ESCUELA POLITECNICA DEL EJÉRCITO

SEDE LATACUNGA

FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA

DESARROLLO E IMPLEMENTACION DEL PORTAL WAP-WML PARA LA ESCUELA POLITECNICA DEL EJERCITO SEDE LATACUNGA, APLICANDO LA METODOLOGIA XPOOHD-WAP

PROYECTO PREVIO A LA OBTENCION DEL TITULO DE INGENIERO DE SISTEMAS E INFORMATICA

DARWIN VINICIO HERRERA VEINTIMILLA

Latacunga, Abril 2008

DEDICATORIA

El siguiente proyecto de investigación y desarrollo lo dedico a Dios por concederme la salud y fortaleza de iniciar y concluir este proyecto, a mi abnegada madre Amelia que con su esmero y ejemplo me inculcó los principio y valores fundamentales en mi vida y mi querida esposa la cual es la base de mi hogar y que supieron mantener vivo su amor paciencia y apoyo en todo momento.

AGRADECIMIENTO

A mi madre que jamás dejo de ser el ejemplo de tesón y anhelo en mi vida, a mi padre a pesar de las adversidades supo seguirme apoyando, a mi esposa y amiga que con su apoyo más allá de todo lo pensado junto a mi lado ha sido uno de los pilares fundamentales para llegar a culminar este proyecto. A mis suegros que han sabido comprender y apoyarme en los momentos difíciles, a mi hermano y su esposa que me apoyan moralmente y están presentes cuando más lo he necesitado. Agradezco a mi director y codirector que gracias a su guía he logrado culminar este proyecto el cual representa la culminación de mi carrera universitaria y el inicio de mi vida profesional.

CAPITULO I

1. APLICACIONES INALÁMBRICAS

1.1.- INTRODUCCIÓN

La necesidad de conectarse a Internet de modo inalámbrico ha sido una inquietud de muchos usuarios, y por ende hasta ahora la solución estaba en usar un computador laptop, conectado a un MODEM y a un teléfono celular, pero esta forma de navegar en la web no era muy efectiva debido a los grandes tiempos de respuesta y a las bajas velocidades de transmisión de datos que caracterizan a la red inalámbrica.

El Protocolo de Aplicaciones Inalámbricas WAP, surge como un estándar universal, independiente, abierto y seguro, cuyo objetivo es proporcionar al usuario el acceso a servicios de información iterativa basados en la WEB, así como también servicios de valor agregado a través de aparatos inalámbricos como son el teléfono celular digital, los pagers y los asistentes digitales personales (PDA), que están limitados por el tamaño de su pantalla, la capacidad de memoria, las velocidades de transmisión de datos, el tiempo de duración de las baterías y el CPU.

El WAP en los aparatos inalámbricos como el teléfono celular va a proporcionar la información y servicio en un formato de texto básico, el cual va a permitir que la carga de información sea más rápida. Cabe destacar que WAP está en sus comienzos y se espera que las mejoras en la tecnología de las redes y aparatos inalámbricos le permitan desarrollarse y así en un lapso de tiempo no muy largo poder llegar a recibir contenidos en multimedia. En la figura 1.1 podemos observar una presentación de WAP en un teléfono celular.

Figura 1.1: Presentación de un Teléfono Celular WAP

Fuente: [http://www.cellular.co.za/wap.htm]

En la actualidad, debido al rápido avance en la tecnología es posible que el teléfono celular tenga varias funciones, además de comunicar a través de la voz. Los teléfonos celulares cuentan con varias características, por ejemplo:

Cobertura: La cobertura del sistema se refiere a las zonas geográficas en las que se va a prestar el servicio. La tecnología más apropiada es aquella que permita una máxima cobertura con un mínimo de estaciones base, manteniendo los parámetros de calidad exigidos por las necesidades de los usuarios. La tendencia en cuanto a cobertura de la red es permitir al usuario acceso a los servicios en cualquier lugar, ya sea local, regional, nacional, e incluso mundial, lo que exige acuerdos de interconexión entre diferentes operadoras para extender el servicio a otras áreas de influencia, diferentes a las áreas donde cada red ha sido diseñada.

Capacidad: Se refiere a la cantidad de usuarios que se pueden atender simultáneamente. Es un factor de elevada relevancia, pues gracias al adecuado dimensionamiento de la capacidad del sistema, depende la calidad del servicio que se preste al usuario. Esta capacidad se puede incrementar mediante el uso de técnicas tales como la reutilización de frecuencias, la asignación adaptiva del canal, el control de potencias, saltos de frecuencias, algoritmos de codificación, diversidad de antenas en la estación móvil, etc.

Últimamente los teléfonos celulares poseen funcionalidades que permiten a los usuarios contar con varios servicios por ejemplo: agenda, calculadora, envío

de correo electrónico, alarmas, navegación por Internet, cámara fotográfica, radio FM, transferencia de información a través de USB, transferencia de información a través de infrarrojo, transferencia de información a través de Bluetooth, etc.

El desarrollo de las redes y de los terminales GSM (*Global System for Mobile Communications*), Sistema Global para comunicaciones móviles, formalmente conocida como *Group Special Movile,* Grupo Especial Móvil, permiten un mayor ancho de banda y un ambiente con mayor capacidad en la ejecución, permitiendo el desarrollo de usos móviles.

El mundo tiene cada vez más necesidad de una computadora y las aplicaciones informáticas ahora se utilizan para un número de tareas tales como comunicaciones, gerencia financiera, recuperación de datos y juegos. Es normal para los usuarios que estos usos estén disponibles en su Terminal móvil.

Los progresos iniciales en usos móviles eran básicos, por ejemplo: el funcionamiento en la tarjeta del GSM usando un kit de herramientas de SIM que se interconectaba a través de las terminales capaces y que usaba SMS para comunicarse con la infraestructura del uso.

Más tarde fueron teniendo cabida la introducción de los navegadores que utilizaban los protocolos móviles especiales (WAP, HDML, etc) y las capacidades básicas de los datos de los terminales del GSM del tiempo.

Lo anterior, permite que el usuario utilice su terminal móvil para tener acceso al contenido básico del formato de texto tal como noticias, deporte información, etc.

1.2.- PROTOCOLO DE APLICACIONES INALÁMBRICAS (WAP)

WAP (Wireless Application Protocol o Protocolo de Aplicaciones Inalámbricas) es una solución unificada para los servicios de valor agregado

existentes y futuros para la telefonía móvil. El protocolo incluye funcionalidades de seguridad. WAP también define un entorno de aplicaciones.

Es escalable, permitiendo así a las aplicaciones disponer de las capacidades de pantalla y recursos de red según su necesidad y en una gran variedad de tipos de terminales. Los servicios podrán ser aplicables a pantallas de una sola línea o a terminales mucho más complejas como las PDAs.

A diferencia de las tecnologías de Internet para PCs, WAP está pensado para dispositivos que tienen algunas limitaciones técnicas inherentes a la tecnología actual como son:

- Menor potencia de procesamiento
- Menor capacidad en memoria (ROM-RAM)
- Restricciones de suministro de potencia
- Despliegues pequeños
- Dispositivos de entrada diferentes

1.2.1.- WAP, FUNCIONAMIENTO Y OPERACIÓN:

- El usuario solicita la página WAP que quiera ver.
- El micronavegador del móvil envía la petición con la dirección (URL) de la página solicitada y la información sobre el abonado al Gateway WAP (software capaz de conectarse a la red de telefonía móvil y a Internet).
- El Gateway examina la petición y la envía al servidor donde se encuentra la información solicitada.
- El servidor añade la información http o HTTPS pertinente y envía la información de vuelta al Gateway. En el Gateway se examina la respuesta del servidor, se valida el código WML en busca de errores y se genera la respuesta que se envía al móvil.

- El micronavegador examina la información recibida y si el código es correcto lo muestra en pantalla.

1.2.2.- WAP, MERCADO Y APLICACIONES:

WAP ofrece infinidad de posibilidades, tanto para empresas y profesionales, como para el consumidor: Algunos ejemplos son:

- Agendas corporativas WAP.
- Gestión de pedidos (fuerza de ventas).
- Servicios de localización.
- Gestión de flotas.
- Servicios de mensajería.
- Tiendas virtuales.
- Comercio electrónico móvil.
- Servicios de banca on-line (mobile home-banking, bolsa, ...).
- Venta y reserva de billetes (transportes).
- Información del tiempo, tráfico, horarios, guía turística, entre otros.

1.2.3.- CONSIDERACIONES TÉCNICAS DE WAP

El protocolo de Aplicaciones inalámbricas WAP emplea equivalentes inalámbricos al HTTP y HTML.

En WAP el equivalente para el HTTP viene a ser el WSP (Wireless Session Protocol) y esta basado en HTTP/1.1 y define el protocolo de comunicación entre el dispositivo inalámbrico y un WAP Servidor o un WAP Gateway.

El WSP parte del mismo concepto de un requerimiento y una respuesta, donde cada una va estar conformada por un encabezado (header) y un cuerpo (body).

El WSP permite la transferencia de datos de múltiples partes, por ejemplo cuando un deck específico es solicitado el servidor puede responder con todo el contenido. Esto elimina la necesidad de subsecuentes requerimientos desde el cliente, lo cual contribuye a disminuir los retardos que se generan entre requerimientos y respuestas.

En WAP el equivalente de HTML es el WML (Wireless Markup Language) es el nuevo lenguaje markup que va a permitir mostrar la información e interactuar con el usuario. WML esta basado en HTTP y esta altamente influenciado por el HDML.

Tanto el WSP y WML están altamente enfocados a lo limitado del tamaño de la pantalla y a las bajas velocidades de transmisión de datos.

1.3.- COMPONENTES DE LA ARQUITECTURA (WAP)

La arquitectura WAP está pensada para proporcionar un "entorno escalable y extensible para el desarrollo de aplicaciones para dispositivos de comunicación móvil". Para ello, se define una estructura en capas, en la cual cada capa es accesible por la capa superior así como por otros servicios y aplicaciones a través de un conjunto de interfaces muy bien definidos y especificados. Las capas de la arquitectura WAP se recogen en el siguiente diagrama:

Figura 1. 2: Capas de la arquitectura WAP

Fuente: [http://www.cellular.co.za/wap.htm**]**

1.3.1.1.- Capa de Aplicación (WAE)

Es un entorno de aplicación de propósito general, basado en la combinación del World Wide Web y tecnologías de Comunicaciones Móviles.

Este entorno incluye un micro navegador, que posee las siguientes funcionalidades:

- El lenguaje WML, del cual ya hablaremos posteriormente
- El lenguaje WMLS, similar al JavaScript.
- WTA (Wireless Telephony Applicactions), es un entorno para aplicaciones o servicios de telefonía.
- WTAI (Wireless Telephony Application Interface), es una interfaz utilizada en los terminales móviles para operaciones locales de control de llamadas (recepción, iniciación y terminación) y acceso a listines telefónicos.
- Una serie de formatos de contenido, que son un conjunto de datos definidos, entre los que se encuentran: imágenes, información de calendario.

1.3.1.2.- Capa de sesión (WSP)

Este protocolo proporciona a la Capa de Aplicación (WAE) interfaz con dos servicios de sesión:

- Un servicio orientado a conexión que funciona por encima de la Capa de Transacciones (WTP).
- Un servicio no orientado a conexión que funciona por encima de la Capa de Transporte (WTP), y que proporciona servicio de datagramas seguro o no seguro.

Esta capa proporciona las siguientes funcionalidades:

- Establecimiento y liberación de conexiones entre cliente y servidor.,
- Intercambio de información entre cliente y servidor.
- Negociación de las características del protocolo.
- Suspensión y reanudación de la sesión.

1.3.1.3.- Capa de transacciones (WTP)

Este protocolo funciona por encima de un servicio de datagramas ya sean seguros como no seguros, y proporciona las siguientes funcionalidades:

a) Proporciona los servicios necesarios para soportar las transacciones, estos servicios pueden ser de tres clases:

- Peticiones inseguras de un solo camino
- Peticiones seguras de un solo camino
- Transacciones seguras de dos caminos

b) También proporciona seguridad en las transacciones.

1.3.1.4.- Capa de seguridad (WTLS)

La Capa Inalámbrica de Seguridad de transporte (WTLS) es un protocolo basado en el estándar SSL, utilizado en el entorno Web, para la seguridad en la transferencia de datos, esta capa proporciona a las capas de nivel superior de Wap una interfaz de servicio de transporte seguro, que lo resguarde de una interfaz de transporte inferior. Las funcionalidades de esta capa son las siguientes:

- **Integridad de los datos:** se asegura que la información intercambiada entre el Terminal y el servidor de aplicaciones, no haya sido modificada.

- **Privacidad de los datos:** se asegura que la información intercambiada entre el Terminal y el servidor de aplicaciones, no pueda ser captada ni entendida por elementos externos a la comunicación.
- **Autentificación:** se ofrecen servicios para determinar la autenticidad del terminal y del servidor de aplicaciones.

También puede ser utilizado para el establecimiento de una comunicación segura entre terminales.

1.3.1.5.- Capa de transporte (WDP)

El Protocolo Inalámbrico de Datagramas (WDP) proporciona las siguientes funcionalidades:

- Proporciona un servicio fiable a los protocolos de las capas superiores de WAP.
- Permite la comunicación de forma transparente sobre los protocolos portadores: CDMA, SMS, GSM.

