BENEMÉRITA UNIVERSIDAD

AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Lógica Móvil"

Tesis Profesional que para obtener el título de:
Licenciado en Ciencias de la Computación

Presenta:
Alberto Ramos Carbente

Asesor:
Dr. Fernando Zacarías Flores

Primavera 2010

INDICE

INTRODUCCIÓN

DLV es una de los sistemas basados en el paradigma ASP más ampliamente usado y exitoso a nivel internacional. Esta herramienta se basa en la semántica de modelos estables, y soporta un lenguaje poderoso que extiende la programación lógica disyuntiva con muchos constructores expresivos, incluyendo agregaciones, restricciones débiles y fuertes, funciones, listas y conjuntos.

Actualmente, DLV es empleado aplicaciones industriales, e incluso está listo para distribución comercial. Así lo demuestran diversos proyectos basados en esta herramienta tales como: “Potenziamento e Applicazioni della Programmazione Logica Disgiuntiva” and “Sistemi basati sulla logica per la rappresentazione di conoscenza: estensioni e tecniche di ottimizzazione.”

Además, los programas lógicos disyuntivos (DLV) son una poderosa herramienta en la representación de conocimiento y razonamiento de sentido común. Esta herramienta permite resolver problemas complejos. Sin embargo, los sistemas para programas lógicos disyuntivos carecen actualmente de una interface que permita a los investigadores del área poder contar con esta herramienta en todo momento y en cualquier lugar, lo que dificulta el quehacer científico y de investigación, así como el desarrollo de aplicaciones de manera permanente.

Así, en el presente trabajo pretendemos explotar el potencial que las tecnologías de la información y comunicaciones nos ofrecen actualmente. De manera particular pretendemos llevar el revolucionario paradigma de programación lógica disyuntiva a la mano de cualquier persona que desee involucrarse en el aprendizaje de la lógica de manera continua y permanente (Life-Long-Learning en inglés) a través de dispositivos móviles.

El desarrollo de aplicaciones móviles simples facilita el aprendizaje de cualquier área o tema, ya que pueden ser más atractivos si ofrecen la posibilidad de ser utilizados en diferentes dispositivos móviles, como un teléfono móvil, por ejemplo.

Se empleará como plataforma de desarrollo Netbeans y el kit de desarrollo Wireless Toolkit de Sun Java™. Además, se desarrollaran diversos programas de acuerdo a los publicados en diversos artículos internacionales, y así permitir a los usuarios la interacción e investigación en todo lugar, en cualquier momento y en casi cualquier dispositivo.

Se espera que este tipo de aplicaciones móviles lleguen a más aprendices, profesores y/o investigadores, aprovechando el auge que los teléfonos móviles han tenido en los últimos días, pues de entrada contaremos con 55 millones de usuarios potenciales (en México), es decir, el desarrollo de este tipo de aplicaciones móviles podrá estar disponible en la mano (teléfono móvil) de cualquier aprendiz interesado en su educación superior e investigación de primer nivel.

Existen diversas propuestas para llevar el conocimiento a todas las personas de nuestro país. El motivo principal, lograr que la gente aprenda, adquiera conocimiento y genere ideas a través de nuestra aplicación. Se han planteado propuestas en nuevos medios, tal es el caso del internet. Con esta posibilidad ha sido posible llevar este nuevo tipo de aplicaciones a diversos contextos. Así, nuestro propósito es el poder llevar nuestra aplicación a dispositivos móviles de forma tal que nos permitan el intercambio e interacción en los procesos de investigación.

Por otro lado, los dispositivos móviles ya forman parte de los accesorios básicos y necesarios en el trabajo, en el estudio, en la investigación actual, en el hogar, para la diversión, etc. Por lo que es importantísimo poder agregarles

un plus en su uso, convirtiéndolo en un medio estimulante para fomentar un proceso de aprendizaje e investigación permanente.

PLANTEAMIENTO DEL PROBLEMA

¿Cuántas personas que realizan investigación desean contar con las herramientas adecuadas en todo momento y en cualquier lugar?

Recordemos que la investigación y surgimiento de ideas no tiene hora ni lugar para presentarse, pero si que debemos estar preparados. Y la lógica no es la excepción.

Para responder a esta pregunta solo basta con conocer las estadísticas sobre el número de usuarios de telefonía celular o móvil. Hasta el año 2007, las estadísticas de usuarios de una sola compañía indicaban 55 millones de usuarios. Esto significa que el desarrollo de herramientas móviles tiene el potencial para llegar a todos estos millones de usuarios cautivos y muy probablemente, ávidos de conocimiento.

Por consiguiente, es de vital importancia el desarrollo de este trabajo de tesis, con el objetivo principal de contar con tecnología que permita explotar los dispositivos móviles que de antemano el usuario ya cuenta con ellos y, no será necesario que adquieran este tipo de dispositivos.

JUSTIFICACIÓN

Uno de los problemas actuales en nuestro país es la falta de entornos adecuados para poder realizar una investigación de primer nivel, tal como se realiza en países de primer mundo y a todos los niveles: básica, media, superior y de posgrado. El proyecto propuesto en la presente tesis tiene como objetivo, brindar la oportunidad a todos los usuarios de telefonía móvil el beneficio de traer en su teléfono un paradigma de programación lógica que les permitan experimentar en todo momento y en cualquier lugar. La idea general es la siguiente: Demostrar la viabilidad de la lógica móvil, es decir, hacer parte de su vida cotidiana la disponibilidad del paradigma de programación lógico a través de DLV móvil, de tal forma que puedan estudiar e investigar en todo momento y en cualquier lugar. Para esto, nuestra aplicación constará de una aplicación que residirá en el dispositivo móvil y otra en el servidor a la que podrá accederse a través de internet vía el protocolo WAP y el servicio GPRS. De esta manera, cualquier usuario de telefonía móvil podrá llevar DLV (paradigma de programación lógica) en su teléfono con los beneficios que este tipo de tecnologías brinda.

OBJETIVOS

Objetivo General

- Demostrar la viabilidad de llevar el paradigma de programación lógica a dispositivos móviles.

Objetivos Específicos

- Implementar una aplicación móvil (MIDP) para realizar las tareas de captura de programas lógicos (ASP).
- Construir una aplicación web (SERVLET) la cual dará respuesta a las solicitudes de ejecución de programas lógicos DLV. Servirá como intermediario en la comunicación entre la Aplicación Móvil y la aplicación WEB que ejecutara DLV.
- Implementar la comunicación entre las peticiones de la Aplicación Móvil y la aplicación Web vía WAP-GPRS.

Capitulo 1

ANSWER SET PROGRAMMING

Answer set programming (ASP – Stable model semantics o A-Prolog) definida en 1988 por Michael Gelfond y Vladimir Lifschitz en [1], es un paradigma lógico que en los últimos años ha tenido un gran desarrollo [9, 10, 11, 12, 13, 14]. Además, se han obtenido resultados relevantes que establecen relaciones directas entre ASP y la lógica intuicionista y otras lógicas, lo que permite hacer uso de toda la maquinaria desarrollada en Intuicionismo. Esto ha dado a este paradigma un gran impulso entre los miembros de la comunidad científica. También, han surgido diversas aplicaciones en el campo de la Inteligencia Artificial, razonamiento no-monótono, etc. Aquí, presentamos un panorama general sobre los conceptos básicos utilizados en ASP, con la finalidad de establecer el marco teórico empleado en nuestro trabajo.

En ASP, una teoría está compuesta por tres componentes fundamentales que son:

- El conjunto de axiomas lógicos

Estos corresponden a los axiomas que conforman la lógica que elijamos, como puede ser clásica, intuicionismo, modal, etc.

- El conjunto de axiomas propios

En nuestro contexto lo conforman el conjunto de cláusulas comprendidas en un programa lógico, estas deben ser definite o positivas.

- Las reglas de inferencia

Comúnmente empleadas para su aplicación en la obtención de nuevo conocimiento, en nuestro caso: la regla de “modus ponens”.

Más adelante, definimos de manera formal estos tres ingredientes que son la parte fundamental del conocimiento básico necesario para entender este trabajo. Enseguida, damos algunos conceptos y definiciones que serán utilizados a lo largo de este trabajo.

1.1 Lógica proposicional

El lenguaje de la lógica proposicional tiene un alfabeto que consiste de:

- Símbolos Proposicionales p0, p1, ...
- Conectivos $\wedge$, $\vee$, $\rightarrow$, $\bot$
- Símbolos auxiliares “(“, “)”, “,” , “.”

donde $\wedge$, $\vee$, $\rightarrow$, son conectivos binarios y $\bot$ es un conectivo sin argumentos. También, el conectivo $\bot$ denota falso. Los símbolos proposicionales son también llamados átomos o proposiciones atómicas.

Una fórmula T (también llamada Top) es introducida como una abreviación de $\bot \leftarrow \bot$, así, tenemos que podemos definir Top como sigue:

$$T: \bot \leftarrow \bot$$

Representaremos la negación clásica o negación de default con $\neg$ (también conocida con la palabra reservada “not”). Consideramos que un programa es un conjunto de cláusulas donde éstas son escritas tomando en consideración la siguiente convención:

$$B \rightarrow A.$$

Que por cuestiones de eficiencia, en programación lógica la denotaremos como sigue:

$$A \leftarrow B.$$

De manera general denotamos una cláusula de la siguiente manera:

$$H \leftarrow B$$

Refiriéndose a H como la cabeza (Head) y a B como el cuerpo (Body) de la regla o cláusula.

1.2 Tipos de cláusulas

Enseguida, definimos los diferentes tipos de cláusulas que se pueden encontrar en la literatura [11].

- Si H = A y $B=\overset{m}{\underset{i=1}{\wedge}} Bi$, Bi letra proposicional entonces H ← B se le denomina cláusula positiva.

- Si H =⊥ y $B=\overset{m}{\underset{i=1}{\wedge}} Bi$, Bi letra proposicional entonces H ← B se le denomina cláusula Horn.

- Si H ← B donde H = A, B = $\overset{m}{\underset{i=1}{\wedge}} Bi$, $\overset{n}{\underset{j=m+1}{\wedge}} \neg Bj$ es decir, A ← B1, B2, ..., Bm, $\neg B_{m+1}$,..., $\neg B_{m+j}$ le denominamos cláusula normal.

- Si tenemos H ← B donde H = $\overset{m}{\underset{k=1}{\vee}}$, B = $\overset{n}{\underset{j=m+1}{\wedge}} Bj$, $\overset{l}{\underset{i=n+1}{\wedge}} \neg Bl$ es decir, A1, ..., Ak ← B1, B2, ..., Bm, ¬Bm+1, ..., ¬Bj le llamamos cláusula disyuntiva.

- Si tenemos H ← B donde H = $\overset{m}{\underset{k=1}{\vee}} Ak$, $\overset{n}{\underset{i=m+1}{\vee}} Ai$, B = $\overset{t}{\underset{j=n+1}{\wedge}} Bi$, $\overset{s}{\underset{l=t+1}{\wedge}} \neg Bj$, i. e. A1 v ... v Am v ¬Am+1 v ... v ¬An ← B1, B2, ..., Bm, ¬Bm+1, ..., ¬Bj le denominamos cláusula libre.

- Si H = B, H y B son anidaciones de: ∨, ∧ y ¬, es una cláusula aumentada.

- Cada uno de estos tipos de cláusulas definen una clase de programa [11], por ejemplo:

  $a \vee b \leftarrow e \wedge d \wedge \neg c$ Disyuntiva, Libre

  $\bot \leftarrow p \vee q$. Restricción

Existen algunos tipos de cláusulas especiales tales como las siguientes [11]:

Si B1, ..., Bm, $\neg$Bm+1, ..., $\neg$Bm+n es T entonces identificamos la regla con A y le llamamos hecho.

Por otro lado, si A es $\bot$ entonces decimos que la regla es una restricción.

Definición.- Una literal L es un átomo A o un átomo fuertemente negado $\sim$A.

Así, un programa P es un conjunto finito de reglas normales, donde A y cada Bi son literales.

A es llamada la cabeza de un regla r, denotada por H(r), y a B1, ..., Bm, $\neg$Bm+1, ..., $\neg$Bm+n le llamamos el cuerpo de la regla r y lo denotamos por B(r).

1.3 Teoría en ASP

Dentro del paradigma ASP consideramos que una teoría se compone de tres elementos fundamentalmente [11] que son:

1. Conjunto de axiomas lógicos, que normalmente son definidos de manera concreta para cada sistema axiomático existente.

2. El conjunto de axiomas propios, que en nuestro caso lo constituye el programa lógico modelado y definido en DLV[1].
3. EL tercer y último componente lo constituye la regla de inferencia llamada “modus ponens”.

Axiomas lógicos.- Existen diferentes sistemas axiomáticos.

Axiomas propios.- Pueden ser las cláusulas de un programa.

Regla de inferencia.- Modus Ponens.

1) $A \rightarrow B$

2) A

3) B

Esta regla significa que si tenemos como hipótesis A y $A \rightarrow B$ entonces podemos concluir B.

Veamos un ejemplo de lo que sería una teoría en nuestro marco de trabajo.

Ejemplo 1.1 Supongamos que P es un programa que solo contiene átomos positivos en el cuerpo, A y B letras proposicionales, entonces:

$P = \{ \ A_1 \leftarrow B_{11}, ..., B_{1n}.$

.

.

.

$A_k \leftarrow B_{k1}, ..., B_{kn}. \}$

[1] http://www.dbai.tuwien.ac.at/proj/dlv

Consideremos a la lógica clásica como nuestro contexto teórico, por tanto nuestros axiomas lógicos son:

Axioma A1) $A \rightarrow (B \rightarrow C)$

Axioma A2) $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

Axioma A3) $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$

Y como axiomas propios las cláusulas de nuestro programa P:

$A_1 \leftarrow B_{11}, ..., B_{1n}.$

.

.

.

$A_k \leftarrow B_{k1}, ..., B_{kn}.$

Como única regla de inferencia el Modus Ponens.

Así, estos tres elemntos conforman nuestra teoría.

1.4 Definiciones básicas

Definición 1.1 [11]. FORM (conjunto de formulas bien formadas):

a) A es una fórmula bien formada:

b) Si A y B son fórmulas bien formadas, así también lo son:

$A \vee B$, $A \wedge B$, $A \leftarrow B$, $A \leftrightarrow B$, $A \rightarrow \perp$

Definición 1.2 [11]. Dado σ un conjunto de cláusulas decimos que existe una deducción de A supesto σ, notación $\sigma \vdash A$ si existe $B_1, B_2, ..., B_n$ fórmulas bien formadas tales que para cada $i \in \{1, ..., n\}$, B_i es un axioma (A1, A2, A3) o $B_i \in \sigma$ o existe B_k

$B_k := B_j \rightarrow B_i$

B_j

entonces B_i.

donde $k, j < i$.

Definición 1.3 [11]. Toda letra proposicional, $\perp$ y T, ($T := \perp \rightarrow \perp$) son denominadas fórmulas elementales. Cualquier fórmula construida con $\wedge$ y $\vee$ sobre fórmulas elementales es denominada fórmula básica.

Definición 1.4 [11]. Un conjunto de letras proposicionales X satisface una fórmula básica F ($X \models F$) si, para F fórmula elemental,

$X \models F$ si $F \in X$ o $F = T$.

$X \models F \vee G$ si $X \models F$ y $X \models G$

$X \models F \wedge G$ si $X \models F$ o $X \models G$

Definición 1.5 [11]. Dado P un programa básico constituido por fórmulas básicas. Un conjunto de letras proposicionales X se dice cerrado bajo P si para toda cláusula $H \leftarrow B \in P$ tenemos que $X \models H$ siempre y cuando $X \models B$ esto es, si $X \models B$ entonces $X \models H$.

