



GUÍA 4. Conexión HTTP [VideoTienda]

Objetivos:

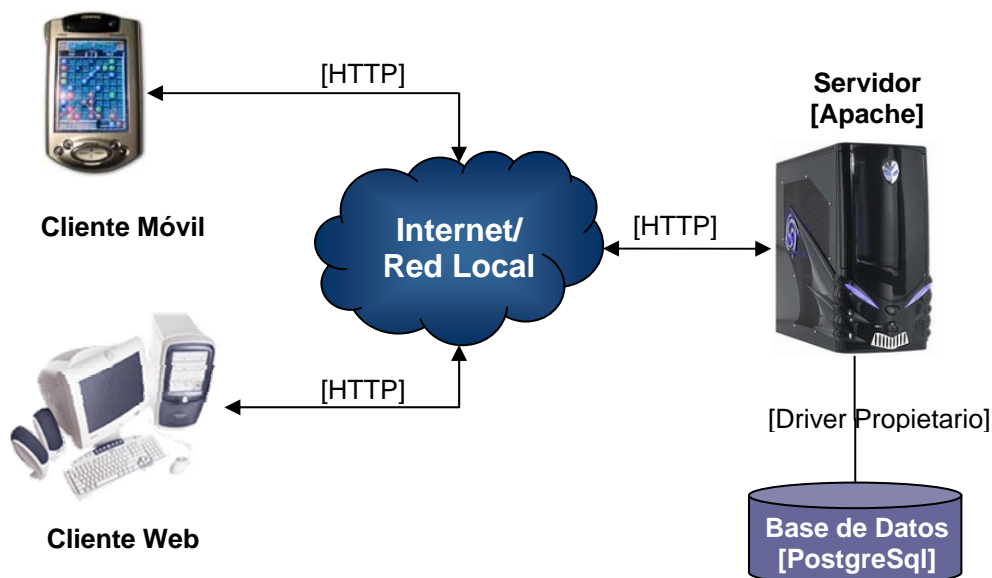
- Establecer una conexión con un servidor HTTP.
- Intercambiar mensajes entre el móvil y el servidor.
- Autenticarse como usuario y recibir información proveniente del servidor http, el cual se encuentra conectado a una base de datos PostgreSQL.

1. Practica 4

En la siguiente práctica vamos a realizar la consulta a una tienda de video por medio del móvil, donde nos autenticaremos y si estamos registrados, se nos desplegará la lista de las películas disponibles. En caso contrario lo dejará en la parte de validación.

El acceso Web esta montado sobre Apache, en el cual se ha desplegado sitio Web llamado *WebVideotienda*. Este sitio mediante las paginas *login.php* (Validación) y *películas.php* (Información de las películas), se encargará de recibir y dar respuesta al móvil.

La arquitectura es la siguiente:



Como se puede observar disponemos de dos clientes (Web y Móvil). Y del servidor ya mencionado. Veamos como están compuestas cada uno de los componentes del diagrama.

2. SERVIDOR HTTP

Para el lado del servidor, como lo hemos mencionado antes, se realizó el montaje de las páginas PHP, sobre Apache. El servidor esta ubicado en la dirección:



<http://172.16.130.164>



Los instaladores se pueden descargar de la página <http://httpd.apache.org/download.cgi>

Y para los instaladores de PHP se pueden descargar de <http://www.php.net>

2.1. Estructura Base de datos

La Base de datos se montó sobre PostgreSQL, los instaladores se pueden descargar desde la página www.postgresql.org. La base de datos se llama **VideoTienda**, y esta compuesta por dos tablas como se ve a continuación:

