Capítulo 7

Planos de código

7.1 Introducción

En este apartado se procede a listar el código correspondiente a las tres aplicaciones que se han usado para probar la plataforma de servicios web XML.

7.2 Prueba 1: Números primos

7.2.1 Cliente JSR 172

El cliente está formado por el archivo *PrimoSampleMIDlet.java*, que contiene la aplicación, y las distintas clases del stub.

7.2.1.1 *PrimoSampleMIDlet.java*

Este cliente JSR 172 ha sido generado automáticamente a partir del archivo WSDL del servicio web XML mediante el generador de stubs de Sun Java One Studio Mobility 6.

```
// This MIDlet was generated by S1S.
// Contents subject to change without notice.

//package defaultpackage;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class PrimoSampleMIDlet extends MIDlet implements CommandListener {
       static public final int maxFieldSize = 255;
       static public final int alertTimeout = Alert.FOREVER;

       // The display for this MIDlet.
       protected Display display;
       protected Command exitCommand;
       protected Command selectCommand;
       protected Command goCommand;
       protected Command cancelCommand;
       protected Command mainCommand;
       private Alert alert;
       private List mainScreen;
       private GoHelper currentHelper;

       // The current command that we are executing as an index
      // into mainScreenChoices.
       protected int commandNumber;
       private TextField[] inputFields;
```

```
	protected defaultpackage.Primo_Stub proxy_Primo_Stub;

	// The top level menu items for which operation to run.
	private String[] mainScreenChoices = {"add", "Change endpoint URL"};

	public PrimoSampleMIDlet() {
		display = Display.getDisplay(this);
		exitCommand = new Command("Exit", Command.EXIT, 2);
		selectCommand = new Command("Select", Command.OK, 1);
		goCommand = new Command("Go", Command.OK, 1);
		cancelCommand = new Command("Cancel", Command.BACK, 2);
		mainCommand = new Command("main", Command.BACK, 2);

		alert = new Alert("", "", null, AlertType.INFO);
		alert.setTimeout(alertTimeout);
		setupProxy();
	}

	// Start up the MIDlet.
	public void startApp() {
		genMainScreen();
	}

	protected Screen genMainScreen() {
		if (mainScreen == null) {
			mainScreen = new List("Menu", List.IMPLICIT,
					mainScreenChoices, null);
			mainScreen.addCommand(selectCommand);
			mainScreen.addCommand(exitCommand);
			mainScreen.setCommandListener(this);
		}
		display.setCurrent(mainScreen);
		return mainScreen;
	}

	// Pause needs to stop background activities and close record stores.
	public void pauseApp() {
		if (currentHelper != null) {
			currentHelper.abort();
			currentHelper = null;
		}
	}

	// Destroy must cleanup everything not
	//handled by the garbage collector.
	public void destroyApp(boolean unconditional) {
		if (currentHelper != null) {
			currentHelper.abort();
			currentHelper = null;
		}
	}

	// Respond to commands, including exit.
	public void commandAction(Command c, Displayable s) {
		if (c == exitCommand) {
			destroyApp(false);
			notifyDestroyed();
		} else if (c == mainCommand) {
			genMainScreen();
		} else if (c == cancelCommand) {
			if (currentHelper != null) {
				currentHelper.abort();
				currentHelper = null;
			}
			genMainScreen();
		} else if ((c == List.SELECT_COMMAND) || (c == selectCommand)) {
			// Save the commandNumber for goCommand() for later
			commandNumber = mainScreen.getSelectedIndex();
```

```
                if (!genCommand()) {
                        // There was nothing to generate, so go ahead now.
                        goCommand();
                }
        } else if (c == goCommand) {
                goCommand();
        }
}

// Generate the input screen for the current command.
// If return value is false, then no input is needed.
public boolean genCommand() {
        switch (commandNumber) {
                case 0:
                        return gen_add();
                case 1:
                        genChangeEndpoint();
                        return true;
                default:
                        displayAlert(AlertType.ERROR,
                              "Unexpected index in main screen:
                              "+commandNumber, mainScreen);
                        return true;
        }
}

// Given user input (if needed), actually execute the command.
public class GoHelper implements Runnable {
        private int command;
        private boolean aborted = false;

        public GoHelper(int command) {
                this.command = command;
        }

        // run can be executed in the current thread (synchronous) or
        // in another thread (asynchronous).
        public void run() {
                try {
                        currentHelper = this;
                        switch (command) {
                                case 0:
                                        go_add(this);
                                        break;
                        }
                        currentHelper = null;
                } catch (java.lang.RuntimeException e) {
                        //e.printStackTrace();
                        if (!aborted) {
                                displayAlert(AlertType.ERROR,
                           e.getClass().getName()+": "+
                            +e.getMessage(), mainScreen);
                        }
                } catch (java.rmi.RemoteException e) {
                        //e.printStackTrace();
                        if (!aborted) {
                                displayAlert(AlertType.ERROR,
                            "java.rmi.RemoteException:
                             "+e.getMessage(), mainScreen);
                        }
                }
        }

        public void abort() {
                aborted = true;
        }

        public boolean hasAborted() {
```

```
                return aborted;
        }
    }

    // Execute the current command.
    public void goCommand() {
        if (commandNumber == 1) {
            goChangeEndpoint();
        } else if (commandNumber < 0 || commandNumber > 1) {
            displayAlert(AlertType.ERROR,
                "Unexpected index in commandNumber:
                "+commandNumber, mainScreen);
        } else {
            // Start a new thread to call the web service,
           // so as to not block
            // the UI thread and to provide a 'hour glass'.
            displayWorking();
            GoHelper helper = new GoHelper(commandNumber);
            // Depending on what threading model is desired,
           // we could also have done:
            //   display.callSerially(helper);
            // or
            //   helper.run();
            Thread thread = new Thread(helper);
            thread.start();
        }
    }

    public void genChangeEndpoint() {
        Form form = new Form("Change endpoint");
        form.addCommand(cancelCommand);
        form.addCommand(goCommand);
        form.setCommandListener(this);
        TextField tf;
        tf = new TextField("URL",
            (String)((javax.xml.rpc.Stub)proxy_Primo_Stub)._getProperty(
             javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY),
             maxFieldSize, TextField.URL);
        form.append(tf);
        inputFields = new TextField[1];
        inputFields[0] = tf;
        display.setCurrent(form);
    }

    public void goChangeEndpoint() {
        ((javax.xml.rpc.Stub)proxy_Primo_Stub)._setProperty(
            javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
            inputFields[0].getString());
        genMainScreen();
    }

    protected void displayResult(String title, String result) {
        TextBox t = new TextBox(title, result, result.length(), 0);

        t.addCommand(mainCommand);
        t.setCommandListener(this);
        display.setCurrent(t);
    }

    private void setupProxy() {
        proxy_Primo_Stub = new defaultpackage.Primo_Stub();
    }

    protected void displayAlert(AlertType type, String msg, Screen s) {
        alert.setString(msg);
        alert.setType(type);
        if (type == AlertType.ERROR) {
            alert.setTitle("Error!");
```

```
            } else if (type == AlertType.INFO) {
                  alert.setTitle("Info");
            }
            display.setCurrent(alert, s == null ? display.getCurrent() : s);
      }

      public void displayWorking() {
            Form form = new Form("Working...");
            Ticker tick = new Ticker("./\\../\\..");
            form.setTicker(tick);
            form.addCommand(cancelCommand);
            form.setCommandListener(this);
            display.setCurrent(form);
      }

      public boolean gen_add() {
            Form form = new Form("add");
            form.addCommand(cancelCommand);
            form.addCommand(goCommand);
            form.setCommandListener(this);
            java.util.Vector fields = new java.util.Vector();
            TextField field;
            field = new TextField("a", "", 11, TextField.NUMERIC);
            form.append(field);
            fields.addElement(field);
            inputFields = new TextField[fields.size()];
            fields.copyInto(inputFields);
            display.setCurrent(form);
            return true;
      }

      public void go_add(GoHelper helper) throws java.rmi.RemoteException {
            java.lang.String a = inputFields[0].getString();
            java.lang.String _returnValue = proxy_Primo_Stub.esPrimo(a);
            if (helper.hasAborted()) {
                  return;
            }

            StringBuffer result = new StringBuffer();
            result.append(
                  (((_returnValue) == null) ? "null" : _returnValue));
            displayResult("add", result.toString());
      }

}
```

7.2.1.2 *EsPrimo.java*

```
// This class was generated by the JAXRPC SI, do not edit.
// Contents subject to change without notice.
// JSR-172 Reference Implementation wscompile 1.0, using: JAX-RPC Standard
Implementation (1.1, build R59)

package defaultpackage;


public class EsPrimo {
    protected java.lang.String a;

    public EsPrimo() {
    }

    public EsPrimo(java.lang.String a) {
        this.a = a;
    }
```

```
    public java.lang.String getA() {
        return a;
    }

    public void setA(java.lang.String a) {
        this.a = a;
    }
}
```

7.2.1.3 *EsPrimoResponse.java*

```
// This class was generated by the JAXRPC SI, do not edit.
// Contents subject to change without notice.
// JSR-172 Reference Implementation wscompile 1.0, using: JAX-RPC Standard
Implementation (1.1, build R59)

package defaultpackage;

public class EsPrimoResponse {
    protected java.lang.String primoResult;

    public EsPrimoResponse() {
    }

    public EsPrimoResponse(java.lang.String primoResult) {
        this.primoResult = primoResult;
    }

    public java.lang.String getPrimoResult() {
        return primoResult;
    }

    public void setPrimoResult(java.lang.String primoResult) {
        this.primoResult = primoResult;
    }
}
```

7.2.1.4 *Primo.java*

```
// This class was generated by 172 StubGenerator.
// Contents subject to change without notice.
// @generated

package defaultpackage;

public interface Primo extends java.rmi.Remote {
       public java.lang.String esPrimo(java.lang.String a) throws
java.rmi.RemoteException;

}
```

7.2.1.5 *Primo_Stub.java*

```
// This class was generated by 172 StubGenerator.
// Contents subject to change without notice.
// @generated

package defaultpackage;
```

```
import javax.xml.rpc.JAXRPCException;
import javax.xml.namespace.QName;
import javax.microedition.xml.rpc.Operation;
import javax.microedition.xml.rpc.Type;
import javax.microedition.xml.rpc.ComplexType;
import javax.microedition.xml.rpc.Element;

public class Primo_Stub implements defaultpackage.Primo,
                           javax.xml.rpc.Stub {
       private String[] _propertyNames;
       private Object[] _propertyValues;

       public Primo_Stub() {
              _propertyNames = new String[] {ENDPOINT_ADDRESS_PROPERTY};
              _propertyValues = new Object[]
                        {"http://localhost:8080/axis/services/Math"};
       }

       public void _setProperty(String name, Object value) {
              int size = _propertyNames.length;
              for (int i = 0; i < size; ++i) {
                     if (_propertyNames[i].equals(name)) {
                            _propertyValues[i] = value;
                            return;
                     }
              }
              // Need to expand our array for a new property
              String[] newPropNames = new String[size + 1];
              System.arraycopy(_propertyNames, 0, newPropNames, 0, size);
              _propertyNames = newPropNames;
              Object[] newPropValues = new Object[size + 1];
              System.arraycopy(_propertyValues, 0, newPropValues, 0, size);
              _propertyValues = newPropValues;

              _propertyNames[size] = name;
              _propertyValues[size] = value;
       }

       public Object _getProperty(String name) {
              for (int i = 0; i < _propertyNames.length; ++i) {
                     if (_propertyNames[i].equals(name)) {
                            return _propertyValues[i];
                     }
              }
              if (ENDPOINT_ADDRESS_PROPERTY.equals(name) ||
                  USERNAME_PROPERTY.equals(name) ||
                  PASSWORD_PROPERTY.equals(name)) {
                     return null;
              }
              if (SESSION_MAINTAIN_PROPERTY.equals(name)) {
                     return new java.lang.Boolean(false);
              }
              throw new JAXRPCException("Stub does not recognize property: "+
                     name);
       }

       protected void _prepOperation(Operation op) {
              for (int i = 0; i < _propertyNames.length; ++i) {
                     op.setProperty(_propertyNames[i],
                               _propertyValues[i].toString());
              }
       }

       //
       //  Begin user methods
       //
```

```
	public java.lang.String esPrimo(java.lang.String a)
			throws java.rmi.RemoteException {
		// Copy the incoming values into an Object array if needed.
		Object[] inputObject = new Object[1];
		inputObject[0] = a;

		Operation op = Operation.newInstance(_qname_EsPrimo,
					_type_EsPrimo,
					_type_EsPrimoResponse);
		_prepOperation(op);
		op.setProperty(Operation.SOAPACTION_URI_PROPERTY, "");
		Object resultObj;
		try {
			resultObj = op.invoke(inputObject);
		} catch (JAXRPCException e) {
			Throwable cause = e.getLinkedCause();
			if (cause instanceof java.rmi.RemoteException) {
				throw (java.rmi.RemoteException) cause;
			}
			throw e;
		}
		java.lang.String result;
		// Convert the result into the right Java type.
		// Unwrapped return value
		Object primoResultObj = ((Object[])resultObj)[0];
		result = (java.lang.String)primoResultObj;
		return result;
	}
	//
	//  End user methods
	//

	protected static final QName _qname_A =
			new QName("http://math.samples/", "A");
	protected static final QName _qname_EsPrimo =
			new QName("http://math.samples/", "EsPrimo");
	protected static final QName _qname_EsPrimoResponse =
			new QName("http://math.samples/", "EsPrimoResponse");
	protected static final QName _qname_PrimoResult =
			new QName("http://math.samples/", "PrimoResult");
	protected static final Element _type_EsPrimo;
	protected static final Element _type_EsPrimoResponse;
	static {
		// Create all of the Type's that this stub uses, once.
		Element _type_A;
		_type_A = new Element(_qname_A, Type.STRING);
		ComplexType _complexType_esPrimo;
		_complexType_esPrimo = new ComplexType();
		_complexType_esPrimo.elements = new Element[1];
		_complexType_esPrimo.elements[0] = _type_A;
		_type_EsPrimo = new Element(_qname_EsPrimo,
					_complexType_esPrimo);
		Element _type_PrimoResult;
		_type_PrimoResult = new Element(_qname_PrimoResult,
				Type.STRING);
		ComplexType _complexType_esPrimoResponse;
		_complexType_esPrimoResponse = new ComplexType();
		_complexType_esPrimoResponse.elements = new Element[1];
		_complexType_esPrimoResponse.elements[0] = _type_PrimoResult;
		_type_EsPrimoResponse = new Element(_qname_EsPrimoResponse,
				_complexType_esPrimoResponse);
	}

}
```

7.2.2 Servidor JSR 172

El servidor consta de una sola clase, Primo.java. Aparte también se listan los archivos WSDL y WSDD.