Definición 1.6 [11]. Sea X un conjunto de letras proposicionales y P un progrma constituido por fórmulas básicas, X es denominado un Answer Set de P si el conjunto X es mínimo entre los conjuntos de átomos cerrados bajo P.

Definición 1.7 [11] **(Answer Set)** La reducción de un programa normal (disyuntivo, libre, aumentado) relativo al conjunto de letras proposicionales X, es definido de la siguiente manera:

Para una formula elemental F:

a) $F^x = F$

b) $F \wedge G) = F^x \wedge G^x$

c) $(F \vee G)^x = F^x \vee G^x$

d) $(\neg F)^x = \bot$ si $X \not\models F$; T en otro caso.

e) $P^x = (H \leftarrow B)^x \mid H \leftarrow B \in P$

Definición 1.8 [11]. Sea P un programa y X un conjunto de átomos. X es un answer set de P si este es un answer set de la reducción P^x.

Sea P un progama, $M \subseteq L_P$, M answer set de P y L_P un conjunto de átomos:

$M \cup \neg\{L_P - M\}$ es el modelo de P [11].

Definición 1.9 [11]. Dado un conjunto de literales A y P un programa. Denotamos $\neg A = \{\neg a \mid a \in A\}$. También definimos $\overline{A} = A \cup \neg\{Lit_P - M\}$.

Definición 1.10 [11]. Decimos que dos reglas r1 y r2 están en conflicto (denotado por r1 ∇ ~r2) sí y sólo si ambas tienen la misma cabeza pero una es la contrapuesta de la otra, es decir, si H(r1) = ~H(r2).

Definición 1.11 [Eiter] (Principio de rechazo causal). Sean P1 y P2 programas y sea S answer set de la actualización de P1 y P2. Definimos y denotamos al conjunto de reglas a rechazar como:

Rej(S, (P1, P2)) = {r$\in$ P1 | $\exists$ r'$\in$P2, tal que r ∇ r' y S $\models$ B(r) $\cup$ B(r')} [11]

1.5 Ejemplos del cálculo de answer sets

En esta sección, damos algunos ejemplos que muestran como podemos hallar los answer sets de un programa. Para eso aplicamos la reducción anterior a las reglas de un programa para calcular sus answer sets del programa transformado y así hallar los answer sets del programa original.

Ejemplo 1.2 Consideremos el ejemplo siguiente.

Sea P:

$a \leftarrow \neg\neg a.$

$\neg b \leftarrow c \vee b.$

Iniciamos proponiendo que nuestro answer set sea X = $\varnothing$ dado que es el conjunto mínimo, que es lo que buscamos entonces, aplicando la reducción a nuestro programa P para verificar si lo satisface.

P^x:

$a^x \leftarrow \neg\neg a^x$

$\neg b^x \leftarrow (c \vee b)^x$

P^x:

$a \leftarrow T$

$T \leftarrow c \vee b$

Así, $X = \varnothing$ no satisface a P^x por lo tanto X no es un answer set de P

Ahora, si suponemos $X = \{a\}$

P^x:

$a^x \leftarrow \neg\neg a^x$

$\neg b^x \leftarrow (c \vee b)$ *

P^x:

$a^x \leftarrow T$

$T \leftarrow c \vee b$

por lo tanto X es un answer set de P^x, entonces tambien lo es de P.

Ejemplo 1.3 Ahora, consideremos el siguiente ejemplo.

P: $a \leftarrow \neg a$

Sea X = ∅

P^x: $a^x \leftarrow (\neg a)^x$

P^x: $a \leftarrow T$

Por lo tanto X no es un answer set de P.

Ejemplo 1.4 Consideremos el siguiente ejemplo.

Sea P:

$a \leftarrow \neg b$

$b \leftarrow \neg a$

Sea X = ∅

P^x:

$a \leftarrow T$

$b \leftarrow T$

Por lo tanto X = ∅ no es un answer set de P

Sea X = {a}

P^x: $a \leftarrow T$

$b \leftarrow \perp$

por lo tanto X = {a} es un answer set de P^x y por tanto es un answer set de P

Sea X = {b}

P^x:

$a \leftarrow \bot$

$b \leftarrow T$

por lo tanto X = {b} también es un answer set de P. En este caso, P tiene dos answer sets, esto significa que hay dos explicaciones posibles para P.

Ejemplo 1.5 Consideremos el siguiente ejemplo.

Sea P:

$a \leftarrow \neg b$

$b \leftarrow \neg a$

$p \leftarrow \neg p$

$p \leftarrow \neg a$

Sea X = $\varnothing$

P^x:

$a \leftarrow T$

$b \leftarrow T$

$b \leftarrow T$

$b \leftarrow a$

por lo tanto X = ∅ no es un answer set de P.

Sea X = {a}

P^x:

a ← T

b ←⊥

b ←T

b ← a

por lo tanto X = {a} no es un answer set de P.

Sea X = {b}

P^x:

a ← ⊥

b ← T

b ← T

b ← a

por lo tanto X = {b} no es un answer set de P

Sea X = {p}

P^x:

$a \leftarrow T$

$b \leftarrow T$

$b \leftarrow \bot$

$b \leftarrow a$

por lo tanto X = {p} no es un answer set de P

Sea X = {p, a}

P^x:

$a \leftarrow T$

$b \leftarrow \bot$

$b \leftarrow \bot$

$b \leftarrow a$

por lo tanto X = {p, a} si es un answer set de P

Sea X = {p, b}

P^x:

$a \leftarrow \bot$

$b \leftarrow T$

$b \leftarrow \bot$

$b \leftarrow a$

por lo tanto X = {p, b} no es un answer set de P

Ejemplo 1.6 Consideremos el siguiente ejemplo.

Sea P:

$\neg b \leftarrow \neg a$

Sea X = $\varnothing$

P^x:

$T \leftarrow T$

por lo tanto X = $\varnothing$ es un modelo de P.

1.6 Intuicionismo

La lógica intuicionista tiene características excelentes para el desarrollo de la teoría de answer set programming. De la misma forma es la base para desarrollar teorías que son usadas en la caracterización de programas lógicos y transformaciones de estas [11].

Intuicionismo esta basado en en concepto de prueba, más que en verdad como en el caso de lógica clásica para explicar el significado y uso de conectivos lógicos. Enseguida damos la definición formal de Intuicionismo [15].

Definición 2.12 La teoría formal del cálculo proposicional intuicionista es: Símbolos lógicos $\leftarrow \vee$, $\wedge$ y la negación clásica “not” (denotada también como $\neg$).

Modus ponens como la única regla de inferencia y el siguiente esquema de axiomas [15]:

1. $A \rightarrow (B \rightarrow A)$
2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
3. $A \wedge B \rightarrow A$
4. $A \wedge B \rightarrow B$
5. $A \rightarrow (B \rightarrow (A \wedge B))$
6. $A \rightarrow (A \vee B)$
7. $B \rightarrow (A \vee B)$
8. $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow ((A \vee B) \rightarrow C))$
9. $(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$
10. $\neg A \rightarrow (A \rightarrow B)$

David Pearce [Pearce] demostró que M un conjunto de átomos, M es un answer set de P si y solo si $P \cup \neg \overline{\overline{M}} \vdash M$.

Nosotros hemos seleccionado answer set programming en esta tesis debido a que esta es la realización de mucho trabajo teórico sobre razonamiento no monótono y aplicaciones de inteligencia artificial de programación lógica. Además, existen contribuciones recientes y relevantes que establecen relaciones estrechas entre answer set programming y lógica intuicionista. Esto nos permite que podamos utilizar toda la maquinaria existente de intuicionismo en ASP.

Capítulo 2

NETBEANS EN EL DESARROLLO DE APLICACIONES

Para conseguir trabajar en aplicaciones móviles (J2ME) una opción es la instalación de herramientas (Toolkits) Java [16, 17]. Sin embargo, si se recurre a ese tipo de instalaciones, se está bastante limitado para desarrollos posteriores y además no te darán una visión global del lenguaje Java. Por ello, en el presente trabajo de tesis instalamos herramientas profesionales, algunas bastante complejas, pero tocando solamente los aspectos J2ME para facilitar su uso. Se necesita, para ello, la instalación de los siguientes componentes: El Java Standard Edition (J2SE), en su versión Java Development Kit (JDK), en su versión 4 o superior. Lo puedes descargar desde la web de Sun: www.sun.com no confundir con el J2SE (Java Standard Edition), en su versión Java Runtime Environment (JRE), ya que este no nos serviría, pues es solo el entorno de ejecución, y nosotros lo que necesitamos es el entorno de desarrollo). También, empleamos un IDE completo, gratuito cuyo nombre es NetBeans (ver figura 2.1).

2.1 NetBeans

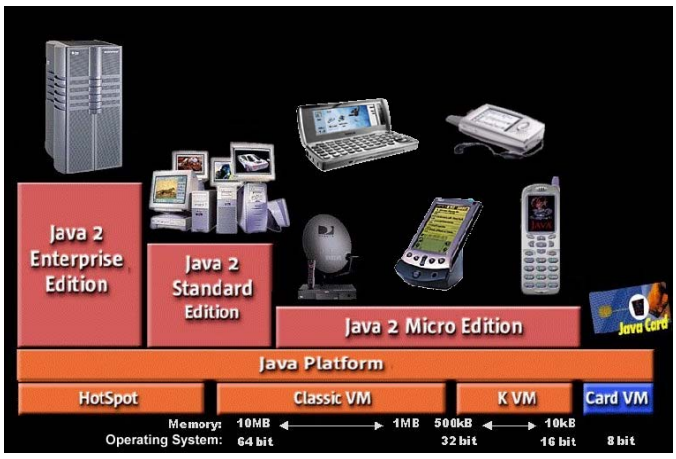

Figura 2.1 Relaciones entre ediciones de Java

NetBeans es Open Source Integrated Development Environment, escrito en Java, usado como plataforma para desarrollar programas en Java. Es decir, con NetBeans no solo podremos desarrollar para J2ME, sino también

programas complejos para diferentes entornos y sistemas operativos (Windows, Linux, Mac…) y aplicaciones cliente servidor. La instalación de todos estos programas es sobre el sistema operativo Windows XP presentando un ambiente completo, tal como se muestra en la figura 2.2.

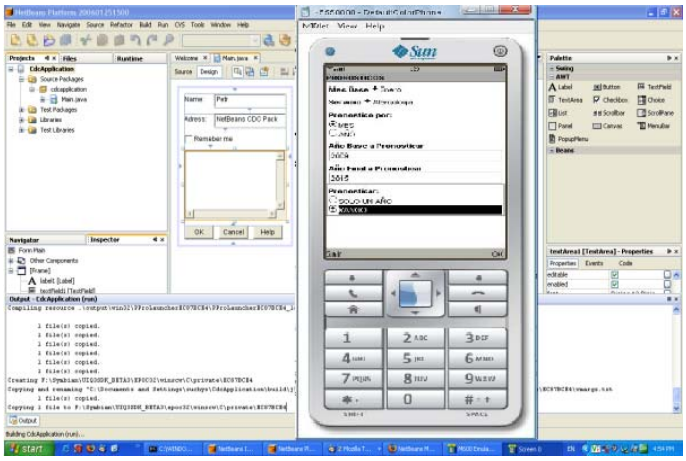

Figura 2.2 Ambiente de desarrollo Netbeans

NetBeans es Open Source Integrated Development Environment, escrito en Java, usado como plataforma para desarrollar programas en Java. Es decir, con NetBeans no solo podremos desarrollar para J2ME, sino también programas complejos para diferentes entornos y sistemas operativos (Windows, Linux, Mac…) y aplicaciones cliente servidor. La instalación de todos estos programas es sobre el sistema operativo Windows XP presentando un ambiente completo, tal como se muestra en la figura 2.1.

Con NetBeans podremos:

- Crear un proyecto
- Ejecutar un proyecto
- Depurar un proyecto.

2.2 NetBenas y modulo para movilidad

Con NetBeans Mobility Pack [18] se tiene la capacidad para desarrollar aplicaciones para dispositivos móviles, con características de optimización para el Mobile Information Device Profile (MIDP), y para el Connected Limited Device Configuration (CLDC). Además, integra en NetBeans características de Visual Mobile Designer (diseño visual móvil) (ver figura 2.3); Wireles

Connection Wizards: para acceder fácilmente a servicios vía servlets, con soporte para el estándar JSR-172; Fragmentación de Dispositivos: de modo que se puedan desarrollar aplicaciones para ejecutarlas en un dispositivo en concreto; Integración con el J2ME Wireles Toolkit: la herramienta de certificación y desarrollo J2ME, que integra emulación OTA (Over-The-Air), mensaje inalámbrico y APIs multimedia, y WMA emulación para mensajes SMS y CBS. Además, integra un Ofuscador (criptógrafo de código), y OTA Download Testing, que emula dicha capacidad sobre un móvil real.

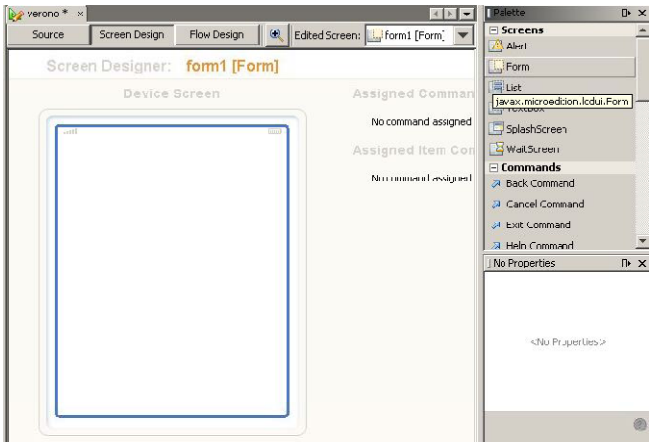

Figura 2.3 Interfaz para diseño visual móvil

Todas estas cualidades hacen que el Mobility Pack sea una herramienta potente y completa para desarrollar aplicaciones en J2ME.

2.3 Tecnología móvil

El uso de la tecnología móvil inalámbrica tal como PDA's, telefonía móvil, ipods o computadoras Notebook (ver figura 2.4) tanto en nuestros medios de trabajo como en nuestra vida diaria [19, 20, 21] está haciendo nuestras tareas cotidianas más flexibles y lo que es más importante, están pudiendo ser empleadas en donde sea y en cualquier momento. El desarrollo de aplicaciones móviles es novedoso ya que este facilita la entrega de información a la persona correcta, en el momento oportuno y en el lugar correcto usando dispositivos electrónicos portables. En un futuro cercano, las aplicaciones móviles vendrán a ser consideradas una parte normal de nuestra vida cotidiana. En este proyecto se desarrollan y evalúan metodologías innovadoras

para la construcción de sistemas basados en dispositivos móviles apoyados por aplicaciones en el Web.

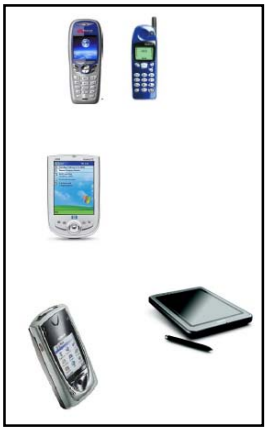

Figura 2.4 Dispositivos móviles

La investigación muestra que los dispositivos móviles pueden ser más fácilmente integrados a la currícula que los dispositivos de escritorio. Esto es posible dado que muchos de los estudiantes cuentan actualmente con dispositivos móviles más que con computadoras de escritorio. Además, los dispositivos móviles no necesitan infraestructura como los dispositivos de escritorio. Otro punto a favor de los dispositivos móviles es que las investigaciones y estudios muestran ganancia en la motivación del estudiante, habilidades analíticas y organizacionales, y algo muy importante, la comprensión; habilidades de escritura y gramática y originalidad. La movilidad permitida por los dispositivos móviles puede crear un sentimiento mayor de propiedad de trabajo para los estudiantes. Es claro, que es necesario un tiempo de adaptación y familiaridad adecuada con los nuevos dispositivos móviles y aquí es donde el staff técnico debe estar disponible para apoyar.