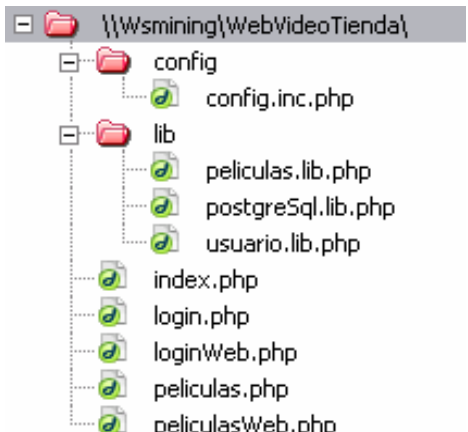
| VIDEOTIENDA | | | | | | | | | | | | | | | | | | | | | | | |
|--|----------------|--|-------------------|---------|--------|----------------|----------|----------------|-------|---------|-------|----------------|----------|----------------|--|-----------|--|--------|----------------|-----|---------------|--------|----------------|
| <table><tr><td>usuario</td><td></td></tr><tr><td>idusuario Integer</td><td>NN (PK)</td></tr><tr><td>nombre</td><td>Varchar(n)(15)</td></tr><tr><td>apellido</td><td>Varchar(n)(15)</td></tr><tr><td>email</td><td>Varchar</td></tr><tr><td>login</td><td>Varchar(n)(20)</td></tr><tr><td>password</td><td>Varchar(n)(20)</td></tr></table> | usuario | | idusuario Integer | NN (PK) | nombre | Varchar(n)(15) | apellido | Varchar(n)(15) | email | Varchar | login | Varchar(n)(20) | password | Varchar(n)(20) | <table><tr><td>películas</td><td></td></tr><tr><td>titulo</td><td>Varchar(n)(15)</td></tr><tr><td>ano</td><td>Varchar(n)(5)</td></tr><tr><td>genero</td><td>Varchar(n)(10)</td></tr></table> | películas | | titulo | Varchar(n)(15) | ano | Varchar(n)(5) | genero | Varchar(n)(10) |
| usuario | | | | | | | | | | | | | | | | | | | | | | | |
| idusuario Integer | NN (PK) | | | | | | | | | | | | | | | | | | | | | | |
| nombre | Varchar(n)(15) | | | | | | | | | | | | | | | | | | | | | | |
| apellido | Varchar(n)(15) | | | | | | | | | | | | | | | | | | | | | | |
| email | Varchar | | | | | | | | | | | | | | | | | | | | | | |
| login | Varchar(n)(20) | | | | | | | | | | | | | | | | | | | | | | |
| password | Varchar(n)(20) | | | | | | | | | | | | | | | | | | | | | | |
| películas | | | | | | | | | | | | | | | | | | | | | | | |
| titulo | Varchar(n)(15) | | | | | | | | | | | | | | | | | | | | | | |
| ano | Varchar(n)(5) | | | | | | | | | | | | | | | | | | | | | | |
| genero | Varchar(n)(10) | | | | | | | | | | | | | | | | | | | | | | |

2.2. Estructura del sitio Web

La estructura esta compuesta de la siguiente forma:

Como se observa esta compuesta por dos carpetas **Config** (Configuración de la conexión a la base de datos) y **Lib** (Archivos utilizados para la conexión, y para realizar las consultas a las diferentes tablas.)

En Raíz tenemos los archivos los cuales conforman la lógica del negocio. Se distribuye de la siguiente forma:

| | | |
|--|-------------------------|--|
|  <p>\\Wsmining\WebVideoTienda\</p> <ul style="list-style-type: none"> config <ul style="list-style-type: none"> config.inc.php lib <ul style="list-style-type: none"> películas.lib.php postgreSql.lib.php usuario.lib.php index.php login.php loginWeb.php películas.php películasWeb.php | Acceso Web | |
| | Index.php | Pagina principal para la validación del usuario. |
| | loginWeb.php | Controla el acceso según si el usuario esta o no registrado. |
| | películasWeb.php | Muestra las películas que están almacenadas en la base de datos. |
| | Acceso Movil | |
| | login.php | Valida al usuario móvil. |
| | Películas.php | Devuelve un arreglo con las películas que contiene la tabla películas. |

Se puede acceder al sitio Web mediante <http://172.16.130.164/WebVideoTienda/index.php>

3. Cliente Web

El login y el password son "aplimovil" y "aplimovil". Si el usuario esta registrado en a base de datos, despliega la lista de las películas disponibles.

Ingreso al sistema

Registro de Usuario

Por favor ingrese su login y su password

Login:

Password:

Video Tienda "El Rapidito"

Películas disponibles

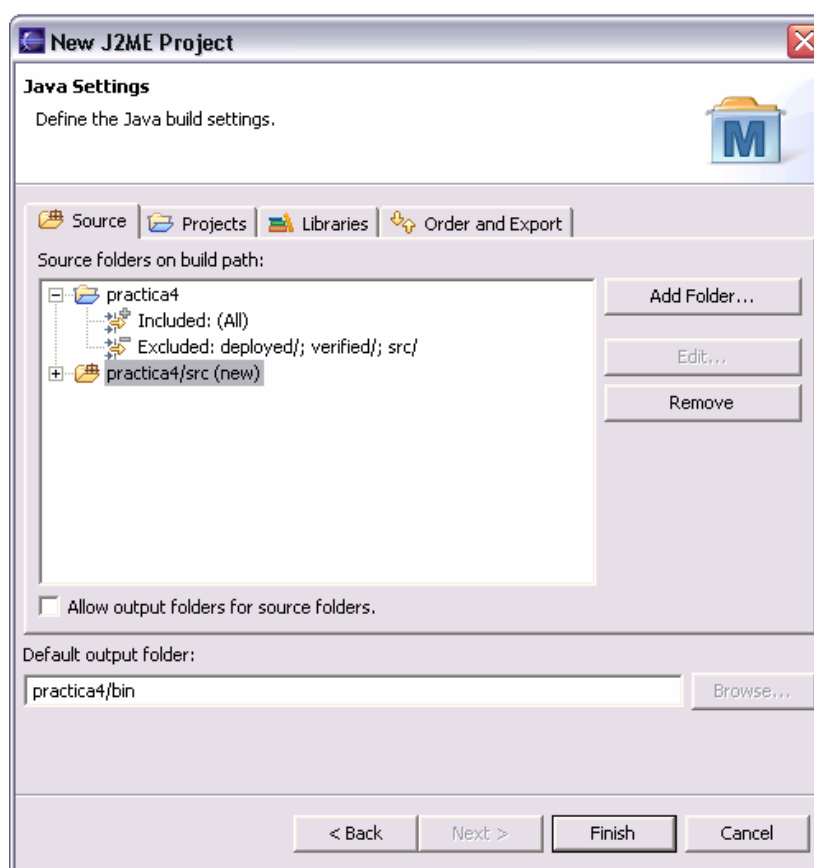
| Titulo | Año | Genero |
|-------------|------|----------|
| Cafelo | 1982 | Triller |
| Garfield | 2004 | Infantil |
| Mindhunters | 2005 | Suspense |
| Saw | 2005 | Accion |
| Titanic | 1997 | Drama |
| Unleashed | 2005 | Suspense |