La especificación de WAP es un conjunto de documentos que define la arquitectura general de WAP: Wireless Application Environment que define el Wireless Markup Language, Script Language, y el interfase para Wireless Telephone Application; una capa de Transporte que incluye soporte para el protocolo UDP, el protocolo TCP y el protocolo IP; la capa de Seguridad; y la especificación del protocolo de Sesión y de Transacciones que permite servicios orientados a conexión y no orientados a ella.

1.4.- ARQUITECTURA DE UN SISTEMA WAP

El modo de solicitar información para un usuario de WAP es a través de las URL, por supuesto estas son presentadas al usuario en forma de hipervínculos y donde la información es almacenada previamente y presentada luego.

Lo que el usuario emplea en su teléfono celular es una aplicación cliente compatible con WAP, para hacer solicitudes y visualizar el contenido. Existen dos maneras en que el usuario puede obtener información:

- Proceso de Comunicación entre un Cliente WAP y un Servidor WAP.
- Proceso de Comunicación entre un Cliente WAP, un WAP Gateway y un Servidor de WEB estándar.

A continuación en la figura 1.3, se puede apreciar el proceso de comunicación entre un cliente WAP y un servidor WAP y en la figura 1.4, se puede apreciar el proceso de comunicación entre un Cliente WAP, un WAP Gateway y un Servidor de WEB estándar.

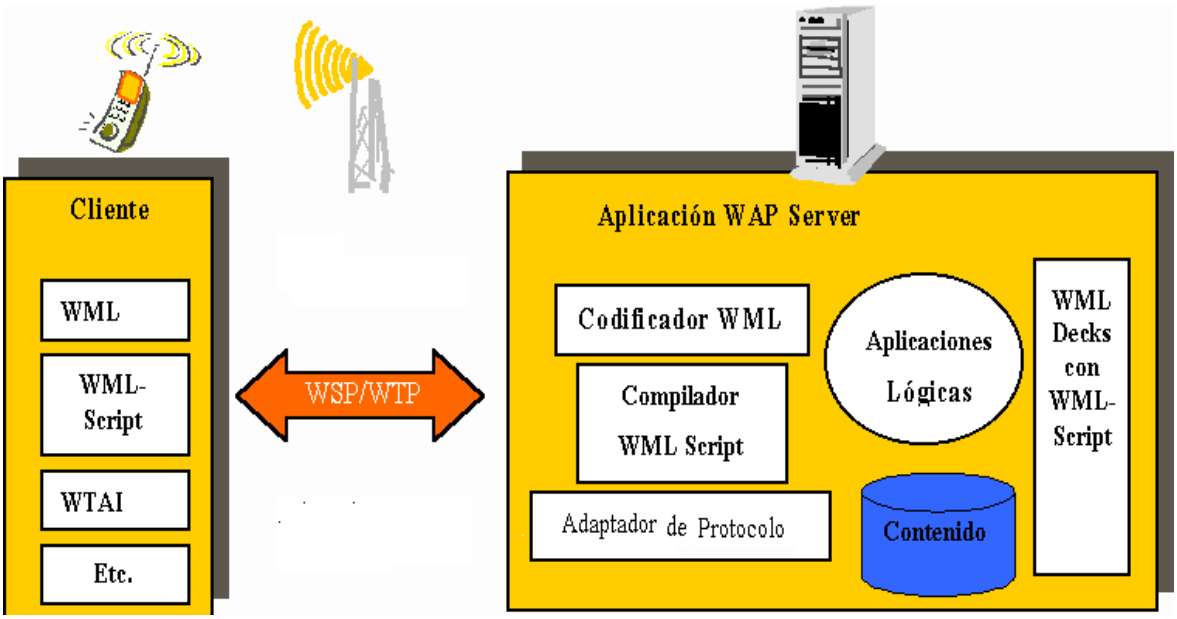


Figura 1.3: Proceso de Comunicación entre un Cliente WAP y un Servidor WAP.

Fuente: [http://www.cellular.co.za/wap.htm]

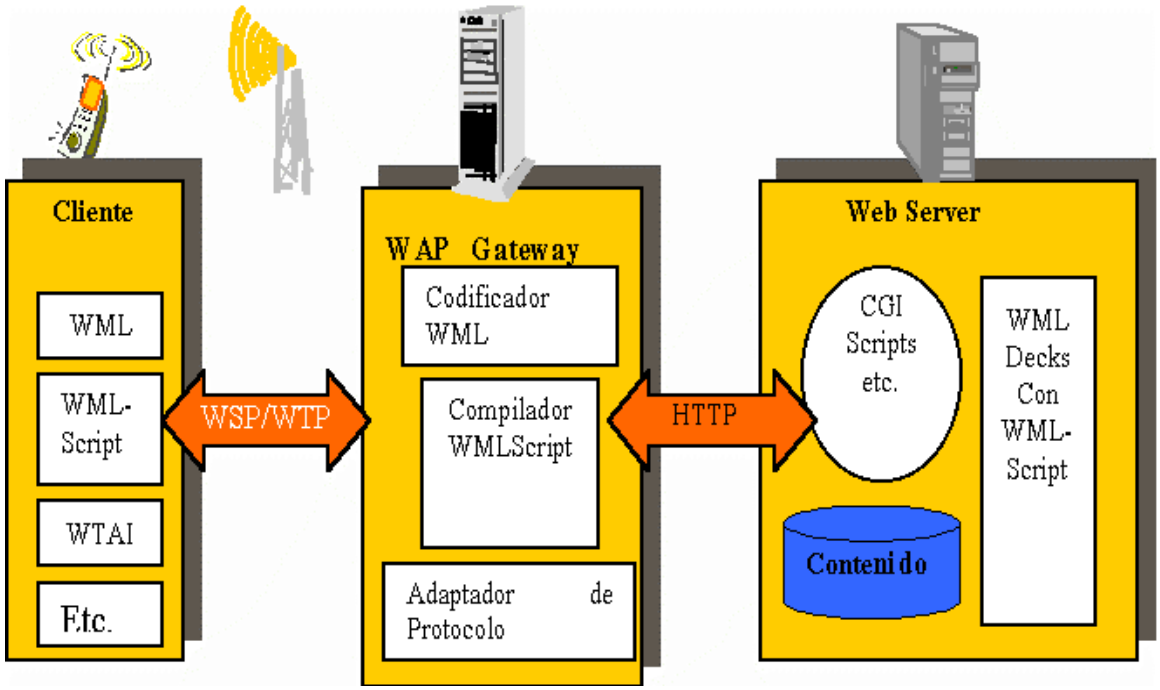


Figura 1.4: Proceso de Comunicación entre un Cliente WAP, un WAP Gateway y un Servidor de WEB estándar.

Fuente: [http://www.cellular.co.za/wap.htm]

1.5.- APLICACIONES DE WAP EN LA TELEFONÍA CELULAR

Las aplicaciones que un teléfono celular puede tener con WAP se debe a WTA (Wireless Telephony Application), la cual junto a WAE (Wireless Application Enviroment) forman la capa más alta de la arquitectura WAP teniendo la función de ser la principal interfaz entre la aplicación cliente, el WML, el WML Script y una interfaz telefónica. WTA va a proporcionar un conjunto de funciones que permiten el control sobre el cliente asumiendo que el aparato inalámbrico es un teléfono.

El WTA esta diseñada principalmente para las redes de los operadores donde una de las consideraciones mayores son la disponibilidad y la seguridad.

A través de WTA en un teléfono celular WAP se pueden tener las siguientes aplicaciones:

- Control de llamadas
- Indicadores de control
- Interfaz con la agenda telefónica
- Navegar en internet/intranet en forma de texto
- Enviar y recibir e-mail
- Realizar comercio electrónico
- Tener acceso a servicios de información y bancarios

Un caso práctico de WTA en un celular WAP, es cuando un usuario desea ir a un restaurante local, el usuario consulta el directorio de las páginas amarillas, donde para hacer la reservación solo debe hacer click en el botón mostrado por la WML card y el teléfono marcará el número telefónico del restaurante para así establecer la llamada.

1.5.1.- TECNOLOGÍAS INALÁMBRICAS QUE SOPORTAN A WAP

Estas tecnologías van a proporcionar los enlaces de datos inalámbricos entre el cliente y un servidor. Entre ellas encontramos las siguientes:

1.5.1.1.- GSM SMS, USSD, C-S Data, GPRS

GSM (siglas derivadas originalmente de Group Speciale Mobile) es tecnología celular desarrollada en Europa considerada como la tecnología celular más madura, con más de 200 millones de usuarios en más de 100 países alrededor del mundo. GSM es un servicio de voz y datos basado en conmutación de circuitos de alta velocidad la cual combina hasta 4 ranuras de tiempo en cada canal de radio.

- Velocidad de transferencia de 9,6 Kbps.
- Tiempo de establecimiento de conexión, de 15 a 30 segundos.
- Pago por tiempo de conexión.

La baja velocidad de transferencia limita la cantidad de servicios que Internet nos ofrece. Por ejemplo, a 9,6 Kbps no se puede navegar por Internet de una manera satisfactoria. Si, además, tenemos en cuenta que estamos pagando por tiempo de conexión, los costos se disparan. La combinación de estos tres factores negativos hace que GSM sea una tecnología mayoritariamente utilizada para la voz y no para los datos.

GPRS (Global Packet Radio Service), es una evolución no traumática de la actual red GSM: no conlleva grandes inversiones y reutiliza parte de las infraestructuras actuales de GSM. Por este motivo, GPRS tendrá, desde sus inicios, la misma cobertura que la actual red GSM. GPRS (Global Packet Radio Service) es una tecnología que subsana las deficiencias de GSM:

- Velocidad de transferencia de hasta 144 Kbps.
- Conexión permanente. Tiempo de establecimiento de conexión inferior al segundo.
- Pago por cantidad de información transmitida, no por tiempo de conexión.

1.5.1.2.- IS-136 R- Data, C-S Data, Packet

IS-136 (conocido también como TIA/EIA-136 o ANSI-136), Interim Standard 136 fue la primera tecnología digital de telefonía celular. IS-136. Es un estándar de telefonía celular digital también basado en TDMA. En realidad este es la versión mejorado del IS-54, primer sistema TDMA en los Estados Unidos.

El estándar IS-136 puede operar automáticamente en cualquiera de las dos bandas 800 y 1900 Mhz (dual band) y permite el acceso a la red analógica y digital (dual mode).

1.5.1.3.- CDMA SMS

CDMA (Code División Múltiple Access). CDMA ofrece una serie de beneficios incluyendo mejor calidad de servicios y operación en forma dual band y dual mode.

1.5.1.4.- PDC C-S Data, Packet

Personal Digital Cellular (PDC) es uno de los tres principales estándares inalámbricos digitales del mundo, junto con GSM y TDMA.

Aunque PDC actualmente sólo se utiliza en Japón, es el segundo estándar digital más grande del mundo con más de 48 millones de suscriptores en julio de 2000; los operadores de otras regiones del mundo se están planteando seriamente el uso de PDC. Al igual que GSM, PDC se basa en la tecnología TDMA.

1.5.1.5.- CDPD

CDPD es la abreviatura de "Cellular Digital Packet Data", que corresponde a la tecnología de transmisión de datos, que permite el fraccionamiento de los mensajes de información, en una serie de paquetes, para ser enviados a través de canales celulares dedicados o disponibles en la red celular de voz, a velocidades de hasta 19.200 bits por segundo (bps).

Un sistema CDPD se basa en la comunicación de datos de bajo tráfico. Esta tecnología en combinación con otras conforma toda una plataforma de red móvil de datos. La red CDPD en su funcionamiento regular envía paquetes de datos a usuarios móviles sobre los canales de la red celular utilizando el mismo espectro de radio del sistema telefónico celular y puede utilizar los mismos diseños de ingeniería de radio que el sistema telefónico celular, así mismo es

posible que empresas de servicios adopten estándares de diseños propios de cualquier compañía en función de mantener la exclusividad.

1.5.1.6.- Data TAC

El Sistema Inalámbrico de Transmisión de Datos de Motorota, denominado DATA TAC, es una red inalámbrica dedicada que permite intercambio de datos en tiempo real, ideal para zonas que no permiten instalaciones físicas y/o fijas. Gracias a su funcionalidad tipo computadora más movilidad, provee conectividad entre aplicaciones de servidor fijo y aplicaciones en terminales de usuarios portátiles y móviles, optimizando los canales de comunicación.

1.5.1.7.- FLEX y REFLEX

FLEX un protocolo de localización de alta velocidad desarrollado por Motorola. FLEX aumenta la capacidad de los canales de localización de 200,000 a 600,000 localizaciones por canal, y funciona a 1600, 3200 y 6400 baudios.

El protocolo ReFLEX es el número uno en lo que se refiere a mensajes alfanuméricos de dos vías. Y es además miembro de la familia de protocolos FLEX de Motorota. El mismo aumenta la utilidad de los sistemas inalámbricos de mensajería tradicionales adicionando un canal de respuesta. Permitiendo así a los operadores el poder ofrecer confirmación de mensaje enviado y un servicio de mensajes de dos vías, incrementando la utilización del sistema reutilizando la frecuencia. Los dispositivos de mensajería ReFLEX utilizan el canal de respuesta no solo para confirmar los mensajes enviados sino también para información general de localización.