2.4 Implicaciones de la tecnología móvil

Es importante notar como el desarrollo tecnológico evoluciona de manera vertiginosa logrando cambiar el comportamiento de las sociedades de manera radical. Apenas hace unos años, los usuarios de dispositivos móviles no eran más que unos simples consumidores, es decir, su actividad se centraba en el entretenimiento y aplicaciones multimedia, sin embargo, estos han revolucionado y han pasado de ser unos simples consumidores a ser en la

actualidad “prosumidores”, lo que significa que ahora, los usuarios de dispositivos móviles no solo ven video clips, y escuchan música, si no que ahora también socializan, comprando, buscando, creando contenidos, etc.

Actualmente, el término prosumidor (tomado de Wikipedia Foundation Inc. http://es.wikipedia.org/wiki/Prosumidor), se aplica en aquellos usuarios que fungen como canales de comunicación humanos, lo que significa que al mismo tiempo de ser consumidores, son a su vez productores de contenidos. Un “prosumer” no tiene fines lucrativos, sólo participa en un medio digital de intercambio de información, tal es el caso del P2P, redes pares intercambiables. La palabra prosumer describe perfectamente a millones de participantes en la revolución del Web 2.0, ya que son cada vez más las personas involucradas que suben información a la red y a su vez son consumidores de la misma, creando así un abanico de información en todos los sentidos.

Uno debería pensar que el concepto de integrar estándares, protocolos y mejores prácticas debería dominar todos y cada uno de los aspectos de un entorno integrado de desarrollo (IDE), pero los desarrolladores de aplicaciones a menudo descubren piezas críticas ausentes en sus presuntamente extensos conjuntos de herramientas. En la figura 2.5 podemos observar una aplicación desarrollada en la herramienta móvil de Netbeans. Como se puede apreciar, la calidad y desempeño de la aplicación son bastante aceptables que cualquier usuario se sentirá a gusto con la aplicación móvil.

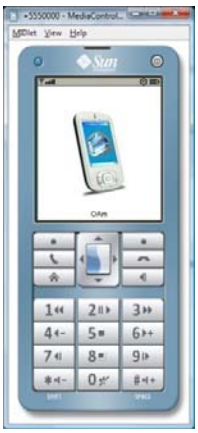

Figura 2.5 Aplicación móvil generada con NetBeans

2.5 J2ME y NetBeans

Sólo de forma muy simplista puede considerarse que J2ME y J2EE son versiones reducida y ampliada respectivamente, de J2SE: en realidad, cada una de las ediciones enfoca ámbitos de aplicación radicalmente distintos e independientes entre sí. Después de todos, las necesidades computacionales y APIs de programación requeridas por, por ejemplo, un juego gráfico ejecutándose sobre una PDA no tienen mucho que ver con los de un servidor distribuido de aplicaciones basado en EJB.

En lo que sigue presentamos con más detalle los componentes que integran la plataforma J2ME, la taxonomía propuesta por Sun de entornos de ejecución basados en las capacidades gráficas y computacionales de los dispositivos y la funcionalidad para la que están diseñados, la integración en J2ME de otras tecnologías Java relacionadas y la relevancia de la iniciativa J2ME para el desarrollo de aplicaciones y servicios embebidos en dispositivos móviles 2,5 y 3G [17].

2.5.1 Componentes de J2ME

Al contrario que en otras tecnologías orientadas a PCs y ordenadores convencionales, en J2ME el espectro de dispositivos considerados varía enormemente en cuanto a capacidad computacional, memoria y capacidades gráficas. Ante la imposibilidad de establecer una arquitectura común que se adecue a esta variedad de entornos hardware, J2ME define una serie de componentes (building blocks) a partir de los cuales se construye una implantación concreta para un dispositivo determinado. Estos componentes se agrupan en los siguientes tipos [22, 23]:

- Máquina virtual
- Configuración
- Perfil
- Paquetes opcionales

Un entorno de ejecución determinado de J2ME se compone entonces de una selección de máquina virtual, configuración y perfil, y posiblemente otros paquetes opcionales. En la figura 2.6 se muestran las relaciones entre los distintos componentes de un entorno de ejecución de J2ME.

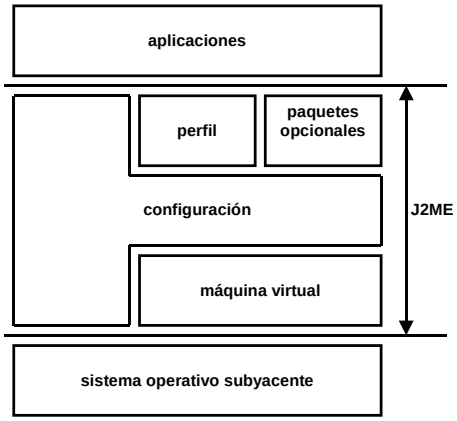

Figura 2.6 Relaciones entre componentes

2.5.2 Máquina virtual

Una máquina virtual de Java (JVM, *Java Virtual Machine*) es un programa encargado de traducir código intermedio (*bytecode*) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java[22]. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente.

Las implementaciones tradicionales de JVMs son en general muy pesadas en cuanto a memoria ocupada (*footprint*) y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos, que en algunos casos se apartan de la especificación estándar relajando algunas exigencias con el fin de obtener implementaciones más ligeras.

2.5.3 Configuración

Una configuración es un conjunto de APIs básicas de Java que definen un entorno generalizado de ejecución [22]. Una configuración no está orientada a ningún campo específico de aplicación, sino que más bien trata de agrupar los dispositivos en cuanto a sus capacidades computacionales y entorno genérico de operación (por ejemplo, si tienen conectividad o no). En particular, las librerías de interfaz gráfica nunca pertenecen a la definición de una configuración J2ME.

La configuración define la plataforma mínima necesaria para un grupo de dispositivos que tienen similar memoria y capacidades de procesamiento. Se compone de una máquina virtual, unas características del lenguaje Java y un conjunto mínimo de clases que soporta ese grupo de dispositivos.

La configuración tomada para la realización de nuestro proyecto es:

- Configuración del dispositivo: CLDC-1.0

2.5.4 Perfil

Un perfil es un conjunto de APIs orientadas a un ámbito de aplicación determinado, para una configuración J2ME dada. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proveen (por ejemplo, teléfonos móviles, electrodomésticos) y el tipo de aplicaciones que típicamente correrán sobre ellos. La librería de interfaz gráfica es generalmente una componente muy importante de la definición de un perfil [22]. Las interfaces gráficas difieren mucho entre distintos perfiles (displays textuales en teléfonos móviles, interfaces basadas en pen para PDAs, etc.), y en algunos perfiles ni siquiera existen (sistemas embebidos).

Un perfil siempre se define a partir de una configuración dada. Así, cabe pensar en un perfil como un conjunto de APIs que dotan a una configuración J2ME de funcionalidad específica para una familia de dispositivos concreta y para un tipo de aplicaciones determinado.

Con objeto de ofrecer un completo entorno de ejecución específico para cada categoría de dispositivo, las configuraciones se deben combinar con un conjunto de APIs de alto nivel (perfiles), que definen el modelo de ciclo de vida de la aplicación, el interfaz de usuario y el acceso a las propiedades específicas del dispositivo. Algunos de los perfiles existentes son (varios todavía en la etapa de definición):

- **Mobile Information Device Profile (MIDP).** Diseñado para teléfonos móviles y PDAs, incluye la interfaz de usuario, conectividad de red, almacenamiento local de datos y gestión de aplicaciones. Combinado con

CLDC, MIDP aporta un entorno de ejecución para Java que minimiza los consumos de memoria y de procesador.

- **Foundation Profile (FP).** Este perfil es el de nivel más bajo asociado a CDC, ya que los perfiles CDC pueden ser vistos como capas que se añaden a para proveer funcionalidades a los diferentes dispositivos. FP suministra una implementación de CDC con capacidades de acceso a red que se utiliza para aplicaciones embebidas en alto grado y sin interfaz de usuario.
- **Personal Profile (PP)** es un perfil diseñado expresamente para dispositivos con un interfaz gráfico completo y con posibilidad de ejecución de applets. Incluye además las bibliotecas del Abstract Windows Toolkit (AWT). Añade, por tanto, un interfaz de usuario básico a FP.
- **Personal Basis Profile (PBP)** es un subconjunto de PP que suministra un entorno para dispositivos que se puedan conectar a una red y que dispongan de un interfaz un poco más desarrollado que aquellos que posean los dispositivos donde PP va dirigido.
- **PDA Profile (PDAP)** es similar al MIDP pero diseñado para PDAs que tengan mejores pantallas y más memoria de los teléfonos móviles.
- **Game Profile (GP)** ofrece la plataforma para escribir juegos en dispositivos CDC.

Actualmente el perfil más utilizado es MIDP, que será el utilizado para el desarrollo de este proyecto, y en concreto usaremos el Perfil del Dispositivo MIDP-2.0.

Las aplicaciones desarrolladas para un determinado perfil serán portables a cualquier dispositivo que soporte ese perfil. Cabe destacar también que un mismo dispositivo puede soportar varios perfiles y que sobre una configuración también pueden residir diversos perfiles.

2.6 Arquitectura J2ME

Como se ha explicado anteriormente, el programa de estandarización ha definido diversos entornos de ejecución de J2ME identificados por una selección particular de máquina virtual, configuración y perfil J2ME. Es de suponer que esta arquitectura evolucionará en dos sentidos: refinamiento de

los entornos de ejecución definidos (nuevas versiones de los componentes) y definición de nuevos entornos de ejecución.

En el siguiente gráfico podemos ver un esquema modular con las diferentes configuraciones, perfiles y máquinas virtuales:

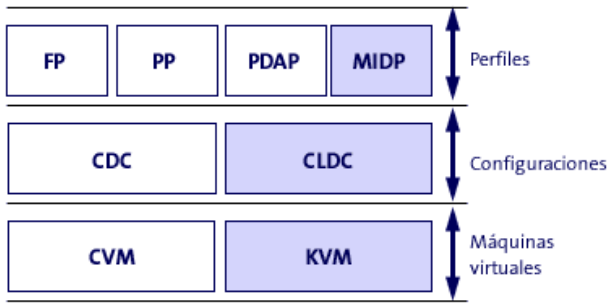

Figura 2.7 Arquitectura J2ME

En la figura 2.7 se muestra el estado actual de la arquitectura J2ME (figura 2.8 ejemplos de Sun). Solamente se han incluido en el diagrama los componentes de J2ME que están estandarizados o muy próximos a ser estandarizados y que tienen especial relevancia para el mercado de los dispositivos para comunicaciones móviles.

J2ME presenta dos configuraciones: CLDC y CDC. La primera se dedica a dispositivos con estrictas limitaciones de memoria, capacidad de cálculo, consumo y conectividad de red. Por otro lado, CDC se encarga de dispositivos con más potencia. Parte de CLDC es un subconjunto de CDC, por lo que la portabilidad de aplicaciones se puede conseguir cuando nos movemos de un entorno más restringido a otro más rico. De la misma manera, y siguiendo en el hilo de la portabilidad, una aplicación en J2ME podrá ejecutarse en J2SE normalmente, salvo que se utilicen las bibliotecas específicas de J2ME.

2.7 MIDlet

A las aplicaciones MIDP se denominan "MIDlet", el cual es un programa en lenguaje de programación Java para dispositivos embedidos (se dedican a una sola actividad), más específicamente para la máquina virtual Java MicroEdition (Java ME). Generalmente son juegos y aplicaciones que corren en un teléfono

móvil y está desarrollada bajo la especificación MIDP (perfil para información de dispositivo móvil) [26].

El ciclo de desarrollo de un MIDlet es el siguiente:

- Editar
- Compilar
- Pre-verificar MIDlet
- Ejecución en el emulador
- Ejecución en el dispositivo

Una vez realizados los MIDlets tienen que ser distribuidos en dos archivos especiales. Son los archivos JAR (Java Archive) y los archivos JAD (Java Application Descriptor). Un archivo JAR es un archivo comprimido (en formato ZIP) que contiene las clases (.class) que ha generado la compilación de nuestro programa. Además puede contener los recursos necesarios para el MIDlet como sonidos, gráficos, etc... Finalmente, contiene un archivo con extensión .mf., es lo que se llama un archivo de manifiesto. Este archivo contiene información sobre las clases contenidas en el archivo JAR.

El segundo archivo necesario para la distribución de MIDlets son los archivos JAD. El archivo JAD contiene información necesaria para la instalación de los MIDlets contenidos en el archivo JAR. Un archivo puede contener más de un MIDlet. Cuando ocurre esto, hablamos de un MIDlet suite (véase figura 2.8). Podemos editar los parámetros contenidos en el archivo JAD mediante el botón Settings de KTooBar. Aquí podemos editar información del MIDlet como el nombre, la versión o el autor del MIDlet (o de los MIDlets).

Sólo nos resta transferir los archivos JAR y JAD al dispositivo móvil con soporte J2ME. Hay varias formas de hacerlo dependiendo de la marca y modelo del dispositivo.
Si el teléfono tiene soporte de infrarrojos o bluetooth, y la computadora tiene puerto IrDA o bluetooth, se podrá transferir fácilmente el archivo sin necesidad de cable alguno. Si no, se tendrá que recurrir a un cable de datos (consulta el manual del teléfono).

Otra posibilidad es poner los archivos JAR y JAD en un servidor wap o un espacio web y descargarlo desde tu móvil. Para ello es necesario que el dispositivo tenga un navegador wap o web y soporte GPRS para una descarga fiable.

Figura 2.8 Ejemplos de aplicaciones MIDlets de Sun

Para que una aplicación MIDP trabaje en un dispositivo móvil, se requiere un dispositivo que implemente JavaME y MIDP (versiones 1.0 y posteriores) para ser utilizada [26]. Como otros programas desarrollados en Java, tienen la característica, "Escribir una vez, ejecutar en cualquier parte" ("Write once, run anywhere"). Para escribir se puede obtener Sun Java Wireless Toolkit o NetBeans con la extensión Mobility Pack. Para las distribución son necesarios dos archivos, archivo .JAR conteniendo el bytecode del programa y un archivo .JAD que describe los contenidos del archivo .JAR.

Un MIDlet tiene que cumplir los siguientes requisitos para poder funcionar en un dispositivo móvil:

- La clase principal necesita ser una subclase de javax.microedition.midlet.MIDlet.
- El MIDlet necesita ser empacado dentro de un archivo .JAR.
- El archivo .JAR necesita ser pre-verificado usando un preverificador.
- En algunos casos, el archivo .JAR necesita ser firmado digitalmente por un proveedor de dispositivos móviles.

En la Figura 2.9 podemos observar el diagrama de estados el cual muestra el comportamiento de un Midlet en su ejecución.

