Código del proyecto "Encuesta"

Encuesta.java

```
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.awt.Scrollbar;

import javax.security.auth.login.FailedLoginException;
import javax.tv.xlet.*;

import org.havi.ui.*;
import org.w3c.dom.events.Event;
import org.dvb.dsmcc.AsynchronousLoadingEventListener;
import org.dvb.dsmcc.DSMCCException;
import org.dvb.dsmcc.DSMCCObject;
import org.dvb.dsmcc.InvalidPathNameException;
import org.dvb.dsmcc.MPEGDeliveryException;
import org.dvb.dsmcc.ServiceDomain;
import org.dvb.dsmcc.ServiceXFRException;
import org.dvb.event.EventManager;
import org.dvb.event.UserEvent;
import org.dvb.event.UserEventRepository;
import org.dvb.net.rc.*;
import org.dvb.ui.*;
import org.davic.media.*;
import org.davic.net.InvalidLocatorException;
import org.davic.resources.ResourceClient;
import org.davic.net.dvb.DvbLocator;


import javax.tv.service.selection.*;
import javax.tv.service.Service;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InterruptedIOException;
import java.util.EventListener;
import java.util.LinkedList;

public class Encuesta extends HContainer implements KeyListener, Xlet,
FocusListener, org.w3c.dom.events.EventListener, ResourceClient {

      private static XletContext context;
      private HScene scene;
      private HContainer contPpal;
      private HContainer contFinal;

      private DatosEncuesta datos;

      //paths
```

```
	private String pathBase="";
	private String pathEncuesta=pathBase+"/info/Encuesta.txt";
	private String pathRedSpot=pathBase+"/images/redspot.png";
	private String pathGreenSpot=pathBase+"/images/greenspot.png";


	private Image icono;

	private HStaticText pregunta;
	private HStaticText resultado;
	private HStaticText voto;
	private HStaticText botonVerdetxt;
	private HTextButton opciones [];
	private HGraphicButton botonVerde;
	private HGraphicButton botonIcono;
	private HTextButton botonInvisible;

	private Image botonSalir;
	private Image botonIngresarImg;


	private boolean esCarrusel = false;


	//Para el carrusel
	private DSMCCObject redSpot;
	private DSMCCObject greenSpot;
	private DSMCCObject encuesta;


	//Para el locator de acceso al carrusel
	private org.davic.net.dvb.DvbLocator locator;
	private ServiceContextFactory scf;
	private ServiceContext sc;



	public Encuesta() {
		super();
	}


	public void initXlet(XletContext ctx) throws XletStateChangeException
{

		setName("Encuesta");
		this.context = ctx;

		//Configuro la scena basica que contendra a todos lo
componentes y containers, porq cada xlet se debe
		//tener un solo Hscene
		HSceneFactory factory = HSceneFactory.getInstance();
		HSceneTemplate hst = new HSceneTemplate();
		hst.setPreference(HSceneTemplate.SCENE_SCREEN_DIMENSION, new
org.havi.ui.HScreenDimension(1, 1), HSceneTemplate.REQUIRED);
```

```
            hst.setPreference(HSceneTemplate.SCENE_SCREEN_LOCATION, new
org.havi.ui.HScreenPoint(0, 0), HSceneTemplate.REQUIRED);
            scene = factory.getBestScene(hst);
            scene.setBounds(0,0,900,900);
            scene.setLayout(null);
            scene.setBackgroundMode(HScene.BACKGROUND_FILL);
            scene.setVisible(true);
            scene.add(this);

            contPpal = new HContainer(10, 10, 900, 900);

            //carga los datos de la encuesta desde un archivo txt
            if (esCarrusel){
                  this.cargarObjetos();
            }else{
                  this.cargarDatos();
            }


            pregunta = new HStaticText(datos.getPregunta(),0,20,700,67 );
            pregunta.setFont(new Font("Verdana",Font.BOLD,35));
            Color color = new Color(238,0,0);
            pregunta.setBackground(Color.BLACK);
            pregunta.setVisible(true);
            pregunta.setName("Pregunta");
            pregunta.setForeground(color);
            contPpal.add(pregunta);


            opciones= new HTextButton [datos.getCantOp()];
            for(int i = 0 ; i < (datos.getCantOp()) ; i++){
                  //verificar cantidad de caracteres y cortar línea
                  opciones[i] = new HTextButton((datos.getOpciones(i)),
100, 100+i*50, 570, 40);
                  opciones[i].setHorizontalAlignment(100);