Si accedemos a login.php y películas.php (cliente móvil), podremos observar que nos devuelve una cadena de caracteres, lo cual es suficiente para ser interpretado por el móvil. Ya veremos como lo hace mas adelante.

4. Cliente Móvil

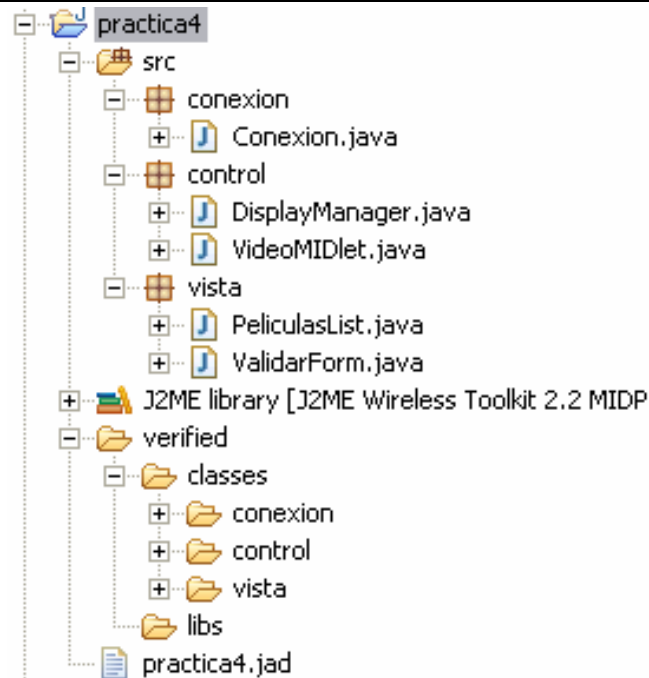
4.1. Creación del Proyecto

Primero creamos el Proyecto: "**practica4**", con la configuración con la que hemos venido trabajando. La plataforma con la cual vamos a trabajar, en este caso es *J2ME Wireless Toolkit 2.2 MIDP 2.0 Platform*. No olvidemos crear las carpetas **src** (Archivos fuente .java) , y **bin** (Archivos .class)



4.2. Estructura del Proyecto

Ahora vamos a crear 3 paquetes llamados "**vista**", "**control**" y "**conexion**", dentro del paquete "**src**". Para ello simplemente damos click derecho sobre "src" y le damos **New/Package**. Y creamos cada paquete quedando la estructura de la siguiente manera:



4.3. VideoMIDlet.java

Este es el MIDlet encargado de desplegar las diferentes ventanas de la aplicación. Como en prácticas anteriores vamos a crear un objeto *DisplayManager* el cual nos permitirá desplegar las diferentes opciones al usuario. Comenzando con un objeto *ValidarForm*, que nos permite validar un usuario dentro de nuestro sistema de VideoTienda.

Creamos el MIDlet, para ello vamos a *Package Explorer* y le damos clic derecho a “*vista*” (o según sea el paquete), allí se abre un submenú, _escojemos **New>Other...** Seleccionamos **J2ME Midlet** y presionamos <Next>