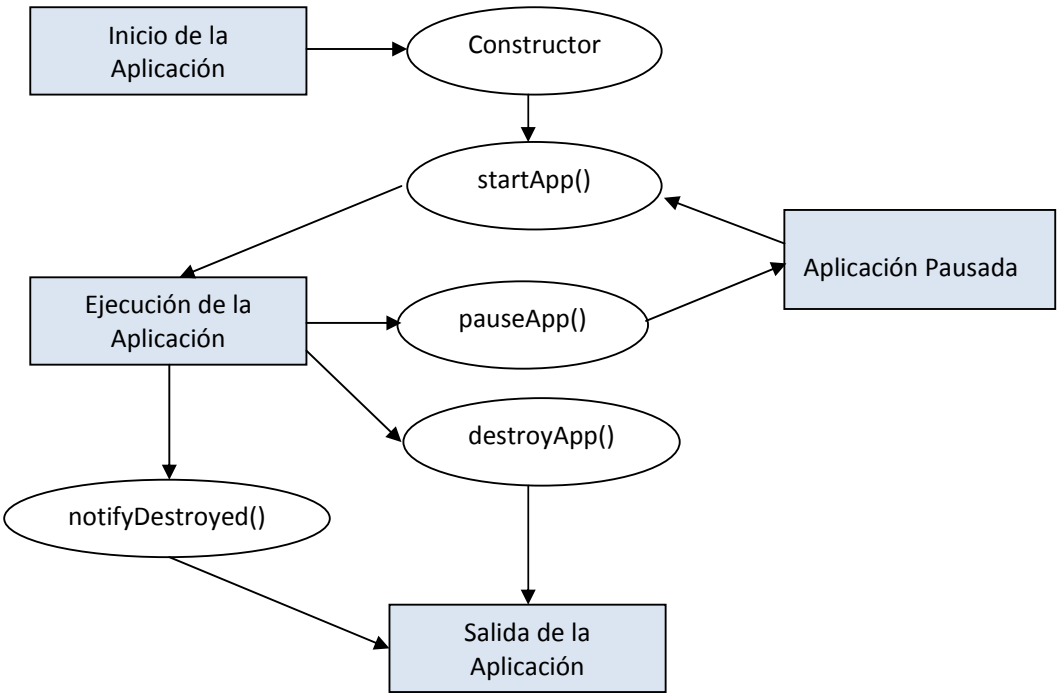

Figura 2.9. Diagrama de Estados de un Midlet

2.8 GPRS

Nuestra aplicación móvil utiliza el servicio GPRS para realizar la descarga de los Objetos de Aprendizaje Móviles, dicho servicio deberá estar configurado el dispositivo móvil que contiene nuestra aplicación.

El sistema GSM es el sistema de comunicación de móviles digital de 2ª generación basado en células de radio. Apareció para dar respuestas a los problemas de los sistemas analógicos.

Fue diseñado para la transmisión de voz por lo que se basa en la conmutación de circuitos, aspecto del que se diferencia del sistema GPRS. Al realizar la transmisión mediante conmutación de circuitos los recursos quedan ocupados durante toda la comunicación y la tarificación es por tiempo. Más adelante veremos como estas limitaciones hacen ineficiente la transmisión de datos con GSM y como GPRS lo soluciona.

Nuestra aplicación realiza la descarga de los Objetos de Aprendizaje Móviles vía GPRS el cual es una nueva tecnología que comparte el rango de frecuencias de la red GSM utilizando una transmisión de datos por medio de 'paquetes'. La conmutación de paquetes es un procedimiento más adecuado para transmitir datos, hasta ahora los datos se habían transmitido mediante conmutación de circuitos, procedimiento más adecuado para la transmisión de voz.

2.8.1 Ventajas del GPRS para el usuario.

Las ventajas que obtiene el usuario con el sistema GPRS son consecuencia directa de las características vistas en el punto anterior.

- Característica de "Always connected": un usuario GPRS puede estar conectado todo el tiempo que desee, puesto que no hace uso de recursos de red (y por tanto no paga) mientras no esté recibiendo ni transmitiendo datos.
- Tarificación por volumen de datos transferidos, en lugar de por tiempo (Por ejemplo: una compañía mexicana cobra $ 0.04 pesos por Kilobyte descargado).
- Coste nulo de establecimiento de conexión a la red GPRS, frente a los quantum de conexiones existentes actualmente en GSM.
- Mayor velocidad de transmisión. En GSM sólo se puede tener un canal asignado (un "timeslot"), sin embargo, en GPRS, se pueden tener varios canales asignados, tanto en el sentido de transmisión del móvil a la estación base como de la estación base al móvil. La velocidad de transmisión aumentará con el número de canales asignados. Además, GPRS permite el uso de esquemas de codificación de datos que permiten una velocidad de transferencia de datos mayor que en GSM.
- Posibilidad de realizar/recibir llamadas de voz mientras se está conectado o utilizando cualquiera de los servicios disponibles con esta tecnología.
- Modo de transmisión asimétrico, más adaptado al tipo de tráfico de navegación html o wml (un terminal GPRS 4+1 (4 slots downlink y 1 uplink) tendrá cuatro veces mayor capacidad de transmisión de bajada que de subida).

2.9 Conexión a redes

Uno de los aspectos más interesantes de los MIDlets es su acceso a Internet a través de una conexión inalámbrica. Las clases e interfaces usadas en MIDP para acceso a red son muy similares a las usadas en J2SE y J2EE.

El CLDC delega las funciones de red en el GCF(Generic Connection Framework). El propósito del GCF es proporcionar una abstracción de los servicios de red. En lugar de forzar a todos los dispositivos a soportar todos los protocolos de red, el GCF describe un marco de trabajo en red del que el dispositivo seleccionará los protocolos y servicios que es capaz de soportar. En realidad, no es el dispositivo el que selecciona las capacidades de red que soportará, sino que esto lo hará el perfil del dispositivo. La responsabilidad del GCF es describir las capacidades de red disponibles en todos los dispositivos así como un sistema extensible en el que los diferentes perfiles de dispositivo pueden soportar un subconjunto de dichas capacidades.

El GCF describe una clase fundamental denominada *Connector*, que será usada para establecer todas las conexiones de red. Los tipos específicos de conexiones son modelados por las interfaces del GCF que son obtenidos a través de la clase *Connector*. Todas estas clases e interfaces están dentro del paquete *javax.microedition.io*. Tal y como vimos anteriormente, las interfaces son las siguientes [23]:

- **Connection –** Una conexión básica que sólo puede ser abierta y cerrada.
- **ContentConnection –** Un flujo (stream) de conexión que proporciona acceso a datos web.
- **DatagramConnection –** Una conexión para manejar comunicaciones orientadas a paquetes.
- **InputConnection –** Una conexión de entrada para las comunicaciones del dispositivo.

- **OutputConnection –** Una conexión de salida para las comunicaciones del dispositivo.

- **StreamConnection –** Una conexión en ambas direcciones para las comunicaciones del dispositivo.

- **StreamConnectionNotifier –** Una conexión especial para notificaciones, que es usada para esperar que se establezca una conexión.

La descripción de las capacidades especificas en cada caso, la tenemos en los perfiles. Por ejemplo, el API de MIDP construye sobre estas interfaces el soporte para el trabajo en red de los MIDlets. El API de MIDP añade la interfaz HttpConnection, que proporciona un componente nuevo en el GCF para conexiones HTTP. Estas conexiones permiten a los MIDlets conectarse con páginas web. La especificación MIDP indica que las conexiones HTTP son el único tipo de conexiones obligatorio en las implementaciones de MIDP.

Siempre usaremos la clase Connector para establecer conexiones, independientemente del tipo de conexión. Todos los métodos de la clase Connector son estáticos. El más importante de ellos es el método open(). Las tres versiones de este método son:

static Connection open (String name)
static Connection open (String name, int mode)
static Connection open (String name, int mode, boolean timeouts)

El primer parámetro de estos métodos es la cadena de conexión. Este es el parámetro más importante porque determina el tipo de conexión que se realizará. La cadena de conexión tendrá el siguiente formato:

Protocolo:Destino[;Parámetros]

La primera parte define el protocolo de red, como http o ftp. El destino es típicamente el nombre de la dirección de red. El último parámetro es una lista de parámetros asociados a la conexión.

Algunos ejemplos de diferentes tipos de cadenas de conexión [25]:

- HTTP – http://www.usal.es/
- Socket – socket://www.usal.es:80
- Datagram – datagram://:9000
- File – file:/datos.txt
- Port – comm:0;baudrate=9600

La segunda y la tercera versión del método open() esperan un segundo parámetro denominado modo (mode), que describe el modo de conexión. Este modo indica si la conexión es abierta para leer, escribir o para ambas cosas. Las siguientes constantes, definidas en la clase Conector, pueden ser usadas para describir el tipo de conexión:

- READ
- WRITE
- READ_WRITE

La última versión del método open() acepta un tercer parámetro, que es un flag que indica si el código que ha llamado el método es capaz de controlar una excepción por tiempo excedido (timeout). Esta excepción es lanzada si el periodo de timeout para establecer la conexión falla. Los detalles de este periodo y cómo es gestionado, es labor del protocolo específico, por lo que no tenemos control sobre el mismo.

El método open() devuelve un objeto de tipo Connection, que es la interface básica para el resto de interfaces de conexión. Para usar undeterminado tipo de interface de conexión, debemos convertir la interfaz Connection que devuelve el método open() al tipo apropiado:

```
StreamConnection conn =
(StreamConnection)Connector.open(http://www.usal.es/);
```

2.9.1 Entrada/Salida desde el MIDlet

La clase Connector y las interfaces asociadas a esta son usadas para obtener una conexión a la red. Una vez que la conexión está establecida, debemos usar las clases de entrada/salida para leer y escribir datos a través de la conexión. Estas clases son aportadas por el paquete java.io:

- ByteArrayInputStream – Un flujo (stream) de entrada que se gestiona internamente como una matriz de bytes.
- ByteArrayOutputStream – Un flujo (stream) de salida que se gestiona internamente como una matriz de bytes.
- DataInputStream - Un flujo (stream) desde el cual se leen datos como tipos primitivos.
- DataOutputStream – Escribe datos en tipos primitivos en un flujo (stream) en formato binario.
- InputStream – La clase base para todos los flujos (streams) de entrada
- InputStreamReader – Un flujo (stream) desde el que se puede nleer caracteres de texto.
- OutputStream – La clase base para todos los flujos (streams) de salida
- OutputStreamWriter – Un flujo (stream) en el que se pueden escribir caracteres de texto.
- PrintStream – Un flujo (stream) de escritura que facilita el “envío” de datos en forma de tipos primitivos.
- Reader – Una clase abstracta para leer flujos (streams) de lectura
- Writer – Una clase abstracta para leer flujos (streams) de escritura

Las dos clases más importantes de esta lista son InputStream y OutputStream, que proporcionan la funcionalidad básica para recibir y enviar datos.

2.9.2 La clase InputStream

La clase InputStream es una clase abstracta que sirve como base para otras clases de streams de entrada en MIDP. Esta clase define un interfaz básico para leer bytes en modo de stream. El uso normal será crear un objeto de tipo InputStream desde una conexión y entonces recoger información a través del

método read(). Si no hay información disponible en este momento, el stream de entrada usa una técnica como conocida como bloqueo (bloquing) para esperar hasta que haya datos disponibles.

La clase InputStream define los siguientes métodos:

- read()
- read(byte b[])
- read(byte b[], int off, int len)
- skip(long n)
- available()
- mark(int readlimit)
- reset()
- markSupported()
- close()

2.9.3 La clase OutputStream

La clase OutputStream es el homólogo de salida de la clase InputStream y sirve como base para todas las clases de streams de salida de MIDP. La clase OutputStream define el protocolo básico para escribir datos a través de un stream de salida. Normalmente crearemos un objeto de tipo OutputStream en una conexión y entonces llamaremos al método write [23, 24]. La clase OutputStream usa la técnica de bloqueo de forma similar a lo visto para la clase InputStream, es decir, se bloquea hasta que los datos son escritos en el stream. Durante el bloqueo la clase OutputStream no permite que más datos sean escritos.

Los métodos de la clase son:

- write(int b)
- write(byte b[])
- write(byte b[], int off, int len)
- flush()
- close()

2.9.4 Haciendo conexión

El primer paso en el desarrollo de cualquier tipo de comunicación de red en un MIDlet involucra el establecimiento de la conexión, esto se realiza a través de static open() el cual es método de la clase Connector [24]. El siguiente ejemplo muestra como abrir una conexión HTTP de una página web:

```
String url = "http://localhost:8080/mi_servlet/OAmt?por="+por+
"&tipo="+tipo+"&busqueda="+enviar;
HttpConnection c = (HttpConnection)Connector.open(url);
```

La conexión lanza un objeto HttpConnection para que se pueda realizar la comunicación conexión por un Outputstream o Inputstream. El siguiente paso es obtener un Outputstream o Inputstream sobre la conexión, el cual es necesario actualmente para ejecutar cualquier I/O.

2.9.5 Obtener un Input Stream

El obtener un Stream de una conexión es un poco avanzado. Para el propósito de lo desarrollado en este proyecto de tesis, demostraré como obtener un InputStream, el cual podrá ser usado para leer la página web asociada a la conexión HTTP. El código necesario para obtener un InputStream de una conexión a través del cual se puedan leer datos, es el siguiente:

```
InputStream in = null;
in = conn.openInputStream();
```

El método *openInputStream()* es todo lo necesario para establecer un InputStream. Lo siguiente a este punto es obtener el Output Stream para que se pueda iniciar la lectura de los datos de la conexión HTTP.

2.9.6 Leyendo desde el Stream

La clave para la lectura de datos de un objeto IntputStream es el uso del método *read()*. La versión básica del método *read()* lee un byte de datos desde el InputStream y regresa a este como un entero. Las llamadas subsecuentes al método *read()* resulta en bytes adicionales de datos iniciando la lectura desde

el Stream, continuando hasta el final Stream, el cual resulta en el retorno de el método *read().* El siguiente es un ejemplo de cómo el código dentro de un ciclo (loop) lee líneas de datos de un InutStream e imprime la lectura:

```
StringBuffer data = new StringBuffer();
int ch;
while ((ch = in.read()) != -1) {
if (ch != '\n') {
// Lee la línea caracter por caracter
data.append((char)ch);
}
else {
//  Imprime la línea como cadena de datos
System.out.println(data.toString());
// Limpia la cadena para la siguiente línea
data = new StringBuffer();
}
}
```

En éste código la variable *data* es un *string buffer* que es usado para retener por un momento una línea de datos. El método *read()* es llamada para leer caracteres del Stream. Dentro del ciclo *while*, el actual caracter es comparado con '/n' (nueva línea) para ver si el final de la línea ha sido alcanzado. Si es así, ya sabemos que tenemos acumulada una línea de texto completa, y si no hay error se imprime y se limpia el *string buffer. [22,24]*

Este código representa la premisa básica de cómo podemos leer los datos a través de un Input Stream de una conexión HTTP.

Capítulo 3
DISPOSITIVOS MÓVILES

3.1 Telefonía Móvil [20]

G1

La primera generación de telefonía móvil (**G1**) funcionaba por medio de comunicaciones analógicas y dispositivos portátiles que eran relativamente grandes. Esta generación utilizaba principalmente los siguientes estándares:

- **AMPS** (Sistema telefónico móvil avanzado): Se presentó en 1976 en Estados Unidos y fue el primer estándar de redes celulares. Utilizada principalmente en el continente americano, Rusia y Asia, la primera generación de redes analógicas contaba con mecanismos de seguridad endebles que permitían hackear las líneas telefónicas.
- **TACS** (Sistema de comunicaciones de acceso total): Es la versión europea del modelo AMPS. Este sistema fue muy usado en Inglaterra y luego en Asia (Hong-Kong y Japón) y utilizaba la banda de frecuencia de 900 MHz.
- **ETACS** (Sistema de comunicaciones de acceso total extendido): Es una versión mejorada del estándar TACS desarrollado en el Reino Unido que utiliza una gran cantidad de canales de comunicación.