7.2.2.1 *Primo.java*

```
package samples.math;

public class Primo {
    public String esPrimo(String a) {
        double b = Double.valueOf(a).doubleValue();
        int num = 2;
        String resultado;
        boolean esPrimo = true;
        do
        {
            if (b % num == 0)
                esPrimo = false;
            num++;
        } while (num <= java.lang.Math.sqrt(b));
        if (esPrimo)
            resultado = "El numero es primo";
        else
            resultado = "El numero no es primo";
        return resultado;
    }
}
```

7.2.2.2 *Math.wsdl*

```
<wsdl:definitions
 targetNamespace="http://math.samples/">

      <!--
        WSDL created by Apache Axis version: 1.2RC2
        Built on Nov 16, 2004 (12:19:44 EST)
       -->

  <wsdl:types>
    <schema
      elementFormDefault="qualified"
      targetNamespace="http://math.samples/">

      <element name="EsPrimo">
        <complexType>
          <sequence>
            <element name="A" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>

      <element name="EsPrimoResponse">
        <complexType>
          <sequence>
            <element name="PrimoResult" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>
    </schema>
  </wsdl:types>
```

```
  <wsdl:message name="EsPrimoResponse">
    <wsdl:part element="impl:EsPrimoResponse" name="parameters"/>
  </wsdl:message>

  <wsdl:message name="EsPrimoRequest">
    <wsdl:part element="impl:EsPrimo" name="parameters"/>
  </wsdl:message>

  <wsdl:portType name="Primo">
    <wsdl:operation name="EsPrimo">
      <wsdl:input message="impl:EsPrimoRequest" name="EsPrimoRequest"/>
      <wsdl:output message="impl:EsPrimoResponse" name="EsPrimoResponse"/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="MathSoapBinding" type="impl:Primo">
    <wsdlsoap:binding style="document"
       transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="EsPrimo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="EsPrimoRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>

      <wsdl:output name="EsPrimoResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="PrimoService">
    <wsdl:port binding="impl:MathSoapBinding" name="Math">
      <wsdlsoap:address location="http://localhost:8080/axis/services/Math"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

7.2.2.3 *Deploy.wsdd*

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="Math" provider="java:RPC" style="wrapped" use="literal">
      <parameter name="wsdlTargetNamespace" value="http://math.samples/"/>
      <parameter name="className" value="samples.math.Primo"/>
      <operation name="esPrimo" qname="operNS:EsPrimo"
                 xmlns:operNS="http://math.samples/"
                 returnQName="retNS:PrimoResult"
                 xmlns:retNS="http://math.samples/"
                 returnType="rtns:string"
                 xmlns:rtns="http://www.w3.org/2001/XMLSchema" >
        <parameter qname="pns:A" xmlns:pns="http://math.samples/"
                   type="tns:string"
xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
      </operation>
      <parameter name="allowedMethods" value="esPrimo"/>
  </service>
</deployment>
```

7.2.2.4 *Undeploy.wsdd*

```
<undeployment
    xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="Math"/>
</undeployment>
```

7.2.3 Cliente kSOAP

Este cliente es muy similar al de JSR 172, ya que se ha basado en éste, realizando las modificaciones necesarias para adaptarlo a kSOAP.

```
// This MIDlet was generated by S1S.  Contents subject to change without
notice.

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import org.ksoap.*;
import org.ksoap.transport.*;
import org.ksoap.SoapObject;
public class PrimoSampleMIDlet extends MIDlet implements CommandListener {
       static public final int maxFieldSize = 255;
       static public final int alertTimeout = Alert.FOREVER;

       // The display for this MIDlet.
       protected Display display;
       protected Command exitCommand;
       protected Command selectCommand;
       protected Command goCommand;
       protected Command cancelCommand;
       protected Command mainCommand;
       private Alert alert;
       private List mainScreen;
       private GoHelper currentHelper;
        private String soapUrl= "http://localhost:8080/axis/services/Mathks";

      // The current command that we are executing as an
      // index into mainScreenChoices.

       protected int commandNumber;
       private TextField[] inputFields;

       // The top level menu items for which operation to run.
       private String[] mainScreenChoices = {"add", "Change endpoint URL"};

       public PrimoSampleMIDlet() {
              display = Display.getDisplay(this);
              exitCommand = new Command("Exit", Command.EXIT, 2);
              selectCommand = new Command("Select", Command.OK, 1);
              goCommand = new Command("Go", Command.OK, 1);
              cancelCommand = new Command("Cancel", Command.BACK, 2);
              mainCommand = new Command("main", Command.BACK, 2);

              alert = new Alert("", "", null, AlertType.INFO);
              alert.setTimeout(alertTimeout);
       }

       // Start up the MIDlet.
       public void startApp() {
              genMainScreen();
       }

       protected Screen genMainScreen() {
```

```
            if (mainScreen == null) {
                  mainScreen = new List("Menu", List.IMPLICIT,
                          mainScreenChoices, null);
                  mainScreen.addCommand(selectCommand);
                  mainScreen.addCommand(exitCommand);
                  mainScreen.setCommandListener(this);
            }
            display.setCurrent(mainScreen);
            return mainScreen;
      }

      // Pause needs to stop background activities and close record stores.
      public void pauseApp() {
            if (currentHelper != null) {
                  currentHelper.abort();
                  currentHelper = null;
            }
      }

       // Destroy must cleanup everything not handled by
       // the garbage collector.
      public void destroyApp(boolean unconditional) {
            if (currentHelper != null) {
                  currentHelper.abort();
                  currentHelper = null;
            }
      }

      // Respond to commands, including exit.
      public void commandAction(Command c, Displayable s) {
            if (c == exitCommand) {
                  destroyApp(false);
                  notifyDestroyed();
            } else if (c == mainCommand) {
                  genMainScreen();
            } else if (c == cancelCommand) {
                  if (currentHelper != null) {
                        currentHelper.abort();
                        currentHelper = null;
                  }
                  genMainScreen();
            } else if ((c == List.SELECT_COMMAND) || (c == selectCommand)) {
                  // Save the commandNumber for goCommand() for later
                  commandNumber = mainScreen.getSelectedIndex();
                  if (!genCommand()) {
                        // There was nothing to generate, so go ahead now.
                        goCommand();
                  }
            } else if (c == goCommand) {
                  goCommand();
            }
      }

      // Generate the input screen for the current command.
      // If return value is false, then no input is needed.
      public boolean genCommand() {
            switch (commandNumber) {
                  case 0:
                        return gen_add();
                  case 1:
                        genChangeEndpoint();
                        return true;
                  default:
                        displayAlert(AlertType.ERROR,
                              "Unexpected index in main screen: "+
                               commandNumber, mainScreen);
                        return true;
            }
```

```
	}

	// Given user input (if needed), actually execute the command.
	public class GoHelper implements Runnable {
		private int command;
		private boolean aborted = false;

		public GoHelper(int command) {
			this.command = command;
		}

		// run can be executed in the current thread (synchronous) or
		// in another thread (asynchronous).
		public void run() {
			try {
				currentHelper = this;
				switch (command) {
					case 0:
						go_add(this);
						break;
				}
				currentHelper = null;
			} catch (java.lang.RuntimeException e) {
				//e.printStackTrace();
				if (!aborted) {
					displayAlert(AlertType.ERROR,
						e.getClass().getName()+
						": "+e.getMessage(), mainScreen);
				}
			} catch (Exception e) {
				//e.printStackTrace();
				if (!aborted) {
					displayAlert(AlertType.ERROR,
					     "java.rmi.RemoteException: "+
					     e.getMessage(), mainScreen);
				}
			}
		}

		public void abort() {
			aborted = true;
		}

		public boolean hasAborted() {
			return aborted;
		}
	}

	// Execute the current command.
	public void goCommand() {
		if (commandNumber == 1) {
			goChangeEndpoint();
		} else if (commandNumber < 0 || commandNumber > 1) {
			displayAlert(AlertType.ERROR,
				     "Unexpected index in commandNumber: "
				     +commandNumber, mainScreen);
		} else {
			 // Start a new thread to call the web service,
			// so as to not block
			// the UI thread and to provide a 'hour glass'.
			displayWorking();
			GoHelper helper = new GoHelper(commandNumber)
			// Depending on what threading model is desired,
			// we could also have done:
			//   display.callSerially(helper);
			// or
			//   helper.run();
			Thread thread = new Thread(helper);
```

```
                thread.start();
        }
    }

    public void genChangeEndpoint() {
        Form form = new Form("Change endpoint");
        form.addCommand(cancelCommand);
        form.addCommand(goCommand);
        form.setCommandListener(this);
        inputFields = new TextField[1];
        display.setCurrent(form);
    }

    public void goChangeEndpoint() {
        genMainScreen();
    }

    protected void displayResult(String title, String result) {
        TextBox t = new TextBox(title, result, result.length(), 0);

        t.addCommand(mainCommand);
        t.setCommandListener(this);
        display.setCurrent(t);
    }


    protected void displayAlert(AlertType type, String msg, Screen s) {
        alert.setString(msg);
        alert.setType(type);
        if (type == AlertType.ERROR) {
            alert.setTitle("Error!");
        } else if (type == AlertType.INFO) {
            alert.setTitle("Info");
        }
        display.setCurrent(alert, s == null ? display.getCurrent() : s);
    }

    public void displayWorking() {
        Form form = new Form("Working...");
        Ticker tick = new Ticker("./\\../\\..");
        form.setTicker(tick);
        form.addCommand(cancelCommand);
        form.setCommandListener(this);
        display.setCurrent(form);
    }

    public boolean gen_add() {
        Form form = new Form("add");
        form.addCommand(cancelCommand);
        form.addCommand(goCommand);
        form.setCommandListener(this);
        java.util.Vector fields = new java.util.Vector();
        TextField field;
        field = new TextField("a", "", 11, TextField.NUMERIC);
        form.append(field);
        fields.addElement(field);
        inputFields = new TextField[fields.size()];
        fields.copyInto(inputFields);
        display.setCurrent(form);
        return true;
    }

    public void go_add(GoHelper helper) throws Exception{
        int a = java.lang.Integer.parseInt(inputFields[0].getString());

      SoapObject client = new SoapObject ( "urn:Mathks", "esPrimo");
      client.addProperty("a", Integer.toString(a));
      HttpTransport ht = new HttpTransport(soapUrl,"esPrimo");
```

```
            java.lang.String _returnValue = ht.call(client).toString();

            if (helper.hasAborted()) {
                  return;
            }

            StringBuffer result = new StringBuffer();
            result.append(
                     (((_returnValue) == null) ? "null" : _returnValue));
            displayResult("esPrimo", result.toString());
      }

}
```

7.2.4 Servidor kSOAP

El código es muy similar al servidor JSR 172, aunque los archivos WSDD y WSDL cambian bastante.

7.2.4.1 *Primo.java*

Es básicamente igual que el servidor JSR 172, tan sólo cambia el paquete al que pertenece.

```
package samples.mathks;

public class Primo {
    public String esPrimo(String a) {
        double b = Double.valueOf(a).doubleValue();
        int num = 2;
        String resultado;
        boolean esPrimo = true;
        do
        {
            if (b % num == 0)
                esPrimo = false;
            num++;
        } while (num <= java.lang.Math.sqrt(b));
        if (esPrimo)
            resultado = "El numero es primo";
        else
            resultado = "El numero no es primo";
        return resultado;
    }
}
```

7.2.4.2 *Mathks.wsdl*

```
<wsdl:definitions targetNamespace="http://math.samples/">

       <!--
         WSDL created by Apache Axis version: 1.2RC2
         Built on Nov 16, 2004 (12:19:44 EST)
        -->

  <wsdl:message name="esPrimoRequest">
    <wsdl:part name="in0" type="soapenc:string"/>
  </wsdl:message>

  <wsdl:message name="esPrimoResponse">
    <wsdl:part name="esPrimoReturn" type="soapenc:string"/>
```

```
  </wsdl:message>

  <wsdl:portType name="Primo">
    <wsdl:operation name="esPrimo" parameterOrder="in0">
      <wsdl:input message="impl:esPrimoRequest" name="esPrimoRequest"/>
      <wsdl:output message="impl:esPrimoResponse" name="esPrimoResponse"/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="MathksSoapBinding" type="impl:Primo">
    <wsdlsoap:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="esPrimo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="esPrimoRequest">
        <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://mathks.samples" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="esPrimoResponse">
        <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://math.samples/" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="PrimoService">
    <wsdl:port binding="impl:MathksSoapBinding" name="Mathks">
      <wsdlsoap:address location=
             "http://localhost:8080/axis/services/Mathks"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

7.2.4.3 *Deploy.wsdd*

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="Mathks" provider="java:RPC">
      <parameter name="wsdlTargetNamespace" value="http://math.samples/"/>
      <parameter name="className" value="samples.mathks.Primo"/>
      <parameter name="allowedMethods" value="esPrimo"/>
  </service>
</deployment>
```

7.2.4.4 *Undeploy.wsdd*

```
<undeployment
    xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="Mathks"/>
</undeployment>
```

7.3 Prueba 2: Cartelera de cine

7.3.1 Cliente JSR 172

Formado por el archivo *CineSampleMidlet.java* y las clases del stub.