El código del MIDlet es el siguiente:

```
/**
 * Practica 4. Conexión HTTP mediante el Movil [VideoTienda]
 * Electiva: Desarrollo de Aplicaciones para Dispositivos Móviles [Telemática]
 * Universidad del Cauca
 * Ing. Oscar Mauricio Caicedo
 * Monitor: Carlos Felipe López
 * Created on may-2005
 */
```

```
package control;
```

```
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Image;
import javax.microedition.midlet.MIDlet;
import vista.*;
```



```
public class VideoMIDlet extends MIDlet {

    private Display display;
    public static VideoMIDlet midlet;
    public DisplayManager displayMgr;
    private Image OK;
    private String respuesta;

    public void VideoMIDlet() {

    }

    /**
     * Invocada cuando es creado el midlet.
     */
    public void startApp() {

        display = Display.getDisplay(this);
        midlet = this;
        displayMgr = new DisplayManager(display, new ValidarForm());
        displayMgr.home();
    }

    /**
     * Invocada cuando es cerrada la aplicacion.
     */
    public void destroyApp( boolean unconditional ) {
        midlet.notifyDestroyed();
        midlet = null;
    }

    public void pauseApp() {}

    public void setResp(String resp){
        respuesta=resp;
    }
}
```

4.4. *DisplayManager.java*

Esta clase, nos permite llevar un control sobre el flujo de tiene el usuario en la aplicación. Hereda de la clase *java.util.Stack*.

El código es el siguiente:

```
/**
 * Practica 4. Conexion HTTP mediante el Movil [VideoTienda]
 * Electiva: Desarrollo de Aplicaciones para Dispositivos Móviles [Telemática]
 * Universidad del Cauca
 * Ing. Oscar Mauricio Caicedo
 *
 * Created on may-2005
 */
```

Monitor: Carlos Felipe Lopez



package control;

```
import javax.microedition.lcdui.*;  
import java.util.Stack;
```

```
/**  
 * Permite administrar las ventanas de la aplicación. Se usa un objeto de tipo "stack" para desplegar los objetos de  
 tipo displayable.
```

```
 * @author GIT - UNICAUCA
```

```
 * @version 1.0
```

```
 */
```

```
public class DisplayManager extends Stack {  
    private Display display; // Referencia a un Objeto Display  
    private Displayable mainDisplayable; // Displayable principal del MIDlet  
    private Alert alStackError; // Alerta para errores
```

```
    /**
```

```
     * Constructor.
```

```
     * @param display MIDlet display object
```

```
     * @param mainDisplayable Main screen to be displayed. It becomes the home screen
```

```
     */
```

```
    public DisplayManager(Display display, Displayable mainDisplayable) {  
        //Solo un objeto display por Midlet,  
        this.display = display;  
        this.mainDisplayable = mainDisplayable;  
        // Crea una alerta cuando hay algun problema al cargar el objeto displayable.  
        alStackError = new Alert("Displayable Stack Error");  
        alStackError.setTimeout(Alert.FOREVER); // Modal  
    }
```

```
    /**
```

```
     * Agrega el objeto displayable actual al tope del Stack y lo despliega.
```

```
     ** @param newDisplayable Nueva ventana a ser desplegada
```

```
     */
```

```
    public void pushDisplayable(Displayable newDisplayable) {  
        push(display.getCurrent());  
        display.setCurrent(newDisplayable);  
    }
```

```
    /**
```

```
     * Agrega el objeto displayable actual al tope del Stack y lo despliega.
```

```
     * Pero Primero despliega una Alarma.
```

```
     ** @param newDisplayable Nueva ventana a ser desplegada
```

```
     */
```

```
    public void pushDisplayable(Alert alert, Displayable newDisplayable) {  
        push(display.getCurrent());  
        display.setCurrent(alert,newDisplayable);  
    }
```

```
    /**
```

```
     * Despliega la ventana definida anteriormente en la constructora.
```

```
     */
```

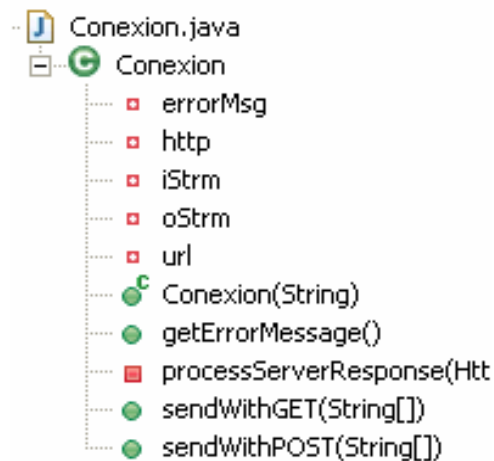
```
    public void home() {  
        while (elementCount > 1)  
            pop();  
        display.setCurrent(mainDisplayable);  
    }
```

```
    /**
```



```
* Despliega el objeto que esta en el tope y lo elimina del Stack.  
*/  
public void popDisplayable() {  
  
    // Si el Stack no esta vacio, saca el objeto.  
    if (empty() == false){  
        display.setCurrent((Displayable)pop());  
    }else{  
        // Cuando hay un error despliega la alerta  
        // Cuando sale la alerta, coloca el mainDisplayable  
        display.setCurrent(alStackError, mainDisplayable);  
    }  
  
}
```

4.5. Conexion,java



Conexion va a ser la encargada de enviar y recibir las respuestas, desde el dispositivo móvil hasta el servidor HTTP y viceversa.