Con la aparición de una segunda generación totalmente digital, la primera generación de redes celulares se volvió obsoleta.

G2

La segunda generación de redes móviles (**G2**) marcó un quiebre con la primera generación de teléfonos celulares al pasar de tecnología analógica a digital.

Los principales estándares de telefonía móvil de G2 son:

- **GSM** (*Sistema global para las comunicaciones móviles*): El estándar más usado en Europa a fines de siglo XX y también se admite en Estados Unidos. Este estándar utiliza las bandas de frecuencia de 900 MHz y de 1800 MHz en Europa. Sin embargo, en Estados Unidos la banda de frecuencia utilizada es la de 1900 MHz. Por lo tanto, los teléfonos móviles que pueden funcionar tanto en Europa como en Estados Unidos se denominan teléfonos de **tribanda**.

- **CDMA** (*Acceso múltiple por división de código*): Utiliza una tecnología de espectro ensanchado que permite transmitir una señal de radio a través de un rango de frecuencia amplio.

- **TDMA** (*Acceso múltiple por división de tiempo*): Emplea una técnica de división de tiempo de los canales de comunicación para aumentar el volumen de los datos que se transmiten simultáneamente. Esta tecnología se usa, principalmente, en el continente americano, Nueva Zelanda y en la región del Pacífico asiático.

Gracias a la G2, es posible transmitir voz y datos digitales de volúmenes bajos, por ejemplo, mensajes de texto (**SMS** siglas en inglés de *Servicio de mensajes cortos*) o mensajes multimedia (**MMS** siglas en inglés de *Servicio de mensajes multimedia*). El estándar GSM permite una velocidad de datos máxima de 9,6 kbps.

Se han hecho ampliaciones al estándar GSM con el fin de mejorar el rendimiento. Una de esas extensiones es el servicio **GPRS** (*Servicio general de paquetes de radio*) que permite velocidades de datos teóricas en el orden de los 114 Kbits/s pero con un rendimiento cercano a los 40 Kbits/s en la práctica. Como esta tecnología no se encuentra dentro de la categoría "G3", se la llama **G2.5**.

El estándar **EDGE** (*Velocidades de datos mejoradas para la evolución global*) anunciado como **G2.75**, cuadriplica las mejoras en el rendimiento de GPRS con

la tasa de datos teóricos anunciados de 384 Kbps, por lo tanto, admite aplicaciones de multimedia. En realidad, el estándar EDGE permite velocidades de datos teóricas de 473 Kbits/s pero ha sido limitado para cumplir con las especificaciones IMT-2000 (*Telecomunicaciones móviles internacionales-2000*) de la ITU (*Unión internacional de telecomunicaciones*).

G3

Las especificaciones IMT-2000 (*Telecomunicaciones móviles internacionales para el año 2000*) de la Unión internacional de telecomunicaciones (ITU) definieron las características de la **G3** (tercera generación de telefonía móvil). Las características más importantes son:

- Alta velocidad de transmisión de datos :
- 144 Kbps con cobertura total para uso móvil.
- 384 Kbps con cobertura media para uso de peatones.
- 2 Mbps con áreas de cobertura reducida para uso fijo.
- Compatibilidad mundial.
- Compatibilidad de los servicios móviles de G3 con las redes de segunda generación.

La G3 ofrece velocidades de datos de más de 144 Kbit/s y de este modo brinda la posibilidad de usos multimedia, por ejemplo, transmisión de videos, video conferencias o acceso a Internet de alta velocidad. Las redes de G3 utilizan bandas con diferentes frecuencias a las redes anteriores: 1885 a 2025 MHz y 2110 a 2200 MHz.

El estándar G3 más importante que se usa en Europa se llama **UMTS** (*Sistema universal de telecomunicaciones móviles*) y emplea codificación **W-CDMA** (*Acceso múltiple por división de código de banda ancha*). La tecnología UMTS usa bandas de 5 MHz para transferir voz y datos con velocidades de datos que

van desde los 384 Kbps a los 2 Mbps. El **HSDPA** (*Acceso de alta velocidad del paquete de Downlink*) es un protocolo de telefonía móvil de tercera generación, apodado "G3.5", que puede alcanzar velocidades de datos en el orden de los 8 a 10 Mbps. La tecnología HSDPA usa la banda de frecuencia de 5 GHz y codificación W-CDMA.

Tabla 3.1 Cuadro sinóptico de Tecnologías Móviles.

Estándar	Generación	Banda de frecuencia	Rendimiento	
GSM	G2	Permite la transferencia de voz o datos digitales de bajo volumen.	9,6 kbps	9,6 kbps
GPRS	G2.5	Permite la transferencia de voz o datos digitales de volumen moderado.	21,4 a 171,2 kbps	48 kbps
EDGE	G2.75	Permite la transferencia simultánea de voz y datos digitales.	43,2 a 345,6 kbps	171 kbps
UMTS	G3	Permite la transferencia simultánea de voz y datos digitales a alta velocidad.	0,144 a 2 Mbps	384 kbps

3.2 GPRS (General Packet Radio Service)

Para que el dispositivo móvil pudiera efectuar los protocolos de comunicación mencionados a lo largo de la presente Tesis, se contrato el servicio de navegación en Internet con la empresa prestadora del servicio, en modo GPRS, lo cual consta de un paquete de datos que cuesta 4 Centavos por cada Kb recibido, para tal efecto nuestra aplicación recibe alrededor de .775 Kb, demostrando así que aunque es una aplicación para altos ejecutivos, su costo de operación no es excesivo [41].

Definición

Protocolo inalámbrico, no de voz, que ofrece acceso a redes de datos mediante la conmutación de paquetes. El servicio ofrece tasas de transmisión de datos de hasta 170 Kbps. La característica mas notable de esta tecnología es que provee una conexión permanente entre la red y la Terminal móvil. El término permanente se refiere a que la transmisión de datos no es orientada a conexión. Es decir, en este tipo de tecnología no necesita abrir y cerrar una conexión, únicamente debe preocuparse por el envió de paquetes.

El uso de esta tecnología requiere de equipos móviles capaces de interactuar con sitios web que ofrecen servicios multimedia y que contienen aplicaciones que permitan el manejo de estos servicios. [21]

Arquitectura

La popularidad que ha adquirido GPRS junto a GSM es notable. Desde hace algunos años la tecnología GSM ha empezado su auge en México. GSM en conjunción con GPRS brinda a los usuarios nuevos servicios nunca antes brindados mediante la tecnología celular digital. La arquitectura del sistema que permite ofrecer estos servicios esta representada en la figura (4.2.1).

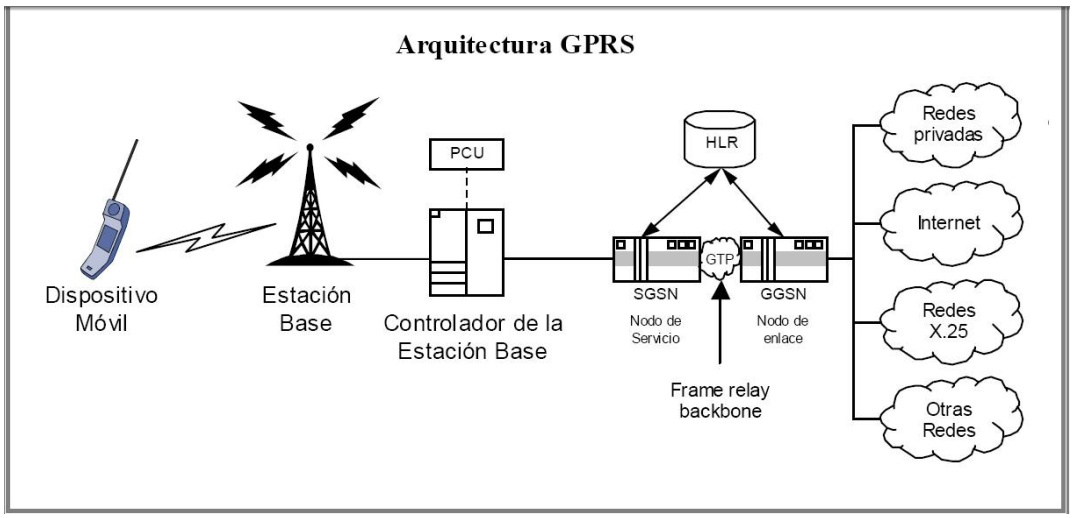

Figura 3.1 Arquitectura GPRS.

Los servicios que provee GPRS utilizan la misma interfaz aérea que GSM. Con el fin de proveer la transmisión de datos GPRS define una infraestructura extra, sobre la infraestructura GSM existente. Así, un dispositivo móvil se comunica con el Controlador de la Estación Base mediante la Estación Base. El CEB de GSM existente solo maneja voz, adicionalmente requiere de una unidad de control de paquetes (PCU), quien será encargada de desviar los paquetes GPRS hasta el nodo de soporte de servicios GPRS (SGSN). El nodo SGSN se encarga de mandar y recibir los datos a los dispositivos móviles, además interactúa con un registro de ubicación local (HLR) para identificar a los dispositivos disponibles en su área de servicio y así hacerse cargo del hand-off de las sesiones GPRS. Este nodo a su vez se conecta al nodo de soporte gateway GPRS (GGSN) mediante frame delay. En este punto la comunicación se realiza mediante un protocolo llamado GPRS tunel protocol (GTP) que encapsula paquetes IP o X.25 para ser transmitidos entre el SGSN y el GGSN. Finalmente, el GGSN es quien mantiene al sistema conectado a redes como Internet, X.25 y otras redes privadas. Este dispositivo actúa, además, como firewall protegiendo a la infraestructura GPRS de cualquier ataque que pueda provenir por la Internet. También participa junto con el HLR para desviar el tráfico SGSN adecuado, ya que puede haber más de un SGSN en la infraestructura GPRS [41], [42].

3.3 NetBeans de Sun Microsystems

NetBeans es un IDE (*Integrated Development Environment* o, en español, Entorno Integrado de Desarrollo) escrito en Java que agrupa un conjunto de utilidades que facilitan la edición, compilación, depuración, análisis y ejecución de un programa Java. [22]

Con él se pueden diseñar interfaces gráficas de usuario casi sin escribir código, al más puro estilo visual de Delphi o C++, sólo con hacer clic en los componentes sobre la ventana seleccionada.

Además de ser gratuito y de acogerse al proyecto "*open-source project*", NetBeans se caracteriza por su entorno de desarrollo intuitivo, personalizable,

modular y extensible. Es precisamente esta última característica la que permite ampliar el paquete de clases que trae por defecto NetBeans 5.5.1 con una extensión especial para programadores J2ME que, no sólo cuenta con todas las clases necesarias para crear un *midlet* sino que añade al IDE una serie de herramientas para facilitar la integración con las aplicaciones J2SE o J2EE [44].

Al ser NetBeans la herramienta que mas se nos facilito de aprender, y para la que encontramos mas tutoriales sobre su aprendizaje, para llevar a buen puerto este proyecto, creemos conveniente exponer aquí unas nociones sobre cómo crear una pequeña aplicación visual.

Creación de una aplicación Web

Una aplicación Web, es un contenedor de servlets, a los cuales se puede acceder desde nuestra aplicación móvil, y a continuación se describe la manera de crearlo fácil y rápido, utilizando las pantallas que nos proporciona nuestro IDE.

Dar click en Nuevo Proyecto-->Web--->Aplicación Web y Botón de Siguiente (Figura 3.2).

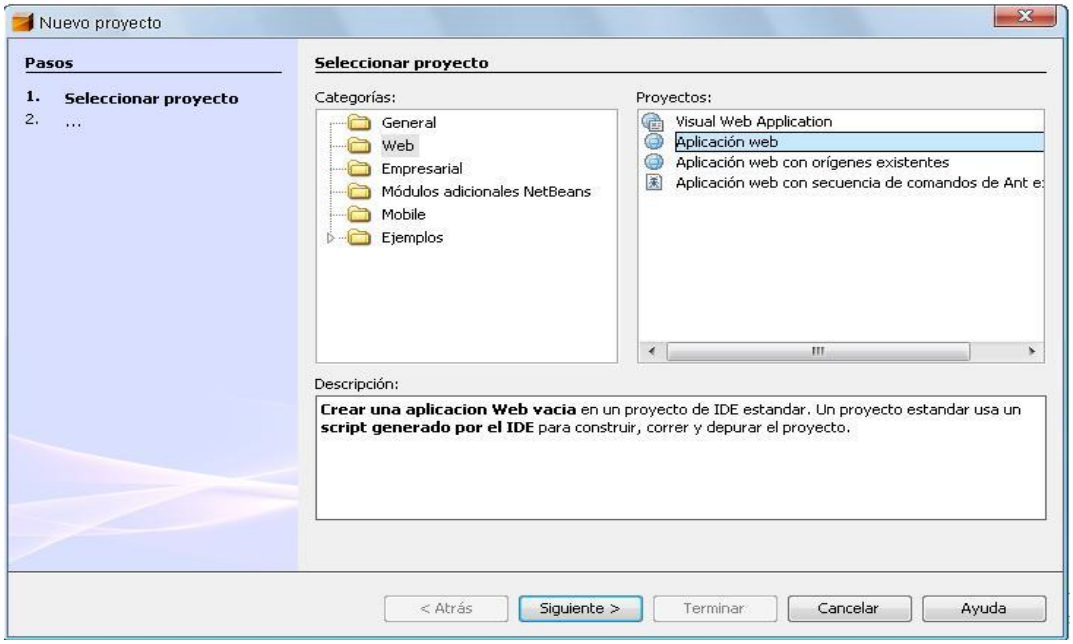

Figura 3.2 Pantalla Nuevo Proyecto.

Introducimos el nombre que tendrá, y la dirección donde estará alojada, lo demás lo dejamos como esta por default, y por ultimo damos en el botón de Finalizar (Figura 3.3)

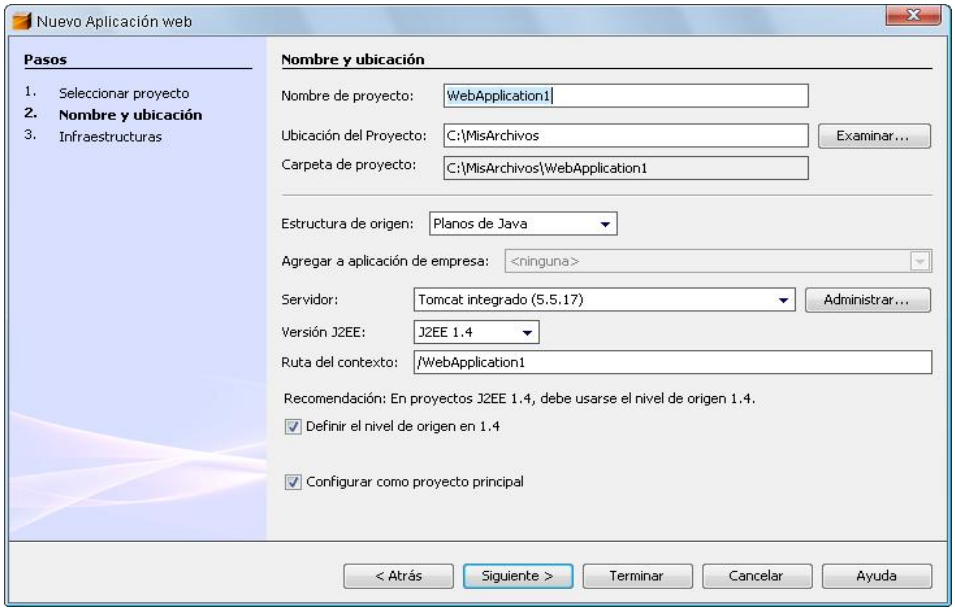

Figura 3.3 Pantalla Nuevo Aplicación Web.