7.3.1.1 *CineSampleMidlet.java*

Este archivo ha sido generado automáticamente por el generador de stubs del Mobility Studio 6

```
// This MIDlet was generated by S1S.  Contents subject to
// change without notice.

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class CineSampleMIDlet extends MIDlet implements CommandListener {
	static public final int maxFieldSize = 255;
	static public final int alertTimeout = Alert.FOREVER;

	// The display for this MIDlet.
	protected Display display;
	protected Command exitCommand;
	protected Command selectCommand;
	protected Command goCommand;
	protected Command cancelCommand;
	protected Command mainCommand;
	private Alert alert;
	private List mainScreen;
	private GoHelper currentHelper;

	// The current command that we are executing as an index into
	 // mainScreenChoices.
	protected int commandNumber;
	private TextField[] inputFields;
	protected defaultpackage.Cine_Stub proxy_Cine_Stub;

	// The top level menu items for which operation to run.
	private String[] mainScreenChoices = {"peli", "Change endpoint URL"};

	public CineSampleMIDlet() {
		display = Display.getDisplay(this);
		exitCommand = new Command("Exit", Command.EXIT, 2);
		selectCommand = new Command("Select", Command.OK, 1);
		goCommand = new Command("Go", Command.OK, 1);
		cancelCommand = new Command("Cancel", Command.BACK, 2);
		mainCommand = new Command("main", Command.BACK, 2);

		alert = new Alert("", "", null, AlertType.INFO);
		alert.setTimeout(alertTimeout);
		setupProxy();
	}

	// Start up the MIDlet.
	public void startApp() {
		genMainScreen();
	}

	protected Screen genMainScreen() {
		if (mainScreen == null) {
			mainScreen = new List("Menu", List.IMPLICIT,
					mainScreenChoices, null);
			mainScreen.addCommand(selectCommand);
			mainScreen.addCommand(exitCommand);
			mainScreen.setCommandListener(this);
		}
		display.setCurrent(mainScreen);
		return mainScreen;
	}
```

```
	// Pause needs to stop background activities and close record stores.
	public void pauseApp() {
		if (currentHelper != null) {
			currentHelper.abort();
			currentHelper = null;
		}
	}

	// Destroy must cleanup everything not handled by
	// the garbage collector.
	public void destroyApp(boolean unconditional) {
		if (currentHelper != null) {
			currentHelper.abort();
			currentHelper = null;
		}
	}

	// Respond to commands, including exit.
	public void commandAction(Command c, Displayable s) {
		if (c == exitCommand) {
			destroyApp(false);
			notifyDestroyed();
		} else if (c == mainCommand) {
			genMainScreen();
		} else if (c == cancelCommand) {
			if (currentHelper != null) {
				currentHelper.abort();
				currentHelper = null;
			}
			genMainScreen();
		} else if ((c == List.SELECT_COMMAND) || (c == selectCommand)) {
			// Save the commandNumber for goCommand() for later
			commandNumber = mainScreen.getSelectedIndex();
			if (!genCommand()) {
				// There was nothing to generate, so go ahead now.
				goCommand();
			}
		} else if (c == goCommand) {
			goCommand();
		}
	}

	// Generate the input screen for the current command.
	// If return value is false, then no input is needed.
	public boolean genCommand() {
		switch (commandNumber) {
			case 0:
				return gen_peli();
			case 1:
				genChangeEndpoint();
				return true;
			default:
				displayAlert(AlertType.ERROR,
					"Unexpected index in main screen: "
					 +commandNumber, mainScreen);
				return true;
		}
	}

	// Given user input (if needed), actually execute the command.
	public class GoHelper implements Runnable {
		private int command;
		private boolean aborted = false;

		public GoHelper(int command) {
			this.command = command;
		}
```

```
            // run can be executed in the current thread (synchronous) or
            // in another thread (asynchronous).
            public void run() {
                  try {
                        currentHelper = this;
                        switch (command) {
                              case 0:
                                    go_peli(this);
                                    break;
                        }
                        currentHelper = null;
                  } catch (java.lang.RuntimeException e) {
                        //e.printStackTrace();
                        if (!aborted) {
                              displayAlert(AlertType.ERROR,
                                          e.getClass().getName()+
                                          ": "+e.getMessage(), mainScreen);
                        }
                  } catch (java.rmi.RemoteException e) {
                        //e.printStackTrace();
                        if (!aborted) {
                              displayAlert(AlertType.ERROR,
                                "java.rmi.RemoteException:
                                "+e.getMessage(), mainScreen);
                        }
                  }
            }

            public void abort() {
                  aborted = true;
            }

            public boolean hasAborted() {
                  return aborted;
            }
      }

      // Execute the current command.
      public void goCommand() {
            if (commandNumber == 1) {
                  goChangeEndpoint();
            } else if (commandNumber < 0 || commandNumber > 1) {
                  displayAlert(AlertType.ERROR,
                                "Unexpected index in commandNumber: "
                                +commandNumber, mainScreen);
            } else {
                  // Start a new thread to call the web service,
                  // so as to not block
                  // the UI thread and to provide a 'hour glass'.
                  displayWorking();
                  GoHelper helper = new GoHelper(commandNumber);
                  // Depending on what threading model is desired,
                  // we could also have done:
                  //   display.callSerially(helper);
                  // or
                  //   helper.run();
                  Thread thread = new Thread(helper);
                  thread.start();
            }
      }

      public void genChangeEndpoint() {
            Form form = new Form("Change endpoint");
            form.addCommand(cancelCommand);
            form.addCommand(goCommand);
            form.setCommandListener(this);
            TextField tf;
            tf = new TextField("URL",(String)((javax.xml.rpc.Stub)
```

```
                    proxy_Cine_Stub)._getProperty(
                    javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY),
                    maxFieldSize, TextField.URL);
            form.append(tf);
            inputFields = new TextField[1];
            inputFields[0] = tf;
            display.setCurrent(form);
      }

      public void goChangeEndpoint() {
            ((javax.xml.rpc.Stub)
             proxy_Cine_Stub)._setProperty(
             javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
            inputFields[0].getString());
            genMainScreen();
      }

      protected void displayResult(String title, String result) {
            TextBox t = new TextBox(title, result, result.length(), 0);

            t.addCommand(mainCommand);
            t.setCommandListener(this);
            display.setCurrent(t);
      }

      private void setupProxy() {
            proxy_Cine_Stub = new defaultpackage.Cine_Stub();
      }

      protected void displayAlert(AlertType type, String msg, Screen s) {
            alert.setString(msg);
            alert.setType(type);
            if (type == AlertType.ERROR) {
                  alert.setTitle("Error!");
            } else if (type == AlertType.INFO) {
                  alert.setTitle("Info");
            }
            display.setCurrent(alert, s == null ? display.getCurrent() : s);
      }

      public void displayWorking() {
            Form form = new Form("Working...");
            Ticker tick = new Ticker("./\\../\\..");
            form.setTicker(tick);
            form.addCommand(cancelCommand);
            form.setCommandListener(this);
            display.setCurrent(form);
      }

      public boolean gen_peli() {
            Form form = new Form("peli");
            form.addCommand(cancelCommand);
            form.addCommand(goCommand);
            form.setCommandListener(this);
            java.util.Vector fields = new java.util.Vector();
            TextField field;
            field = new TextField("datos", "", maxFieldSize, TextField.ANY);
            form.append(field);
            fields.addElement(field);
            inputFields = new TextField[fields.size()];
            fields.copyInto(inputFields);
            display.setCurrent(form);
            return true;
      }

      public void go_peli(GoHelper helper) throws java.rmi.RemoteException {
            java.lang.String datos = inputFields[0].getString();
            java.lang.String[] _returnValue = proxy_Cine_Stub.peli(datos);
```

```
            if (helper.hasAborted()) {
                  return;
            }

            StringBuffer result = new StringBuffer();
            if (_returnValue != null) {
                  for (int ii_java_lang_String = 0;
                        ii_java_lang_String < _returnValue.length;
                        ++ii_java_lang_String) {
                        if (ii_java_lang_String > 0) {
                              result.append("\n");
                        }
                        result.append(
                   (((_returnValue[ii_java_lang_String]) == null) ? "null" :
                    _returnValue[ii_java_lang_String]));
                     }
            }
            displayResult("peli", result.toString());
      }

}
```

7.3.1.2 *Cine.java*

```
// This class was generated by 172 StubGenerator.
// Contents subject to change without notice.
// @generated

package defaultpackage;

public interface Cine extends java.rmi.Remote {
      public java.lang.String[] peli(java.lang.String datos) throws
java.rmi.RemoteException;

}
```

7.3.1.3 *Cine_Stub.java*

```
// This class was generated by 172 StubGenerator.
// Contents subject to change without notice.
// @generated

package defaultpackage;

import javax.xml.rpc.JAXRPCException;
import javax.xml.namespace.QName;
import javax.microedition.xml.rpc.Operation;
import javax.microedition.xml.rpc.Type;
import javax.microedition.xml.rpc.ComplexType;
import javax.microedition.xml.rpc.Element;

public class Cine_Stub implements defaultpackage.Cine, javax.xml.rpc.Stub {
      private String[] _propertyNames;
      private Object[] _propertyValues;

      public Cine_Stub() {
            _propertyNames = new String[] {ENDPOINT_ADDRESS_PROPERTY};
            _propertyValues =
                 new Object[] {"http://localhost:8080/axis/services/Cine"};
      }

      public void _setProperty(String name, Object value) {
            int size = _propertyNames.length;
```

```
            for (int i = 0; i < size; ++i) {
                  if (_propertyNames[i].equals(name)) {
                        _propertyValues[i] = value;
                        return;
                  }
            }
            // Need to expand our array for a new property
            String[] newPropNames = new String[size + 1];
            System.arraycopy(_propertyNames, 0, newPropNames, 0, size);
            _propertyNames = newPropNames;
            Object[] newPropValues = new Object[size + 1];
            System.arraycopy(_propertyValues, 0, newPropValues, 0, size);
            _propertyValues = newPropValues;

            _propertyNames[size] = name;
            _propertyValues[size] = value;
      }

      public Object _getProperty(String name) {
            for (int i = 0; i < _propertyNames.length; ++i) {
                  if (_propertyNames[i].equals(name)) {
                        return _propertyValues[i];
                  }
            }
            if (ENDPOINT_ADDRESS_PROPERTY.equals(name) ||
                USERNAME_PROPERTY.equals(name) ||
                PASSWORD_PROPERTY.equals(name)) {
                  return null;
            }
            if (SESSION_MAINTAIN_PROPERTY.equals(name)) {
                  return new java.lang.Boolean(false);
            }
            throw new JAXRPCException("Stub does not recognize property: "+
                                      name);
      }

      protected void _prepOperation(Operation op) {
            for (int i = 0; i < _propertyNames.length; ++i) {
                  op.setProperty(_propertyNames[i],
                                 _propertyValues[i].toString());
            }
      }

      //
      //  Begin user methods
      //

      public java.lang.String[] peli(java.lang.String datos)
                throws java.rmi.RemoteException {
            // Copy the incoming values into an Object array if needed.
            Object[] inputObject = new Object[1];
            inputObject[0] = datos;

            Operation op = Operation.newInstance(_qname_Peli, _type_Peli,
                                                 _type_PeliResponse);
            _prepOperation(op);
            op.setProperty(Operation.SOAPACTION_URI_PROPERTY, "");
            Object resultObj;
            try {
                  resultObj = op.invoke(inputObject);
            } catch (JAXRPCException e) {
                  Throwable cause = e.getLinkedCause();
                  if (cause instanceof java.rmi.RemoteException) {
                        throw (java.rmi.RemoteException) cause;
                  }
                  throw e;
            }
            java.lang.String[] result;
```

```
            // Convert the result into the right Java type.
            // Unwrapped return value
            Object primoResultObj = ((Object[])resultObj)[0];
            result = (java.lang.String[]) primoResultObj;
            return result;
      }
      //
      //  End user methods
      //

      protected static final QName _qname_Peli =
                  new QName("http://math.samples/", "Peli");
      protected static final QName _qname_PeliResponse =
                  new QName("http://math.samples/", "PeliResponse");
      protected static final QName _qname_PrimoResult =
                  new QName("http://math.samples/", "PrimoResult");
      protected static final QName _qname_datos =
                  new QName("http://math.samples/", "datos");
      protected static final Element _type_Peli;
      protected static final Element _type_PeliResponse;
      static {
            // Create all of the Type's that this stub uses, once.
            Element _type_datos;
            _type_datos = new Element(_qname_datos, Type.STRING);
            ComplexType _complexType_peli;
            _complexType_peli = new ComplexType();
            _complexType_peli.elements = new Element[1];
            _complexType_peli.elements[0] = _type_datos;
            _type_Peli = new Element(_qname_Peli, _complexType_peli);
            Element _type_PrimoResult;
            _type_PrimoResult = new Element(_qname_PrimoResult,
                                            Type.STRING, 1, -1, false);
            ComplexType _complexType_peliResponse;
            _complexType_peliResponse = new ComplexType();
            _complexType_peliResponse.elements = new Element[1];
            _complexType_peliResponse.elements[0] = _type_PrimoResult;
            _type_PeliResponse = new Element(_qname_PeliResponse,
                                             _complexType_peliResponse);
      }

}
```

7.3.1.4 *Peli.java*

```
// This class was generated by the JAXRPC SI, do not edit.
// Contents subject to change without notice.
// JSR-172 Reference Implementation wscompile 1.0, using: JAX-RPC Standard
Implementation (1.1, build R59)

package defaultpackage;


public class Peli {
    protected java.lang.String datos;

    public Peli() {
    }

    public Peli(java.lang.String datos) {
        this.datos = datos;
    }

    public java.lang.String getDatos() {
        return datos;
    }
```

```
    public void setDatos(java.lang.String datos) {
        this.datos = datos;
    }
}
```

7.3.1.5 *PeliResponse.java*

```
      // This class was generated by the JAXRPC SI, do not edit.
// Contents subject to change without notice.
// JSR-172 Reference Implementation wscompile 1.0, using: JAX-RPC Standard
Implementation (1.1, build R59)

package defaultpackage;


public class PeliResponse {
    protected java.lang.String[] primoResult;

    public PeliResponse() {
    }

    public PeliResponse(java.lang.String[] primoResult) {
        this.primoResult = primoResult;
    }

    public java.lang.String[] getPrimoResult() {
        return primoResult;
    }

    public void setPrimoResult(java.lang.String[] primoResult) {
        this.primoResult = primoResult;
    }
}
```

7.3.2 Servidor JSR 172

El servidor es muy simple; consta de la clase Cine.java que contiene el método peli. También se listan los archivos WSDL y WSDD

7.3.2.1 *Cine.java*

```
package samples.cine;

public class Cine {
    public String[] peli(String a) {
          String[] resultado = new String[9];
        if (a.toLowerCase().equals("sin city"))
        {

       resultado[0]="SIN CITY";
       resultado[1]="TITULO ORIGINAL: Sin City";
       resultado[2]="DIRECTOR: Frank Miller, Robert Rodriguez";
       resultado[3]="ACTORES: Bruce Willis, Jessica Alba, " +
                  "Mickey Rourke, Rosario Dawson";
       resultado[4]="GUIONISTA: Frank Miller, Robert Rodriguez";
       resultado[5]="BASADO EN: Sin City, La Gran Matanza y "+
                 "Ese Cobarde Bastardo, de Frank Miller";
```

```
        resultado[6]="PUNTUACION: 8/10";
        resultado[7]="ARGUMENTO: Tres historias distintas ambientadas en " +
                  "Sin City, con la violencia y la corrupcion como "+
                  "denominador comun.";
        resultado[8]="CRITICA: Visualmente demoledora y sadicamente violenta";
         }
         else
         {
             resultado[0]="La película no consta en nuestra base de datos.";
             resultado[1]="Disculpe las molestias.";
             for (int i=2;i<9;i++)
                 resultado[i]=".";
         }
         return resultado;
    }
}
```

