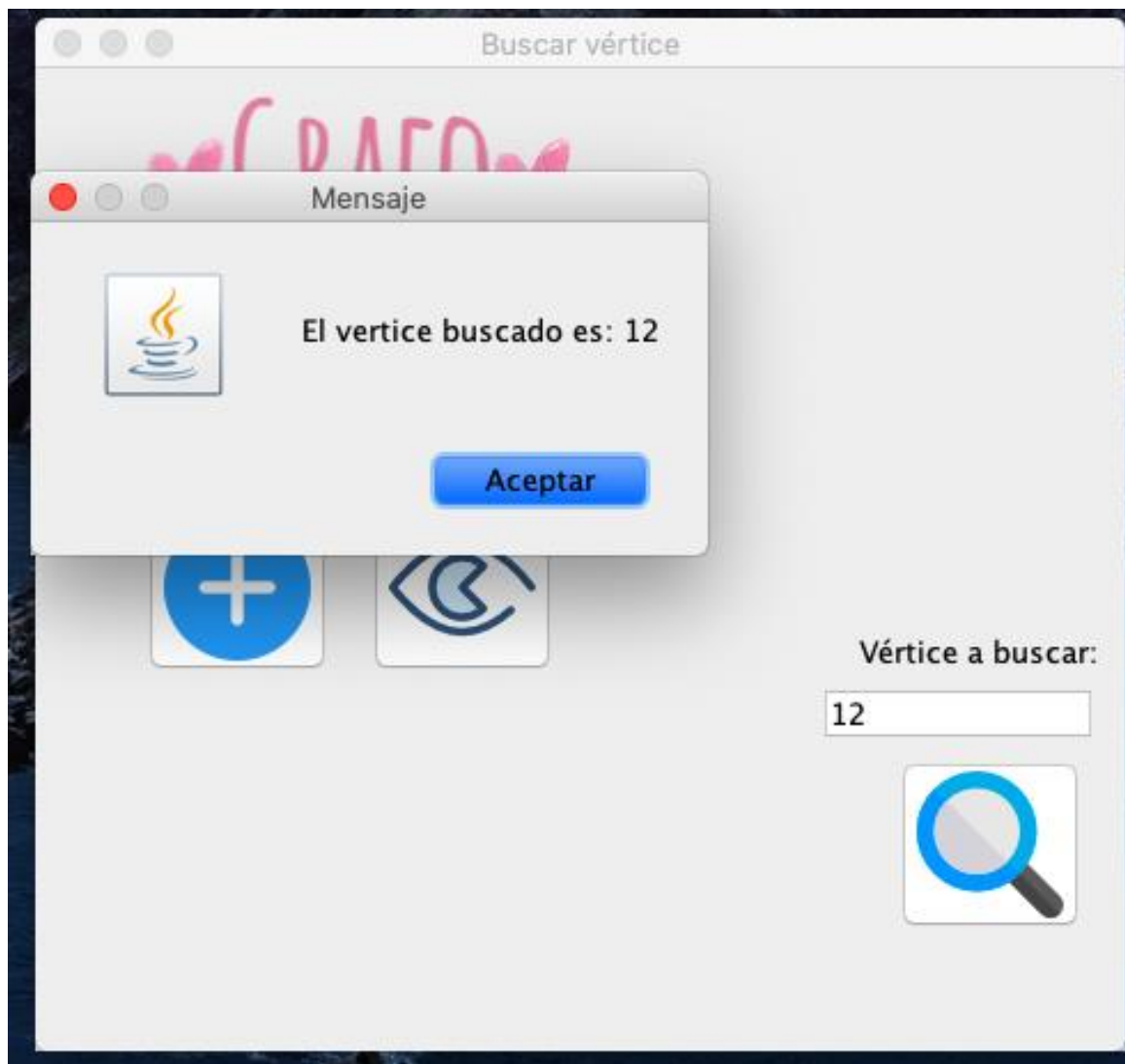


# Funcionabilidad











## Lista de Adyacencia

Vertice origen	Vertice destino
76	
55	
88	
99	
12	76->



# CONCLUSIÓN

Durante el desarrollo de esta actividad pude reafirmar los conocimientos que obtuve a lo largo del cuatrimestre, en especial reafirmé y puse en práctica los que obtuve en la materia de Programación Visual.

Al principio de esta semana me costó mucho trabajo empezar a desarrollar este proyecto, ya que las vacaciones me afectaron un poco y sólo ví series y una que otra película, pero todo cambió el lunes cuando empecé a pensar sobre cómo haría el diseño y la codificación de éste.

Para el desarrollo primero pensé en realizar seis vistas, una para el menú, otra para agregar el vértice, etc., pero después pensé en realizar dos y utilizar diferentes paneles y redimensionar las ventanas.

Una de mis partes favoritas de la materia Programación Visual es diseñar, buscar paletas de colores e íconos para las vistas, lo cual también se convirtió en mi parte favorita de este proyecto, y cuando llegó el momento de programar honestamente creí que sería sencillo, dado a que ya contaba con los métodos a utilizar y creo que ya estoy acostumbrada a usar la arquitectura Modelo-Vista-Controlador, es por eso que cuando ví error tras error me agüité y decidí no hacer el proyecto. Broma.

Uno de los mayores desafíos que he tenido y que sigo teniendo es la resolución de problemas, veo un error y al momento entro en pánico y a veces al tratar de arreglar los errores termino “rompiendo” más el código.

Es por eso que quiero agradecer a Arely por resolver mis dudas, explicarme y ayudarme cuando más lo necesito.