Añadir nuevos servlets

Una vez que hemos creado y ejecutado nuestra aplicación web sin ningún error, podemos agregarle los servlets necesarios para establecer la comunicación con nuestra aplicación web.

Desplegamos nuestro árbol de Proyecto de nuestra aplicación Web, desglosamos nuestro Paquete de Origen, y en el Paquete Determinado damos click derecho --->>> Nuevo --->>> Servlet. (Figura 3.4).

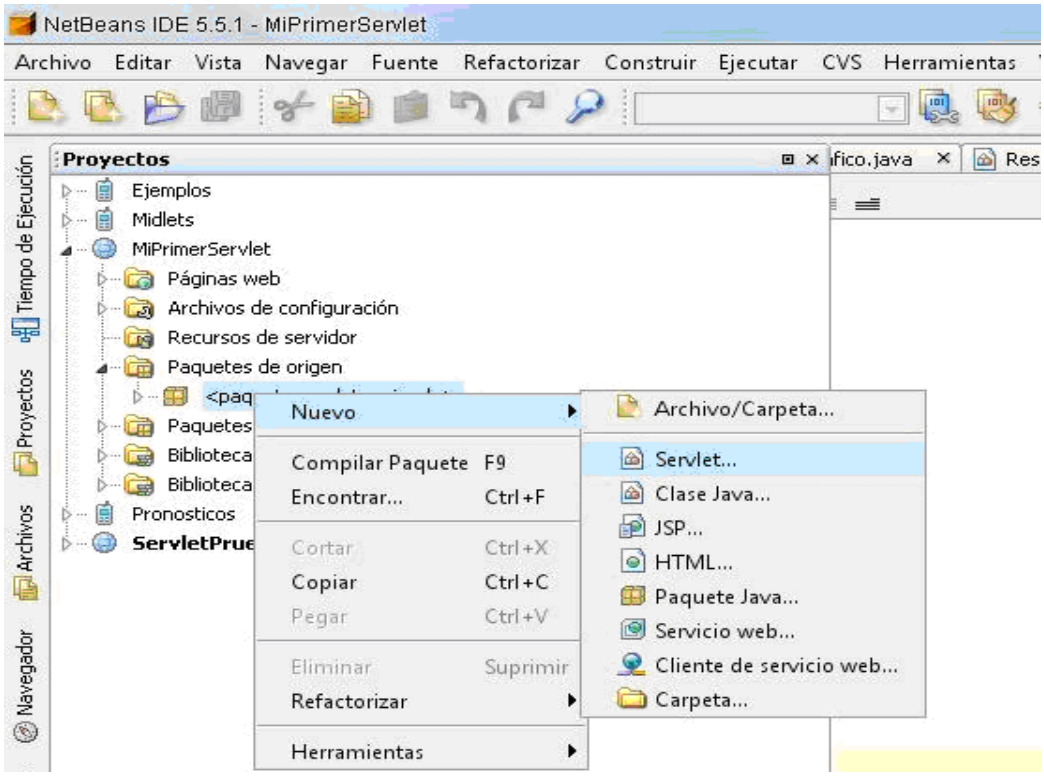

Figura 3.4 Pantalla IDE Nuevo→Servlet.

Escribimos el nombre de nuestro servlet, y damos click en finalizar, y nos aparecerá el esqueleto de nuestro servlet, el cual podremos modificar (Figura 3.5).

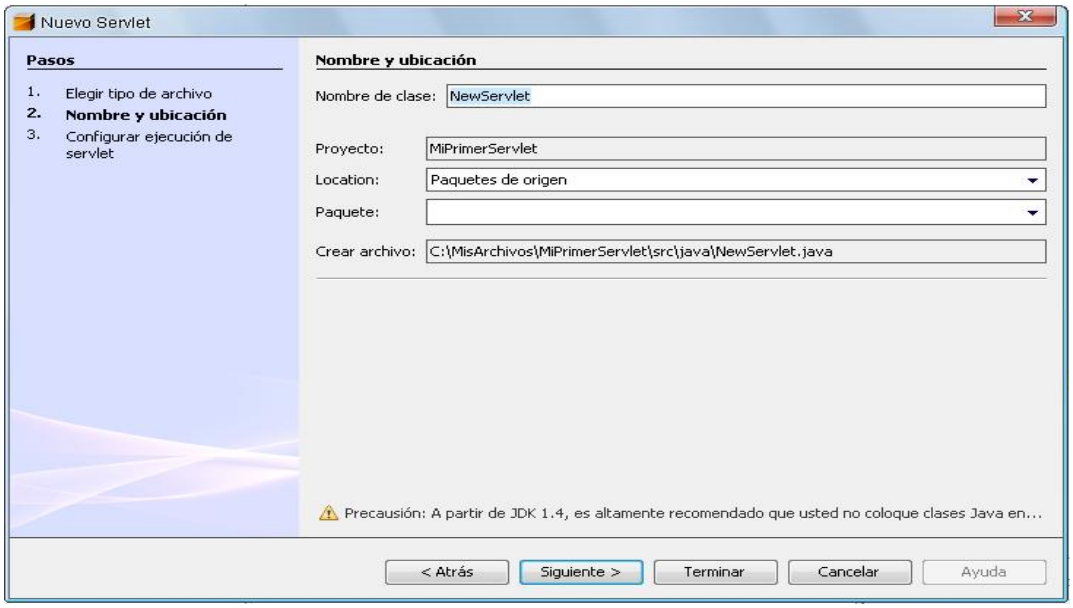

Figura 3.5 Pantalla Nuevo→Servlet.

Creación de un MIDlet con conexión a Internet

Ya que hemos agregado nuestro primer servlet de prueba, ahora crearemos un MIDlet para poder acceder de manera local desde nuestro emulador [43].

```
import java.io.*;

import javax.microedition.io.*;

import javax.microedition.lcdui.*;

import javax.microedition.midlet.*;


/**

* An example MIDlet to invoke a CGI script.

*/

public class FirstMidletServlet extends MIDlet {

private Display display;


String url = "http://localhost:8084/ServletPrueba/HelloServlet";

public FirstMidletServlet() {

display = Display.getDisplay(this);

}


/**
```

```
 * Initialization. Invoked when we activate the MIDlet.    */
public void startApp() {
        try {
      invokeServlet(url);
        } catch (IOException e) {
           System.out.println("IOException " + e);
           e.printStackTrace();
        }
  }
/**    * Pause, discontinue ....       */
public void pauseApp() {
}
/**
 * Destroy must cleanup everything.
 */
public void destroyApp(boolean unconditional) {}
/**     * Prepare connection and streams then invoke servlet.     */
 void invokeServlet(String url) throws IOException {
   HttpConnection c = null;
   InputStream is = null;
   StringBuffer b = new StringBuffer();
   TextBox t = null;
   try {
    c = (HttpConnection)Connector.open(url);
    c.setRequestMethod(HttpConnection.GET);
    c.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
    c.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-1.0");
    c.setRequestProperty("Content-Language", "en-CA");
    is = c.openDataInputStream();
```

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * The simplest possible servlet.
 */

public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws IOException, ServletException {
    response.setContentType("text/plain");
    PrintWriter out = response.getWriter();
    out.println("Servlet invoked!");
    out.println(new Date());
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
       String firstName = request.getParameter("firstName").toString();
       System.out.println("firstName = " + firstName);
       processRequest(request, response);
    }
```

Y por el otro lado, nuestra aplicación web, contendrá el servlet HelloServlet:

J2ME versus NetBeans

Aprovechando las ventajas que nos ofrece Nuestro IDE, hemos decidido, además de albergar nuestra aplicación Web, también utilizarlo para crear nuestro proyecto, ya que de otra manera debemos, en primer lugar crear un archivo con nuestro MIDlet, después compilarlo, y por ultimo ejecutarlo en nuestro emulador, pero cada una de estas fases se presentan por separado, siendo muy engorroso al trabajar de esta manera, es así como NetBeans nos ofrece de manera grafica y en una sola pantalla cada una de estas fases, pasando fácilmente de una a la otra sin necesidad de abrir pantallas adicionales para nuestro proyecto.

3.4 El servidor de servlets: Tomcat 5

Tomcat (también llamado *Jakarta Tomcat* o *Apache Tomcat*) funciona como contenedor de servlets y es desarrollado bajo el proyecto Jakarta [22] en la *Apache Software Foundation*. Implementa las tecnologías *Java Servlet* 2.4 y *JavaServer Pages* 2.0 (JSP) de *Sun Microsystems*.

Tomcat es un servidor de aplicaciones que, a diferencia de un servidor Web, como es por ejemplo *Apache*, incluye un contenedor Web que puede servir páginas dinámicas (a diferencia del servidor Web, que solo sirve páginas HTML estáticas). Incluye el compilador Jasper, que compila páginas JSPs y las convierte en servlets. Además, funciona con cualquier sistema operativo que disponga de máquina virtual Java, ya que fue escrito en este mismo lenguaje.

Historia

James Duncan Davidson se puede considerar el padre de Tomcat. Trabajaba como arquitecto de software para Sun Microsystems en el momento en que decidió donar el proyecto a la Apache Software Foundation .

Eligió el nombre de Tomcat (gato) pretendiendo representar la capacidad del programa de ser independiente, de cuidarse por sí mismo. Aunque el verdadero motivo es que Duncan esperaba que Tomcat se convirtiese en *open source* y que O`Reilly

(famosa editorial norteamericana dedicada a la informática) publicase un libro sobre su proyecto. O`Reilly es conocida por asociar animales a las portadas de sus libros.

En el momento de escribir este documento Apache Tomcat se encuentra en su

Versión 5.5.1

3.5 NetBeans Versión 5.5.1 con MySQL

Conexión de NetBeans 5.5.1 con MySQL

Ahora que tenemos configurado nuestra aplicación Móvil con nuestra aplicación Web, necesitamos configurar la conexión de nuestro IDE con MySQL, ya que haremos uso de nuestra Base de Datos para realizar las consultas necesarias para generar los datos base para crear nuestro escenario de pronostico.

Para ello necesitamos el mismo conector que utilizamos para Java, y aquí se explica brevemente en que consiste:

La intención es crear una nueva librería a nuestro proyecto de modo que contendrá todas y cada una de las clases necesarias para realizar la conexión con la BD.

1. Primero crearemos un proyecto normal (Aplicación Java)

2. Abrimos el manejador de bibliotecas. **Menu Herramientas ->**

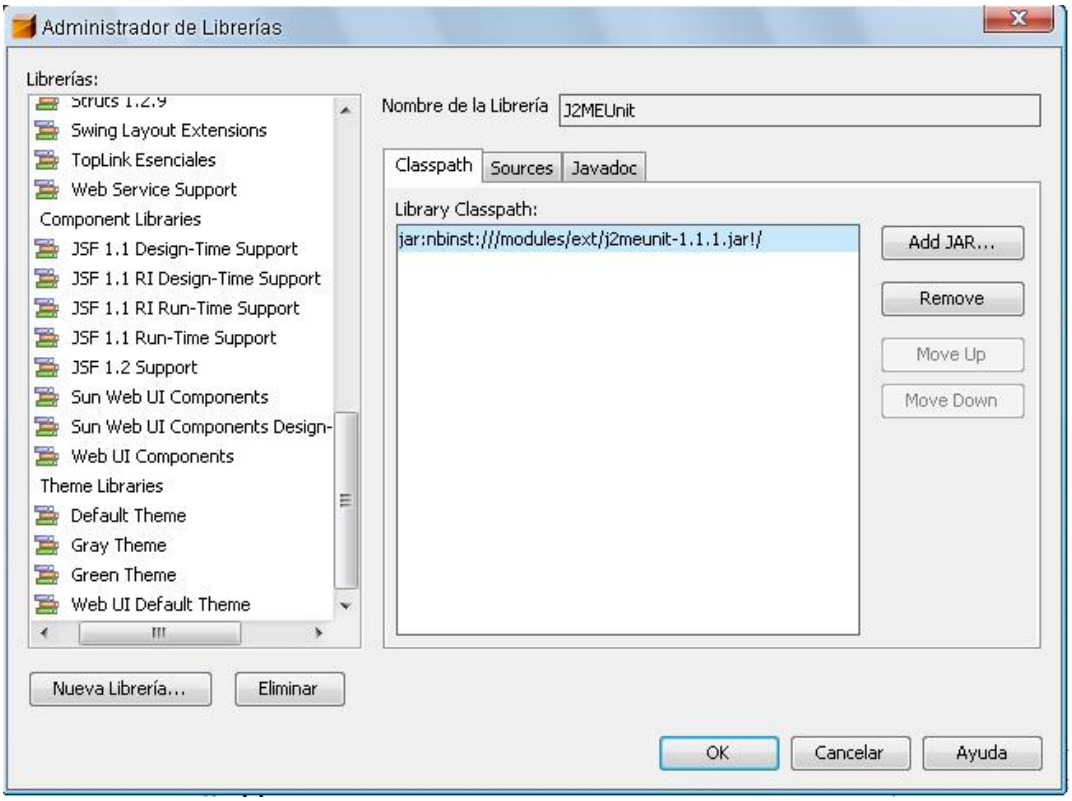

Figura 3.6 Administrador de Librerías.

3. Nos aparecerá el cuadro de dialogo Administrador de Librerías. Ahora debemos dar de alta una nueva biblioteca dando click en **Nueva Librería...**

4. En el cuadro que nos aparece, debemos llenarlo de la siguiente forma: (Figura 4.7).

   Nombre de la Librería: MySQL (no puede tener espacios)

   Tipo de Librería: Librería de clases.

Figura 3.7 Nueva Librería

Click en el botón OK

5. Ahora debemos agregar unos detalles a la biblioteca. Asegúrese que la nueva biblioteca creada este seleccionada (izquierda).

6. Seleccionamos la pestaña Class Path y damos clic en el botón **Agregar archivo JAR/carpeta**

7. Se abre un dialogo llamado Browse JAR/Folder que permite seleccionar archivos. Buscamos el jar descargado para conectarnos a MySQL (mysql-3.1.11.jar) los seleccionamos y damos click en el botón **Agregar archivo JAR/carpeta**

8. Aparecerá en la sección Class Path el archivo que acabamos de agregar, damos clic en Ok del cuadro de dialogo Administrador de Librerías.

9. Ahora debemos agregar la librería al proyecto. Dentro del proyecto (explorador del proyecto) seleccionar el nodo Bibliotecas. (Figura 4.8)

Dar click derecho para mostrar el menú contextual

- Seleccionar opción Agregar Biblioteca
- Seleccionar de la lista, la biblioteca recién creada con el nombre MySQL

Debe aparecer como una nueva entrada dentro del nodo Bibliotecas. Y Listo.

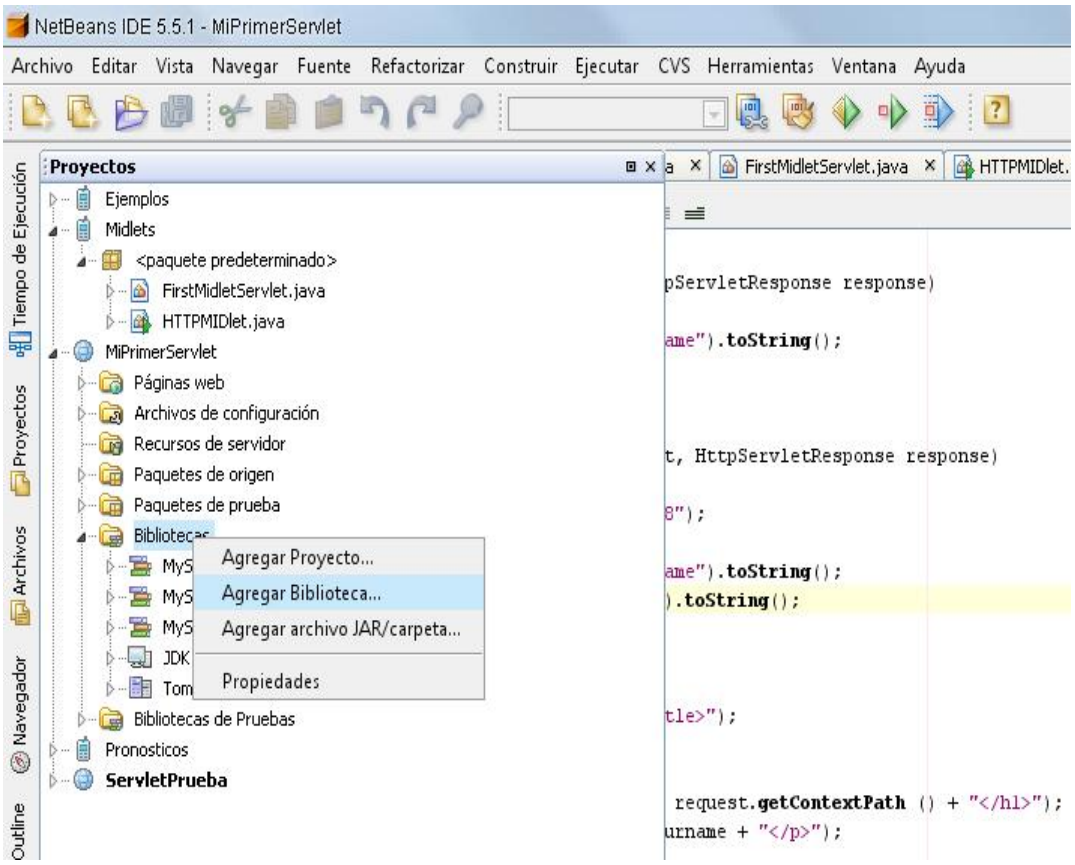

Figura 3.8 Agregar Biblioteca.

Podemos escribir el siguiente programa para verificar que todo funciona.