7.3.2.2 *Cine.wsdl*

```
<wsdl:definitions targetNamespace="http://math.samples/">

       <!--
          WSDL created by Apache Axis version: 1.2RC2
          Built on Nov 16, 2004 (12:19:44 EST)
        -->

  <wsdl:types>

    <schema elementFormDefault="qualified"
            targetNamespace="http://math.samples/">

      <element name="Peli">
        <complexType>
          <sequence>
            <element name="datos" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>

      <element name="PeliResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded"
                     name="PrimoResult" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>

    </schema>
  </wsdl:types>

  <wsdl:message name="PeliResponse">
    <wsdl:part element="impl:PeliResponse" name="parameters"/>
  </wsdl:message>

  <wsdl:message name="PeliRequest">
    <wsdl:part element="impl:Peli" name="parameters"/>
  </wsdl:message>

  <wsdl:portType name="Cine">
    <wsdl:operation name="Peli">
      <wsdl:input message="impl:PeliRequest" name="PeliRequest"/>
      <wsdl:output message="impl:PeliResponse" name="PeliResponse"/>
    </wsdl:operation>
  </wsdl:portType>
```

```
  <wsdl:binding name="CineSoapBinding" type="impl:Cine">
    <wsdlsoap:binding style="document"
              transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Peli">
      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="PeliRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>

      <wsdl:output name="PeliResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>

    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="CineService">
    <wsdl:port binding="impl:CineSoapBinding" name="Cine">
      <wsdlsoap:address location="http://localhost:8080/axis/services/Cine"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

7.3.2.3 *Deploy.wsdd*

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="Cine" provider="java:RPC" style="wrapped" use="literal">
      <parameter name="wsdlTargetNamespace" value="http://math.samples/"/>
      <parameter name="className" value="samples.cine.Cine"/>
      <operation name="peli" qname="operNS:Peli"
                 xmlns:operNS="http://math.samples/"
                 returnQName="retNS:PrimoResult"
                 xmlns:retNS="http://math.samples/"
                 returnType="rtns:string[]"
                 xmlns:rtns="http://www.w3.org/2001/XMLSchema" >
        <parameter qname="pns:datos" xmlns:pns="http://math.samples/"
                   type="tns:string"
                   xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
      </operation>
      <parameter name="allowedMethods" value="peli"/>
  </service>
</deployment>
```

7.3.2.4 *Undeploy.wsdd*

```
<undeployment
    xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="Cine"/>
</undeployment>
```

7.3.3 Cliente kSOAP

Basado en el cliente JSR 172, con algunas modificaciones realizadas para hacerlo compatible con kSOAP.

```
// This MIDlet was generated by S1S.  Contents subject to change without
notice.

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import org.ksoap.*;
import org.ksoap.transport.*;
import org.ksoap.SoapObject;
import java.util.*;

public class CineSampleMIDlet extends MIDlet implements CommandListener {
       static public final int maxFieldSize = 255;
       static public final int alertTimeout = Alert.FOREVER;

       // The display for this MIDlet.
       protected Display display;
       protected Command exitCommand;
       protected Command selectCommand;
       protected Command goCommand;
       protected Command cancelCommand;
       protected Command mainCommand;
       private Alert alert;
       private List mainScreen;
       private GoHelper currentHelper;
      private String soapUrl= "http://localhost:8080/axis/services/Cineks";

       // The current command that we are executing as
        // an index into mainScreenChoices.
       protected int commandNumber;
       private TextField[] inputFields;

       // The top level menu items for which operation to run.
       private String[] mainScreenChoices = {"peli", "Change endpoint URL"};

       public CineSampleMIDlet() {
              display = Display.getDisplay(this);
              exitCommand = new Command("Exit", Command.EXIT, 2);
              selectCommand = new Command("Select", Command.OK, 1);
              goCommand = new Command("Go", Command.OK, 1);
              cancelCommand = new Command("Cancel", Command.BACK, 2);
              mainCommand = new Command("main", Command.BACK, 2);

              alert = new Alert("", "", null, AlertType.INFO);
              alert.setTimeout(alertTimeout);
       }

       // Start up the MIDlet.
       public void startApp() {
              genMainScreen();
       }

       protected Screen genMainScreen() {
              if (mainScreen == null) {
                     mainScreen = new List("Menu", List.IMPLICIT,
                                           mainScreenChoices, null);
                     mainScreen.addCommand(selectCommand);
                     mainScreen.addCommand(exitCommand);
                     mainScreen.setCommandListener(this);
              }
              display.setCurrent(mainScreen);
              return mainScreen;
       }

       // Pause needs to stop background activities and close record stores.
       public void pauseApp() {
              if (currentHelper != null) {
                     currentHelper.abort();
```

```
                currentHelper = null;
        }
    }

    // Destroy must cleanup everything not handled by
    // the garbage collector.
    public void destroyApp(boolean unconditional) {
        if (currentHelper != null) {
                currentHelper.abort();
                currentHelper = null;
        }
    }

    // Respond to commands, including exit.
    public void commandAction(Command c, Displayable s) {
        if (c == exitCommand) {
                destroyApp(false);
                notifyDestroyed();
        } else if (c == mainCommand) {
                genMainScreen();
        } else if (c == cancelCommand) {
                if (currentHelper != null) {
                        currentHelper.abort();
                        currentHelper = null;
                }
                genMainScreen();
        } else if ((c == List.SELECT_COMMAND) || (c == selectCommand)) {
                // Save the commandNumber for goCommand() for later
                commandNumber = mainScreen.getSelectedIndex();
                if (!genCommand()) {
                        // There was nothing to generate, so go ahead now.
                        goCommand();
                }
        } else if (c == goCommand) {
                goCommand();
        }
    }

    // Generate the input screen for the current command.
    // If return value is false, then no input is needed.
    public boolean genCommand() {
        switch (commandNumber) {
                case 0:
                        return gen_peli();
                case 1:
                        genChangeEndpoint();
                        return true;
                default:
                        displayAlert(AlertType.ERROR,
                                    "Unexpected index in main screen: "+
                                    commandNumber, mainScreen);
                        return true;
        }
    }

    // Given user input (if needed), actually execute the command.
    public class GoHelper implements Runnable {
        private int command;
        private boolean aborted = false;

        public GoHelper(int command) {
                this.command = command;
        }

        // run can be executed in the current thread (synchronous) or
        // in another thread (asynchronous).
        public void run() {
                try {
```

```
                        currentHelper = this;
                        switch (command) {
                              case 0:
                                    go_peli(this);
                                    break;
                        }
                        currentHelper = null;
                  } catch (java.lang.RuntimeException e) {
                        //e.printStackTrace();
                        if (!aborted) {
                              displayAlert(AlertType.ERROR,
                                          e.getClass().getName()
                                          +": "+e.getMessage(),
                                          mainScreen);
                        }
                  } catch (Exception e) {
                        //e.printStackTrace();
                        if (!aborted) {
                              displayAlert(AlertType.ERROR,
                                          "Exception: "+e.getMessage(),
                                          mainScreen);
                        }
                  }
            }

            public void abort() {
                  aborted = true;
            }

            public boolean hasAborted() {
                  return aborted;
            }
      }

      // Execute the current command.
      public void goCommand() {
            if (commandNumber == 1) {
                  goChangeEndpoint();
            } else if (commandNumber < 0 || commandNumber > 1) {
                  displayAlert(AlertType.ERROR,
                              "Unexpected index in commandNumber: "+
                              commandNumber, mainScreen);
            } else {
                  // Start a new thread to call the web service,
                  // so as to not block
                  // the UI thread and to provide a 'hour glass'.
                  displayWorking();
                  GoHelper helper = new GoHelper(commandNumber);
                  // Depending on what threading model is desired,
                  // we could also have done:
                  //   display.callSerially(helper);
                  // or
                  //   helper.run();
                  Thread thread = new Thread(helper);
                  thread.start();
            }
      }

      public void genChangeEndpoint() {
            Form form = new Form("Change endpoint");
            form.addCommand(cancelCommand);
            form.addCommand(goCommand);
            form.setCommandListener(this);

            inputFields = new TextField[1];
            display.setCurrent(form);
      }
```

```
	public void goChangeEndpoint() {
		genMainScreen();
	}

	protected void displayResult(String title, String result) {
		TextBox t = new TextBox(title, result, result.length(), 0);

		t.addCommand(mainCommand);
		t.setCommandListener(this);
		display.setCurrent(t);
	}



	protected void displayAlert(AlertType type, String msg, Screen s) {
		alert.setString(msg);
		alert.setType(type);
		if (type == AlertType.ERROR) {
			alert.setTitle("Error!");
		} else if (type == AlertType.INFO) {
			alert.setTitle("Info");
		}
		display.setCurrent(alert, s == null ? display.getCurrent() : s);
	}

	public void displayWorking() {
		Form form = new Form("Working...");
		Ticker tick = new Ticker("./\\../\\..");
		form.setTicker(tick);
		form.addCommand(cancelCommand);
		form.setCommandListener(this);
		display.setCurrent(form);
	}

	public boolean gen_peli() {
		Form form = new Form("peli");
		form.addCommand(cancelCommand);
		form.addCommand(goCommand);
		form.setCommandListener(this);
		java.util.Vector fields = new java.util.Vector();
		TextField field;
		field = new TextField("datos", "", maxFieldSize, TextField.ANY);
		form.append(field);
		fields.addElement(field);
		inputFields = new TextField[fields.size()];
		fields.copyInto(inputFields);
		display.setCurrent(form);
		return true;
	}

	public void go_peli(GoHelper helper) throws Exception {
		java.lang.String datos = inputFields[0].getString();


	      SoapObject client = new SoapObject ( "urn:Cineks", "peli");
	      client.addProperty("a", datos);
	      HttpTransport ht = new HttpTransport(soapUrl,"peli");
	       String[] _returnValue = new String[9];
	      Vector resObj = new Vector();
	      resObj =(Vector)ht.call(client);


	  for (int j=0;j<resObj.size();j++){
	     _returnValue [j] = resObj.elementAt(j).toString();
	  }


		if (helper.hasAborted()) {
```

```
                    return;
              }

              StringBuffer result = new StringBuffer();
              if (_returnValue != null) {
                    for (int ii_java_lang_String = 0;
                          ii_java_lang_String < _returnValue.length;
                          ++ii_java_lang_String) {
                          if (ii_java_lang_String > 0) {
                                result.append("\n");
                          }
                          result.append(
                              (((_returnValue[ii_java_lang_String])
                               == null) ? "null" :
                               _returnValue[ii_java_lang_String]));
                    }
              }
              displayResult("peli", result.toString());
        }
}
```

7.3.4 Servidor kSOAP

El código es muy parecido al de JSR 172, aunque los archivos WSDL y WSDD cambian bastante.

7.3.4.1 *Cine.java*

```
package samples.cineks;

public class Cine {
    public String[] peli(String a) {
          String[] resultado = new String[9];
        if (a.toLowerCase().equals("sin city"))
        {

        resultado[0]="SIN CITY";
        resultado[1]="TITULO ORIGINAL: Sin City";
        resultado[2]="DIRECTOR: Frank Miller, Robert Rodriguez";
        resultado[3]="ACTORES: Bruce Willis, Jessica Alba, " +
                  "Mickey Rourke, Rosario Dawson";
        resultado[4]="GUIONISTA: Frank Miller, Robert Rodriguez";
        resultado[5]="BASADO EN: Sin City, La Gran Matanza y "+
                  "Ese Cobarde Bastardo, de Frank Miller";
        resultado[6]="PUNTUACION: 8/10";
        resultado[7]="ARGUMENTO: Tres historias distintas ambientadas en " +
                  "Sin City, con la violencia y la corrupcion como "+
                  "denominador comun.";
        resultado[8]="CRITICA: Visualmente demoledora y sadicamente violenta";
        }
        else
        {
            resultado[0]="La película no consta en nuestra base de datos.";
            resultado[1]="Disculpe las molestias.";
            for (int i=2;i<9;i++)
                resultado[i]=".";
        }
        return resultado;
    }
}
```

7.3.4.2 *Cineks.wsdl*

```
<wsdl:definitions targetNamespace="http://math.samples/">

       <!--
         WSDL created by Apache Axis version: 1.2RC2
         Built on Nov 16, 2004 (12:19:44 EST)
        -->

  <wsdl:types>
    <schema targetNamespace="http://math.samples/">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <complexType name="ArrayOf_soapenc_string">
        <complexContent>
          <restriction base="soapenc:Array">
             <attribute ref="soapenc:arrayType"
                        wsdl:arrayType="soapenc:string[]"/>
          </restriction>
        </complexContent>
      </complexType>
    </schema>
  </wsdl:types>

  <wsdl:message name="peliRequest">
    <wsdl:part name="in0" type="soapenc:string"/>
  </wsdl:message>

  <wsdl:message name="peliResponse">
    <wsdl:part name="peliReturn" type="impl:ArrayOf_soapenc_string"/>
  </wsdl:message>

  <wsdl:portType name="Cine">
    <wsdl:operation name="peli" parameterOrder="in0">
      <wsdl:input message="impl:peliRequest" name="peliRequest"/>
      <wsdl:output message="impl:peliResponse" name="peliResponse"/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="CineksSoapBinding" type="impl:Cine">
    <wsdlsoap:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="peli">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="peliRequest">
        <wsdlsoap:body
                  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://cineks.samples" use="encoded"/>
      </wsdl:input>

      <wsdl:output name="peliResponse">
        <wsdlsoap:body
                  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://math.samples/" use="encoded"/>
      </wsdl:output>

    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="CineService">
    <wsdl:port binding="impl:CineksSoapBinding" name="Cineks">
      <wsdlsoap:address
                location="http://localhost:8080/axis/services/Cineks"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

7.3.4.3 *Deploy.wsdd*

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="Cineks" provider="java:RPC">
      <parameter name="wsdlTargetNamespace" value="http://math.samples/"/>
      <parameter name="className" value="samples.cineks.Cine"/>
      <parameter name="allowedMethods" value="peli"/>
  </service>
</deployment>
```

7.3.4.4 *Undeploy.wsdd*

```
<undeployment
    xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="Cineks"/>
</undeployment>
```

7.4 Prueba 3: Servicio de directorio

7.4.1 Cliente JSR 172

El cliente JSR 172 está formado por la clase principal *ClienteMid.java*, las clases del stub y el objeto *Atributo.java*.