CAPITULO II

2. LENGUAJE DE MARCAS PARA INALÁMBRICOS (WML)

2.1.- INTRODUCCION

El WML es un lenguaje de marcas basado en el lenguaje de marcación extensible (XML, *Extensible Markup Language*) y fue desarrollado para especificar contenidos e interfaces de usuario para terminales de banda estrecha tales como teléfonos móviles e intérpretes de páginas.

El WML está diseñado para trabajar con dispositivos inalámbricos pequeños que poseen cuatro características:

- Pantalla pequeña de baja resolución. Por ejemplo, la mayoría de los teléfonos móviles pueden sólo mostrar unas pocas líneas de texto, y cada una de ellas puede contener solamente de 8 a 12 caracteres.
- Los aparatos de entrada tienen una capacidad limitada, o están diseñados para un propósito determinado. Un teléfono móvil tiene comúnmente teclas numéricas y un reducido número de teclas adicionales con funciones especificas. Dispositivos más sofisticados pueden poseer teclas programables de software, pero no un ratón ni otros dispositivos de selección.
- Los recursos computacionales están limitados por una CPU de baja potencia, una memoria reducida y una potencia restringida.
- La red ofrece un reducido ancho de banda y una alta latencia. No son infrecuentes los aparatos con conexiones de red de 300 bit/s a 10 Kbit/s y latencia "*round-trip*" de entre 5 y 10 segundos.

Las características del lenguaje WML pueden agruparse en cuatro áreas principales:

- El WML ofrece un soporte de texto e imagen y tiene una amplia variedad de formatos y comandos.
- Las cartas WML se agrupan en barajas. Una baraja WML es similar a una página HTML identificada por un URL (*Uniform Resource Locator*, localizador de recursos uniforme) y es la unidad básica de transmisión de contenidos.
- El WML ofrece soporte para gestión de navegación entre cartas y barajas, e incluye comandos para su manejo. Estos pueden usarse para navegar o ejecutar "scripts". El WML también provee de conexiones de anclaje similares a las usadas en el HTML versión 4.
- Se pueden establecer parámetros para todas las barajas de WML usando un modelo establecido. Se pueden usar variables en lugar de cadenas y sustituirse en el tiempo de ejecución. Esta forma de establecer parámetros permite que los recursos de la red sean usados de forma eficiente.

Toda la información de WML se transmite en formato codificado por la red inalámbrica.

2.2.- TIPOS DE DATOS EN EL NUCLEO DEL WML

Se describe los tipos de datos que acepta el núcleo de WML para WAP. Estos tipos de datos se utilizan en las descripciones de elementos de WML a lo largo de todo este capítulo.

2.2.1.- TIPOS DE CARACTERES

Todos los tipos de caracteres de WML se definen en términos de tipos de datos de XML. Los tipos de datos de caracteres se resumen en la tabla 2.1:

Tabla 2.1: Tipos de datos de Caracteres

Tipos de Datos	Explicación
CDATA	Texto que puede contener caracteres alfanuméricos. *Cdata* Sólo puede desempeñar la función de valor de un atributo. Ejemplos. “$(value)” name=”value”
PCDATA	CDATA analítico. Texto que puede contener caracteres alfanuméricos. Este texto también puede contener etiquetas. *PCDATA* se usa sólo entre elementos. Ejemplo: Text written <big> I CAPS </big> (Texto escrito en mayúsculas)
NMTOKEN	Un testigo de nombre, que contiene cualquier combinación de números, letras y algunos caracteres de puntuación ". “, " - ", " _ ", y “: ". Así, datos de este tipo pueden comenzar con signos de puntuación. Un NMTOKEN no puede usarse en referencias a variables ni en nombres de elementos. Ejemplos: “text” _card 1 a.name.token a-perfectly-valid.name.token
Id	Identificador del elemento, que además es único. Ejemplo: <card id=”card1”/>

2.2.2.- LONGITUD

El tipo ***%length*** puede ser especificado bien como un entero que represente el número de píxeles de la pantalla, o bien como un porcentaje de espacio horizontal o vertical disponible. De este modo, el valor "50" significa 50 píxeles.

Para la anchura, el valor "50%" significa la mitad del espacio horizontal disponible. Y para la altura, el valor "50%" significa la mitad de espacio vertical disponible.

El valor entero consiste en uno o más dígitos decimales ([0-9]) seguido por un carácter porcentual opcional (%). El tipo *length* sólo se emplea como valor de un atributo.

Tabla 2.2: Tipo de dato length

Nombre	Tipo	Utilización
%length	CDATA	[0-9] para píxeles o [0-9] + % para porcentaje de tamaño

Ejemplo 2.1:

Los valores de los atributos ***hspace*** y ***vspace*** son *%length*:

```
<img src="bitmaps/moon.wbmp" hspace="1" vspace="1"/>
<IMG SRC="BITMAPS/MOON.WBMP" HSPACE="2%" VSPACE="2%"/>
```

2.2.3.- VDATA

El tipo *%vdata* representa una cadena de caracteres (*string*) que puede contener referencias a variables.

Este tipo se usa solo en valores de atributos.

Tabla 2.3: Tipo de dato vdata

Nombre	Tipo	Utilización
%vdata	CDATA	Valor de atributo que puede contener referencias a variables

Ejemplo 2.2:

```
<card id="card1" title="$(showme)">
```

2.2.4.- FLOW, INLINE Y LAYOUT

El tipo ***%flow*** (flujo) representa la información a "nivel de carta" y el tipo ***%inline*** (en línea) representa a "nivel de texto".

En general, el **%flow** se utiliza en cualquier lugar en el que se puede incluir una marca general.

El tipo **%inline** indica las áreas donde se maneja texto o referencias a variables.

Tabla 2.4: Tipo de datos Flow, Inline Y Layout

Nombre	Tipo	Utilización
%layout	Br	Disposición del texto, tales como interrupciones de renglón.
%inline	%text %layout	Indica las áreas donde se maneja solo texto o Referencias a variables.
%flow	%inline Img	Cubre elementos a nivel de carta, tales como texto o imágenes.

Ejemplo 2.3:

Los tipos de datos **%flow** y **%inline** se usan para texto general que puede tener atributos de formato tales como itálica, subrayado y negrita.

An emphasized line.
<big>
A big and emphasized line.
</big>

A line with no text formatting.

2.2.5.- TEXT

El tipo ***%text*** (texto) puede incluir las siguientes entidades:

Tabla 2.5: Tipo de dato text

Nombre	Tipo	Utilización
%text	#PCDATA %emph	Indica texto que contiene formato.

Ejemplo 2.4:

A line with plain text.

A line with strong emphasis.

2.2.6.- HREF

El tipo %**href** se refiere tanto a un identificador de recurso uniforme (URI, Uniform Resource Identifier) absoluto como a uno relativo.

Tabla 2.6: Tipo de dato href

Nombre	Tipo	Utilización
%href	%vdata	URI, URL (localizador de recursos uniforme), o de diseño de nudos de hipertexto. Puede contener referencias a variables.

Ejemplo 2.5:

<go href="http://wapforum.org/"/>

<go href="file:///d:\dir\file.wml"/>

<go href="app.wml"/>

Los valores de los eventos intrínsecos son URL:

<card onenterforward="#card2"/>

El atributo *src* de un elemento *img* es un URL:

2.2.7.- BOOLEAN

El tipo **%boolean** se refiere a valores lógicos de verdadero o falso.

Tabla 2.7: Tipo de dato boolean

Nombre	Tipo	Utilización
%Boolean	true, false	Valores lógicos de verdadero o falso.

Ejemplo 2.6:

```
<card newcontext="true"/>
<do optional="true" type="accept"/>
```

2.2.8.- NUMBER

El tipo **%number** (numero) representa un valor entero mayor o igual a cero.

Tabla 2.8: Tipo de dato number

Nombre	Tipo	Utilización
%number	NMTOKEN	Un número entre [0-9]

Ejemplo 2.7:

```
<select tabindex="2"/>
<input name="setvar" size="4" maxlength="20" tabindex="3"/>
```

2.2.9.- EMPHASIS

El tipo **%emph** representa etiquetas de formato de texto descritas en la tabla 2.9:

Tabla 2.9: Tipo de dato Emphasis

Nombre	Tipo	Utilización
%emph	Em, strong, b, I, u, big, small	Formateo de texto, por ejemplo itálico o subrayado.

Ejemplo 2.8:

```
<em>
        An emphasized line.
        <big>
                A big and emphasized line.
        </big>
</em>
```

2.3.- SINTAXIS DE WML

WML hereda la mayoría de sus construcciones sintácticas de XML.

2.3.1.- ENTIDADES

El texto WML puede contener entidades de caracteres numéricos o de nombre las cuales especifican caracteres específicos en el juego de caracteres de documento.

Estas entidades especiales se utilizan para especificar caracteres que deben ser solventados en WML o que pueden ser difíciles de introducir en un editor de texto.

Por ejemplo, el ampersand (&) se representa por la entidad de nombre & .Todas las entidades empiezan con un ampersand y terminan con un punto y seguido.

WML es un lenguaje XML. Esto implica que los caracteres ampersand y “menor que” deben ser simulados cuando se utilicen en datos de texto, es decir, esos caracteres pueden aparecer en su forma literal solo cuando se utilicen como delimitadores de marca, dentro de un comentario, etc.

Ejemplo 2.9:

```
<card id="card1">
        <p>
                Ampersand = &amp;amp; <br/>
                Quote = &amp;quot; <br/>
                Less than = &amp;lt; <br/>
        </p>
</card>
```

La baraja genera el siguiente interfaz de usuario en el terminal de destino:

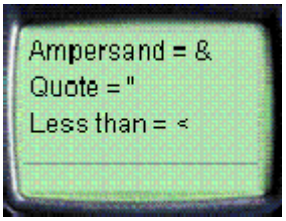


Figura 2.1. Ejemplo de entidades.

Fuente: [www.wapeton.com]

2.3.2.- ETIQUETAS

Una etiqueta (tag) es un descriptor de elementos del lenguaje. Una etiqueta describe un elemento y contiene un nombre de tipo de elemento y un identificador único. Una etiqueta podría también incluir atributos describiendo otras propiedades.

El código WML consiste en un conjunto de etiquetas, cada una de ellas encerrada en un par de corchetes, < y >.

<tag>Comienzo de elemento. La etiqueta de comienzo puede contener atributos.

</tag>Etiqueta que indica el fin de un elemento.

<tag/> Indica un elemento vacío, por ejemplo
, que indica una salto de línea.

2.3.3.- ELEMENTOS

Los elementos especifican todas las marcas e información estructural de una baraja de WML. Los elementos pueden contener una etiqueta de comienzo, contenido, otros elementos y una etiqueta de final. Los elementos tienen una o dos estructuras:

<tag> content **</tag>** ó
<tag/>

Los elementos que incluyen contenido u otros elementos están identificados por una etiqueta de comienzo <tag> y una etiqueta de finalización </tag>. Una etiqueta de elemento vacío <tag/> identifica elementos sin contenido.

2.3.4.- ATRIBUTOS

Los atributos WML especifican información adicional para un elemento. Los atributos se especifican siempre en la etiqueta de comienzo de un elemento. Por ejemplo,

```
<tag attr="value"/>
```

Nota: Los nombres de los atributos deber estar en minúsculas.

WML requiere que todos los valores del atributo sean entrecomillados usando tanto dobles comillas (") como comilla simple ('). Las comillas simples pueden incluirse dentro del valor de atributo cuando el valor se delimite por comillas dobles y viceversa. Las entidades de caracteres pueden incluirse en un valor de un atributo.

Algunos atributos son obligatorios, y están resaltados en negrita en las descripciones de elemento y las tablas de atributo mostradas posteriormente en esta guía. Por ejemplo, el elemento *go* requiere el atributo *href:*

```
<go href="http://www.acme.com"/>
```

2.3.5.- COMENTARIOS

Los comentarios en WML siguen el estilo de comentario de XML y tienen la siguiente sintaxis:

```
<!-- a comment -->
```

Los comentarios son una ayuda para el programador del documento WML y no se muestran al usuario en el terminal de destino. Hay que señalar que los comentarios WML no pueden ser almacenados.

2.3.6.- VARIABLES

Los parámetros pueden ser incluidos en las cartas y barajas de WML usando variables. Para sustituir una variable (por su valor) dentro de una carta o baraja, se utilizan las siguientes sintaxis:

```
$identifier
$(identifier)
$(identifier: conversion)
```

Se requieren paréntesis si el espacio en blanco no indica el final de una variable, es decir si el nombre de la variable lo conforman varias palabras separadas por espacios en blanco.

La sintaxis de variable tiene la mayor prioridad en WML, esto es, en cualquier lugar donde la sintaxis de variable sea legítima, un carácter “sin escapar” '*$*' conlleva a una sustitución de una variable. Las referencias de variables son legítimas en cualquier *PCDATA* y en cualquier valor de atributo que sea de tipo entero *vdata*.

Ejemplo 2.10:

Una secuencia de dos signos de dólar ($$) representa un carácter de signo de dólar sencillo.

