Ingeniería del Software II.

Curso 2003/2004.

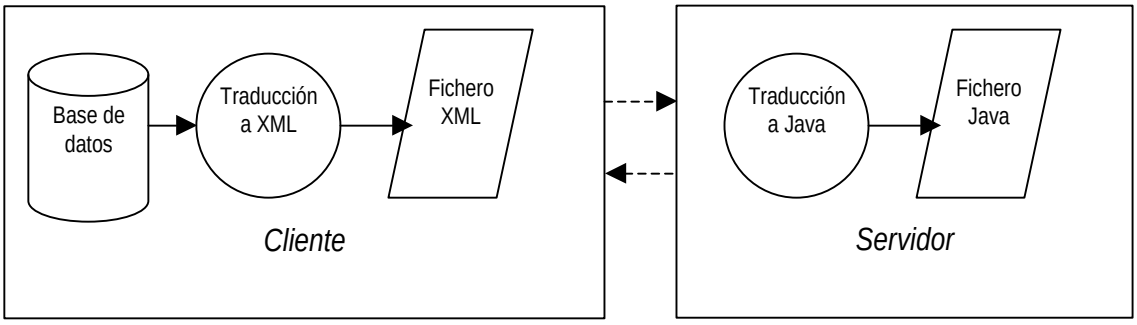
Enunciado de la práctica del Primer parcial.

Se desea desarrollar un sistema CASE, compuesto por dos o más aplicaciones, que permita obtener clases Java a partir de tablas de una base de datos.

La idea es la siguiente:

1) En una aplicación *cliente,* un programador elige una base de datos.
2) La aplicación cliente recupera cierta información sobre la estructura de tablas y de relaciones entre tablas de esa base de datos.
3) La aplicación cliente transforma dicha base de datos a un documento XML, que almacena la misma información que se recuperó en el paso 2.
4) Obtenido el fichero XML, la aplicación cliente lo envía a otra aplicación que actúa como *servidor*.
5) El servidor analiza el fichero XML y genera un conjunto de clases Java, que devuelve al cliente.

Gráficamente:



Ambas aplicaciones deberán ser desarrolladas siguiendo el Proceso Unificado de Desarrollo usando notación UML, e implementadas en lenguaje Java. Se utilizarán las herramientas Oracle JDeveloper como entorno integrado de desarrollo, y Rational Rose y/o Poseidon UML para las fases de análisis y diseño.

Todas las comunicaciones entre el cliente y el servidor se realizarán mediante *rmi* (tema 7 de los apuntes).

La práctica será desarrollada en grupos de 4 personas (excepcionalmente 3 o 5), debiendo 2 dedicarse al desarrollo del cliente y 2 al servidor, aunque tendrán que ponerse de acuerdo desde el principio para determinar las interfaces de comunicación entre ambas aplicaciones.

El último martes de cada mes (el lunes previo, si es festivo) se entregará al profesor un breve informe sobre el estado del proyecto. En el primer informe se especificará qué miembros del grupo van a realizar el cliente y quiénes el servidor, así como las interfaces de comunicación entre ambas aplicaciones y un plan de iteraciones.

Como documento final, se entregará toda la documentación del proyecto: plan de iteraciones, requisitos y casos de uso, contratos, diagramas de secuencia, diagramas de clases, código fuente, y otra información que se considere de interés y que podrá ser tenida en cuenta a la hora de puntuar (decisiones de interés que se hayan tomado, mediciones del código, pruebas a las que se ha sometido a las aplicaciones, etc.).

En los siguientes anexos se describen los formatos de los ficheros que deben manipular las aplicaciones.

Cualquier duda se atiende en horario de tutorías.

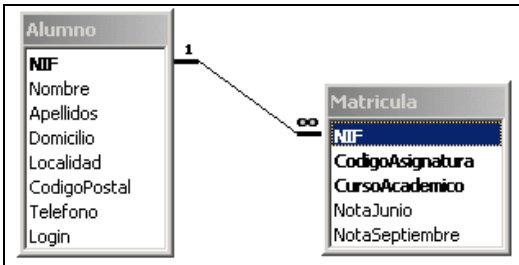
Macario Polo Usaola
Despacho 3.21
Teléfono 926.295300 ext. 3730
macario.polo@uclm.es

Anexo I. DTD de los ficheros XML.

```
<?xml version="1.0"?>
<!ELEMENT bd    (tabla*,relacion*)>
<!ELEMENT tabla (columna+)>
<!ATTLIST columna nombre CDATA #REQUIRED>
<!ATTLIST columna tipo CDATA #REQUIRED>
<!ATTLIST columna esPK CDATA #REQUIRED>
<!ELEMENT relacion  (columnasPrincipal, columnasSecundaria)>
<!ATTLIST relacion tablaPrincipal CDATA #REQUIRED>
<!ATTLIST relacion tablaSecundaria CDATA #REQUIRED>
<!ELEMENT columnasPrincipal (columnaPrincipal+)>
<!ELEMENT columnasSecundaria (columnaSecundaria+)>
<!ATTLIST columnaPrincipal nombre CDATA #REQUIRED>
<!ATTLIST columnaSecundaria nombre CDATA #REQUIRED>
```

Anexo II. Ejemplo.