7.4.1.1 *ClienteMid.java*

El archivo *ClienteMid.java* asociado a JSR 172 es muy similar al original, tan sólo cambia la función *JSR172()* que se encarga de realizar la petición SOAP a través del stub y procesar la respuesta.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
import java.util.*;

// Clase que implementa una pantalla en la que se presenta una
// lista de selección de tipo popup incorporada en un formulario
public class ClienteMid extends MIDlet implements CommandListener {

  // Definimos la variable de la pantalla del dispositivo
  private Display display;
  // Definimos el formulario en donde colocaremos la selección
  private Form Pant_ini;
  private Form Seleccion1;
  private Form Seleccion2;
  private Form Empresa;
  private Form Res_Completo;
  private Form intermedio;
  private Form Conectando;

  // En este list es donde vamos a guardar los resultados cuando la
  // búsqueda tiene mas de un resultado
  private List Resultados;
```

```
// Definimos la selección de las opciones
private ChoiceGroup actividad;
private ChoiceGroup provincias;
private ChoiceGroup localidades;
private ChoiceGroup auxiliar;

// Se usa esta variable cuando se quiere introducir el nombre
// de la empresa
private TextField empresa;

// Definimos los comandos de control de la aplicación
private Command salir;
private Command volver;
private Command entrar;
private Command envio;
private Command envio2;
private Command continuar;
private Command info;
private Command ver;

//donde se va a guardar la imagen
private ImageItem imgItem;

//Direccion donde se encuentra el servidor

//Para poder presentar los diversos mensajes durante la ejecucion
private Ticker infor;

//Con este flag conseguimos indicar si se ha metido un nombre o no
private int flag;
//Con este int guardamos las referencias de las búsquedas
private int[] ref;
//Vamos a llevar a cabo un contador del numero de objetos que se reciben
private int contador;

private boolean prof;

//El stub mediante el que accederemos al servicio
protected Conector.WService_Stub proxy_WService_Stub;

// Constructor de la clase
public ClienteMid() {
    display = Display.getDisplay(this);
    proxy_WService_Stub = new Conector.WService_Stub();
  }

// Método que arranca el midlet creando la caja de texto y
// asignándole el comando de salida
protected void startApp() {
    flag=0;
    contador=0;
    prof=true;
    ref = new int[5];
    intermedio = new Form(null);
    salir = new Command ("Salir",Command.EXIT,0);
    imgItem = new ImageItem(null,null,ImageItem.LAYOUT_CENTER,"[Image]");
    intermedio.append(imgItem);
    Resultados = new List("Resultado posibles",List.EXCLUSIVE);
    Diseña_portada();

  }

// Suspende las acciones en segundo plano y libera recursos
// mientras el midlet no está activo
protected void pauseApp() {}

// Detiene toda actividad del midlet y libera los recursos
protected void destroyApp( boolean flag ) {}
```

```
public Screen Diseña_portada(){
    // Creamos el Form de la pantalla inicial
    Pant_ini = new Form ("Portada");
    //Creamos el icono de la portada
    Image p_ini = null;
    try{
        p_ini = Image.createImage("/Bienvenido.jpg");
    }

    catch (java.io.IOException ex){
        //Si no se ha podido encontrar la imagen se genera la excepcion
        System.err.println ("Excepcion:" + ex);
    }

    ImageItem ImgItem = new ImageItem
             (null,p_ini,ImageItem.LAYOUT_CENTER,"Imagen no disponible");
    entrar = new Command ("Entrar", Command.SCREEN,0);
    salir = new Command ("Salir", Command.EXIT,0);

    // A la pantalla inicial se le añaden todos los comandos
    // que sean necesarios ademas de la imagen. Se añade un
    //CommandListener para ver lo que quiere
    // hacer el usuario
    Pant_ini.append(ImgItem);
    Pant_ini.addCommand(entrar);
    Pant_ini.addCommand(salir);
    Pant_ini.setCommandListener(this);
    display.setCurrent(Pant_ini);
    return (Pant_ini);
}


public Screen Diseña_formulario_entrada(){

    // Creamos el formulario al que vamos a incorporar los
    // elementos y los comandos de control
    Seleccion1 = new Form("Eleccion actividad y ciudad");

    String activ[] = {"Autoescuela","Hoteles","Fontaneria",
           "Joyerias","Limpieza",
    "Mudanzas","Talleres Mecanicos","Urgencias","Veterinarios"};
    String prov[]={"Almeria","Cadiz","Cordoba","Granada","Huelva","Jaen",
          "Malaga","Sevilla"};

    //Dos Choice del tipo POPUP porque se representa
    // mucho mejor en la pantalla
    actividad = new ChoiceGroup("Sel. actividad:",Choice.POPUP,activ,null);
    provincias = new ChoiceGroup("Sel. ciudad:",Choice.POPUP,prov,null);

    //Indicamos unos indices por defecto
    provincias.setSelectedIndex(2,true);
    actividad.setSelectedIndex(2,true );

    // Inicializamos el comando de salida de la aplicación
    salir = new Command( "Salir",Command.EXIT,1 );
    // Creamos el comando de visualización del estado
    continuar = new Command( "Continuar",Command.SCREEN,2 );

    // Se añaden los diferentes elementos al Form ademas de espacios para
    // que la presentacion sea mucho mejor
    Seleccion1.append(new Spacer(100,30));
    Seleccion1.append(actividad);
    Seleccion1.append(new Spacer(50,15));
    Seleccion1.append(provincias);

    Seleccion1.addCommand(salir);
```

```
    Seleccion1.addCommand(continuar);
    Seleccion1.setCommandListener(this);
    display.setCurrent(Seleccion1);
    return(Seleccion1);
}

public Screen Diseña_formulario_entrada2(){

    // Se hace un llamamiento a una funcion para que nos de las
    // diferentes localidades
    // dependiendo de la provincia que se trate
    localidades = Presenta_localidades();
    Seleccion2 = new Form("Paso 2");
    Seleccion2.append( new Spacer(100,30) );
    Seleccion2.append(localidades);

    // Se van a presentar diferentes pantallas cuando se trata de
    // una urgencia
    // o bien se trata de cualquier otra opcion
    if (actividad.getSelectedIndex()==7)
    {
         String aux[] = {"Ambulancias","Bomberos","Hospitales",
        "Proteccion Civil", "Emergencias Sanitarias"};
        auxiliar= new ChoiceGroup ("Sel. urgencia:",Choice.POPUP,aux,null);
        auxiliar.setSelectedIndex(1,true);
        Seleccion2.append(new Spacer(50,15));
        Seleccion2.append(auxiliar);
    }

    // Solo en el caso de tratarse de hoteles se presenta la opcion
    // elegir el numero de tenedores o de estrellas que desea que tenga
    else if (actividad.getSelectedIndex()== 1)
    {
        String aux[] = {"1","2","3","4","5","6"};
        auxiliar= new ChoiceGroup ("Sel. categoria:",Choice.POPUP,aux,null);
        auxiliar.setSelectedIndex(1,true);
        Seleccion2.append(new Spacer(50,15));
        Seleccion2.append(auxiliar);
        infor = new Ticker("Selecciona numero de estrellas");
        Seleccion2.setTicker(infor);
    }

     //Presentamos los comandos que nos interesan que aparezcan
     envio = new Command( "Conectar",Command.SCREEN,0 );
     volver = new Command ("Volver",Command.BACK,0 );

     Seleccion2.addCommand(volver);
     Seleccion2.addCommand(envio);

     //Preparamos el boton de + detalle excepto cuando se tratan de
     //emergencias
     if(actividad.getSelectedIndex()!=7){
         info = new Command ("+ Detalle",Command.SCREEN,0);
         Seleccion2.addCommand(info);
     }

     Seleccion2.setCommandListener(this);
     display.setCurrent(Seleccion2);
     return(Seleccion2);
}

public ChoiceGroup Presenta_localidades(){
    //Segun la provincia se van a presentar las principales localidades
    ChoiceGroup local=null;
    int indice = provincias.getSelectedIndex();

    if (indice == 0){
        String loc[] = {"Almeria","Berja","Nijar","Roquetas"};
```

```
            local = new ChoiceGroup ("Sel. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(0,true);
        }
        else if (indice ==1){
            String loc[] = {"Algeciras","Cadiz","Jerez","Pto.Sta.Maria",
                            "San Fernando"};
            local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(1,true);
        }
        else if (indice==2){
            String loc[]= {"Aguilar","Baena","Cordoba","Cabra","Lucena"};
            local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(2,true);
        }
        else if (indice == 3){
            String loc[]= {"Baza","Granada","Guadix","Motril"};
            local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(1,true);
        }
        else if (indice == 4){
            String loc[]= {"Almonte","Aracena","Huelva",
                           "Isla Cristina","Moguer"};
            local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(2,true);
        }
        else if (indice == 5){
            String loc[]= {"Andujar","Bailen","Cazorla","Jaen","Linares"};
            local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(3,true);
        }
        else if (indice == 6){
            String loc[]= {"Antequera","Malaga","Marbella","Ronda",
                           "Torremolinos"};
            local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(1,true);
        }
        else {
            String loc[]= {"Carmona","Dos Hermanas","Ecija",
                           "Lora del Rio","Sevilla"};
            local = new ChoiceGroup("Selec. localidad:",Choice.POPUP,loc,null);
            local.setSelectedIndex(4,true);
        }
        return local;
      }

    // Funcion que realiza un formulario en el cual se va a poder introducir
    // El nombre de la empresa
    public Screen Diseña_formulario_empresa(){

        Empresa = new Form("Empresa");
        Empresa.append( new Spacer(100,30) );
        infor = new Ticker("Introduzca nombre de la empresa");
        empresa = new TextField("Empresa","",20,TextField.ANY);
        Empresa.setTicker(infor);

        envio = new Command( "Conectar",Command.SCREEN,0 );
        volver = new Command ("Volver",Command.BACK,0 );

        Empresa.append(empresa);
        Empresa.addCommand(envio);
        Empresa.addCommand(volver);

        Empresa.setCommandListener(this);
        display.setCurrent(Empresa);
        return(Empresa);
    }

    // Si se trata del caso en el cual hay mas de un objeto como resultado de la
```

```
// búsqueda se van añadiendo los objetos a una lista a medida que son leidos
public void Diseña_resultado_parcial(List Res,Atributo atrib)
{
    ref[contador]=atrib.getReferencia();
    Res.append("Nombre: "+atrib.getNombre() +"\nTel: " +
                atrib.getTelefono() +
                "\nDir: "+ atrib.getDireccion(),null);

}

//Cuando ya esta finalizada la lista se va a mostrar por pantalla
public Screen Presenta_resultado_parcial(List Res)
{
    salir= new Command("Salir",Command.EXIT,1);
    ver = new Command("Ver Mapa",Command.SCREEN,0);
    Res.addCommand(ver);
    Res.addCommand(salir);
    Res.setCommandListener((CommandListener)this);
    display.setCurrent(Res);
    return(Res);
}

//Se va a presentar todo el resultado, y para ello se hace uso de
//toda la informacion disponible
public Screen Diseña_resultado_completo(Atributo atrib)
{
    String aux1= atrib.getActividad();
    Res_Completo = new Form("Resultado");
    infor = new Ticker("Resultado de la busqueda");
    Res_Completo.setTicker(infor);
    ref[contador]= atrib.getReferencia();
    Res_Completo.append("\n");
    Res_Completo.append("Nombre: ");
    Res_Completo.append(atrib.getNombre());
    Res_Completo.append("\n");
    Res_Completo.append("Direccion: ");
    Res_Completo.append(atrib.getDireccion());
    Res_Completo.append("\n");
    Res_Completo.append("Telefono: ");
    Res_Completo.append(atrib.getTelefono());
    Res_Completo.append("\n");
    Res_Completo.append("Localidad: ");
    Res_Completo.append(atrib.getLocalidad());
    Res_Completo.append("\n");

    // Solo si se trata de un hotel se va a presentar mas informacion
    if(aux1.equals("Hoteles"))
    {
        Res_Completo.append("Categoria: ");
        Res_Completo.append(atrib.getCategoria() + " estrellas");

    }

    salir = new Command("Salir",Command.EXIT,0 );
    Res_Completo.addCommand(salir);

     if (!aux1.equals("Urgencias"))
     {
        ver = new Command ("Ver Mapa",Command.SCREEN,1 );
        Res_Completo.addCommand(ver);
     }

    Res_Completo.setCommandListener((CommandListener)this);
    display.setCurrent(Res_Completo);
    return(Res_Completo);
}
```

```
//Metodo que realiza la llamada al servicio web
public void JSR172(){
       try
      {
       boolean varios = false;
      int c;
       Atributo atrib = new Atributo();

      //Se guardan los resultados del formulario en strings intermedios.
       String act = actividad.getString(actividad.getSelectedIndex());
       String prov = provincias.getString(provincias.getSelectedIndex());
       String loc = localidades.getString(localidades.getSelectedIndex());
       String emp = new String();
      String cat = new String();

      //El nombre de la empresa y la categoria se rellenan solo si se han
      // cumplimentado
       if (actividad.getSelectedIndex()==7)
           emp = auxiliar.getString(auxiliar.getSelectedIndex());
      else
       {
           if (flag==1)
               emp = empresa.getString();

           if(actividad.getSelectedIndex()==1)
               cat = String.valueOf(auxiliar.getSelectedIndex() + 1);
       }

      //Se realiza la invocacion del metodo remoto
       String[] retorno = proxy_WService_Stub.query(act,prov,loc,cat,emp);

       //Rellenamos el objeto con los resultados obtenidos
       int i = 0;
       int cont = retorno.length / 8;
       while (cont-- > 0)
       {
           if (i != 0) {
               Diseña_resultado_parcial(Resultados,atrib);
               varios = true;
           }
           atrib.setActividad(retorno[i++]);
           atrib.setProvincia(retorno[i++]);
           atrib.setLocalidad(retorno[i++]);
           atrib.setCategoria(Integer.parseInt(retorno[i++]));
           atrib.setNombre(retorno[i++]);
           atrib.setTelefono(retorno[i++]);
           atrib.setDireccion(retorno[i++]);
           atrib.setReferencia(Integer.parseInt(retorno[i++]));
           atrib.setLocalidad(localidades.getString(
                               localidades.getSelectedIndex()));
       }
       if (varios)
       {
           Diseña_resultado_parcial(Resultados, atrib);

           Presenta_resultado_parcial(Resultados);
       }
       else
           Diseña_resultado_completo (atrib);
  }
  catch( Exception e )
  {
      System.out.println(e);
      Alert aviso = new Alert("Resultado", "Su búsqueda no ha producido " +
      "resultados. Por favor, pruebe otra vez.",null,null);
      aviso.setTimeout(Alert.FOREVER);
      display.setCurrent(aviso,display.getCurrent());
  }
```

```
}

// Método que controla los comandos de la pantalla
public void commandAction( Command c,Displayable d ) {
  // Si se quiere ver el estado de las opciones
  if(d == Pant_ini)
  {
      if (c == entrar)
          Diseña_formulario_entrada();
      else if (c==salir){
         // Concluimos la ejecución y liberamos recursos
         destroyApp( true );
         // Notificamos a la plataforma que el midlet desaparece
         notifyDestroyed();
      }
  }

  //Cuando se trata de la primera pantalla de seleccion se puede pasar a la
  //segunda o bien se puede salir de la aplicacion
  else if(d == Seleccion1 )
  {
      if (c == continuar)
          Diseña_formulario_entrada2();
      else if (c == salir){
          // Concluimos la ejecución y liberamos recursos
         destroyApp( true );
         // Notificamos a la plataforma que el midlet desaparece
         notifyDestroyed();
      }
    }