El código WML podría tomar la siguiente forma:

```
This is a $$ character.
The value is $(amount)$$.
```

En el terminal del usuario, esto se mostraría como:

```
This is a $ character.
The value is 5000$.
```

2.3.7.- SENSIBILIDAD DE FORMATO DE LETRA

El lenguaje XML es un lenguaje sensible al formato de letra (mayúsculas-minúsculas) y WML ha heredado dicha característica. Cuando se analiza una baraja de WML no se realiza aceptando el doble formato como uno, es decir reconoce como diferentes lo que está en mayúsculas y lo que está en minúsculas. Esto implica que todas las etiquetas, atributos y contenidos de WML son sensibles al formato de letra. Además cualquier valor de atributo enumerado es sensible al formato. Los siguientes valores de atributo son todos diferentes:

Ejemplo 2.11:

id="Card1"
id="card1"
id="CARD1"

2.3.8.- SECCIÓN CDATA

Las secciones *Cdata* se utilizan para “escapar” bloques de texto y son legítimas en cualquier sección *pcdata*, por ejemplo, dentro de un elemento. Las secciones Cdata empiezan con una cadena "<! [cdata [" y terminan con una cadena "]] >".

Ejemplo 2.12:

<*![CDATA [*This is <b> a test.*]]*>

Cualquier texto dentro de una sección cdata se trata como texto literal y no será procesado. Las secciones cdata son útiles en cualquier lugar donde sea conveniente el texto literal.

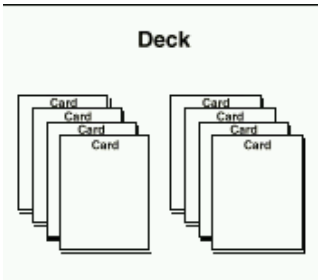
2.4.- LENGUAJE DE WML

2.4.1.- CARTAS Y BARAJAS

Todo el código WML se organiza dentro de una colección de cartas y barajas. Las cartas especifican una o más unidades de interacción con el usuario, por ejemplo un menú de opciones, una pantalla de texto o un campo de entrada de texto. Lógicamente, un usuario navega a través de una serie de cartas de WML, revisando el contenido de cada una de ellas, introduciendo la información requerida, realizando elecciones y moviéndose a otra carta.

Las cartas están agrupadas dentro de barajas. Una baraja es la unidad más pequeña WML que un servidor puede enviar al terminal del usuario.

La siguiente figura 2.2 ilustra el significado de carta y baraja:



Cartas (*card*) y barajas (*deck*) en el lenguaje WML.

Figura 2.2. Primer ejemplo de programación en WML
Fuente: [www.wapeton.com**]**

Nuestro primer ejemplo de WML introduce una baraja WML simple conteniendo dos cartas. Cuando el usuario presiona la tecla ACCEPT (Aceptar), rotulada "Next"(Siguiente), el usuario navega a la segunda carta de la baraja y muestra su contenido.

```
<?xml version="1.0"?> <!-- 1 -->
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">                    <!-- 2 -->
<wml>                                                         <!-- 3 -->
<card id="First_Card">                                        <!-- 4 -->
<do type="accept" label="Next">                               <!-- 5 -->
<go href="#Second_Card"/>                                     <!-- 6 -->
</do>                                                         <!-- 7 -->
<p>                                                           <!-- 8 -->
Select <b>Next</b> to display the next card.                  <!-- 9 -->
</p>                                                          <!-- 10 -->
</card>                                                       <!-- 11 -->
<card id="Second_Card">                                       <!-- 12 -->
```

```
<p>                                                    <!-- 13 -->
        This card contains the following:...           <!-- 14 -->
</p>                                                   <!-- 15 -->
</card>                                                <!-- 16 -->
</wml>                                                 <!-- 17 -->
```

Esta baraja genera el siguiente interfaz de usuario:

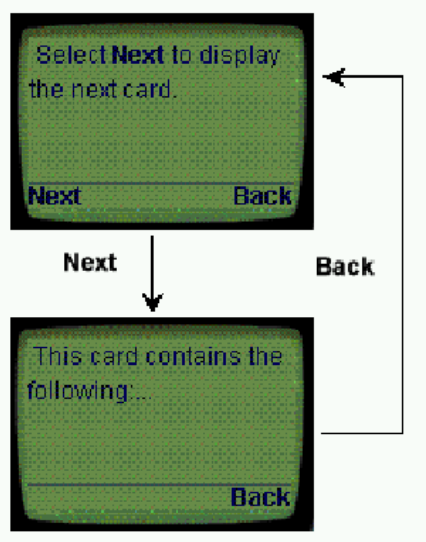


Figura 2.3. Baraja que contiene dos cartas

Fuente: [www.wapeton.com**]**

A continuación se realiza una explicación detallada de este ejemplo:

1. Las dos primeras líneas definen la cabecera del documento que identifica el subjuego XML. Esta cabecera debe incluirse al comienzo de cada baraja de WML, es decir, antes de cada etiqueta <wml>.
2. La tercera línea define el encabezado de la baraja WML. Todas las barajas WML deben empezar con una etiqueta <wml> y terminar con una etiqueta </wml>. Para más información, ver el "elemento wml " en esta guía.

3. La cuarta línea de la baraja especifica el encabezado e identificador de la primera carta. Análogamente a las barajas, las cartas también requieren etiquetas de comienzo y de finalización, <card> y </card>.

La mayoría de los elementos de WML permiten especificar atributos. Los atributos se introducen en la forma atribute=value (atributo=valor), donde *atribute* es el nombre de atributo y *value* es un valor alfabético o numérico que toma el atributo.

Para más información, ver el punto "el elemento carta" del tema siguiente.

4. La quinta línea define una acción, la cual especifica lo que el terminal del usuario debiera hacer al presionar una determinada tecla de función. El atributo *type* identifica la tecla (*accept*) y el atributo *label* asigna un rotulo (*Next*) a la tecla especificada.
5. La sexta línea indica una acción de relación entre cartas, páginas webs, etc. El atributo *href* indica el destino URI donde dicha acción nos va a llevar (en este ejemplo éste será la carta “Second-Card”).

2.4.1.1.- Atributos comunes.

Todos los elementos de WML tienen dos atributos centrales, ***id*** y ***class,*** que pueden usarse para tareas como transformaciones en el servidor. El atributo *id* proporciona a un elemento un nombre único dentro de una baraja sencilla. El atributo *class* afilia un elemento con una o más clases.

A los elementos múltiples se les puede dar el mismo nombre de clase. Todos los elementos de una baraja sencilla con un nombre de clase común se consideran parte de la misma clase.

Los nombres de clase son sensibles al formato de letra (mayúsculas-minúsculas). Un elemento puede pertenecer a varias clases si tiene definidos varios nombres de clases en su atributo ***class*****.** Si dentro de un atributo incluimos varios nombres de clases, estos se deben separarse con un espacio en blanco. Los nombres de clase redundantes así como espacios en blanco no significativos entre nombres de clase deben eliminarse. El terminal WML debería ignorar este atributo.

Todos los elementos que contengan texto pueden contener el atributo ***xml:lang*****.** Este atributo especifica el lenguaje natural de un elemento o sus atributos. El atributo indica al terminal del usuario el lenguaje utilizado para texto, tales como un contenido del elemento y valores del atributo, que puede ser mostrado al usuario.

2.4.1.2.- Cabecera del documento.

Una baraja de WML válida es un documento XML válido y por ello debe contener una declaración XML y una declaración de tipo de documento. A continuación se expone un encabezado típico de un programa WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

Se puede guardar la declaración de tipo de documento en un archivo, permitiéndose asi para una traducción a *bytecodes* más rápida.

Hay que distinguir los siguientes identificadores de documento:

- El identificador público SGML es `"-//WAPFORUM//DTD WML 1.1//EN"`.
- El identificador tipos de medios de difusión de WML es:
  - En forma textual: `"text/vnd.wap.wml"`.

- En forma de signo: `"application/vnd.wap.wmlc"`.

2.4.1.3.- El elemento wml

El elemento ***wml*** define una baraja y encierra toda la información y las cartas de la baraja.

- Elementos contenidos
  - head ?
  - template?
  - card +
- Sintaxis

  El atributo *xml:lang* del elemento *wml* se explica en la tabla 2.10:

Tabla: 2.10: Atributo del elemento wml

Atributo	Explicación
Xml : lang=nmtoken	Este atributo especifica el lenguaje natural o formal en el cual el punto el documento está escrito.

Ejemplo 2.13:

A continuación se muestra una baraja que contiene dos cartas, cada una representada por la etiqueta *card*. Después de cargar la baraja, el terminal del usuario muestra la primera carta. Si el usuario activa el elemento *do*, el terminal muestra la segunda carta.
El atributo *xml : lang* especifica que el documento está escrito en inglés americano.

```
<wml xml:lang="en-us">
<card id="card1" title="Card 1">
        <do type="accept">
```

```
            <go href="#card2"/>
        </do>
        <p>
            Hello world!
            This is the first card...
        </p>
</card>
<card id="card2" title="Card 2">
        <p>
            This is the second card.
            Goodbye.
        </p>
</card>
</wml>
```

La baraja genera la siguiente interfaz de usuario en el terminal:

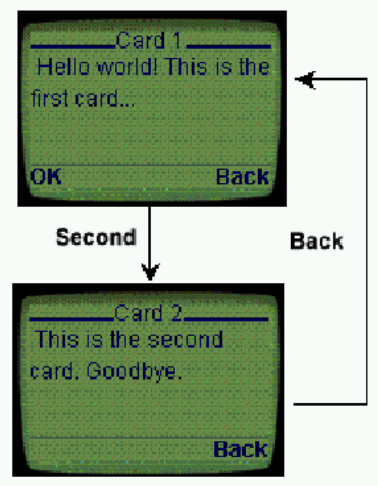


Figura 2.4. Baraja que contiene dos cartas

Fuente: [www.wapeton.com**]**

2.4.1.4.- El elemento card

Una baraja de WML contiene una colección de cartas. Hay diferentes tipos de cartas, cada una especificando un modo diferente interacción con el usuario.

El elemento ***card*** es un contenedor para texto y elementos de entrada, lo suficientemente flexible para permitir la presentación en una amplia variedad de dispositivos, con una gran variedad de características de entrada y de presentación. El elemento ***card*** indica la disposición general y los campos de entrada requeridos, mientras da una considerable libertad para la disposición instrumentalizada y los esquemas de entrada del terminal del usuario. Por ejemplo, un ***card*** puede ser presentado como una página sencilla en un dispositivo de pantalla grande o como una serie de páginas más pequeñas en un aparato de pantalla pequeña.

Un card puede contener marcas, campos de entrada y elementos que indican la estructura de la carta. Hay que señalar que el orden de los elementos en la carta es significativo. Se puede usar un *id* de una carta como un anclaje de fragmento.

- Elementos contenidos
  - onevent *
  - timer ?
  - do *
  - p *
- Sintaxis

  Los atributos del elemento card se explican en la tabla 2.11:

Tabla: 2.11: Atributos del elemento card

Atributo	Explicación
Title=vdata	Este atributo especifica información adicional sobre la carta.
Newcontext=boolean	Si se coloca este atributo como true (verdadero), el estado actual del browser es reinicializado cuando se ejecute esta carta, realizando las operaciones siguientes: Borra el estado del historial de navegación. Resetea el estado específico de implementación a un valor conocido.
Order=boolean	Este atributo da una indicación al Terminal de sobre como se organiza el contenido card. Esta indicación puede usarse para organizar la presentación del contenido para influir en la disposición de la carta. *ordered="true"* La carta se organiza como una secuencia lineal de campos, por ejemplo, un conjunto de preguntas o campos los cuales son manejados por el usuario en el mismo orden en que han sido especificados en el grupo. Este estilo es el mejor para forma donde no hayan campos opcionales. Por ejemplo, enviar un mensaje de correo electrónico (email) requiere un Dirigido a:, un Tema, un mensaje, y están lógicamente especificados en este orden. *ordered="false"* Esta carta es una colección de elementos de campo sin un orden natural. Esto es útil para las colecciones de campos que contienen componentes opcionales y desordenados o para datos grabados simples donde el usuario ajusta campos de entrada individuales.
Onenterforward=href	El suceso *onenterforward* ocurre cuando el usuario navega dentro de una carta usando una tarea *go*.
Onenterbackward=href	El suceso *onenterbackward* ocurre cuando el usuario navega en una carta usando una tarea *prev*.
ontimer=ref.	El suceso *ontimer* ocurre cuando un timer (contador de tiempo) expira, para más información, ver el punto "el suceso ontimer".

Ejemplo 2.14:

Lo que aparece a continuación es un ejemplo de un elemento card encerrado en una baraja de WML.
Esta carta contiene texto, el cual se muestra en la pantalla del terminal del usuario.
Cuando el usuario navega a esta carta, el contesto del browser es reinicializado, esto implica que todas las variables son borradas y la pila del historial es vaciada.

```
<wml>
<template>
        <do type="accept" name="exit" label="EXIT">
                <prev/>
        </do>
</template>
<card id="card_1" title="Welcome" newcontext="true">
        <p> Hello World! </p>
</card>
</wml>
```

Esta baraja genera el siguiente interfaz con el usuario en su terminal:

Figura 2.5. Una carta sencilla
Fuente: [www.wapeton.com]

2.4.1.5.- El elemento template

El elemento ***template*** (patrón) declara un patrón para las cartas de la baraja. Las acciones especificadas en el elemento template (por ejemplo, ***do*** o ***onevent***) se aplican a todas las cartas de la baraja. Especificar un suceso en el elemento template es equivalente a especificarlo en cada carta. De todos modos, un elemento ***card*** puede supeditar las acciones especificadas en el elemento ***template***. En particular, pueden señalarse las siguientes reglas de supeditación:

- Los elementos ***do*** especificados en el *template* puede ser supeditados en cartas individuales si ambos elementos tienen el mismo valor de atributo ***name***.