                  opciones[i].addKeyListener(this);
                  opciones[i].addFocusListener((FocusListener) this);
                  opciones[i].setFocusable(true);
                  opciones[i].setForeground(Color.BLACK);
                  opciones[i].setBackground(Color.WHITE);
                  Font fuente = new Font("Verdana",Font.BOLD,20);
                  opciones[i].setFont(fuente);
                  opciones[i].setVisible(true);
                  icono =
Toolkit.getDefaultToolkit().getImage(datos.getIcono());
                  botonIcono =new HGraphicButton(icono, 20,20,80,80);
                  contPpal.add(opciones[i]);
            }
            for (int i=0; i < opciones.length ; i++){

                  if (i == opciones.length -1){//Si estoy en el ultimo
                        opciones[i].setFocusTraversal(opciones[i-1],
opciones[0], null, null);
                  }else{
                        if (i==0){//Si soy el primero
```

```
      opciones[i].setFocusTraversal(opciones[opciones.length -1],
opciones[i+1], null, null);
                        }else{
                              opciones[i].setFocusTraversal(opciones[i-1],
opciones[i+1], null, null);
                        }

                  }

                              }
            //Agrego el boton verde
            if (esCarrusel){
                  botonIngresarImg =
Toolkit.getDefaultToolkit().getImage(greenSpot.getPath());
            }else{
                  botonIngresarImg =
Toolkit.getDefaultToolkit().getImage(pathGreenSpot);
            }

            botonVerde =  new HGraphicButton(botonIngresarImg,
550,510,40,40);
            contPpal.add(botonVerde);
            botonVerdetxt = new HTextButton("Enviar");
            botonVerdetxt.setVisible(true);
            botonVerdetxt.setBounds(598,490, 100, 80);
            botonVerdetxt.setForeground(Color.BLACK);
            botonVerdetxt.setFont(new Font("Verdana",Font.BOLD,30));
            botonVerdetxt.setBackgroundMode(Color.TRANSLUCENT);
            contPpal.add(botonVerdetxt);

            //agregar los campos de input de aut si corresponde.

            //Habilito en evento de teclado para escuchar los eventos que
provienen del control remoto

            scene.addKeyListener((KeyListener)this);
            scene.add(contPpal);
            scene.setVisible(true);
            this.setSize(scene.getSize());



      }

      public void startXlet() throws XletStateChangeException {
            validate();
            scene.setVisible(true);
            scene.requestFocus();
            opciones[0].requestFocus();


      }

      public void pauseXlet() {
```

```
            scene.setVisible(false);
      }

      public void destroyXlet(boolean b) {
            if (scene != null) {
                  removeKeyListener(this);
                  scene.remove(this);
                  scene.setVisible(false);
                  HSceneFactory.getInstance().dispose(scene);
                  scene = null;
            }
            context.notifyDestroyed();
      }



      public void paint(Graphics g) {
            super.paint(g);
            cargarBotones(g);

      }

private void cargarBotones(Graphics g) {
      g.setColor(new Color(0x3f,0x00,0x7f));//Azul
      g.setColor(new Color(0x00,0x00,0x00));
      g.setColor(Color.BLACK);
      g.drawLine(0,28,719,28);
      g.drawLine(0,29,719,29);
      g.drawLine(0,95,719,95);
      g.drawLine(0,96,719,96);

      //Aca no es necesario verificar lo del carusel ya que el path sale
del archivo Encuesta.txt
      icono = Toolkit.getDefaultToolkit().getImage(datos.getIcono());


      g.drawImage(icono, 20,20,80,80, null);


      g.setColor(Color.black);

      if (esCarrusel){
            botonSalir =
Toolkit.getDefaultToolkit().getImage(redSpot.getPath());
      }else{
            botonSalir = Toolkit.getDefaultToolkit().getImage(pathRedSpot);
      }
      g.drawImage(botonSalir,20,520,40,40,this);
      g.setColor(Color.BLACK);
      g.setFont(new Font("Verdana",Font.BOLD,30));
      g.drawString("Salir",70,550);



}
//    carga la pregunta, icono, opciones y si aplica autenticacion.