```
import java.sql.*;
public class TestConnection {
static String bd = "tu_BD";
static String login = "usuario";
static String password = "contraseña";
static String url = "jdbc:mysql://localhost/"+bd;

        public static void main(String[] args) throws Exception { Connection conn = null;
        try {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        conn = DriverManager.getConnection(url,login,password);
        if (conn != null) {
                System.out.println("Conexión a base de datos "+url+"          ... Ok");
                conn.close();
}
}
catch(SQLException ex) {
          System.out.println("Hubo un problema al intentar conectarse con   la base de datos "+url);
}
catch(ClassNotFoundException ex) {
System.out.println(ex);
}
}//MAIN
}//CLASS
```

Deberá aparecer conexión a base de datos "+url+" ... Ok

Capítulo 4
LOGICA MÓVIL

DLV es una de los sistemas basados en el paradigma ASP más ampliamente usado y exitoso a nivel internacional. Esta herramienta se basa en la semántica de modelos estables, y soporta un lenguaje poderoso que extiende la programación lógica disyuntiva con muchos constructores expresivos, incluyendo agregaciones, restricciones débiles y fuertes, funciones, listas y conjuntos.

Actualmente, DLV es empleado aplicaciones industriales, e incluso está listo para distribución comercial. Así lo demuestran diversos proyectos basados en esta herramienta tales como: “Potenziamento e Applicazioni della Programmazione Logica Disgiuntiva” and “Sistemi basati sulla logica per la rappresentazione di conoscenza: estensioni e tecniche di ottimizzazione.” Además, los programas lógicos disyuntivos (DLV) son una poderosa herramienta en la representación de conocimiento y razonamiento de sentido común. Esta herramienta permite resolver problemas complejos. Sin embargo, los sistemas para programas lógicos disyuntivos carecen actualmente de una interface que permita a los investigadores del área poder contar con esta herramienta en todo momento y en cualquier lugar, lo que dificulta el quehacer científico y de investigación, así como el desarrollo de aplicaciones de manera permanente.

Así, en el presente capítulo mostramos nuestra aplicación que permite contar con un sistema de lógica móvil, es decir, llevar a la mano de cualquier usuario de telefonía móvil la posibilidad de contar con DLV en su teléfono móvil. Esto permite tanto a estudiantes, profesores e investigadores del área de lógica (DLV) poder experimentar sus ideas en cualquier lugar y en todo momento.

4.1 La revolución móvil

Si con la llegada de las computadoras e Internet llegó el Aprendizaje Electrónico (mejor conocido en inglés como e-Learning), con la revolución móvil actual llega el computador Móvil. El computador móvil, no es más que la consecuencia lógica y natural de la evolución que la tecnología experimenta día con día.

Con el desarrollo del Bluetooth, WAP, GPRS y UMTS, el movimiento de la tecnología y la informática hacia un entorno inalámbrico es irreversible. Tecnologías y aplicaciones son sustituidas o completadas por otras nuevas e inalámbricas: e-Business por m-Business, e-Commerce por m-Commerce y también e-Learning por m-Learning, de igual forma proponemos llevar la lógica a m-Logic, es decir DLV a m-DLV.

4.2 Aplicación Móvil

En esta sección presentamos nuestra aplicación basada en dispositivos móviles. En particular hemos diseñado nuestra aplicación móvil de DLV para teléfonos móviles a través de J2ME. Esto obedece a que actualmente existen más de 3.5 billones de usuarios potenciales en el mundo que ya cuentan con dispositivos móviles y nos ofrece un gran campo de desarrollo. Además, la mayoría de los usuarios existentes son jóvenes y esto le da un valor agregado a las aplicaciones desarrolladas para móviles y más aún a los sistemas de la relevancia de DLV.

En la figura 4.1 mostramos las interfaces principales de nuestra aplicación. Como se puede observar, la primera solo es la pantalla de bienvenida que permite al usuario identificar que el sistema DLV (programas lógicos disyuntivos) se está cargando para preparar la interfaz donde se pueda capturar el programa o reglas del problema que nos intereza.

Figura 4.1 Interfaz inicial del sistema DLV móvil

En la figura 4.2 se presenta la interfaz que permite al usuario poder escribir las reglas disyuntivas que modelen el problema que desea experimentar. Además, la interfaz permite que el usuario pueda dar los parámetros que dlv requiera para la ejecución del programa lógico disyuntivo.

Figura 4.2 Interfaz para la captura del programa lógico

Enseguida, tal como se puede observar en la figura 4.3, una vez que las reglas son escritas, el usuario puede decidir la ejecución de estas vía GPRS. En el ejemplo particular que se muestra en esta interfaz (figura 4.2) representa lo siguiente:

Sea P el programa mostrado en la figura 4.3

p :- not q.

q :- not p.

Tales reglas podemos interpretarlas de la manera siguiente:

Sea q: “es de noche” y sea

p: “es de dia”

Así, las dos reglas se interpretan como sigue:

Si no es de noche entonces es de día

Si no es de día entonces es de noche

Es decir, las reglas representan un “or” exclusivo.

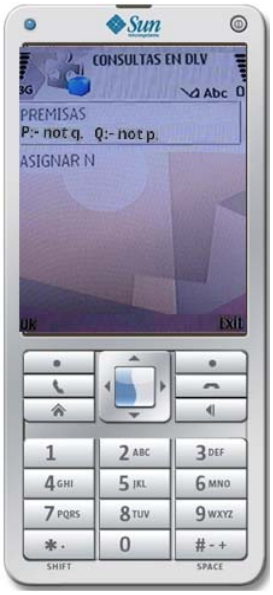

Figura 4.3 Interfaz con reglas escritas en DLV

Una vez que el usuario ha terminado de escribir su problema a través de reglas lógicas disyuntivas, se encuentra en la posibilidad de poder ejecutar éstas usando DLV móvil. Para esto, tal como se muestra en la figura 4.4 el usuario deberá presionar el botón de “OK” para que la aplicación haga lo solicitud de conexión vía internet. Para esto, la aplicación usa el protocolo de comunicación WAP y el servicio de comunicación GPRS. Esta forma de comunicación hace que nuestra aplicación implique un costo bajo por el uso de DLV móvil. En promedio, un usuario de DLV móvil gasta 40 centavos por la ejecución (solución) de un programa escrito en DLV móvil.

Figura 4.4 Aplicación móvil solicitando acceso a través del protocolo WAP

La solicitud de comunicación y respuesta de nuestra aplicación tarda en promedio alrededor de 30 segundos. Cabe mencionar que para poder mostrar las soluciones calculadas a través de DLV, la aplicación móvil realiza una conexión inalámbrica (mencionado en el Capítulo 2, subtema 2.8) a través del servicio GPRS logrando comunicarse con un servidor web (aplicación servlet).

Toda solicitud es atendida por nuestra aplicación instalada en nuestro servidor de aplicaciones a través de un servlet, el cual a su vez se conecta con DLV y

realiza la búsqueda de la solución al problema modelado y enviado por el usuario de la aplicación móvil.

Figura 4.5 Resultado obtenido y mostrado por DLV móvil

En la figura 4.5 se muestra nuestra aplicación mostrando el resultado obtenido a través de DLV y enviado al dispositivo móvil vía GPRS. Como podemos observar, las dos soluciones halladas por DLV son:

Solo puede suceder una de las dos proposiciones, es decir

"Es de día o es de noche, pero no ambas"

Así, nuestra aplicación permite a los usuarios poder modelar cualquier problema en su dispositivo móvil y en cuestión de 30 segundos tener la solución lista en nuestro dispositivo.

4.3 Arquitectura de DLV móvil

La arquitectura (figura 4.6) empleada para nuestra aplicación consta de un servidor de aplicaciones (tomcat) en el que tenemos instalada nuestra aplicación servidor. Esta aplicación es la encargada de atender las solicitudes realizadas por los usuarios de "lógica móvil" y tiene como objetivo lanzar a

ejecución el sistema DLV con el correspondiente programa recibido del usuario móvil. Una ves obtenido el resultado del sistema DLV, esta es enviada al usuario móvil a través de un servlet responsable de la comunicación GPRS.

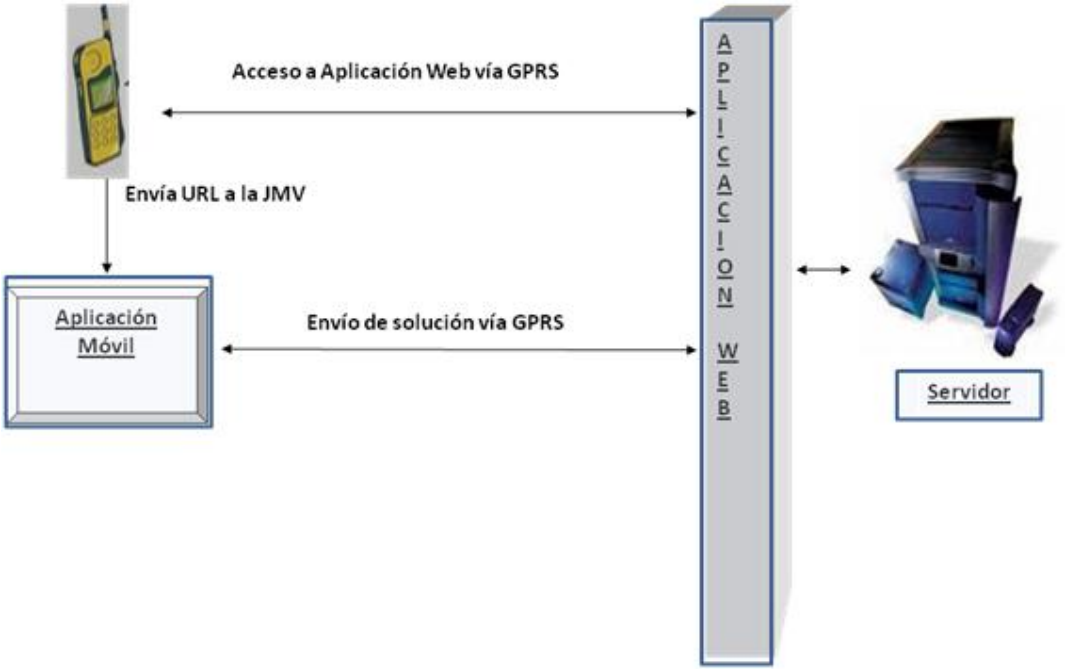

Figura 4.6 Arquitectura general de lógica móvil

Este tipo de arquitectura nos da beneficios significativos que hacen que nuestra aplicación sea atractiva y eficiente.

1. El costo por solicitud en promedio es de 40 centavos
2. El tiempo por solicitud es de 20 a 30 segundos
3. El tamaño del problema no es limitado
4. Solo se requiere de la activación del servicio GPRS (el cual no tiene costo)
5. Disponibilidad permanente y en cualquier lugar

4.4 Especificaciones Técnicas:

Para el funcionamiento de nuestra aplicación móvil, el dispositivo móvil que se utiliza deberá contar como mínimo con las siguientes características:

APIs	
Tecnología Java	
JSR 139 Connected, Limited Device Configuration (CLDC) 1.0	
JSR 118 MIDP 2.0	
JSR 185 Java™ Technology for Wireless Industry	
JSR 172 J2ME™ Web Services Specification	
Navegador	
Detalles del Navegador:	WAP 2.0 (con GPRS activado)
Memoria	
Espacio libre en memoria mínimo:	1 Mb
Tamaño máximo de archives JAR:	1 Mb

4.5 Desarrollo de aplicación WEB

Nuestra aplicación Web (Servlet) fue desarrollada bajo la plataforma Netbeans (véase Capitulo 2) el cual nos da las herramientas necesarias para su construcción, conexión y ejecución.

Cuando se habla de aplicaciones para Web comúnmente escuchamos hablar de JAVA, y con ello sus aplicaciones más conocidas, los Applets, que son programas que se pueden cargar a través de una red y que se ejecutan de igual forma en cualquier plataforma, todo ello gracias a las potentes características de JAVA. Hasta hace poco, JAVA se utilizaba básicamente para dotar a las páginas WEB de una mayor interactividad mediante los Applets, y por tanto solo actuaba sobre el lado cliente. Pero el lado servidor también puede beneficiarse de todas las ventajas que ofrece JAVA, gracias a los *Servlets*.

Los *Servlets* se diferencian de los Applets básicamente en que se ejecutan en el servidor y en que no presentan ningún tipo de interfaz gráfica, puesto que se encargan de hacer el trabajo oculto.

Los servlets son programas que atienden peticiones de un cliente teniendo al servidor como el encargado, pero escritos en Java y con la ventaja de explotar todas las bondades de java. Por ejemplo, un servlet puede ser responsable de tomar los datos de un formulario HTML y enviarlos a una base de datos para actualización de la misma.

4.6 Concepto de Servlet

La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor.

El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

4.7 Ventajas de los Servlets

Las ventajas de los Servlets ante los CGI son:

Desempeño: Los servlets son más rápidos que los CGI debido a que utilizan *threads* en lugar de *procesos*.

Una de las quejas más comunes de los CGI/Perl es el desperdicio de los recursos del sistema. En un ambiente típico CGI, cada petición crea un nuevo proceso el cual carga el interprete de Perl. El intérprete de Perl carga el script CGI, lo compila, y lo ejecuta. Además, si la aplicación se comunica con la BD, una nueva conexión será necesaria para ser hecha por cada CGI.

Los Servlets inician un nuevo thread, *hilo*, (máss que un nuevo proceso) con cada petición. Cada servlet es cargado una vez y usado más y más. Nota que a diferencia de los CGI, los servlets requieren que se cuente con una Máquina

Virtual de Java (JVM) corriendo sobre el servidor todo el tiempo. Para sitios ocupados, ésta permite a los servlets usar mucho menos los recursos del sistema e incrementar el desempeño.

Portabilidad: Los servlets son tan portables como cualquier otra aplicación de Java. Muchos sitios están buscando una solución de portabilidad. Tal vez desearían para su desarrollo usar Linux o Windows NT mientras su servidor de producción corra bajo Solaris. Ellos pueden querer vender su aplicación Web para muchos clientes como para plataformas sea posible. Hay versiones de Perl para muchas variantes de Unix como para Windows 95 y NT. Mientras que Perl para Unix no es ciento por ciento compatible con el Perl para NT, los CGIs escritos en Perl pueden ser escritos para trabajar sobre ambos. El llevar CGIs a Windows NT no es difícil de hacer, pero nunca funcionarán sin realizar al menos una modificación. Por otro lado, los CGI pueden contener comandos de Unix que no trabajarán sobre Windows NT.

La portabilidad con los servlets de Java es más simple. Java fue diseñado para ser portable a través de todas las plataformas, permitiendo que las aplicaciones sean movidas fácilmente de un sistema operativo a otro.

Seguridad: La seguridad puede ser un problema cuando se desarrollan CGIs. El área de mas preocupación es el proceso de estrada del usuario. Este podría ser desde formas o desde datos dentro de un URL. Muchos CGIs escritos en Perl son vulnerables a ataques donde la finalidad del usuario es trucar el CGI ejecutando algún comando sobre el servidor. Los Servlets no están en riesgo de correr comando de shell no planeados. Los lenguajes compilados como Java (o C) proveen mejor seguridad que los lenguajes que interpretan scripts.

Los Servlets son archivos de clases compilados mientras que un CGI/Perl es manipulado en su forma de código fuente. Dependiendo quien tenga acceso al servidor Web, se puede elegir entre instalar o no el código fuente.

Desarrollo: El desarrollo con Perl es simple. Lo único que se necesita es un editor y mucha documentación de Perl.. Mientras que a muchos desarrolladores les agrada este ambiente, hay una creciente proliferación de

herramientas IDE (Integrated Development Environment) para desarrollo. En ese campo, Java por encima de Perl. Hay IDEs disponibles desde Sun, Microsoft, Symantec y muchas otras, proporcionando simples entornos GUI. Las IDEs pueden proveer útiles herramientas fáciles de aprender para nuevos desarrolladores.

Compartiendo el trabajo entre los desarrolladores es además es más fácil con Java, o puede asignarse en paquetes individuales. Cuando se trabaja con una aplicación CGI/Perl donde dos o mas desarrolladores estuvieron trabajando simultáneamente, la aplicación normalmente es manejada de un desarrollador a otro. El desarrollo de aplicaciones Java es mucho mejor para proyectos largos.

4.8 Arquitectura de los Servlets

El principal componente de la Servlet API es la interfaz Servlet. Todos los servlets implementan esta interfaz directamente, por medio de la extensión de la clase que la implementa, HttpServlet. Esta interfaz está provista de métodos que manipulan a los servlets y la comunicación con sus clientes. Ésta clase contiene una serie de métodos predefinidos, que son invocados en respuesta a eventos:

1. Inicialización. Se carga en memoria una instancia del servlet. La instanciación se produce la primera vez que el servlet es cargado en memoria por parte del servidor de aplicaciones. Como consecuencia de este evento se ejecuta el siguiente método:

public void init(ServletConfig config) throws ServletException

2. Petición: el cliente realiza la petición al servlet. Dentro de la versión 1.1 y en función del método de invocación usado en HTML (DO o GET) se ejecutará uno de los siguientes:

public void doGet(HttpServletRequest peticion, HttpServletResponse respuesta) throws ServletException, IOException

public void doPost(HttpServletRequest peticion, HttpServletResponse respuesta) throws ServletException, IOException

El objeto 'peticion' es enviado por la JVM del servidor de aplicaciones y representa (encapsula) información de la petición HTTP. Más adelante veremos para que puede servir el objeto 'respuesta', por de pronto nos sirve decir que en este objeto se encapsulan los servicios para generar la salida. También tiene la posibilidad de usar:

public void service(HttpServletRequest peticion, HttpServletResponse respuesta) throws ServletException, IOException

3. Destrucción del servlet: la contraimagen de *init()*, el servlet va a ser destruido (normalmente por la detención del servidor). No se invoca cuando se termina la petición, sino cuando se va a descargar el servlet de memoria:

public void destroy()

Cuando un servlet es llamado desde un cliente, este recibe dos objetos: ServletRequest y ServletResponse. La interfaz ServletRequest se encarga la comunicación desde el cliente al servidor, mientras que la interfaz ServletResponse atiende la comunicación desde servlet al cliente.

La interfaz ServletRequest permite al servlet acceder a información como, los nombres de parámetros pasados por el cliente, el protocolo usado por el cliente, y los nombres de los host remotos que hacen la solicitud y el servidor que la recibe. Est interfaz permite a los servlets el acceso a métodos que permiten manejar la presentación de la respuesta como salida en el navegador, a través de los cuales consiguen los datos desde el cliente que usa protocolos como HTTP POST , etc..

La interfaz ServletResponse proporciona al servlet los métodos para contestarle al cliente. Permite al servlet configurar la forma de salida de los datos para el cliente, ServletOutputStream que permite enviar la replica de datos como respuesta. La subclases de ServletResponse le dan más capacidad al servlet para responder.

Por ejemplo:

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;


public class ObjetosAprendizaje extends HttpServlet{

   PrintWriter out;

   public void init(){   } // motor del servlet para cargar la petición


public void doGet(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{

 try{

       String tipo=req.getParameter("formato ");

       String busqueda=req.getParameter("busqueda");

       res.setContentType("text/html");

         PrintWriter out = response.getWriter();

          out.println("Busqueda:" + busqueda + "\n Formato: "+formato);

            out.close();


   } catch(Exception ee){out.write(ee.toString());}


Public void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{


try{
```

```
      doGet(request, response);
    }catch (Exception e) { e.printStackTrace(); }
 }

Public void destroy(){} //destruimos y liberamos los recursos abiertos
}
```