- Los enlaces de sucesos intrínsecos especificados en el ***template*** pueden ser supeditados por enlaces de sucesos de una carta.

- Elementos contenidos
  - do *
  - onevent *
- Sintaxis

  Los atributos del elemento template se explican en la tabla 2.12:

Tabla: 2.12: Atributos del elemento template

Atributo	Explicación
Onenterforward=href	Especifica un suceso intrínseco que instruye al terminal del usuario a ir al URL especificado cuando el usuario navega hacia delante a esta carta.
onenterbackward=href	Especifica un suceso intrínseco que instruye al terminal a ir al URL especificado cuando el usuario navega hacia atrás a esta carta, por ejemplo, usando una tarea prev.
ontimer=ref.	Especifica un suceso intrínseco que instruye al terminal a ir al URL especificado después de que el timer(contador de tiempo) ha expirado.

Ejemplo 2.15:

El siguiente elemento ***template*** incluye un elemento *do* indicando la navegación a la carta anterior de la baraja.

```
<template>
        <do type="prev" label="Previous">
                <prev/>
        </do>
</template>
```

2.4.1.6.- El elemento head

El elemento ***head*** contiene información relativa a la baraja como un todo, incluyendo los meta-datos y los elementos de control de acceso.

Se debe incluir uno de los siguientes elementos al menos una vez dentro de un elemento ***head:***

- access
- meta

Ejemplo 2.16:

Ver los ejemplos de los elementos **access** y **meta.**

2.4.1.7.- El elemento access

El elemento ***access*** específica la información del control de acceso para la baraja entera. Hay que advertir que es un error sintáctico en WML el que una baraja contenga más de un elemento access. Si una baraja no incluye un elemento ***access,*** el control de acceso está inhabilitado. Cuando el control de

acceso está inhabilitado, las cartas de cualquier baraja pueden acceder a esta baraja.

La colocación por omisión del control de acceso permite acceder a cualquiera de los URI (Uniform Resource Identifier, identificador de recursos uniforme) del mismo dominio.

El terminal del usuario utiliza una pareja de prefijos para comparar los URI de las barajas que están intentando acceder a una baraja con valores de atributo que se definen.

La tabla 2.13 muestra un listado de los elementos que permiten navegar entre barajas y los requerimientos de acceso asociados que la baraja "blanco/diana" debe especificar.

Tabla: 2.13: Elemento que permiten navegar entre cartas

Elemento	Requerimientos de acceso
Prev	Ninguno
Go href=ref.	La baraja localizada en el URI especificado debe especificar el atributo *domain* (dominio) y/o el atributo *path*(vía) que emparejan el URI de la baraja requerida.

- Sintaxis

  Los atributos del elemento access se explican en la tabla 2.14:

Tabla: 2.14: Atributos del elemento access

Atributo	Explicación
domain=cdata	Los atributos *domain* y *path* de una baraja especifican las otras barajas que pueden acceder a ella. Mientras el terminal del usuario navega de una baraja a otra, se realizan comprobaciones (chequeos) del control de acceso para determinar si la baraja destino permite el acceso desde la baraja actual. Si una baraja tiene un atributo *domain* y/o *path*, el URL de la baraja de referencia debe emparejar los valores de los atributos.

	El emparejamiento se realiza como sigue: el dominio de acceso es emparejado por el sufijo frente a la porción de nombre de dominio del URL de referencia y la vía de acceso es emparejada por el prefijo frente a la porción de vía del URL de referencia. El atributo *domain* toma por defecto el dominio de la baraja actual. El atributo *path* tiene en "/" su valor por defecto. Para simplificar el desarrollo de aplicaciones que puedan no conocer el camino exacto a la baraja actual, el atributo *path* acepta URL relativos. El terminal del usuario convierte la vía relativa a una absoluta y entonces realiza el emparejamiento de prefijo frente al atributo path. El atributo domain y el path siguen las normas de capitalización de URL.

Ejemplo 2.17:

Por ejemplo, dados los atributos de control de acceso siguientes:

domain="acmecorp.com"

path="/pub"

Las siguientes referencias a URLs deberían poder acceder a la baraja:

acmecorp.com/pub/stocks.cgi

www.acmecorp.com/pub/demos/packages.cgi

Por el contrario, las siguientes referencias a URLs no deberían tener permiso de acceso a la baraja:

www.test.net/pub

www.acmecorp.com/internal/foo.wml

El elemento *head* siguiente incluye un elemento de control de acceso que indica que solo las barajas del directorio WML del dominio mycompany.com pueden acceder a esta baraja:

```
<head>
    <!-- NOTE: The DOMAIN and PATH must be customized for
    your network location of the WML decks -->
    <access domain="mycompany.com" path="/WML" >
</head>
```

2.4.1.8.- El elemento meta

El elemento ***meta*** contiene información genérica adicional relativa a las barajas de WML. La información adicional es especificada con nombres de propiedades y valores.

Hay que señalar que es un error de sintaxis en WML que un elemento ***meta*** contenga más de un atributo especificando el nombre de una propiedad, es decir, más de un atributo del juego siguiente: ***name*** **y** ***http-equiv*****.**

- Elementos contenidos

  Ninguno.

- Sintaxis

  Los atributos del elemento meta se indican en la siguiente tabla.

Tabla: 2.15: Atributos del elemento meta

Atributo	Explicación
Content=cdata	Este atributo especifica el valor de la propiedad. Es obligatorio.
Name=cdata	Este atributo especifica el nombre de la propiedad. El Terminal ignora los meta datos nombrados. Los servidores de la red no emiten contenidos WML que contienen metadatos nombrados con este atributo.
http=equiv=cdata	Este atributo puede ser utilizado en lugar de *name*; esto indica que la propiedad debería ser interpretada como un encabezado de HTTP. Los meta datos nombrados con este atributo se convierten en un encabezado de respuesta de WSP o HTTP si el contenido es indicado antes de que llegue al terminal.
Forua=bolean	Este atributo especifica que el autor intenta la propiedad de alcanzar al terminal del usuario. Si el valor es *false*, un agente intermedio debe eliminar el elemento meta antes de que el documento se envíe al cliente. Si el valor es *true*, los meta datos del elemento deben ser enviados al terminal del usuario. El método de envío puede variar. Por ejemplo, los meta datos *http-equiv* pueden ser enviados usando los encabezados de http o WSP.
Écheme=cdata	Este atributo especifica una forma o estructura que puede usarse para interpretar el valor de la propiedad.

Ejemplo 2.18:

El siguiente elemento ***head*** incluye un elemento de control de acceso, descrito anteriormente en el ejemplo anterior, y un elemento *meta* que proporciona información al terminal del usuario sobre el juego de caracteres usado en la baraja de WML.

```
<head>
    <!-- NOTE: The domain and path must be customized for
    your network location of the WML decks -->
    <access domain="mycompany.com" path="/WML" >
    <meta content="charset" user agent="character-set=UTF-8"/>
</head>
```

2.4.2.- SUCESOS

WML incluye varios elementos que permiten manejar la navegación y los sucesos especificando el procesamiento de dichos eventos en el terminal del usuario. Por ejemplo, se pueden asociar sucesos con tareas de modo que cuando ocurra un suceso, la tarea asociada sea ejecutada. Se puede especificar una variedad de tareas, como por ejemplo la navegación a un URL. El enlace de sucesos puede implementarse con ayuda de varios elementos, incluyendo ***do*** y ***onevent***.

2.4.2.1.- Elemento do

El elemento ***do*** da al usuario un mecanismo general para realizar acciones en la carta actual, esto es, un elemento de interfaz con el usuario a nivel de carta.

El elemento ***do*** está mapeado a un único interfaz de usuario *widget*, el cual el usuario puede activar. Por ejemplo, el mapeado *widget* puede ser para un botón restaurado (formateado y presentado) gráficamente, para una tecla blanda

o una tecla de función, una secuencia de comando activada por la voz o cualquier otro interfaz que tenga una operación "de activar" sencilla con ningún estado persistente de ínter operación.

El atributo ***type*** proporciona una indicación al terminal del usuario de como el programador de WML propone el elemento para ser utilizado, y es tomado por el terminal para proporcionar un mapeo adecuado sobre la construcción del interfaz usuario físico. Hay que señalar que los programadores no deben confiar en la semántica o el comportamiento de un valor ***type*** concreto, o en el mapeo de ***type*** a una construcción física particular.

El elemento ***do*** puede aparecer tanto a nivel de carta como a nivel de baraja:

A nivel de carta: El elemento ***do*** puede aparecer dentro de un elemento ***card*** y puede localizarse en cualquier lugar del texto. Si el terminal propone restituir (formatear y presentar) el elemento ***do*** en línea (es decir, en el flujo de texto), debería usar el punto de anclaje del elemento como el punto de restitución. Hay que señalar que los autores de WML no deben asumir que la restitución en línea del elemento ***do*** es correcta; ni deben asumir que la restitución se posiciona correctamente.

A nivel de baraja: El elemento ***do*** puede aparecer dentro del elemento ***template***, indicando un elemento ***do*** a nivel de baraja. El elemento ***do*** a nivel de baraja se aplica a todas las cartas de la baraja, es decir, equivale a especificar el elemento ***do*** dentro de cada carta. Para los propósitos de restitución en línea, el terminal usuario se comparta como si los elementos ***do*** a nivel de baraja estuvieran localizados al final del flujo de texto de la carta.

En particular, se debería prestar atención a lo siguiente:

- Un elemento ***do*** a nivel de carta supedita un elemento ***do*** a nivel de baraja si tienen el mismo nombre. Para una carta sencilla, los elementos ***do*** activos se definen como los elementos ***do*** especificados en la carta, además cualesquiera de los elementos ***do*** especificados en el elemento ***template*** de la baraja no se supeditan en la carta.
- Los elementos *do* inactivos y los activos con un elemento de tarea ***noop*** no se presentan al usuario.
- El usuario puede acceder a todos los elementos ***do*** con una tarea que no sea ***noop***. El usuario puede activar estas propiedades de interfase cuando ve que la carta contiene elementos **do** activos. Cuando el usuario activa un elemento ***do***, la tarea asociada se ejecuta.
- Elementos contenidos
  - go| prev | noop | refresh
- Sintaxis

  Los atributos del elemento do se explican en la tabla 2.16:

Tabla: 2.16: Atributos del elemento do

Atributo	Explicación
type=cdata	El tipo del elemento *do*. Este atributo proporciona un indicación al terminal del usuario sobre como el autor de WML propone que el elemento sea usado, y como debería ser mapeado a una construcción de interfaz de usuario física. Todos los tipos están reservados, excepto para aquellos marcados como experimentales. Este atributo es requerido. Los terminales aceptan cualquier *type*, pero pueden tratar cualquier type desconocido como el equivalente a *unknown*. A continuación, el carácter * representa cualquier cadena. Por ejemplo, Test * indica cualquier cadena que empiece con la palabra Test. accept Reconocimiento positivo (aceptación). prev Navega hacia atrás en el historial. Help Petición de ayuda. Puede ser sensible al contexto. reset Borrar o reiniciar el estado.

	options Petición sensible al contexto para opciones u operaciones adicionales. delete Borrado de las elecciones realizadas. unknown Elemento *do* genérico. equivalente a una cadena vacía p.ej. type=" ". X-*, x-* Tipos experimentales. Este juego no está reservado.
label=vdata	Este atributo especifica una cadena de texto para rotular el interfaz de usuario *widget*. Si un elemento no puede ser rotulado de forma dinámica, este atributo se ignora. Para trabajar bien, los rótulos no deber tener más de seis caracteres.
name=nmtoken	Este atributo especifica el nombre del enlace del suceso *do*. Si dos elementos *do* son especificados con el mismo nombre, se refieren al mismo enlace. Si los elementos *do* son especificados ambos a nivel de carta (en una carta) y a nivel de baraja(con un *template*) y ambos elementos tienen el mismo nombre, el elemento do a nivel de baraja es ignorado. Es un error sintáctico de WML especificar dos o más elementos do con el mismo nombre en una carta sencilla o en un *template*. Un nombre con un valor vacío equivale a un atributo *name* no especificado. Un *name* no especificado toma por defecto al valor del atributo *type*.
optional=boolean	Si se coloca este atributo en verdadero (*true*), el terminal puede ignorar este elemento. Por defecto su valor es falso(*false*).

Ejemplo 2.19:

Este ejemplo muestra un elemento *DO* que incluye una tarea *go*. Cuando el usuario activa el elemento seleccionando *Next*, el terminal va a la carta2 de la baraja actual y la muestra al usuario.

```
<card id="card1">
<do type="accept" label="Next">
<go href="#card2"/>
</do>
<p>
Select <b> Next </b> to go to the next card.
</p>
</card>
```

<card id="card2">

This is card 2.