```

```
	public void cargarDatos(){

		datos= new DatosEncuesta();

		//lee el archivo .txt que trae los datos de la encuesta
		BufferedReader in;
		String texto="";

		try {
			if(esCarrusel){
				in = new BufferedReader(new
FileReader(encuesta.getPath()));
			}else{
				in = new BufferedReader(new
FileReader(pathEncuesta));
			}

			String line="";
			while((line = in.readLine()) != null){
				texto=texto+line;

			}
		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		String [] info = texto.split(";;;;");
		datos.setPregunta(info[0]);

		String [] opciones = info[1].split("::");
		datos.setCantOp(opciones.length);
		datos.setOpciones(opciones);
		datos.setIcono(info[2]);
		datos.setTels(info[3].split("::"));

	}

	public void focusGained(FocusEvent e) {

	for( int a=0; a < (datos.getCantOp()) ; a++){
		if(opciones[a].isSelected()){
			opciones[a].setBackground(new Color(221,221,221));
			for( int i=0; i < (datos.getCantOp()) ; i++){
				if(i!=a){
					opciones[i].setBackground(Color.WHITE);
				}
			}

		}
	}
```

```
	}

	public void conectar(int nroOpcion) throws IOException {
		String tel= datos.getTels(nroOpcion);

		RCInterfaceManager rcm = RCInterfaceManager.getInstance();

		//Se obteiene la lista de interfases disponibles para la
aplicación
		RCInterface[] interfaces = rcm.getInterfaces();

		/*Se selecciona una interface, pero antes se verifica que es
del tipo deseado "ConnectionRCInterface"*/

		if (interfaces[0] instanceof ConnectionRCInterface) {
			//Si ese entra aca debo conectar la interfase

			ConnectionRCInterface myInterface;
			myInterface = (ConnectionRCInterface)interfaces[0];

			//usamos la interfase
			try {
				// Reservamos la conexión
				myInterface.reserve(this,null);


				// Seteo de parámetros parameters
				ConnectionParameters myConnectionParameters;
				myConnectionParameters = new
ConnectionParameters(tel, "usuario", "contraseña");

				// Seteamos el target de la interface y nos
conectamos
				myInterface.setTarget(myConnectionParameters);
				myInterface.connect();//Lanza la excepcion
IOException

				//enviar lo que se desea

				// desconexión, liberacion del recurso
				myInterface.disconnect();
				myInterface.release();
				System.out.println("Finalizando...");

			//}
			} catch (PermissionDeniedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (IncompleteTargetException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			//} catch (IOException e) {
				// TODO Auto-generated catch block
				//e.printStackTrace();
```

```
                  }
            }
            else {
                  //LA CONEXIÓN ES PERMANENTE, USAR
            }

      }

      public void keyPressed(KeyEvent key) {

            switch(key.getKeyCode()) {
            case 403://boton rojo

                  destroyXlet(true);
                  break;


            case 404://boton verde
                  for( int a=0; a < (datos.getCantOp()) ; a++){
                        if(opciones[a].isSelected()){
                              try{
                                    this.conectar(a);
                                    scene.remove(contPpal);
                                    contFinal= new HContainer(10, 10, 900,
900);
                                    resultado = new HStaticText("Encuesta
Completada",0,20,700,67 );
                                    resultado.setFont(new
Font("Verdana",Font.BOLD,35));
                                    Color color = new Color(238,0,0);
                                    resultado.setBackground(Color.BLACK);
                                    resultado.setVisible(true);
                                    resultado.setForeground(color);
                                    contFinal.add(resultado);
                                    voto = new HStaticText("Voto
Enviado"+"\r\n"+"\r\n"+"Muchas Gracias",250,200,230,90);
                                    voto.setFont(new
Font("Verdana",Font.BOLD,28));
                                    voto.setVisible(true);
                                    voto.setForeground(color);
                                    voto.setFocusable(true);
                                    voto.requestFocus();
                                    contFinal.add(voto);
                                    botonInvisible = new