Dentro de su estructura hay tres métodos que destacan, dos de ellos (*sendWithPOST* y *sendWithGET*) son utilizados para enviar las solicitudes (request) al servidor y con *processServerResponse*, analizamos lo que el servidor nos manda como respuesta.

Esta es una clase sencilla.

El código es el siguiente:

```
package conexion;  
  
import javax.microedition.io.*;  
import java.io.*;  
  
/**  
 * Practica 4. Conexion HTTP mediante el Movil [VideoTienda]  
 * Electiva: Desarrollo de Aplicaciones para Dispositivos Móviles [Telemática]  
 * Universidad del Cauca  
 * @author Ing. Francisco Martinez  
 */  
public class Conexion {  
  
    private HttpURLConnection http;  
    private InputStream iStrm;
```




```
private OutputStream oStrm;
private String errorMsg;
private String url;

public Conexion(String _url) {
    http = null;
    iStrm = null;
    errorMsg = null;
    url = _url;
}

/*-----
 * Access servlet using GET
 *-----*/
public String sendWithGET(String[] parameters) {

    String ret = null;
    // Data is passed at the end of url for GET
    StringBuffer urlGET = new StringBuffer(url);
    urlGET.append("?");
    urlGET.append(parameters[0]);
    for (int i = 1; i < parameters.length; i++) {
        urlGET.append("&");
        urlGET.append(parameters[i]);
    }

    try {
        http = (HttpConnection) Connector.open(urlGET.toString());
        //-----
        // Client Request
        //-----
        // 1) Send request method
        http.setRequestMethod(HttpConnection.GET);
        // 2) Send header information - none
        // 3) Send body/data - data is at the end of URL
        //-----
        // Server Response
        //-----
        iStrm = http.openInputStream();
        // Three steps are processed in this method call
        ret = processServerResponse(http, iStrm);
    }
    catch (IOException e) {
        e.printStackTrace();
        errorMsg="Don't Connection";
    }
    finally {
        try {
            // Clean up
            if (iStrm != null) {
                iStrm.close();
            }
            if (http != null) {
                http.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
            errorMsg="Don't Streams";
        }
    }
}
```



```
}
return ret;
}

/*-----
 * Access servlet using POST
 *-----*/
public String sendWithPOST(String[] parameters) throws IOException {

    String ret = null;

    try {
        http = (HttpURLConnection) Connector.open(url);
        oStrm = http.openOutputStream();
        //-----
        // Client Request
        //-----
        // 1) Send request type
        http.setRequestMethod(HttpURLConnection.POST);
        // 2) Send header information. Required for POST to work!
        http.setRequestProperty("Content-Type",
                                "application/x-www-form-urlencoded");
        http.setRequestProperty("User-Agent",
                                "Profile/MIDP-2.0 Configuration/CLDC-1.1");
        // If you experience connection/IO problems, try
        // removing the comment from the following line
        //http.setRequestProperty("Connection", "close");
        // 3) Send data/body
        // Write account number
        byte data[] = parameters[0].getBytes();
        oStrm.write(data);

        for (int i = 1; i < parameters.length; i++) {
            data = ("&" + parameters[i]).getBytes();
            oStrm.write(data);
        }

        //-----
        // Server Response
        //-----
        iStrm = http.openInputStream();
        // Three steps are processed in this method call
        ret = processServerResponse(http, iStrm);
    }
    finally {
        // Clean up
        if (iStrm != null) {
            iStrm.close();
        }
        if (oStrm != null) {
            oStrm.close();
        }
        if (http != null) {
            http.close();
        }
    }
    return ret;
}
```



```
/*-----  
 * Process a response from a server  
 *-----*/  
private String processServerResponse(HttpConnection http,  
                                     InputStream iStrm) throws IOException {  
    String str = null;  
    // 1) Get status Line  
    if (http.getResponseCode() == HttpURLConnection.HTTP_OK) {  
        // 2) Get header information - none  
        // 3) Get body (data)  
        int length = (int) http.getLength();  
  
        if (length != -1) {  
            byte servletData[] = new byte[length];  
            iStrm.read(servletData);  
            str = new String(servletData);  
        }  
        else { // Length not available...  
            ByteArrayOutputStream bStrm = new ByteArrayOutputStream();  
            int ch;  
            while ( (ch = iStrm.read()) != -1) {  
                bStrm.write(ch);  
            }  
            str = new String(bStrm.toByteArray());  
            bStrm.close();  
        }  
    }  
    else {  
        // Use message from the servlet  
        errorMsg = new String(http.getResponseMessage());  
    }  
    return str;  
}  
  
public String getErrorMessage() {  
    return errorMsg;  
}  
}
```

4.6. ValidarForm.java

Es un formulario en el cual se valida al usuario, mediante una consulta a la base de datos. Para ello debe implementar *Runnable*, dado que esta consulta, se realiza dentro de un hilo, en el cual el servidor dará la respuesta. Y dependiendo de la respuesta, procedemos con el flujo normal de la aplicación. Por esto el command listener se implementa en una clase aparte para luego ser agregado dentro del form y así poder utilizarlo.