</card>

2.4.2.2.- Evento ontimer

El evento ***ontimer*** puede ser especificado dentro de los elementos siguientes:

- card
- template

El suceso ocurre cuando un timer (contador de tiempo) expira.

- Sintaxis
  Los atributos del evento ***ontimer*** se explican en la tabla 2.17:

Tabla: 2.17: Atributo del elemento ontimer

Atributo	Explicación
ontimer=href	Especifica un suceso intrínseco que instruye al agente usuario a ir al URI especificado después de que el timer haya expirado

Ejemplo 2.20:

A continuación se muestra un ejemplo sencillo de un elemento *ontimer*:
<card id="cardname" ontimer="http://wapserver/hello.wml" title="title">
<timer value="50"/>

Hello World!

</card>

2.4.2.3.- Evento onenterforward

El evento ***onenterforward*** ocurre cuando el usuario introduce una carta utilizando la tarea ***task*** o cualquier otro método con idéntica semántica. El suceso intrínseco ***onenterforward*** puede ser especificado dentro de los siguientes elementos:

- card
- template

Los enlaces de sucesos especificados en el ***template*** se aplican a todas las cartas de la baraja y pueden ser supeditadas como se especifica en el item "2.4.2.9.- Supeditación de tareas en cartas y barajas".

- Sintaxis
  Los atributos del evento ***onenterforward*** se explican en la tabla 2.18:

Tabla: 2.18: Atributo del elemento onenterforward

Atributo	Explicación
Onenterforward =href	Especifica un suceso intrínseco que instruye al terminal a ir al URI especificado cuando el usuario introduce esta carta

Ejemplo 2.21:

Se puede especificar que ciertas tareas sean ejecutadas cuando un evento intrínseco ocurra de dos maneras:

Primera, se puede especificar para que se navegue a URI cuando ocurra un suceso. Este enlace de suceso se especifica en un atributo de elemento especifico bien definido el cual equivale a una tarea *go*. Por ejemplo:

```
<card onenterforward="http://wapserver/hello.wml">
        Hello World! You came back
</card>
```

Hay que señalar que el mensaje "Hello World! You came back", no se muestra cuando se navega hacia adelante en la carta. Igualmente, cuando se navega hacia la carta hacia atrás la carta es mostrada.

Segunda, se puede usar una versión expandida del método anterior, el cual da más control sobre el comportamiento del agente usuario.
Un elemento ***onevent*** se implementa dentro de un elemento *parent* (padre), especificando la unión de sucesos completa para un suceso intrínseco particular. El siguiente ejemplo, es idéntico al ejemplo anterior:

```
<card>
        <onevent type="onenterforward">
                <go href=" http://wapserver/hello.wml "/>
        </onevent>
        <p>
                Hello World! You came back.
        </p>
</card>
```

Sin embargo, el agente usuario trata la sintaxis del atributo como una forma abreviada del elemento *onevent* donde el nombre del atributo es mapeado al tipo ***onevent*.**

2.4.2.4.- Suceso nenterbackward

El suceso **onenterbackward** ocurre cuando el usuario navega dentro de una carta usando una tarea ***prev*** o cualquier método de idéntica semántica. En

otras palabras, el suceso ***onenterbackward*** ocurre cuando el usuario navega dentro de una carta usando un URL recuperado de la pila del historial.

El suceso intrínseco ***onenterbackward*** puede estar especificado dentro de los siguientes elementos:

- card
- template

Los enlaces de sucesos especificados en el elemento ***template*** se aplican a todas las cartas de la baraja y pueden ser supeditadas como se especifica, un poco más adelante en el item “2.4.2.9.- Supeditación de tareas de cartas y barajas".

- Sintaxis
  Los atributos del suceso *onenterbackward* se explican en la tabla 2.19:

Tabla: 2.19: Atributos del suceso onenterbackward

Atributo	Explicación
onenterbackward =ref.	Especifica un suceso intrínseco que instruye al agente usuario a ir al URI especificado cuando el usuario navega hacia atrás hacia esta carta, por ejemplo usando una tarea prev.

Ejemplo 2.22:

En el siguiente ejemplo, el suceso ***onenterbackward*** hace que el terminal del usuario navegue a la carta2 cuando el usuario introduce esta carta usando una tarea *prev* o navegando hacia atrás en la pila del historial.

Esto significa que la carta2 se muestra al usuario en lugar de la carta1. Hay que señalar que si el usuario navega hacia delante hacia esta carta usando la tarea ***go***, por ejemplo, se mostrará la carta1.

```
<card id="card1">
        <onevent type="onenterbackward">
                <go href="#card2"/>
        </onevent>
        <p>
                Hello World!
        </p>
</card>

<card id="card2">
        <p>
                You came back!
        </p>
</card>
```

Igual que en el ejemplo anterior, la carta1 podría también presentarse como se muestra a continuación:

```
<card id="card1" onenterforward="#card2"> Hello World! </card>
```

2.4.2.5.- Evento onpick

El evento ***onpick*** ocurre cuando el usuario elige o deshace la elección del objeto/elemento en el que dicho evento está especificado. El suceso intrínseco ***onpick*** puede ser especificado dentro del elemento ***option*****.**

- Sintaxis

  Los atributos del suceso ***onpick*** son explicados en la tabla 2.20:

Tabla: 2.20: Atributos del suceso onpick

Atributo	Explicación
onpick =ref.	Especifica un suceso intrínseco que instruye al agente usuario a ir al URI especificado cuando el usuario selecciona la opción (o la deselecciona si el elemento select “seleccionar” permite múltiples elecciones) en la cuál se especifíca el suceso.

Ejemplo 2.23:

En el ejemplo siguiente, cuando el usuario selecciona el objeto desde la opción *list*, el terminal del usuario navega a la baraja apropiada del mismo dominio. Por ejemplo, si el usuario selecciona el objeto 1, información sobre gatos (cats), el terminal navega a la carta *cat.wml* que contiene la información sobre los gatos.

```
<card id="Card_1">
<p>
Select your favorite animal:
<select name="animal">
<option value="1" onpick="cat.wml"> Cat </option>
<option value="2" onpick="dog.wml"> Dog </option>
<option value="3" onpick="horse.wml"> Horse </option>
</select>
</p>
</card>
```

2.4.2.6.- Elemento onevent

El elemento ***onevent*** enlaza una tarea a un suceso intrínseco particular para el elemento inmediatamente adjunto; es decir, especifican un elemento ***onevent*** dentro de un elemento que asocia un enlace de sucesos intrínsecos con ese elemento. El terminal del usuario ignora cualquier elemento ***onevent***

especificando un tipo que no corresponda al suceso legítimo para el elemento adjunto.

- Elementos contenidos
  - go | prev | noop | refresh
- Sintaxis

  Los atributos del elemento ***onevent*** se explicados en la tabla 2.21:

Tabla: 2.21: Atributo del elemento onevent

Atributo	Explicación
type=CDATA	El atributo *type* indica el nombre del suceso intrínseco.

Ejemplo 2.24:

La siguiente carta indica que cuando el usuario navega hacia atrás en ella, el suceso ***onenterbackward*** se activa y el mensaje de la carta no se muestra al usuario. En vez de eso, el agente usuario navega a la baraja *foo.wml* localizada en el dominio www.acme.com.
Hay que señalar que si el usuario navega hacia delante en esta carta, se muestra su contenido.

```
<card>
<onenvent type="onenterbackward">
<go href="http://www.acme.com/foo.wml"/>
</onevent>
<p>
Welcome to a new age!
</p>
</card>
```

2.4.2.7.- Elemento postfield

El elemento ***postfield*** (campo enviado) especifica un nombre de campo y un valor para la transmisión a un servidor durante una petición de URL. La codificación real del nombre y el valor dependen del método utilizado para comunicarse con el servidor.

- Elementos contenidos
  - Ninguno
- Sintaxis

  Los atributos del elemento *postfield* se explican en la tabla 2.22:

Tabla: 2.22: Atributo del elemento postfield

Atributo	Explicación
name=vdata	El atributo *name* especifica el nombre del campo. Este atributo es requerido
value=vadata	El atributo *value* especifica el valor de campo. Este atributo es requerido.

Ejemplo 2.25:

El siguiente ejemplo envía tres valores de nombre al servidor de web, permitiendo que se envíe de vuelta datos desde el cliente. EL método *HTTP POST* se utiliza para enviar los datos.

```
<go method="post" href="http://hostname/servlet/bank">
<postfield name="money" value="100"/>
<postfield name="account" value="12345"/>
<postfield name="operation" value="deposit"/>
</go>
```

2.4.2.8.- Sucesos intrínsicos de cartas y barajas

Se pueden especificar los sucesos intrínsecos ***onenterforward*** y ***onenterbackward*** tanto a nivel de carta como al de baraja utilizando semánticas de supeditación definidas en el item "2.4.2.9. Supeditación de tareas de cartas y barajas" un poco más adelante. Los sucesos intrínsecos pueden ser supeditados respecto a la sintaxis utilizada para especificarlos. Un manipulador de sucesos a nivel de baraja especificado con el elemento ***onevent*** puede ser supeditado por el elemento ***oneneterforward*** y viceversa.

2.4.2.9.- Supeditación de tareas en cartas y barajas

Se pueden utilizar una variedad de elementos para crear un enlace de suceso en una carta. Estos enlaces pueden también ser determinados a nivel de baraja:

- **A nivel de carta:** El elemento manipulador de sucesos puede aparecer dentro de un elemento ***card*** y especificar el comportamiento en el procesamiento de sucesos para esa carta particular.
- **A nivel de baraja:** El elemento manipulador de sucesos puede aparecer dentro de un elemento ***template*** y especificar el comportamiento en el procesamiento de sucesos de todas las cartas de la baraja. Un elemento manipulador de sucesos a nivel de baraja es equivalente a especificar el elemento en cada carta de la baraja.

Más concretamente, debe señalarse las siguientes normas de supeditación:

- Un elemento **manipulador de sucesos** a nivel de carta supedita (se muestra por encima) a uno a nivel de baraja si ambos especifican el mismo suceso.

- Un elemento ***onevent*** a nivel de carta supedita (prevalece sobre) un elemento ***onevent*** a nivel de baraja si ambos tienen el mismo tipo (*type*).
- Un elemento ***do*** a nivel de carta prevalece sobre uno a nivel de baraja si ambos tienen el mismo nombre.

Si un elemento a nivel de carta prevalece sobre uno a nivel de baraja y el elemento a nivel de carta especifica la tarea ***noop*****,** el enlace de sucesos para ese suceso debe ser completamente marcado. En esta situación los elementos a nivel de baraja y a nivel de carta serán ignorados y ningún efecto lateral se producirá en el envío de sucesos. En este caso, el terminal del usuario no mostrará el elemento al usuario, por ejemplo, en la restitución (formateo y presentación) de un control de UI. En efecto, la tarea ***noop*** elimina el elemento de la carta.

Ejemplo 2.26:

En el siguiente ejemplo, un elemento *do* a nivel de baraja indica que una tarea *prev* deber ejecutarse en respuesta a una acción de usuario determinada.

- La primera carta hereda el elemento *do* especificado en el *template* y muestra el *do* al usuario.
- La segunda carta predomina sobre el *do* a nivel de baraja con una tarea *noop*. El terminal no muestra el elemento *do* cuando muestra la segunda carta.
- La tercera carta predomina sobre el elemento *do* a nivel de baraja, provocando que el terminal muestre el rotulo alternativo y realice la tarea *go* si se selecciona el elemento *do*.

```
<wml>
    <template>
        <do type="options" name="do1" label="default">
            <prev/>
        </do>
    </template>
<card id="first">
    <!-- deck-level do not overridden. The card exposes the deck-level do
    as part of the current card. -->
    <!-- rest of the card -->
</card>
<card id="second">
    <!-- deck-level do is overridden with noop.
        It is not exposed to the user. -->
    <do type="options" name="do1">
        <noop/>
    </do>
    <!-- rest of the card -->
</card>
<card id="third">
    <!-- deck-level do is overridden.
        It is replaced by a card-level do -->
    <do type="options" name="do1" label="options">
        <go href="/options"/>
    </do>
    <!-- rest of the card -->
</card>
</wml>
```

2.4.3.- TAREAS

2.4.3.1.- Tarea go

El elemento ***go*** determina una tarea ***go*** (“ir”), indicando la navegación a un URL. Si el URL nombra a una baraja o a una carta de WML, ésta es mostrada. Una tarea ***go*** ejecuta una operación de empuje sobre la pila del historial del *browser* (navegador).