A partir de la base de datos de la izquierda, el cliente obtendría el código XML que aparece a la derecha:




```
<?xml version='1.0' encoding='windows-1252'?>
<!DOCTYPE bd SYSTEM "bd.dtd">
<bd>
<tabla nombre='Alumno'>
        <columna nombre='NIF' tipo='texto' esPK='si'/>
        <columna nombre='Nombre' tipo='texto' esPK='no'/>
        <columna nombre='Apellidos' tipo='texto' esPK='no'/>
        <columna nombre='Domicilio' tipo='texto' esPK='no'/>
        <columna nombre='Localidad' tipo='texto' esPK='no'/>
        <columna nombre='CodigoPostal' tipo='texto' esPK='no'/>
        <columna nombre='Telefono' tipo='texto' esPK='no'/>
        <columna nombre='Login' tipo='texto' esPK='no'/>
</tabla>
<tabla nombre='Matricula'>
        <columna nombre='NIF' tipo='texto' esPK='si'/>
        <columna nombre='CodigoAsignatura' tipo='texto'
esPK='si'/>
        <columna nombre='CursoAcademico' tipo='texto' esPK='si'/>
        <columna nombre='NotaJunio' tipo='doble' esPK='no'/>
        <columna nombre='NotaSeptiembre' tipo='doble'
esPK='no'/>
</tabla>
<relacion tablaPrincipal='Alumno' tablaSecundaria='Matricula'>
<columnasPrincipal>
        <columnaPrincipal nombre='NIF'/>
</columnasPrincipal>
<columnasSecundaria>
        <columnaSecundaria nombre='NIF'/>
</columnasSecundaria>
</relacion>
</bd>
```

El servidor procesaría el fichero XML anterior y devolvería al cliente los ficheros *Alumno.java* y *Matricula.java*, que se corresponderían a clases persistentes con un constructor que da un valor por defecto a los campos, un constructor materializador, un método *insert* y un método *delete*. Como ejemplo, *Alumno.java* tendría el siguiente formato:

```
package dominio;

import java.sql.*;
```

```
public class Alumno
{
	protected String mPKNIF;
	protected String mNombre;
	protected String mApellidos;
	protected String mDomicilio;
	protected String mLocalidad;
	protected String mCodigoPostal;
	protected String mTelefono;
	protected String mLogin;
	Matricula[] mMatriculas;

	public Alumno()
	{
		mPKNIF=null;
		mNombre=null;
		mApellidos=null;
		mDomicilio=null;
		mLocalidad=null;
		mCodigoPostal=null;
		mTelefono=null;
		mLogin=null;
	}

	public Alumno(Connection bd, String nif) throws SQLException
	{
		String SQL="Select * from Alumno where NIF=?";
		PreparedStatement p=bd.prepareStatement(SQL);
		p.setString(1, nif);
		ResultSet r=p.executeQuery();
		if (r.next())
		{
			mPKNIF=r.getString(1);
			mNombre=r.getString(2);
			mApellidos=r.getString(3);
			mDomicilio=r.getString(4);
			mLocalidad=r.getString(5);
			mCodigoPostal=r.getString(6);
			mTelefono=r.getString(7);
			mLogin=r.getString(8);
		}
		p.close();
	}

	public void insert(Connection bd) throws SQLException
	{
		String SQL="Insert into Alumno values (?, ?, ?, ?, ?, ?, ?, ?)";
		PreparedStatement p=bd.prepareStatement(SQL);
		p.setString(1, this.mPKNIF);
		p.setString(2, this.mNombre);
		p.setString(3, this.mApellidos);
		p.setString(4, this.mDomicilio);
		p.setString(5, this.mLocalidad);
		p.setString(6, this.mCodigoPostal);
		p.setString(7, this.mTelefono);
		p.setString(8, this.mLogin);
		p.executeUpdate();
		p.close();
	}
```

```
	public void delete(Connection bd) throws SQLException
	{
		String SQL="Delete from Alumno where NIF=?";
		PreparedStatement p=bd.prepareStatement(SQL);
		p.setString(1, this.mPKNIF);
		p.executeUpdate();
		p.close();
	}
}
```