  // Es lo mismo que en seleccion1 pero aqui hay mas opciones
  else if (d == Seleccion2)
  {
  // Si se quiere enviar la informacion se debe hacer en un thread diferente
  // porque los tiempos de conexion y desconexion pueden hacer que el
  // telefono se bloquee
      if (c == envio){
          Diseña_Conectando();
          Thread t =  new Thread() {
              public void run() {
                  // Efectuamos la llamada al servidor
                  JSR172();
              }
          };
          t.start();
      }

      else if (c == volver)
          Diseña_formulario_entrada();

      else {
          Diseña_formulario_empresa();
          flag = 1;
      }

  }
  else if (d == Empresa)
  {
      if (c == envio){
          Diseña_Conectando();
          Thread t =  new Thread() {
              public void run() {
                  // Efectuamos la llamada al servidor
                  JSR172();
              }
          };
```

```
            t.start();
        }
        else
            Diseña_formulario_entrada2();
    }
    else if (d == Res_Completo){
        if(c == salir){
             // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();

        }

        // Esta es la parte en la cual se conecta al servidor para conseguir
        // las imagenes
        else
        {

            Thread h = new Thread(){
                public void run(){
                    getImagen(ref[0]);
                }
            };
            h.start();
        }

    }
    else if (d== intermedio)
    {
        if(c==salir)
        {
        // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();
        }

    }
    else if (c == Resultados.SELECT_COMMAND | c == ver)
    {
        final int indice= Resultados.getSelectedIndex();
        Diseña_Conectando2();
        Thread h = new Thread(){
            public void run(){
                getImagen(ref[indice]);
            }
        };
        h.start();
    }

    else if ( c== salir)
    {
        // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();

    }
}
private void getImagen(int refer) {
    ContentConnection con = null;
    InputStream is = null;
    byte[] datos = null;
    Image imagen = null;
    try {
     //Seria esta la cadena real
         String url = "http://localhost:8080/SerialServlet/"+ refer+".png";
```

```
            con = (ContentConnection)Connector.open( url );
            is = con.openInputStream();
            int longitud = (int)con.getLength();
            if( longitud > 0 ) {
                int offset = 0;
                datos = new byte[longitud];
                while( offset < longitud ) {
                    int leer = 1024;
                    if( (longitud-offset) < 1024 )
                        leer = longitud-offset;
                    int nb = is.read( datos,offset,leer );
                    offset += nb;
                }
            }
            else {
                datos = new byte[68*1024];
                int ch;
                longitud = 0;
                while( (ch = is.read()) != -1 ) {
                    datos[longitud++] = (byte)ch;
                    if ( longitud >= datos.length ) {
                        byte[] temp = new byte[datos.length+1024];
                        for( int j=0; j < datos.length; ++j ) {
                            temp[j] = datos[j];
                        }
                        longitud = datos.length;
                        datos = temp;
                    }
                }
            }
            // Creamos la imagen y la presentamos en pantalla
            imagen = Image.createImage( datos,0,longitud );
            salir= new Command ("Salir",Command.EXIT,0);
            intermedio.addCommand(salir);
            intermedio.setCommandListener(this);
            imgItem.setImage(imagen);
            display.setCurrent( intermedio);
        } catch( Exception e ) {
        } finally {
            // Cerramos las conexiones siempre que haya una interrupción
            // o salida
            try {
                if( is != null ) is.close();
                if( con != null ) con.close();
            } catch( IOException e ) {}
        }
        // Colocamos la imagen recibida en el formulario y presentamos
        // la pantalla
        imgItem.setImage( imagen );
        display.setCurrent( intermedio);
    }
    public Screen Diseña_Conectando(){

        //Creamos el Form para la pantalla cuando se está conectando
        Conectando = new Form(null);

        //Creamos el texto desplazandose por la ventana
        infor = new Ticker("Conectando ...");

        //Se crea el icono de conexion
        Image conectando = null;
        try{
            conectando = Image.createImage("/Conectando.png");
        }catch(java.io.IOException e){
            System.err.println("Excepcion:" + e);
        }
```

```
      //Creamos el item que contiene a la imagen
      ImageItem ImgItem2 = new
ImageItem(null,conectando,ImageItem.LAYOUT_CENTER,
      "Imagen no disponible");

      Conectando.setTicker(infor);
      Conectando.append(ImgItem2);
      Conectando.setCommandListener(this);
      display.setCurrent(Conectando);
      return Conectando;

  }
  public Screen Diseña_Conectando2(){
      display.setCurrent(Conectando);
      return Conectando;
  }

}
```

7.4.1.2 *Query.java*

```
// This class was generated by the JAXRPC SI, do not edit.
// Contents subject to change without notice.
// JSR-172 Reference Implementation wscompile 1.0, using: JAX-RPC Standard
Implementation (1.1, build R59)

package Conector;


public class Query {
    protected java.lang.String act;
    protected java.lang.String pro;
    protected java.lang.String loc;
    protected java.lang.String cat;
    protected java.lang.String emp;

    public Query() {
    }

    public Query(java.lang.String act, java.lang.String pro, java.lang.String
loc, java.lang.String cat, java.lang.String emp) {
        this.act = act;
        this.pro = pro;
        this.loc = loc;
        this.cat = cat;
        this.emp = emp;
    }

    public java.lang.String getAct() {
        return act;
    }

    public void setAct(java.lang.String act) {
        this.act = act;
    }

    public java.lang.String getPro() {
        return pro;
    }

    public void setPro(java.lang.String pro) {
        this.pro = pro;
    }

    public java.lang.String getLoc() {
        return loc;
```

```
    }

    public void setLoc(java.lang.String loc) {
        this.loc = loc;
    }

    public java.lang.String getCat() {
        return cat;
    }

    public void setCat(java.lang.String cat) {
        this.cat = cat;
    }

    public java.lang.String getEmp() {
        return emp;
    }

    public void setEmp(java.lang.String emp) {
        this.emp = emp;
    }
}
```

7.4.1.3 *QueryResponse.java*

```
// This class was generated by the JAXRPC SI, do not edit.
// Contents subject to change without notice.
// JSR-172 Reference Implementation wscompile 1.0, using: JAX-RPC Standard
Implementation (1.1, build R59)

package Conector;


public class QueryResponse {
    protected java.lang.String[] result;

    public QueryResponse() {
    }

    public QueryResponse(java.lang.String[] result) {
        this.result = result;
    }

    public java.lang.String[] getResult() {
        return result;
    }

    public void setResult(java.lang.String[] result) {
        this.result = result;
    }
}
```

7.4.1.4 *WService.java*

```
// This class was generated by 172 StubGenerator.
// Contents subject to change without notice.
// @generated

package Conector;

public interface WService extends java.rmi.Remote {
```

```
        public java.lang.String[] query(java.lang.String act,
                                        java.lang.String pro,
                                        java.lang.String loc,
                                        java.lang.String cat,
                                        java.lang.String emp)
                                  throws java.rmi.RemoteException;
}
```

7.4.1.5 *WService_Stub.java*

```
// This class was generated by 172 StubGenerator.
// Contents subject to change without notice.
// @generated

package Conector;

import javax.xml.rpc.JAXRPCException;
import javax.xml.namespace.QName;
import javax.microedition.xml.rpc.Operation;
import javax.microedition.xml.rpc.Type;
import javax.microedition.xml.rpc.ComplexType;
import javax.microedition.xml.rpc.Element;

public class WService_Stub implements Conector.WService, javax.xml.rpc.Stub {
       private String[] _propertyNames;
       private Object[] _propertyValues;

       public WService_Stub() {
              _propertyNames = new String[] {ENDPOINT_ADDRESS_PROPERTY};
              _propertyValues = new Object[]
{"http://localhost:8080/axis/services/WService"};
       }

       public void _setProperty(String name, Object value) {
              int size = _propertyNames.length;
              for (int i = 0; i < size; ++i) {
                     if (_propertyNames[i].equals(name)) {
                            _propertyValues[i] = value;
                            return;
                     }
              }
              // Need to expand our array for a new property
              String[] newPropNames = new String[size + 1];
              System.arraycopy(_propertyNames, 0, newPropNames, 0, size);
              _propertyNames = newPropNames;
              Object[] newPropValues = new Object[size + 1];
              System.arraycopy(_propertyValues, 0, newPropValues, 0, size);
              _propertyValues = newPropValues;

              _propertyNames[size] = name;
              _propertyValues[size] = value;
       }

       public Object _getProperty(String name) {
              for (int i = 0; i < _propertyNames.length; ++i) {
                     if (_propertyNames[i].equals(name)) {
                            return _propertyValues[i];
                     }
              }
              if (ENDPOINT_ADDRESS_PROPERTY.equals(name) ||
USERNAME_PROPERTY.equals(name) || PASSWORD_PROPERTY.equals(name)) {
                     return null;
              }
              if (SESSION_MAINTAIN_PROPERTY.equals(name)) {
                     return new java.lang.Boolean(false);
              }
```

```
            throw new JAXRPCException("Stub does not recognize property:
"+name);
      }

      protected void _prepOperation(Operation op) {
            for (int i = 0; i < _propertyNames.length; ++i) {
                  op.setProperty(_propertyNames[i],
_propertyValues[i].toString());
            }
      }

      //
      //  Begin user methods
      //

      public java.lang.String[] query(java.lang.String act, java.lang.String
pro, java.lang.String loc, java.lang.String cat, java.lang.String emp) throws
java.rmi.RemoteException {
            // Copy the incoming values into an Object array if needed.
            Object[] inputObject = new Object[5];
            inputObject[0] = act;
            inputObject[1] = pro;
            inputObject[2] = loc;
            inputObject[3] = cat;
            inputObject[4] = emp;

            Operation op = Operation.newInstance(_qname_Query, _type_Query,
_type_QueryResponse);
            _prepOperation(op);
            op.setProperty(Operation.SOAPACTION_URI_PROPERTY, "");
            Object resultObj;
            try {
                  resultObj = op.invoke(inputObject);
            } catch (JAXRPCException e) {
                  Throwable cause = e.getLinkedCause();
                  if (cause instanceof java.rmi.RemoteException) {
                        throw (java.rmi.RemoteException) cause;
                  }
                  throw e;
            }
            java.lang.String[] result;
            // Convert the result into the right Java type.
            // Unwrapped return value
            Object resultObj2 = ((Object[])resultObj)[0];
            result = (java.lang.String[]) resultObj2;
            return result;
      }
      //
      //  End user methods
      //

      protected static final QName _qname_Act = new
QName("http://db.samples/", "Act");
      protected static final QName _qname_Cat = new
QName("http://db.samples/", "Cat");
      protected static final QName _qname_Emp = new
QName("http://db.samples/", "Emp");
      protected static final QName _qname_Loc = new
QName("http://db.samples/", "Loc");
      protected static final QName _qname_Pro = new
QName("http://db.samples/", "Pro");
      protected static final QName _qname_Query = new
QName("http://db.samples/", "Query");
      protected static final QName _qname_QueryResponse = new
QName("http://db.samples/", "QueryResponse");
      protected static final QName _qname_Result = new
QName("http://db.samples/", "Result");
      protected static final Element _type_Query;
```

```
        protected static final Element _type_QueryResponse;
        static {
                // Create all of the Type's that this stub uses, once.
                Element _type_Act;
                _type_Act = new Element(_qname_Act, Type.STRING);
                Element _type_Pro;
                _type_Pro = new Element(_qname_Pro, Type.STRING);
                Element _type_Loc;
                _type_Loc = new Element(_qname_Loc, Type.STRING);
                Element _type_Cat;
                _type_Cat = new Element(_qname_Cat, Type.STRING);
                Element _type_Emp;
                _type_Emp = new Element(_qname_Emp, Type.STRING);
                ComplexType _complexType_query;
                _complexType_query = new ComplexType();
                _complexType_query.elements = new Element[5];
                _complexType_query.elements[0] = _type_Act;
                _complexType_query.elements[1] = _type_Pro;
                _complexType_query.elements[2] = _type_Loc;
                _complexType_query.elements[3] = _type_Cat;
                _complexType_query.elements[4] = _type_Emp;
                _type_Query = new Element(_qname_Query, _complexType_query);
                Element _type_Result;
                _type_Result = new Element(_qname_Result, Type.STRING, 1, -1,
                                           false);
                ComplexType _complexType_queryResponse;
                _complexType_queryResponse = new ComplexType();
                _complexType_queryResponse.elements = new Element[1];
                _complexType_queryResponse.elements[0] = _type_Result;
                _type_QueryResponse = new Element(_qname_QueryResponse,
                                                  _complexType_queryResponse);
        }

}
```

7.4.2 Servidor JSR 172

El servidor tan sólo consta de un archivo Java. Sin embargo, se incluyen también en éste apartado los archivos para desplegar y retirar el servicio y el archivo WSDL:

7.4.2.1 *WService.java*

El archivo *WService.java* contiene el método *query* que será el que realice la petición a la base de datos con los datos del mensaje SOAP, y devolverá la tabla de *string* con las respuestas.