- Elementos contenidos
  - setvar *
  - postfield *
- Sintaxis

  Los atributos del elemento ***go*** se explican en la tabla 2.23:

Tabla: 2.23: Atributos del elemento go

Atributo	Explicación
href=ref.	Este atributo especifica el URI de destino, por ejemplo el URI de la carta a mostrar. Este atributo es requerido.
sendreferer=boolean	Si se coloca el atributo en *true* (verdadero), el terminal incluye el URL de la baraja que contiene esta tarea en el requerimiento de URI. Esto permite al servidor realizar una forma de control de acceso en los URIs, basado en las barajas que están vinculadas a ellos. Especificando el *sendreferer=true* provoca que el agente usuario coloque un encabezado de referencia de HTTP para el URI relativo más pequeño de la baraja requerida. Hay que señalar que si se quiere restringir el acceso a los servicios, las barajas que requieren los URIs de dichos servicios deben colocarse en esta opción como verdadero (trae). Por defecto el valor es falso (*false*).
Meted=(post | get)	Este atributo especifica el método de sumisión de HTTP. Actualmente, los valores de *get* y *post* son aceptados y provocan que el terminal del usuario realice un get o un

	post de HTTP respectivamente. El valor por defecto es *get*.
Accept-charset=CDATA	Este atributo especifica el listado de codificaciones de caracteres de datos que el servidor web debe aceptar cuando se realiza el procesado. El valor de este atributo tomado de la lista de nombres codificados de caracteres (*charset*), está separado por comas o espacios como se especifica en *RFC2045* y *RFC2068*. El registro de juego de caracteres IANA define el registro público para valores de *charset*. Este listado es un listado ORexclusivo, es decir, que el servidor debe aceptar todos los codificadores de caracteres aceptables. El valor por defecto para este atributo es la cadena reservada *unknown* (desconocida). El terminal del usuario interpreta este valor como el codificador de caracteres que se uso para transmitir la baraja de WML que contenía este atributo.

El elemento ***go*** puede contener uno o más elementos ***postfield*****.** Estos elementos especifican información para ser expuesta al servidor de origen durante la solicitud. La exposición de los datos de campo es realizada de la siguiente manera:

- Los pares nombre/valor de campo son identificados y todas las variables son sustituidas.
- El terminal deber transcodificar los valores y nombres de campo al juego de caracteres correcto, como se especifica explícitamente por el accept-charset (nombres de codificación de caracteres-aceptar) o implícitamente por el codificador del documento.
- Los nombres y valores de campo son exportados usando la URL y se ensamblan dentro de un documento de tipo formulario: applicacion/x-www-form-urlencoded. Los exportadores de URL están definidos en RFC2396 y el resumen del contenido se especifica en RFC2070.
- EL requerimiento se realiza según el valor de atributo method (método):
- get- Si el atributo method tiene un valor de get y el valor del atributo href es un URI de HTTP, los datos de la sumisión se añaden al componente

de query (pregunta) del URI. Una operación HTTP GET se realiza sobre el URL resultante.

- post- Si el atributo method tiene un valor de post y el valor del atributo href es un URI de HTTP, los datos de sumisión se envían al servidor de origen con una operación HTTP POST. La sumisión se identifica con un contenido de tipo aplicación/x-www-form-urlencoded, con un parámetro charset indicando el codificador de caracteres.

Ejemplo 2.27:

En el siguiente ejemplo, el elemento *go* produciría una petición HTTP GET al URL *"/ foo? x=1"*:

```
<go href="/foo">
        <postfield name="x" value="1"/>
</go>
```

El siguiente ejemplo producirá un HTTP POST al URL "/bar" con un mensaje conteniendo *"w=12&y=test"*:

```
<go href="/bar" method="post">
        <postfield name="w" value="12"/>
        <postfield name="y" value="test"/>
</go>
```

El siguiente ejemplo crea un interfaz de usuario *widget* con el rotulo Help y va a la carta denominada “help” cuando se activa:

```
<card id="card1">
        <do type="help" label="Help">
                <go href="#help"/>
```

```
            </do>
</card>
<card id="help">
        <p>
            Help topics:
        </p>
</card>
```

2.4.3.2.- Tarea prev

El elemento ***prev*** (de 'previous', previo, anterior) indica la navegación al URI anterior de la pila del historial.

Una tarea **prev** produce una operación pop (de salto) en la pila del historial y elimina el URI actual de la pila. Si no hay URI anteriores en la pila del historial, el elemento *prev* no tiene efecto.

Ejemplo 2.28:

El siguiente ejemplo crea un *widget* de interfaz de usuario con el rotulo "Back" y va a la carta anterior cuando se activa.

```
<do type="accept" label="Back">
        <prev/>
</do>
<p> Hello, World! </p>
```

2.4.3.3.- Tarea refresh

El elemento ***refresh*** (refrescar, renovar) determina una tarea que da lugar a una puesta al día de las variables de la carta especificada. Los efectos colaterales de los cambios de estado que son visibles para el usuario (por

ejemplo, un cambio al mostrar la pantalla) ocurre durante el procesamiento de la tarea ***refresh***. Esta renovación o puesta al día y sus efectos secundarios deben suceder incluso si los elementos no tienen elementos ***setvar*** dado que el contexto puede cambiar por otros medios (por ej. *timer*).

- Elementos contenidos
  - setvar *

Ejemplo 2.29:

El siguiente programa WML asigna un valor *foo* a la variable *product* y refresca la pantalla para actualizar los valores de las variables.

```
<refresh>
        <setvar name="product" value="foo"/>
</refresh>
```

2.4.3.4.- Tarea noop

El elemento ***noop*** especifica que no se debe hacer nada, es decir "no operación". Este elemento es útil para pasar por alto el elemento ***do*** a nivel de baraja.

2.4.4.- VARIABLES

Los parámetros pueden ser colocados para todos los contenidos de WML, aportando una enorme flexibilidad en la creación de barajas y cartas que cambian de forma dinámica el contenido mostrado y la navegación basada en la entrada de usuario. Las variables de WML pueden usarse en lugar de las cadenas y sustituirse en tiempo de ejecución con sus valores actuales.

Una variable es colocada si tiene un valor distinto a una cadena vacía. Un valor no se coloca si tiene un valor igual a una cadena vacía, o es por otro lado desconocida o indefinida en el contexto de browser actual.

2.4.4.1.- Elemento setvar

El elemento ***setvar*** (de "set variable", colocar variable) especifica la variable a colocar en el ***browser*** (navegador) actual como un efecto secundario de la ejecución de una tarea. El elemento es ignorado si el atributo ***name*** no referencia a un nombre de variable legítimo en el tiempo de ejecución.

- Elementos contenidos
  - Ninguno
- Sintaxis

  Los atributos de las variables se explican en la tabla 2.24:

Tabla: 2.24: Atributos de las variables

Atributo	Explicación
Name=vdata	Especifica el nombre de la variable. Este atributo es requerido.
value=vdata	Especifica el valor que se ha designar a la variable, este atributo es requerido.

Ejemplo 2.30:

El siguiente ejemplo coloca a una variable denominada *product* el valor por defecto *WapSdk* y asignan la etiqueta "Clear" a una tecla. Cuando el usuario selecciona "Clear", el valor de la variable *product* es borrado y se muestra la baraja actual que está actualizada con el valor *vacío*.

```
<card id="Product" title="Choose Product">
        <p>
                Product name:
```

```
            <input title="Product name" name="product"
        value="WapSdk"/>
                    <do type=”accept” label="Clear">
                        <refresh>
                            <setvar name="product" value=""/>
                        </refresh>
                    </do>
        </p>
    </card>
```

2.4.4.2.- Poner nombre a las variables

Los nombres de las variables en WML consisten en letras US-ASCII o signos de puntuación seguidas por un cero o más letras, dígitos o signos. Cualquier otro carácter no es legítimo y da como resultado un error. Hay que señalar también que los nombres de las variables son sensibles al formato de letra (mayúsculas-minúsculas), reconociendo como distinto el mismo nombre en mayúsculas y el nombre en minúsculas.

Se requieren paréntesis en cualquier lugar al final de una variable que no pueda ser inferida desde el contexto que la rodea, por ejemplo, los finales de las variables con carácter ilegítimo tal como un espacio en blanco. Los siguientes ejemplos muestran variables legítimas:

```
This is a $var
This is another $(var).
This is an escaped $(var:e).
Long form of escaped $(var:escape).
Long form of unescape $(var:unesc).
Short form of no-escape $(var:N).
Other legal variable forms: $_X $X32 $Test_9A
```

Un efecto secundario de las normas de análisis es que el signo dólar literal debe ser codificado con un par de entidades de signo de dólar. Una entidad de signo de dólar simple, incluso cuando se especifica como $ produce una sustitución de variables.

- Para incluir un carácter $ en una baraja WML, se debe utilizar la siguiente sintaxis: $$
- Dos signos de dólar en fila son remplazados por un carácter $ simple. Por ejemplo: This is a $$ character.
- Se mostrará como: This is a $ character.
- Para incluir el carácter $ en las cadenas URL-exportadas, se especifica con una forma URL-exportado al %24.

2.4.4.3.- Validación de variables

Dentro de un documento de WML, cualquier cadena seguida de un signo de dólar simple ('$') debe tratarse como una referencia de variable y validada. Cada referencia debe usar una sintaxis de nombre de variable propia. Cada referencia deber ser colocada también dentro del texto de la carta (#PCDATA) o dentro de los valores de atributo *%vdata* o *% href*. La baraja da error si cualquier referencia de variable usa una sintaxis no válida o se coloca en una posición no válida.

Ejemplo 2.31:

Los siguientes ejemplos muestran usos de variables no válidos.

```
<!-- bad variable syntx -->
Balance left is $10.00
<!--bad placement (in the type attribute) -->
<do type="x-$(type)" label="$type">
```

2.4.4.4.- Restricción del contexto de variables

Los terminales puede proporcionar medios al usuario para referenciar y navegar a recursos independientes del contenido actual. Por ejemplo, los terminales pueden proporcionar marcadores o un diálogo de entrada de URL. Siempre que un terminal navegue a un recurso que no sea el resultado de una interacción con el contenido del contexto actual, el terminal debe establecer otro contexto para esa navegación. El terminal puede terminar el contexto actual antes del establecimiento de otro por la otra tentativa de navegación.

2.4.4.5.- Establecimiento de variables

Hay varias maneras de establecer el valor de una variable. Cuando una variable está colocada y ya está definida en el contexto del browser, el valor actual es puesto al día.

El elemento ***setvar*** permite colocar el estado de la variable como un efecto colateral de la navegación. El elemento ***setvar*** puede estar especificado en los siguientes elementos de tarea:

go
prev
refresh

Las variables pueden ser colocadas en las situaciones siguientes:

Los elementos de entrada (*input*) colocan la variable identificada por el atributo *name* para cualquier información introducida por el usuario.

- El elemento ***input*** asigna el texto introducido a la variable.
- El elemento ***select*** asigna el valor actual en el atributo *value* del elemento *option* elegido.

Hay que señalar que la entrada del usuario se escribe para variables cuando el usuario encarga la entrada a los elementos ***input* o *select*.**

El nombre de la variable especificado en el atributo ***name*** (por ejemplo, *location*, "posición") es colocado como un efecto secundario de la navegación. Para más información sobre el procesamiento del elemento ***setvar*.** Cuando se colocan variables se debe tener en cuenta lo siguiente:

- Se pueden cambiar valores de variables colocadas en el leguaje WML con los lenguajes WMLScript y viceversa. Esto indica que el WML y el WMLScript utilizan las mismas variables.
- Se pueden colocar y editar variables en la vista de variables del juego de herramientas WAP.
- Se puede usar el atributo ***newcontext*** del elemento ***card*** para borrar todos los valores de variable del contexto actual del browser.
- Si un atributo ***href*** de un elemento ***go*** tiene referencias de variable, éstas son sólo determinadas frente a las variables del contexto del ***browser*.** Esas referencias de variable no son determinadas frente a cualquier elemento ***setvar*** dentro del elemento ***go*.**
- Si un elemento ***go*** contiene elementos ***setvar*** y el nombre o valor de los elementos ***setvar*** contiene referencias de variable, éstas son determinadas frente a las variables del contexto del *browser*. Esas referencias de variable no se determinan frente a elementos ***setvar*** dentro del elemento ***go*.**

2.4.4.6.- Sustitución de variables

Se pueden sustituir valores de variables dentro de texto formateado (*PCDATA*), valores de opción (*vdata*) y atributos ***href*** en los elementos WML. Sin embargo, hay que señalar que sólo el texto puede ser sustituido, es decir, no es

posible ninguna sustitución de elementos o atributos. La sustitución de valores de variables se produce en tiempo de ejecución en el agente usuario.

Mientras que la sustitución es una mera operación de sustitución de cadenas, no afecta el valor actual de la variable. Si una variable indefinida está referenciada, ello conduce a la sustitución de una cadena vacía.

El valor de las variables puede convertirse en una forma diferente mientras son sustituidas. Una conversión puede ser especificada en la referencia de la variable seguida por dos puntos(:).

La tabla 2.25 resume las conversiones actuales y sus abreviaturas legítimas:

Tabla: 2.25: conversiones actuales y sus abreviaturas legítimas

Variable de Referencia	**Explicación**
$ setvar o $ (setvar)	Sustituye el valor de la variable *setvar*. El terminal del usuario exporta la variable usando la convención de exportación del URL en el contexto apropiado
$ (setvar: e) o $ (setvar: escape)	Sustituye el valor de *servar*, exportando caracteres no alfanuméricos según las convenciones de codificación de URL.
$ (setvar : unesc)	Sustituye el valor de setvar, desexportando caracteres no alfanuméricos según las convenciones de codificación de URL.
$ (setvar :N) o$(setvar : noesc)	Sustituye el valor de setvar, sin exportar caracteres no alfanuméricos.

Hay que destacar que el uso de una conversión durante la sustitución de variables no afecta el valor actual de la variable.