El código es el siguiente:

```
/**  
 * Practica 4. Conexion HTTP mediante el Movil [VideoTienda]  
 * Electiva: Desarrollo de Aplicaciones para Dispositivos Móviles [Telemática]  
 * Universidad del Cauca  
 * Ing. Oscar Mauricio Caicedo  
 * Monitor: Carlos Felipe Lopez
```



*

* Created on may-2005

*/

package vista;

import javax.microedition.lcdui.*;

import conexion.Conexion;

import control.*;

public class ValidarForm extends Form implements Runnable{

```
private Command salir, conectar;
private TextField Login, Contraseña;
private String URL;
private Conexion cx; //Establece la conexion con el servidor HTTP.
private StringItem result;
private String parameters[];
private String action;
private String response;
private Alert Error;
```

```
public ValidarForm() {
    super("TIENDA DE VIDEOS");
```

```
    try {
        init();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}
```

/**

* Inicializa la aplicacion. Aqui se instancian los elementos que conforman el Form.

*/

```
private void init() throws Exception {
```

```
    //Mensajes
```

```
    StringItem mensaje = new StringItem("Películas <El Rapidito>", "");
```

```
    mensaje.setLayout(StringItem.LAYOUT_CENTER);
```

```
    StringItem mensaje1 = new StringItem("", "Por favor ingrese su Cedula y su Password.");
```

```
    result = new StringItem("", "");
```

```
    //TextField
```

```
    Login = new TextField("Login: ", "", 15, TextField.ANY);
```

```
    Contraseña = new TextField("Password: ", "", 15, TextField.PASSWORD);
```

```
    //Botones
```

```
    salir = new Command("Cerrar", Command.EXIT, 1);
```

```
    conectar = new Command("Conectar", Command.OK, 1);
```

```
    //Alertas
```

```
    Error = new Alert("Mensaje: ", "", null, AlertType.ERROR);
```

```
    Error.setTimeout(Alert.FOREVER);
```

```
    //Agregamos el listener
```

```
    this.setCommandListener(new ValidarForm_this_commandAdapter(this));
```

```
    //Agregamos los elementos a el Form
```

```
    append(mensaje);
```

```
    append(mensaje1);
```



```
        addCommand(salir);
        addCommand(conectar);
        append(Login);
        append(Contraseña);
        append(result);
    }

    /**
     * Generamos un commandAction para este Formulario.
     * @param command
     * @param displayable
     */

    public void this_commandAction(Command command, Displayable displayable) {
        //Si presiona "Cerrar"
        if (command.getCommandType() == Command.EXIT) {
            VideoMIDlet.midlet.destroyApp(true);
            VideoMIDlet.midlet.notifyDestroyed();
            VideoMIDlet.midlet = null;
        }

        //Si presiona "Conectar"
        else if (command.getCommandType() == Command.OK) {
            System.out.println("*****Validando Usuario*****");
            //Los parametros que vamos a pasar para que el usuario sea validado.
            String param[] = {"login="+Login.getString(), "pwd="+Contraseña.getString()};
            setParameters(param);

            Login.setString("");
            Contraseña.setString("");
            result.setText("");
            Thread t = new Thread(this);
            t.start();
            //Inicializa el Hilo para conectarse con el Modulo Conexion y establecer la conexión HTTP
        }
    }

    /**
     * Inicia el Hilo el cua nos permite realizar las consultas al servidor.
     */
    public void run() {
        //Le indicamos la ubicación de el sitio web.
        Conexion conn = new Conexion("http://172.16.130.164/WebVideoTienda/login.php");
        try{
            response = conn.sendWithGET(parameters);
            System.out.println(response);
        }catch(Exception e){
            Error.setString("Servidor no encontrado!");
            VideoMIDlet.midlet.displayMgr.pushDisplayable(Error,new ValidarForm());
            e.printStackTrace();
        }
        //Si el servidor no nos da una respuesta
        if (response == null) {
            Error.setString(conn.getErrorMessage());
            VideoMIDlet.midlet.displayMgr.pushDisplayable(Error,new ValidarForm());
        }
        else
        {
            //Tenemos una respuesta del servidor
            //Analizamos la respuesta, dado que nos llega con basura, corremos 3 caracteres, para sacar la respuesta
            en limpio.
        }
    }
}
```



```
String resp= response.substring(3);

if (resp.equals("Valido")){
    result.setText("Usuario Valido");
    System.out.println("Usuario Valido");
    //Como el usuario es valido puede ingresar al sistema.
    VideoMIDlet.midlet.displayMgr.pushDisplayable(new PeliculasList());
}
else if (resp.equals("Invalido")){
    result.setLabel("Usuario Invalido");
    result.setText("Inscribite a nuestro servicio por tan solo $50000 mensuales. ");
    System.out.println("Usuario Invalido");
}
else if (resp.equals("Bad Password")){
    result.setLabel("Password Invalido");
    result.setText("Recuerde que el sistema es sensible a las mayusculas.");
    System.out.println("Has ingresado mal el password");
}
}
}

/**
 * Agregamos los parametros para que sean enviados junto a la dirección HTTP.
 * @param parameters
 */
public void setParameters(String [] parameters){
    for(int i=0; i<parameters.length;i++){
        System.out.println(parameters[i]);
    }
    this.parameters=parameters;
}

}

/**
 *
 * Esta clase nos permite implementar el commandListener para recibir las opciones del usuario.
 */

class ValidarForm_this_commandAdapter implements javax.microedition.lcdui.CommandListener {
    ValidarForm adaptee;
    ValidarForm_this_commandAdapter(ValidarForm adaptee) {
        this.adaptee = adaptee;
    }

    /**
     * CommandAction
     */
    public void commandAction(Command command, Displayable displayable) {
        adaptee.this_commandAction(command, displayable);
    }
}
```