```
package WService;

import java.io.IOException;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;

public class WService {
  public String [] query(String Act, String Pro, String Loc, String Cat,
  String Emp) throws Exception {
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/serial", "antonio", "");
    Statement stmt = con.createStatement();
```

```
    String sql ="SELECT DISTINCT * FROM serial WHERE Actividad='" + Act +
        "' and Provincia='" + Pro + "' and Localidad='" + Loc +
        "' and Categoria='" + Cat;
    if (Emp.length()!=0) {
        sql = sql + "' and Empresa='" + Emp;
    }
    sql = sql + "';";

    ResultSet record = stmt.executeQuery(sql);

    int cont=0;
    record.last();
    int Total = 8*record.getRow();
    record.beforeFirst();
    String devolver [] = new String [Total];
    while (record.next()) {
        devolver [cont++]=record.getString ("Actividad");
        devolver [cont++]=record.getString ("Provincia");
        devolver [cont++]=record.getString ("Localidad");
        devolver [cont++]=record.getString ("Categoria");
        devolver [cont++]=record.getString ("Empresa");

        devolver [cont++]=record.getString ("Telefono");
        devolver [cont++]=record.getString ("Direccion");
        devolver [cont++]=record.getString ("Referencia");
    }
    return devolver;

  }

}
```

7.4.2.2 *WService.wsdl*

El archivo WSDL asociado al servicio se crea automáticamente al desplegarlo en Axis.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://db.samples/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://db.samples/" xmlns:intf="http://db.samples/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->

 <wsdl:types>
  <schema elementFormDefault="qualified"
          targetNamespace="http://db.samples/"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="Query">
    <complexType>
     <sequence>
      <element name="Act" type="xsd:string"/>
      <element name="Pro" type="xsd:string"/>
      <element name="Loc" type="xsd:string"/>
      <element name="Cat" type="xsd:string"/>
      <element name="Emp" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="QueryResponse">
    <complexType>
```

```
     <sequence>
      <element maxOccurs="unbounded" name="Result" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>
   <wsdl:message name="QueryResponse">
      <wsdl:part element="impl:QueryResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="QueryRequest">
      <wsdl:part element="impl:Query" name="parameters"/>
   </wsdl:message>
   <wsdl:portType name="WService">
      <wsdl:operation name="Query">
         <wsdl:input message="impl:QueryRequest" name="QueryRequest"/>
         <wsdl:output message="impl:QueryResponse" name="QueryResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="WServiceSoapBinding" type="impl:WService">
      <wsdlsoap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="Query">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="QueryRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="QueryResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="WServiceService">
      <wsdl:port binding="impl:WServiceSoapBinding" name="WService">
         <wsdlsoap:address
                   location="http://localhost:8080/axis/services/WService"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

7.4.2.3 *Deploy.wsdd*

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
 <service name="WService" provider="java:RPC" style="wrapped" use="literal">
      <parameter name="wsdlTargetNamespace" value="http://db.samples/"/>
      <parameter name="className" value="WService.WService"/>
      <operation name="query" qname="operNS:Query"
                 xmlns:operNS="http://db.samples/"
                 returnQName="retNS:Result"
                 xmlns:retNS="http://db.samples/"
                 returnType="rtns:string []"
                 xmlns:rtns="http://www.w3.org/2001/XMLSchema" >
        <parameter qname="pns:Act" xmlns:pns="http://db.samples/"
                   type="tns:string"
                   xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
        <parameter qname="pns:Pro" xmlns:pns="http://db.samples/"
                   type="tns:string"
                   xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
        <parameter qname="pns:Loc" xmlns:pns="http://db.samples/"
                   type="tns:string"
                   xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
        <parameter qname="pns:Cat" xmlns:pns="http://db.samples/"
                   type="tns:string"
                   xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
```

```
        <parameter qname="pns:Emp" xmlns:pns="http://db.samples/"
                   type="tns:string"
                   xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
      </operation>
      <parameter name="allowedMethods" value="query"/>
  </service>
</deployment>
```

7.4.2.4 *Undeploy.wsdd*

```
<undeployment xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="WService"/>
</undeployment>
```

7.4.3 Cliente kSOAP

El cliente kSOAP, al igual que el de JSR 172, es muy similar a la aplicación original, añadiéndose aquí una función *ksoap()* que montará la petición SOAP, la enviará y procesará la respuesta.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
import java.util.*;

import org.ksoap.*;
import org.ksoap.transport.*;
import org.ksoap.SoapObject;
// Clase que implementa una pantalla en la que se presenta una
// lista de selección de tipo popup incorporada en un formulario
public class ClienteMid extends MIDlet implements CommandListener {

  // Definimos la variable de la pantalla del dispositivo
  private Display display;
  // Definimos el formulario en donde colocaremos la selección
  private Form Pant_ini;
  private Form Seleccion1;
  private Form Seleccion2;
  private Form Empresa;
  private Form Res_Completo;
  private Form intermedio;
  private Form Conectando;

  // En este list es donde vamos a guardar los resultados cuando la
  // búsqueda tiene mas de un resultado
  private List Resultados;

  // Definimos la selección de las opciones
  private ChoiceGroup actividad;
  private ChoiceGroup provincias;
  private ChoiceGroup localidades;
  private ChoiceGroup auxiliar;

  // Se usa esta variable cuando se quiere introducir el nombre
  // de la empresa
  private TextField empresa;

  // Definimos los comandos de control de la aplicación
  private Command salir;
  private Command volver;
  private Command entrar;
```

```
  private Command envio;
  private Command envio2;
  private Command continuar;
  private Command info;
  private Command ver;

  //donde se va a guardar la imagen
  private ImageItem imgItem;

  //Direccion donde se encuentra el servidor
  private String soapUrl= "http://localhost:8080/axis/services/WKsoap";

  //Para poder presentar los diversos mensajes durante la ejecucion
  private Ticker infor;

  //Con este flag conseguimos indicar si se ha metido un nombre o no
  private int flag;
  //Con este int guardamos las referencias de las búsquedas
  private int[] ref;
  //Vamos a llevar a cabo un contador del numero de objetos que se reciben
  private int contador;

  private boolean prof;

  // Constructor de la clase
  public ClienteMid() {
      display = Display.getDisplay(this);
    }

  // Método que arranca el midlet creando la caja de texto y
  // asignándole el comando de salida
  protected void startApp() {
      flag=0;
      contador=0;
      prof=true;
      ref = new int[5];
      intermedio = new Form(null);
      salir = new Command ("Salir",Command.EXIT,0);
      imgItem = new ImageItem(null,null,ImageItem.LAYOUT_CENTER,"[Image]");
      intermedio.append(imgItem);
      Resultados = new List("Resultado posibles",List.EXCLUSIVE);
      Diseña_portada();

    }

  // Suspende las acciones en segundo plano y libera recursos
  // mientras el midlet no está activo
  protected void pauseApp() {}

  // Detiene toda actividad del midlet y libera los recursos
  protected void destroyApp( boolean flag ) {}


  public Screen Diseña_portada(){
      // Creamos el Form de la pantalla inicial
      Pant_ini = new Form ("Portada");
      //Creamos el icono de la portada
      Image p_ini = null;
      try{
          p_ini = Image.createImage("/Bienvenido.jpg");
      }

      catch (java.io.IOException ex){
          //Si no se ha podido encontrar la imagen se genera la excepcion
          System.err.println ("Excepcion:" + ex);
      }

      ImageItem ImgItem = new ImageItem (null,p_ini,ImageItem.LAYOUT_CENTER,
```

```
                                        "Imagen no disponible");
    entrar = new Command ("Entrar", Command.SCREEN,0);
    salir = new Command ("Salir", Command.EXIT,0);

    // A la pantalla inicial se le añaden todos los comandos
    // que sean necesarios ademas de la imagen. Se añade un
    // CommandListener para ver lo que quiere
    // hacer el usuario
    Pant_ini.append(ImgItem);
    Pant_ini.addCommand(entrar);
    Pant_ini.addCommand(salir);
    Pant_ini.setCommandListener(this);
    display.setCurrent(Pant_ini);
    return (Pant_ini);
}


public Screen Diseña_formulario_entrada(){

    // Creamos el formulario al que vamos a incorporar los
    // elementos y los comandos de control
    Seleccion1 = new Form("Eleccion actividad y ciudad");

    String activ[] = {"Autoescuela","Hoteles","Fontaneria",
          "Joyerias","Limpieza", "Mudanzas","Talleres Mecanicos",
          "Urgencias","Veterinarios"};
    String prov[]={"Almeria","Cadiz","Cordoba","Granada",
          "Huelva","Jaen","Malaga","Sevilla"};

    //Dos Choice del tipo POPUP porque se
    // representa mucho mejor en la pantalla
    actividad = new ChoiceGroup("Sel. actividad:",Choice.POPUP,
           activ,null);
    provincias = new ChoiceGroup("Sel. ciudad:",Choice.POPUP,prov,null);

    //Indicamos unos indices por defecto
    provincias.setSelectedIndex(2,true);
    actividad.setSelectedIndex(2,true );

    // Inicializamos el comando de salida de la aplicación
    salir = new Command( "Salir",Command.EXIT,1 );
    // Creamos el comando de visualización del estado
    continuar = new Command( "Continuar",Command.SCREEN,2 );

    // Se añaden los diferentes elementos al Form ademas de
    // espacios para que la presentacion sea mucho mejor
    Seleccion1.append(new Spacer(100,30));
    Seleccion1.append(actividad);
    Seleccion1.append(new Spacer(50,15));
    Seleccion1.append(provincias);

    Seleccion1.addCommand(salir);
    Seleccion1.addCommand(continuar);
    Seleccion1.setCommandListener(this);
    display.setCurrent(Seleccion1);
    return(Seleccion1);
}

public Screen Diseña_formulario_entrada2(){

    // Se hace un llamamiento a una funcion para que nos de
    // las diferentes localidades
    // dependiendo de la provincia que se trate
    localidades = Presenta_localidades();
    Seleccion2 = new Form("Paso 2");
    Seleccion2.append( new Spacer(100,30) );
    Seleccion2.append(localidades);
```

```
      // Se van a presentar diferentes pantallas cuando se trata de
      // una urgencia
      // o bien se trata de cualquier otra opcion
      if (actividad.getSelectedIndex()==7)
      {
           String aux[] = {"Ambulancias","Bomberos","Hospitales",
                 "Proteccion Civil",
          "Emergencias Sanitarias"};
          auxiliar= new ChoiceGroup ("Sel. urgencia:",
                 Choice.POPUP,aux,null);
          auxiliar.setSelectedIndex(1,true);
          Seleccion2.append(new Spacer(50,15));
          Seleccion2.append(auxiliar);
      }

      // Solo en el caso de tratarse de hoteles se presenta la opcion
      // elegir el numero de tenedores o de estrellas que desea que tenga
      else if (actividad.getSelectedIndex()== 1)
      {
          String aux[] = {"1","2","3","4","5","6"};
          auxiliar= new ChoiceGroup ("Sel. categoria:",
                 Choice.POPUP,aux,null);
          auxiliar.setSelectedIndex(1,true);
          Seleccion2.append(new Spacer(50,15));
          Seleccion2.append(auxiliar);
          infor = new Ticker("Selecciona numero de estrellas");
          Seleccion2.setTicker(infor);
      }

       //Presentamos los comandos que nos interesan que aparezcan
       envio = new Command( "Conecta",Command.SCREEN,0 );
       volver = new Command ("Volver",Command.BACK,0 );

       Seleccion2.addCommand(volver);
       Seleccion2.addCommand(envio);

       //Preparamos el boton de + detalle excepto cuando se tratan de
       //emergencias
       if(actividad.getSelectedIndex()!=7){
           info = new Command ("+ Detalle",Command.SCREEN,0);
           Seleccion2.addCommand(info);
       }

       Seleccion2.setCommandListener(this);
       display.setCurrent(Seleccion2);
       return(Seleccion2);
  }

  public ChoiceGroup Presenta_localidades(){
      //Segun la provincia se van a presentar las principales localidades
      ChoiceGroup local=null;
      int indice = provincias.getSelectedIndex();

      if (indice == 0){
          String loc[] = {"Almeria","Berja","Nijar","Roquetas"};
          local = new ChoiceGroup ("Sel. localidad:",Choice.POPUP,loc,null);
          local.setSelectedIndex(0,true);
      }
      else if (indice ==1){
          String loc[] = {"Algeciras","Cadiz","Jerez","Pto.Sta.Maria",
                          "San Fernando"};
          local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
          local.setSelectedIndex(1,true);
      }
      else if (indice==2){
          String loc[]= {"Aguilar","Baena","Cordoba","Cabra","Lucena"};
          local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
          local.setSelectedIndex(2,true);
```

```
      }
      else if (indice == 3){
          String loc[]= {"Baza","Granada","Guadix","Motril"};
          local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
          local.setSelectedIndex(1,true);
      }
      else if (indice == 4){
          String loc[]= {"Almonte","Aracena","Huelva",
                         "Isla Cristina","Moguer"};
          local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
          local.setSelectedIndex(2,true);
      }
      else if (indice == 5){
          String loc[]= {"Andujar","Bailen","Cazorla","Jaen","Linares"};
          local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
          local.setSelectedIndex(3,true);
      }
      else if (indice == 6){
          String loc[]= {"Antequera","Malaga","Marbella",
                         "Ronda","Torremolinos"};
          local = new ChoiceGroup("Sel. localidad:",Choice.POPUP,loc,null);
          local.setSelectedIndex(1,true);
      }
      else {
          String loc[]= {"Carmona","Dos Hermanas","Ecija",
                         "Lora del Rio","Sevilla"};
          local = new ChoiceGroup("Selec. localidad:",
                  Choice.POPUP,loc,null);
          local.setSelectedIndex(4,true);
      }
      return local;
    }

  // Funcion que realiza un formulario en el cual se va a poder introducir
  // El nombre de la empresa
  public Screen Diseña_formulario_empresa(){

      Empresa = new Form("Empresa");
      Empresa.append( new Spacer(100,30) );
      infor = new Ticker("Introduzca nombre de la empresa");
      empresa = new TextField("Empresa","",20,TextField.ANY);
      Empresa.setTicker(infor);

      envio = new Command( "Conecta",Command.SCREEN,0 );
      volver = new Command ("Volver",Command.BACK,0 );

      Empresa.append(empresa);
      Empresa.addCommand(envio);
      Empresa.addCommand(volver);

      Empresa.setCommandListener(this);
      display.setCurrent(Empresa);
      return(Empresa);
  }

  // Si se trata del caso en el cual hay mas de un objeto
  // como resultado de la búsqueda se van añadiendo los objetos a una lista
  // a medida que son leidos
  public void Diseña_resultado_parcial(List Res,Atributo atrib)
  {
      ref[contador]=atrib.getReferencia();
      Res.append("Nombre: "+atrib.getNombre() +"\nTel: " +
              atrib.getTelefono()+ "\nDir: "+ atrib.getDireccion(),null);

  }

  //Cuando ya esta finalizada la lista se va a mostrar por pantalla
  public Screen Presenta_resultado_parcial(List Res)
```

```
{
    salir= new Command("Salir",Command.EXIT,1);
    ver = new Command("Ver Mapa",Command.SCREEN,0);
    Res.addCommand(ver);
    Res.addCommand(salir);
    Res.setCommandListener((CommandListener)this);
    display.setCurrent(Res);
    return(Res);
}

//Se va a presentar todo el resultado, y para ello se hace uso de
//toda la informacion disponible
public Screen Diseña_resultado_completo(Atributo atrib)
{
    String aux1= atrib.getActividad();
    Res_Completo = new Form("Resultado");
    infor = new Ticker("Resultado de la busqueda");
    Res_Completo.setTicker(infor);
    ref[contador]= atrib.getReferencia();
    Res_Completo.append("\n");
    Res_Completo.append("Nombre: ");
    Res_Completo.append(atrib.getNombre());
    Res_Completo.append("\n");
    Res_Completo.append("Direccion: ");
    Res_Completo.append(atrib.getDireccion());
    Res_Completo.append("\n");
    Res_Completo.append("Telefono: ");
    Res_Completo.append(atrib.getTelefono());
    Res_Completo.append("\n");
    Res_Completo.append("Localidad: ");
    Res_Completo.append(atrib.getLocalidad());
    Res_Completo.append("\n");

    // Solo si se trata de un hotel se va a presentar mas informacion
    if(aux1.equals("Hoteles"))
    {
        Res_Completo.append("Categoria: ");
        Res_Completo.append(atrib.getCategoria() + " estrellas");