HTextButton("",10,50,50,25);
                                    botonInvisible.addKeyListener(this);

      botonInvisible.addFocusListener((FocusListener) this);
                                    botonInvisible.setFont(new
Font("Verdana",Font.BOLD,5));

      botonInvisible.setBackgroundMode(Color.TRANSLUCENT);

      botonInvisible.setForeground(Color.BLACK);
                                    botonInvisible.setFocusable(true);
```

```
                                    contFinal.add(botonInvisible);
                                    scene.add(contFinal);
                                    botonInvisible.requestFocus();
                              } catch (IOException e) {
                                    scene.remove(contPpal);
                                    contFinal= new HContainer(10, 10, 900,
900);
                                    resultado = new HStaticText("Encuesta NO
completada",0,20,700,67 );
                                    resultado.setFont(new
Font("Verdana",Font.BOLD,35));
                                    Color color = new Color(238,0,0);
                                    resultado.setBackground(Color.BLACK);
                                    resultado.setVisible(true);
                                    resultado.setForeground(color);
                                    contFinal.add(resultado);
                                    voto = new HStaticText("Error!!,
imposible registrar el voto"+"\r\n"+"\r\n"+"Problemas de conexión,
inténtelo más tarde",25,200,650,90);
                                    voto.setFont(new
Font("Verdana",Font.BOLD,28));
                                    voto.setVisible(true);
                                    voto.setForeground(color);
                                    voto.setFocusable(true);
                                    voto.requestFocus();
                                    contFinal.add(voto);
                                    botonInvisible = new
HTextButton("",10,50,50,25);
                                    botonInvisible.addKeyListener(this);

      botonInvisible.addFocusListener((FocusListener) this);
                                    botonInvisible.setFont(new
Font("Verdana",Font.BOLD,5));

      botonInvisible.setBackgroundMode(Color.TRANSLUCENT);

      botonInvisible.setForeground(Color.BLACK);
                                    botonInvisible.setFocusable(true);
                                    contFinal.add(botonInvisible);
                                    scene.add(contFinal);
                                    botonInvisible.requestFocus();

                              }
                        }
                  }

            break;

            default:
                  break;
            }


      }
```

```
	public void keyTyped(KeyEvent ignored) { }


	public void keyReleased(KeyEvent ignored) { }


	public void focusLost(FocusEvent arg0) {
		// TODO Auto-generated method stub

	}



	//Carga de objetos del carrusel

	public void cargarObjetos(){

		ServiceDomain carousel = new ServiceDomain();



		try {

			/*Por lo tanto primero se obtiene una referencia al
service context en el que corre la aplicación. Para hacer esto se pide el
service context que contiene al Xlet context. Es posible que se largue una
excepcion*/
		scf = ServiceContextFactory.getInstance();
		sc = scf.getServiceContext(context);


		//Ahora se obtiene el objeto JavaTV que representa al servicio
		Service s;
		s=sc.getService();

		//Finalmente se obtiene el locator del presente servicio
		locator = (DvbLocator) s.getLocator();

		carousel.attach(locator);


		} catch (ServiceXFRException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (MPEGDeliveryException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (InterruptedIOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (DSMCCException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (ServiceContextException e) {
```

```
                    // No hay nada que se pueda hacer al respecto
                    System.out.println("Error al obtener el service
context!");
                    e.printStackTrace();
              }
              } catch (SecurityException e) {
                    e.printStackTrace();
              }