4.7. *PeliculasList.java*

Este es un tipo *List*, el cual nos permitirá desplegar las películas que tenemos en la base de datos. Claro esta, una vez que el usuario ha sido validado. Dado que recibe un string, ahí esta la información de las películas, por lo que es necesaria decodificarla y almacenarla, para ello están los arreglos que se crean al principio, de titulo, año y genero. Los cuales serán gestionados por el método *organizar()*.

En este caso la consulta se hace al principio para poder cargar las películas.

El código es el siguiente:

```
/**
 * Practica 4. Conexión HTTP mediante el Móvil [VideoTienda]
 * Electiva: Desarrollo de Aplicaciones para Dispositivos Móviles [Telemática]
 * Universidad del Cauca
 * Ing. Oscar Mauricio Caicedo Monitor: Carlos Felipe Lopez
 *
 * Created on may-2005
 */

package vista;

import javax.microedition.lcdui.*;

import control.*;
import conexion.*;

public class PeliculasList extends List implements Runnable {

    private String peliculas;
    private int i = 0;
    private Command cmdSalir;
    private Conexion cx;
    private String parameters[];
    private String action;
    private String response;
    private Alert Error;

    /* Crea los arreglos de string para cargar el titulo, el año, los actores
     * el genero, el formato y la fecha todo en nulo y solo 10 espacios
     */

    public String[] titulo = {
        null, null, null, null, null, null, null, null, null, null};
    public String[] año = {
        null, null, null, null, null, null, null, null, null, null};
    public String[] genero = {
        null, null, null, null, null, null, null, null, null, null};
```



```
/**
 * Constructora. Aquí establecemos la comunicación con el servidor
 * para solicitar las películas disponibles.
 */

public PeliculasList(){

    super("Películas Disponibles",List.IMPLICIT);
    System.out.println("*****Cargando las películas*****");
    String param[] ={" "};
    setParameters(param);
    Thread t = new Thread(this);
    t.start();
    //Establecemos la conexión con el servidor HTTP
}

/**
 * Inicializamos la lista, ya con las películas descargadas.
 * @param resp
 */
public void init(String resp){

    //Capturamos las películas
    this.peliculas = resp;
    //Las organizamos
    this.organizar();

    cmdSalir = new Command("Salir", Command.BACK, 1);

    //Las agregamos a la Lista.
    insert(0, " TÍTULO[AÑO]- GÉNERO ", null);
    for (int n = 0; n < this.i; n++) {
        insert(n + 1, titulo[n] + " [" + año[n] + "]" + " - " + genero[n], null);
    }
    addCommand(cmdSalir);
    this.setCommandListener(new PeliculasList_this_commandAdapter(this));
}

/**
 * Ya con la cadena con las películas, simplemente organizamos en los arreglos que
 * teníamos inicialmente.
 */

public void organizar() {

    while (!peliculas.startsWith("&")) { //Se rompe el ciclo cuando encuentra "&"
        //La cadena es
        asi pelicula;Titanic;19997;Drama:Saw;2005;Accion:Garfield;2004;Comedia:&

        System.out.println("i: " + i);
    }
}
```




```
//almacena en cada uno de los arreglos los datos que hay en cada uno.
this.titulo[i] = this.peliculas.substring(0, this.peliculas.indexOf(";"));
this.peliculas = this.peliculas.substring(this.peliculas.indexOf(";") + 1,
this.peliculas.length());
this.año[i] = this.peliculas.substring(0, this.peliculas.indexOf(";"));