4.9 Estructura básica de un Servlet

Tanto doGet como doPost reciben como parámetros objetos de tipo HttpServletRequest y HttpServletResponse.

De la clase HttpServletRequest nos interesa el método getParameter(String nombreParametro), que devuelve el valor del parámetro de Búsqueda y Formato. Por ejemplo, si visitáramos la URL

http://localhost:8080/mi_servlet/OAm?formato=MP3&busqueda=REDES

(vemos que es una petición GET, porque los parámetros se pasan en la URL) el servlet "ObjetosAprendizaje" (*Figura 3.1*) podría llamar a getParameter("formato") para obtener el Formato o getParameter("Busqueda") para obtener la Búsqueda.

Las clases e interfaces descritas conforman a un servlet básico. Pero existen métodos adicionales que provee la API con la capacidad para controlar sesiones o múltiples conexiones, entre muchas más aplicaciones

El ciclo de vida de un Servlet se divide en los siguientes puntos:

1. El cliente solicita una petición a un servidor vía URL.
2. El servidor recibe la petición.
    1. Si es la primera, se utiliza el motor de Servlets para cargarlo y se llama al método init().

2. Si ya está iniciado, cualquier petición se convierte en un nuevo hilo. Un Servlet puede manejar múltiples peticiones de clientes.

3. Se llama al método service() para procesar la petición devolviendo el resultado al cliente.

4. Cuando se apaga el motor de un Servlet se llama al método destroy(), que lo destruye y libera los recursos abiertos.

4.10 Ejemplos de aplicación de los servlets

Algunas de las aplicaciones de servlets incluyen:

Procesamiento de datos enviados con HTTPS a partir de una forma HTML, incluyendo la orden de compra o datos de una tarjeta de crédito. Un servlet como este podría ser parte de una orden de entrada al sistema de procesamiento de datos, haciendo la respectiva actualización en la base de datos, y quizá hasta un sistema de pago en línea.

Interacción múltiple entre personas. Un servlet puede manipular múltiples peticiones al mismo tiempo; podrían sincronizar dichas solicitudes de sistemas que dan soporte, por ejemplo conferencias en línea.

Redireccionamiento de peticiones. Los servlets pueden reenviar las peticiones a otros servidores y servlets. Esto permite que al ser usados se balancee la carga entre varios servidores que atiendan al misma tarea, según el tipo de proceso que deban atender.

Dado que pueden manejar múltiples peticiones en forma concurrente, es posible implementar aplicaciones de colaboración, como por ejemplo una aplicación de videoconferencia.

CONCLUSIONES

A lo largo de esta tesis, hemos presentado una serie de nuevas tecnologías de la información y comunicación que están revolucionando muchas de las áreas que cotidianamente hemos cultivado. De manera particular, en el presente trabajo nos hemos dado a la tarea de desarrollar un sistema en una de las áreas más prolíferas en la investigación de ciencia básica, programación lógica disyuntiva de nuestros tiempos incorporando para esto las tecnologías móviles.

Hemos logrado integrar un conjunto de herramientas actuales que nos permiten desarrollar y llevar el conocimiento a los usuarios de teléfonos móviles de una manera sencilla. La tecnología NetBeans nos permite incorporar un ambiente de desarrollo para dispositivos móviles, particularmente teléfonos móviles, permitiéndonos llevar DLV a las manos de los investigadores que lo requieran.

Por otro lado, en el presente trabajo de tesis usamos algunas herramientas gestoras de contenido que nos permitieron desarrollar el portal en internet que nos permitió poder llevar DLV a dispositivos móviles.

Además, hemos presentado una aplicación novedosa que incorpora nuevas características y tecnologías al quehacer científico y académico actual. Esta nueva propuesta ha cambiado radicalmente el comportamiento humano, haciendo que las tecnologías móviles sean parte de nuestro atuendo cotidiano. Uno de los principales beneficios que nuestro sistema da al investigador es: contar con el beneficio de tener DLV en todo momento, en cualquier lugar y en “casi” cualquier dispositivo móvil.

De esta manera hemos demostrado la viabilidad de llevar a la mano de los científicos, académicos y aprendices la poderosa herramienta llamada “DLV”, que sin lugar a dudas hará que la lógica sea más ampliamente aceptada por los aprendices e investigadores del área.

REFERENCIAS

[1] Jamie Jaworski, “Java 2 Platform Unleashed”, ISBN:0672316315, 2000

[2] John W. Muchow, “Core J2ME Technology, The Sun Microsystems Press”, Java Series, 2002.

[3] Adam Myatt, “Pro NetBeans IDE 5.5”, Enterprise Edition, 2002.

[4] F. Zacarias, F. Lozano, R. Cuapa and A. Vázquez, “English’s Teaching based on New Technologies”. The International Journal of Technology, Knowledge & Society, Northeastern University in Boston, Massachussetts, USA. ISSN: 1832-3669, Common Ground Publishing, USA 2008.

[5] F. Zacarias, F. Lozano, R. Cuapa, A. Vázquez and D. Zacarias, u-Teacher: “Ubiquitous Learning Approach Springer’s Lecture Notes of Computer Science (LNCS)” ISSN: 0302-9743, ISBN: 3-540-20965-4. Vol. 5093, Nanjin, China 2008.

[6] F. Zacarias, A. Sánchez J., D. Zacarías, A. Méndez, R. Cuapa, “Financial Mobile System based on Intelligent Agents”. Austrian Computer Society book series, 2006.

[7] Sergio Gálvez Rojas, Lucas Ortega Díaz. “Java a Tope: j2me (java 2 micro edition).Edición Electrónica“ Depósito Legal: MA-1769-2003 ISBN: 84-688-4704-6. Universidad de Málaga. 2003

[8] Manuel J. Prieto, “Introducción a J2ME (Java 2 MicroEdition)”, 2008

[9] Jackwind Li Guojie, “Build your stock with J2ME”, developerWorks, your source for great tutorials: http://www.ibm.com/developerworks/

[10] Michael Morrison, “Wireless Java with J2ME in 21Days”, Edit: SAMS, International Standard Book Number: 0-672-32142-4, 201 West 103rd St., Indianapolis, Indiana, 46290 USA, 2007

[11] Mario Jaramillo Restrepo,”The Mobile Information Device Profile and MIDLets, Part 5” Ingeniería de Sistemas-Universidad de Antioquia. Mayo – 2006

[12] Desmond Keegan, “The future on Learning: form e-learning to m-learning”. FernUniversitat.

[13] Konstantinov Andrey, mjSoftware, http://www.mjsoft.nm.ru, 2007

[14] Advanced Distributed Learning, “SCORM® 2004 4th Edition

Run-Time Environment (RTE) Version 1.1". Versión en línea: http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/SCORM%202004%204th%20Ed%20V1.1/Documentation%20Suite/SCORM_2004_4ED_v1_1_Doc_Suite.zip, 2001

[15] Advanced Distributed Learning: "SCORM® 2004 4th Edition Content Aggregation Model (CAM) Version 1.1". Versión en línea: http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/SCORM%202004%204th%20Ed%20V1.1/Documentation%20Suite/SCORM_2004_4ED_v1_1_Doc_Suite.zip, 2009

[16] Álvarez G. Luís A.- Espinoza, Daniela - Duarte, Mauricio.: "Repositorio de Objetos de Aprendizaje Multimediales basados en el estándar SCORM ® 2004". VIII Congreso de Nuevas Tecnologías y Aplicaciones Informáticas. La Habana – Cuba. 9 al 13 de mayo del 2005. Versión en línea. http://www.inf.uach.cl/lalvarez/publicaciones.htm, 2004

[17] Álvarez G. Luís A.- Mónica del C., Gallardo G.: "Diseño de un Repositorio de Objetos de Apoyo al Aprendizaje Colaborativo". CISCI 2004. Orlando - USA. 21 y 25 de julio del 2004. Versión en línea. http://www.iiisci.org/journal/CV$/risci/pdfs/P157490.pdf, 2004

[18] Álvarez G. Luís A.- Mónica del C., Gallardo G.: "Repositorio de objetos de Apoyo al aprendizaje Colaborativo". TISE '03. Santiago - Chile. 26, 27 y 28 de noviembre del 2003. Versión en línea. http://www.tise.cl/archivos/tise2003/papers/repositorio_de_objetos.pdf , 2003

[19] "Draft Standard for Learning Object Metadata. IEEE 1484".12.1-2002, 15 July 2002. Versión en línea. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf, 2002

[20] Merrill, M.D., Li, Z. & Jones, M. "Instructional transaction theory: An introduction. Educational Technology", Versión en línea. http://www.id2.usu.edu/Papers/ITT_Intro.PDF, 1991

[21] Wiley, D. A. "Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy". In D. A. Wiley (Ed.), The Instructional Use of Learning Objects: Versión en línea. desde: http://reusability.org/read/chapters/wiley.doc, 2000

[22] Quinn, C. "mLearning: Mobile, Wireless, In-Your-Pocket Learning". Line zine. Learning in the new economy. http://www.linezine.com/2.1/features/cqmmwiyp.htm, 2000

[23] Malinen, J., Kari, H. & Tiusanen, M. “Wireless networks and their impact on network-based learning content. Enabling network-based learning”. Helsinki University of Technology. http://firgoa.usc.es/drupal/node/5560 , 1999

[24] Gallardo G., Mónica: “Diseño y Construcción de un Repositorio de Herramientas de Apoyo a la Enseñanza/Aprendizaje Virtual”. Tesis de grado para optar al Título Profesional de Ingeniero Civil en Informática. Universidad Austral de Chile. Septiembre 2006.