    }

    salir = new Command("Salir",Command.EXIT,0 );
    Res_Completo.addCommand(salir);

     if (!aux1.equals("Urgencias"))
     {
        ver = new Command ("Ver Mapa",Command.SCREEN,1 );
        Res_Completo.addCommand(ver);
     }

    Res_Completo.setCommandListener((CommandListener)this);
    display.setCurrent(Res_Completo);
    return(Res_Completo);
}

public void ksoap() {
         try{

       boolean varios = false;
      int c;
       Atributo atrib = new Atributo();

      //Se guardan los resultados del formulario en strings intermedios.
       String act = actividad.getString(actividad.getSelectedIndex());
       String prov = provincias.getString(provincias.getSelectedIndex());
       String loc = localidades.getString(localidades.getSelectedIndex());
       String emp = new String();
      String cat = new String();
```

```
      //El nombre de la empresa y la categoria se cumplimentan solo si se
      // han cumplimentado
       if (actividad.getSelectedIndex()==7)
           emp = auxiliar.getString(auxiliar.getSelectedIndex());
      else
       {
           if (flag==1)
               emp = empresa.getString();

           if(actividad.getSelectedIndex()==1)
               cat = String.valueOf(auxiliar.getSelectedIndex() + 1);
       }

       // Se crea el objeto SOAP y se llena con los parametros
       SoapObject client = new SoapObject ( "urn:WKsoap", "query");
        client.addProperty("Act", act);
        client.addProperty("Pro", prov);
        client.addProperty("Loc", loc);
        client.addProperty("Cat", cat);
        client.addProperty("Emp", emp);

       // Enviamos la peticion SOAP al servicio web
       HttpTransport ht = new HttpTransport(soapUrl,"query");
       Vector resobj = new Vector();
       resobj = (Vector)ht.call(client);


       String[] retorno = new String [resobj.size()];

       for (int j=0;j<resobj.size();j++){
          retorno [j] = resobj.elementAt(j).toString();
       }

       int i = 0;
       int cont = resobj.size()/8;

       // rellenamos el objeto resultado y lo añadimos a la lista
       while (cont-- > 0)
       {
           if (i != 0) {
               Diseña_resultado_parcial(Resultados,atrib);
               varios = true;
           }
           atrib.setActividad(retorno[i++]);
           atrib.setProvincia(retorno[i++]);
           atrib.setLocalidad(retorno[i++]);
           atrib.setCategoria(Integer.parseInt(retorno[i++]));
           atrib.setNombre(retorno[i++]);
           atrib.setTelefono(retorno[i++]);
           atrib.setDireccion(retorno[i++]);
           atrib.setReferencia(Integer.parseInt(retorno[i++]));
           atrib.setLocalidad(localidades.getString(
                               localidades.getSelectedIndex()));
       }
       if (varios)
       {
           Diseña_resultado_parcial(Resultados, atrib);

           Presenta_resultado_parcial(Resultados);
       }
       else
           Diseña_resultado_completo (atrib);
   }
   catch( Exception e )
   {
       System.out.println(e);
       Alert aviso = new Alert("Resultado",
```

```
            "Su búsqueda no ha producido " +
        "resultados. Por favor, pruebe otra vez.",null,null);
        aviso.setTimeout(Alert.FOREVER);
        display.setCurrent(aviso,display.getCurrent());
    }
  }

  // Método que controla los comandos de la pantalla
  public void commandAction( Command c,Displayable d ) {
    // Si se quiere ver el estado de las opciones
    if(d == Pant_ini)
    {
        if (c == entrar)
            Diseña_formulario_entrada();
        else if (c==salir){
           // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();
        }
    }

    //Cuando se trata de la primera pantalla de seleccion se
    // puede pasar a la segunda o bien se puede salir de la aplicacion
    else if(d == Seleccion1 )
    {
        if (c == continuar)
            Diseña_formulario_entrada2();
        else if (c == salir){
            // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();
        }
      }

    // Es lo mismo que en seleccion1 pero aqui hay mas opciones
    else if (d == Seleccion2)
    {
        // Si se quiere enviar la informacion se debe hacer en
        //un thread diferente
        // porque los tiempos de conexion y desconexion pueden hacer que el
        // telefono se bloquee
        if (c == envio){
            Diseña_Conectando();
            Thread t =  new Thread() {
                public void run() {
                    // Efectuamos la llamada al servidor
                    ksoap();
                }
            };
            t.start();
        }
        else if (c == volver)
            Diseña_formulario_entrada();

        else {
            Diseña_formulario_empresa();
            flag = 1;
        }

    }
    else if (d == Empresa)
    {
        if (c == envio){
            Diseña_Conectando();
            Thread t =  new Thread() {
                public void run() {
```

```
                    // Efectuamos la llamada al servidor
                    ksoap();
                }
            };
            t.start();
        }
        else
            Diseña_formulario_entrada2();
    }
    else if (d == Res_Completo){
        if(c == salir){
             // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();

        }

        // Esta es la parte en la cual se conecta al servidor para conseguir
        // las imagenes
        else
        {

            Thread h = new Thread(){
                public void run(){
                    getImagen(ref[0]);
                }
            };
            h.start();
        }

    }
    else if (d== intermedio)
    {
        if(c==salir)
        {
        // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();
        }

    }
    else if (c == Resultados.SELECT_COMMAND | c == ver)
    {
        final int indice= Resultados.getSelectedIndex();
        Diseña_Conectando2();
        Thread h = new Thread(){
            public void run(){
                getImagen(ref[indice]);
            }
        };
        h.start();
    }

    else if ( c== salir)
    {
        // Concluimos la ejecución y liberamos recursos
           destroyApp( true );
           // Notificamos a la plataforma que el midlet desaparece
           notifyDestroyed();

    }
}
private void getImagen(int refer) {
    ContentConnection con = null;
    InputStream is = null;
    byte[] datos = null;
```

```
      Image imagen = null;
      try {
       //Seria esta la cadena real
          String url = "http://localhost:8080/SerialServlet/"+ refer+".png";

          con = (ContentConnection)Connector.open( url );
          is = con.openInputStream();
          int longitud = (int)con.getLength();
          if( longitud > 0 ) {
              int offset = 0;
              datos = new byte[longitud];
              while( offset < longitud ) {
                  int leer = 1024;
                  if( (longitud-offset) < 1024 )
                      leer = longitud-offset;
                  int nb = is.read( datos,offset,leer );
                  offset += nb;
              }
          }
          else {
              datos = new byte[68*1024];
              int ch;
              longitud = 0;
              while( (ch = is.read()) != -1 ) {
                  datos[longitud++] = (byte)ch;
                  if ( longitud >= datos.length ) {
                      byte[] temp = new byte[datos.length+1024];
                      for( int j=0; j < datos.length; ++j ) {
                          temp[j] = datos[j];
                      }
                      longitud = datos.length;
                      datos = temp;
                  }
              }
          }
          // Creamos la imagen y la presentamos en pantalla
          imagen = Image.createImage( datos,0,longitud );
          salir= new Command ("Salir",Command.EXIT,0);
          intermedio.addCommand(salir);
          intermedio.setCommandListener(this);
          imgItem.setImage(imagen);
          display.setCurrent( intermedio);
      } catch( Exception e ) {
      } finally {
          // Cerramos las conexiones siempre que haya una interrupción
          // o salida
          try {
              if( is != null ) is.close();
              if( con != null ) con.close();
          } catch( IOException e ) {}
      }
      // Colocamos la imagen recibida en el formulario y presentamos
      // la pantalla
      imgItem.setImage( imagen );
      display.setCurrent( intermedio);
  }
  public Screen Diseña_Conectando(){

      //Creamos el Form para la pantalla cuando se está conectando
      Conectando = new Form(null);

      //Creamos el texto desplazandose por la ventana
      infor = new Ticker("Conectando ...");

      //Se crea el icono de conexion
      Image conectando = null;
      try{
          conectando = Image.createImage("/Conectando.png");
```

```
      }catch(java.io.IOException e){
          System.err.println("Excepcion:" + e);
      }

      //Creamos el item que contiene a la imagen
      ImageItem ImgItem2 = new
ImageItem(null,conectando,ImageItem.LAYOUT_CENTER,
      "Imagen no disponible");

      Conectando.setTicker(infor);
      Conectando.append(ImgItem2);
      Conectando.setCommandListener(this);
      display.setCurrent(Conectando);
      return Conectando;


  }
  public Screen Diseña_Conectando2(){
      display.setCurrent(Conectando);
      return Conectando;
  }

}
```

7.4.4 Servidor kSOAP

Muy parecido en código a la versión JSR 172, pero al ser desplegado como *RPC* en lugar de *wrapped/literal* el WSDD y el WSDL cambian mucho.

7.4.4.1 *WKsoap.java*

El archivo WKsoap.java es casi idéntico a su equivalente en JSR 172, tan sólo cambia el nombre del archivo y del paquete.

```
package WKsoap;

import java.io.IOException;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;

public class WKsoap {
  public String[] query(String Act, String Pro, String Loc, String Cat,
  String Emp) throws Exception {

      //En primer lugar se carga el driver necesario para el acceso MySQL
    Class.forName("com.mysql.jdbc.Driver").newInstance();

      //Se abre una conexión con la base de datos
    Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/serial", "antonio", "");

      //Ahora creamos el statement y la petición SQL
    Statement stmt = con.createStatement();
    String sql ="SELECT DISTINCT * FROM serial WHERE Actividad='" + Act +
        "' and Provincia='" + Pro + "' and Localidad='" + Loc +
        "' and Categoria='" + Cat;

      //El campo Emp es opcional, se comprueba si se ha enviado y
      // si es así se añade
```

```
    if (Emp.length()!=0) {
        sql = sql + "' and Empresa='" + Emp;
    }

      //Se cierra la petición
    sql = sql + "';";

      //Se ejecuta la petición y se almacena el resultado en record
    ResultSet record = stmt.executeQuery(sql);


      //Para saber el tamaño de la tabla necesaria vamos a la última fila
     // y multiplicamos el resultado por el número de cadenas que tiene
     // cada elemento (ocho)
    record.last();
    int Total = 8*record.getRow();

      //Volvemos al estado inicial
    record.beforeFirst();

      //Para cada elemento va leyendo todos los campos y los va
     // almacenando en una tabla
    int cont=0;
    String devolver [] = new String [Total];
    while (record.next()) {
        devolver [cont++]=record.getString ("Actividad");
        devolver [cont++]=record.getString ("Provincia");
        devolver [cont++]=record.getString ("Localidad");
        devolver [cont++]=record.getString ("Categoria");
        devolver [cont++]=record.getString ("Empresa");

        devolver [cont++]=record.getString ("Telefono");
        devolver [cont++]=record.getString ("Direccion");
        devolver [cont++]=record.getString ("Referencia");

     }

      //Por último devolvemos la tabla obtenida con todos los resultados
    return devolver;

  }

}
```

7.4.4.2 *WKsoap.wsdl*

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://db.samples/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://db.samples/" xmlns:intf="http://db.samples/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->

 <wsdl:types>
  <schema targetNamespace="http://db.samples/"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="ArrayOf_soapenc_string">
    <complexContent>
     <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="soapenc:string[]"/>
     </restriction>
```

```
    </complexContent>
   </complexType>
  </schema>
 </wsdl:types>
   <wsdl:message name="queryRequest">
      <wsdl:part name="in0" type="soapenc:string"/>
      <wsdl:part name="in1" type="soapenc:string"/>
      <wsdl:part name="in2" type="soapenc:string"/>
      <wsdl:part name="in3" type="soapenc:string"/>
      <wsdl:part name="in4" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="queryResponse">
      <wsdl:part name="queryReturn" type="impl:ArrayOf_soapenc_string"/>
   </wsdl:message>
   <wsdl:portType name="WKsoap">
      <wsdl:operation name="query" parameterOrder="in0 in1 in2 in3 in4">
         <wsdl:input message="impl:queryRequest" name="queryRequest"/>
         <wsdl:output message="impl:queryResponse" name="queryResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="WKsoapSoapBinding" type="impl:WKsoap">
      <wsdlsoap:binding style="rpc"
                transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="query">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="queryRequest">
            <wsdlsoap:body
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="http://WKsoap" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="queryResponse">
            <wsdlsoap:body
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="http://db.samples/" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="WKsoapService">
      <wsdl:port binding="impl:WKsoapSoapBinding" name="WKsoap">
         <wsdlsoap:address
                   location="http://localhost:8080/axis/services/WKsoap"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

7.4.4.3 *Deploy.wsdd*

```
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
 <service name="WKsoap" provider="java:RPC">
      <parameter name="wsdlTargetNamespace" value="http://db.samples/"/>
      <parameter name="className" value="WKsoap.WKsoap"/>
      <parameter name="allowedMethods" value="query"/>
  </service>
</deployment>
```

7.4.4.4 *Undeploy.wsdd*

```
<undeployment xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="WKsoap"/>
</undeployment>
```

7.4.5 Atributo

La clase *Atributo* será un objeto en que se almacena cada elemento de la respuesta. Es el mismo para el modelo de JSR 172 y para kSOAP.

```
import java.io.*;

 public class Atributo {

     //Son todos los elementos que identifican a un objeto de tipo atributo
    private String actividad=null;
    private String nombre=null;
    private String provincia=null;
    private String direccion=null;
    private String localidad=null;
    private String telefono=null;

    // Categoria se va a usar para indicar la categoria de los hoteles

    private int categoria=0;

    //Nos va a permitir conseguir la referencia al mapa
    private int referencia=0;

    // Restantes servirá cuando estemos recibiendo los objetos en el flujo
    // para saber cuanto objetos restantes quedan en el flujo
    private int restantes=0;

    // Estos tres flags son necesarios para poder distinguir cuando un String
    // esta vacio de cuando lleva algo y asi cuando se serializa que no
    // de fallo

   public String getProvincia() {
        return (provincia);
    }

    public String getLocalidad(){
        return (localidad);
    }

    public String getActividad() {
        return (actividad);
    }

    public String getNombre() {
        return (nombre);
    }

    public String getDireccion(){
        return(direccion);
    }

    public String getTelefono() {
        return (telefono);
    }

    public int getCategoria(){
        return(categoria);
    }

    public int getRestantes(){
        return(restantes);
    }

    public int getReferencia(){
        return(referencia);
```

```
    }

    public void setCategoria(int categ){
        categoria = categ;
    }

    public void setReferencia(int refer){
        referencia=refer;
    }

    public void setRestantes(int s){
        restantes = s;
    }

    public void setTelefono(String telef) {
        telefono = telef;
    }

    public void setActividad(String s) {
        actividad = s;
    }

    public void setNombre(String s){
        nombre = s;
    }

    public void setProvincia(String s){
        provincia = s;
    }

    public void setLocalidad(String s){
        localidad = s;
    }

    public void setDireccion(String s){
        direccion = s;
    }
}
```