this.peliculas = this.peliculas.substring(this.peliculas.indexOf(";") + 1,
this.peliculas.length());
this.genero[i] = this.peliculas.substring(0, this.peliculas.indexOf(":"));
this.peliculas = this.peliculas.substring(this.peliculas.indexOf(":") + 1,
this.peliculas.length());

i++;

}
}

/**
 * Inicia el Hilo el cual nos permite realizar las consultas al servidor.
 */

public void run() {
    Conexion conn = new Conexion("http://172.16.130.164/WebVideoTienda/peliculas.php");
    try{
        response = conn.sendWithGET(parameters);
        System.out.println(response);
    }catch(Exception e){
        Error.setString("Server not found !");
        VideoMIDlet.midlet.displayMgr.pushDisplayable(Error,new PeliculasList());
        e.printStackTrace();
    }
    if (response == null) {
        Error.setString(conn.getErrorMessage());
        VideoMIDlet.midlet.displayMgr.pushDisplayable(Error,new PeliculasList());
    }
    else
    {
        //Tenemos una respuesta del servidor
        //la respuesta en este caso al iniciar la cadena le indica que contiene las peliculas.
        String resp= response.substring(3);
        if (resp.startsWith("pelicula")){

            String var1 = resp.substring(0, resp.indexOf(";"));
            resp = resp.substring(resp.indexOf(";") + 1,resp.length());

            System.out.println("Respuesta de: " + var1);
            System.out.println("Respuesta: " + resp);

            try{
                init(resp);
            }catch (Exception e){}
```



```
    }  
    }  
    }  
  
    /**  
    * Generamos un commandAction para este Formulario.  
    * @param command  
    * @param displayable  
    */  
    public void this_commandAction(Command command, Displayable displayable) {  
        if (command.getCommandType() == Command.BACK) {  
            VideoMIDlet.midlet.displayMgr.popDisplayable();  
        }  
    }  
  
    /**  
    * Agregamos los parametros para que sean enviados junto a la dirección HTTP.  
    * @param parameters  
    */  
    public void setParameters(String [] parameters){  
        for(int i=0; i<parameters.length;i++){  
            System.out.println(parameters[i]);  
        }  
        this.parameters=parameters;  
    }  
}  
  
    /**  
    * Esta clase nos permite implementar el commandListener para recibir las opciones del  
    usuario.  
    */  
    class PeliculasList_this_commandAdapter implements  
    javax.microedition.lcdui.CommandListener {  
        PeliculasList adaptee;  
        PeliculasList_this_commandAdapter(PeliculasList adaptee) {  
            this.adaptee = adaptee;  
        }  
  
        /**  
        * CommandAction  
        */  
  
        public void commandAction(Command command, Displayable displayable) {  
            adaptee.this_commandAction(command, displayable);  
        }  
    }  
}
```

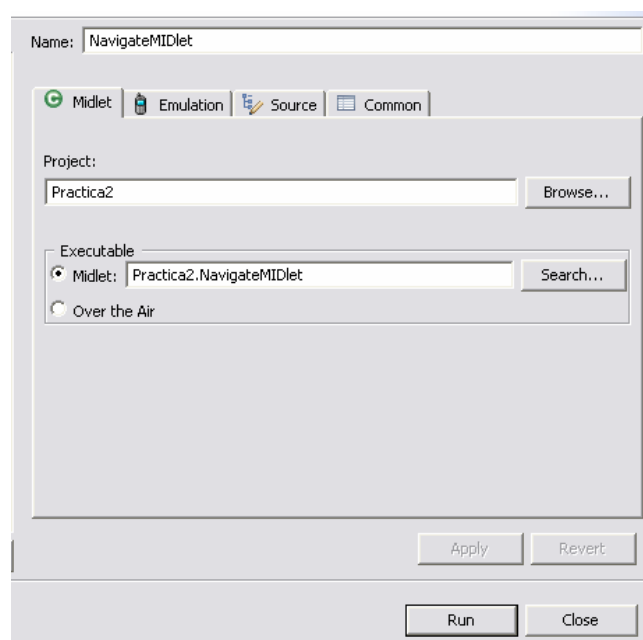


5. Compilando el MIDlet

5.1. Ejecutar el MIDlet

Para ello , vamos a la barra de menú superior en el submenu **Run>Run...** Vamos a la parte Wireless Toolkit Emulator, y presionamos **<New>** para crear el perfil para correr el Midlet, le indicamos el nombre (ejm. VideoMIDlet), el proyecto asociado (practica34 y el ejecutable (practica4.VideoMidlet).

Una vez configurado presionamos **<Apply>** y le damos **<Run>**.



Y cargará el emulador con la Aplicación.