              // tenemos que crear nuestro DSMCCObject con un path absoluto,
lo que
              //significa que tenemos que tener un mount point para el
service domain


              redSpot = new
DSMCCObject(carousel.getMountPoint(),pathRedSpot);
              greenSpot = new
DSMCCObject(carousel.getMountPoint(),pathGreenSpot);
              encuesta = new
DSMCCObject(carousel.getMountPoint(),pathEncuesta);

              AsynchronousLoadingEventListener myListener = null;

              try {
                    redSpot.asynchronousLoad(myListener);
                    greenSpot.asynchronousLoad(myListener);
                    encuesta.asynchronousLoad(myListener);

              } catch (InvalidPathNameException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
              }


                    try {
                          BufferedReader in = new BufferedReader(new
FileReader(redSpot));
                          System.out.println("REdSPot: "+in.read());
                          in = new BufferedReader(new FileReader(greenSpot));

                          System.out.println("greenSPot: "+in.read());

                          in = new BufferedReader(new FileReader(encuesta));

                          System.out.println("encuesta: "+in.read());
                    } catch (FileNotFoundException e) {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }catch (IOException e) {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }
```

```
            //Cargo los datos del archvio Encuesta.txt
            this.cargarDatos();



      }

      public void handleEvent(Event arg0) {
            // TODO Auto-generated method stub
            if (arg0 instanceof ConnectionDroppedEvent){
                  //Si es un dropped event quiere decir quese pudo
registrar el voto y luego se corta la conexion
                  System.out.println("Se pudo registrar el voto");
            }
            if (arg0 instanceof ConnectionFailedEvent){
                  //Si es un Failed event quiere deicr que hubo erores y no
se pudo registrar el voto
                  scene.remove(contPpal);
                  contFinal= new HContainer(10, 10, 900, 900);
                  resultado = new HStaticText("Encuesta NO
completada",0,20,700,67 );
                  resultado.setFont(new Font("Verdana",Font.BOLD,35));
                  Color color = new Color(238,0,0);
                  resultado.setBackground(Color.BLACK);
                  resultado.setVisible(true);
                  resultado.setForeground(color);
                  contFinal.add(resultado);
                  voto = new HStaticText("Error!!, imposible registrar el
voto"+"\r\n"+"\r\n"+"Problemas de conexión, inténtelo más
tarde",25,200,650,90);
                  voto.setFont(new Font("Verdana",Font.BOLD,28));
                  voto.setVisible(true);
                  voto.setForeground(color);
                  voto.setFocusable(true);
                  voto.requestFocus();
                  contFinal.add(voto);
                  botonInvisible = new HTextButton("",10,50,50,25);
                  botonInvisible.addKeyListener(this);
                  botonInvisible.addFocusListener((FocusListener) this);
                  botonInvisible.setFont(new Font("Verdana",Font.BOLD,5));
                  botonInvisible.setBackgroundMode(Color.TRANSLUCENT);
                  botonInvisible.setForeground(Color.BLACK);
                  botonInvisible.setFocusable(true);
                  contFinal.add(botonInvisible);
                  scene.add(contFinal);
                  botonInvisible.requestFocus();

            }
            }


}
```

DatosEncuesta.java

```
public class DatosEncuesta {

	private String i;
	private String p;
	private String op [];
	private String tels[];
	private int cO;

	public DatosEncuesta(){

	}

	public void setIcono(String a){
		this.i=a;
	}

	public String getIcono(){
		return i;
	}

	public void setPregunta(String a){
		this.p=a;
	}

	public String getPregunta(){
		return p;
	}

	public void setCantOp(int a){
		this.cO=a;
	}

	public int getCantOp(){
		return cO;
	}

	public void setOpciones(String [] a){
		this.op=a;
	}

	public String getOpciones(int i){
		return (op[i]);
	}

	public void setTels(String [] a){
		this.tels=a;
	}

	public String getTels(int i){
		return (tels[i]);
	}

}
```