La exportación de URL se detalla en *RFC2396*. Todos los caracteres sensibles léxicamente definidos en WML deben ser exportados, incluidos los

caracteres *URL reserved* (reservado) y *unsafe* (inseguro), como se codifica por *RFC2396*.

Si no se especifica ninguna conversión, la variable es sustituida usando el formato de conversión apropiada al contexto. En los atributos ***onenterbackward*, *onenterforward*, *href*** y ***Sr.***, por defecto se realiza la conversión de escape; en el resto, no se realiza ninguna conversión.

Especificando la conversión *noesc* (sin exportación) deshabilita la exportación sensible al contexto de una variable.

Ejemplo 2.32:

En este ejemplo ilustra el envío de pares de valores de nombre a un servidor Web para que los datos puedan ser enviados desde el cliente:

```
<go method="post" href="http://hostname/servlet/dealer">
<postfield name="make" value="ford"/>
<postfield name="car" value="escort"/>
  </go>
```

2.4.4.7.- Análisis de la sintaxis de la sustitución de variables

La sintaxis de sustitución de variables (por ejemplo, $X) se analiza después de que todo el análisis de XML haya sido completado. Esto implica que toda la sintaxis de variables es analizada después de que las construcciones de XML, tales como elementos y entidades, han sido analizadas. En el contexto del análisis de variables, toda la sintaxis XML tiene precedencia sobre la sintaxis de variables, por ejemplo, la sustitución de entidades sucede antes de que la sintaxis de la sustitución de variables sea analizada.

Ejemplo 2.33:

Los siguientes ejemplos son referencias idénticas a la variable denominada X:

```
$X
&#x24;X
$&#x58;
&#36;&#x58;
```

2.4.5.- ENTRADA DE USUARIO

Las secciones siguientes explican como se manejan las entradas del usuario en WML.

2.4.5.1.- Elemento input

El elemento ***input*** (entrada) especifica un objeto de entrada de texto. Se puede especificar el formato de la entrada del usuario con atributos ***format*** (formato) opcionales. Si una máscara de entrada válida está ligada a un objeto de entrada, el agente usuario debe asegurar que cualquier valor recogido por el objeto de entrada sea conforme a la máscara de entrada ligada. Si la entrada recogida no es conforme a la máscara, el agente usuario no debe entregar esa entrada y debe notificar al usuario que la entrada se rechaza y permitir al usuario para que realice una nueva entrada.

El agente usuario no debe inicializar el objeto de entrada con ningún valor que no esté conforme con la mascara de entrada ligada. En el caso de que los datos de inicialización no sean conformes con la máscara de entrada, el agente usuario debe comportarse como si no hubiera datos de inicialización.

- Elementos contenidos

  Ninguno

- Sintaxis

  Los atributos del elemento input se explican en la tabla 2.26.

Tabla: 2.26: Atributos del elemento input

Atributo	Explicación
name=nmtoden	El atributo *name* especifica el nombre de la variable a colocar con el resultado de la entrada de texto del usuario. El valor de la variable *name* se usa para precargar el objeto de entrada de texto. El atributo *name* es requerido.
value=vdata	El atributo *value* indica el valor por defecto de la variable nombrada en el atributo *name*. Cuando se muestra el elemento y no se fija la variable nombrada mediante *name*, a la variable *name* se le asigna el valor especificado en el atributo *value*. Si la variable *name* ya contiene un valor, el atributo *value* es ignorado. Si el atributo *value* especifica un valor que no es conforme a la máscara de entrada especificada por el atributo *format*, el agente usuario ignora el atributo *value*. En los casos donde no se pueda establecer ningún valor, no se fija la variable *name*.
Type=(text | password)	Este atributo especifica el tipo de área de entrada de texto. Por defecto el tipo es *text*. Se permiten los siguientes valores: text: Un control de entrada de texto. La entrada es repetida de una manera apropiada al terminal del usuario y a la máscara de entrada. Si el valor presentado es conforme a la máscara de entrada existente, el terminal debe almacenar esa entrada inalterada y su totalidad en la variable nombrada en el atributo *name*. Por ejemplo, el terminal no debe disponer la entrada eliminando la guía o ruta del espacio en blanco de la entrada. Si la variable nombrada no es fijada, el terminal debe repetir la cadena vacía de una manera apropiada. Password: Un control de entrada de texto. La entrada de cada carácter es repetida en una forma "oculta", de una manera apropiada al Terminal del usuario. Por ejemplo, los terminales pueden elegir mostrar un asterisco en lugar de un carácter introducido por el usuario. Típicamente, el modo de entrada *password* se indica por una contraseña de entrada u otro dato privado. La entrada

	de *password* es insegura y aplicaciones críticas no deben depender de ella.
Format=cdata	Este atributo especifica una máscara para una entrada de usuario. La cadena consiste en los caracteres de control de la máscara y el texto estático que es mostrado en el área de entrada. Una máscara de entrada solo es válida cuando contiene códigos de formato legítimos. Los terminales deben ignorar máscaras no válidas. Los caracteres de control de formato especifican el formato de los datos que el usuario está esperando introducir. El formato por defecto es "*M*". Los códigos de formato se explican a continuación: A Permite cualquier carácter alfabético en mayúsculas o caracteres de puntuación, es decir, caracteres no numéricos en mayúsculas. a Permite cualquier carácter alfabético o signo de puntuación en minúsculas, es decir, caracteres no numéricos en minúsculas. N Permite cualquier carácter numérico. X Permite cualquier carácter con mayúsculas. x Permite cualquier carácter con minúsculas. M Permite cualquier carácter. El terminal puede asumir que el carácter esté en mayúsculas para propósitos de entrada de datos simples, pero debe permitir entrar cualquier carácter. m Permite cualquier carácter. El terminal puede asumir que el carácter esté en minúsculas para propósitos de entrada de datos simples,
Nf	Permite n caracteres donde n es un numero del 1 al 9; f es uno de los códigos de formato anteriores (excepto **f*) y especifica que clase de caracteres pueden entrar. Este formato puede especificarse una única vez y debe aparecer al final de la cadena de formato.
\c	Muestra el siguiente carácter, *c*, en el campo de entrada. Permite la exportación del código de formato e introducción de caracteres no formateados de modo que puedan ser mostrados dentro del área de entrada. Los caracteres exportados son considerados parte del valor de entrada, y deben ser preservados por el agente usuario.
emptyok=boolean	Si se coloca este atributo como verdadero (*true*) el elemento *input* acepta la entrada aunque una cadena de formato que no esté vacía haya sido especificada. En general, el atributo *emptyok* se indica mediante campos de entrada formateados que son opcionales. Por defecto, los elementos *input* especifican uno *format* que requiere que el usuario introduzca datos de acuerdo con la especificación del

	elemento *format*, que es *emptyok=false* (falso).
size=number	Este atributo especifica la anchura, en caracteres, del área de entrada de texto.
mxalength=number	Este atributo especifica el número máximo de caracteres que el usuario puede entrar en el área de entrada de texto. El valor por defecto de este atributo es un número ilimitado de caracteres.
tittle=vdata	Este atributo especifica un título para el elemento *input*. Es título puede utilizarse en la presentación del objeto.
tabindex=number	El elemento *tabindex* especifica la posición de "*tabbing*" (de fabulación) del elemento actual. Esta posición indica el orden relativo en el cuál los elementos son recorridos cuando se pasa por ellos dentro de una carta WML sencilla. Un valor *tabindex* numéricamente mayor indica un elemento que está después en la secuencia *tab* de un elemento con un valor menor.

Ejemplo 2.34:

El primer ejemplo especifica un elemento *input* que acepta cualquier carácter y muestra la entrada al usuario de una forma que el usuario puede leer. El numero máximo de caracteres que pueden ser introducidos es 32, y la entrada resultante es asignada a la variable denominada X.

```
<input name="X" type="text" maxlength="32"/>
```

El siguiente ejemplo solicita una entrada al usuario y asigna la entrada resultante a la variable *name*. El campo de texto tiene el valor por defecto de "Robert".

```
<input name="name" type="text" value="Robert"/>
```

El ejemplo que se muestra a continuación es una carta que solicita al usuario un nombre, apellido y edad. Hay que tener en cuenta que en el campo edad (*Age*) el usuario puede entrar un numero de dos dígitos.

```
<card>
<p>
```

```
First name: <input type="text" name="first"/><br/>
Last name: <input type="text" name="last"/><br/>
Age: <input type="text" name="age" format="NN"/>
</p>
</card>
```

2.4.5.2.- Elemento select

Seleccionar lista es un elemento de entrada que permite al usuario elegir de una lista de opciones. WML soporta tanto listas de elección simple como de elección múltiple.

El elemento ***select*** permite al usuario escoger de una lista de opciones. Cada opción está especificada por un elemento *option* que tiene una línea de texto formateado. Se puede organizar los elementos *option* dentro de grupos jerarquizados usando el elemento ***optgroup*.**

Se debe incluir uno de los siguientes elementos al menos una vez dentro de un elemento ***select*:**

optgroup
option

En la entrada dentro de una carta que contiene el elemento ***select*,** las opciones iniciales se seleccionan de la siguiente forma:

- Si el atributo ***iname*** existe, los ¡índices de la variable denominada por el iname son usados para seleccionar la opción. Si la variable especificada no está colocada, el índice se presume como 1. Si algún índice es más grande que el número de opciones de la lista de selección, se selecciona la última opción.

- Si el atributo ***iname*** no existe, pero si existe el atributo ***name*****,** el valor de la variable especificada por ***name*** es utilizado para seleccionar las opciones. Si la variable especificada por ***name*** no ha sido fijada o ningún elemento ***option*** tiene un atributo ***value*** emparejando el valor, se selecciona la primera opción.
- Una vez que una opción es seleccionada, la variable denominada por el atributo ***name*** es puesto al día en el valor de la opción.

Los atributos ***name*** **e** ***iname*****,** o ***value*** **e** ***ivalue*** pueden ser especificados. El atributo ***value*** tiene prioridad sobre el ***value***, así como el ***iname*** lo tiene sobre el ***name*****.**

- Elementos contenidos
  - ptgroup +
  - ption +
- Sintaxis

  Los atributos del elemento *select* se explican en la tabla 2.27:

Tabla: 2.27: Atributos del elemento select

Atributo	Explicación
multiple=boolean	Si se coloca este atributo con valor *true*, la lista de selección acepta múltiples selecciones. Si no se coloca, la lista de selección acepta solo una opción de selección simple. El valor por defecto es *false*.
name=nmtoken	El atributo *name* indica el nombre de la variable que toma el valor del elemento elegido. La variable se fija con el valor de la cadena del elemento *option* elegido, el cual se especifica con el atributo *value*. El valor de la variable *name* se utiliza para preseleccionar opciones de la lista de selección.
value=vdata	El atributo *value* indica el valor por defecto de la variable especificada por el atributo *name*. Si la variable especificada por el atributo *name* no tiene un valor cuando se muestra la carta, el terminal le asigna el valor especificado en el atributo *value*. Si la variable *name* ya contiene un valor, se ignora el atributo *value*. Hay que señalar que cualquier aplicación del valor por defecto se realiza antes de que la lista sea

	preselecciónala con el valor de la variable *name*. Si este elemento permite la selección de múltiples opciones, el resultado de la elección del usuario es una lista de todos los valores seleccionados, separados por puntos y aparte. A la variable *name* se le asigna este resultado. Además el atributo *value* se interpreta como una lista de opciones de preselección separados por puntos y aparte.
Iname=nmtoken	El atributo *iname* indica el nombre de una variable que contiene un numero de índice. El terminal del usuario utiliza el número de índice para colocar la opción por defecto. El numero 1 especifica el primer articulo, el numero 2 el segundo, y así sucesivamente. Un numero cero de índice indica que no se selecciona ninguna opción. La numeración del índice empieza en uno y se incrementa monótonamente.
ivalue=vdata	El atributo *ivalue* indica que el ¡índice de el elemento *option* se selecciona por defecto. Si la variable especificada por el atributo *iname* no está fijada cuando la carta es mostrada, se le asigna la entrada seleccionada por defecto. Si la variable ya contiene un valor, se ignora el atributo *ivalue*. Si el atributo *iname* no está especificado, el valor *ivalue* se aplica cada vez que se muestra la carta.
title=vdata	Este atributo especifica un título para el elemento *select*, el cual puede usarse en la presentación de este objeto.
tabindex=number	El elemento *tabindex* especifica la posición "*tabbing*" del elemento actual. Esta posición indica el orden relativo en el cual los elementos son atravesados cuando se va dentro de una carta simple de WML. Un valor *tabindex* numéricamente mayor indica un elemento que va después en la secuencia que un elemento con un valor menor.

Ejemplo 2.35:

El siguiente ejemplo especifica una lista de elección múltiple. Se debe hace notar que:

Las opciones "dog" y "cat" son preselecciónalas si la variable "I" no se ha sido previamente fijada.

Si el usuario eligiera las opciones "cat" y "horse", la variable "X" tomaría el valor "C; H" y la variable "I" !1;3).

```
<wml>
<card>
<p> Please choose <i>all</i> of your favorite animals:
<select name="X" iname="I" ivalue="1;2" multiple="true">
<option value="D">Dog</option>
<option value="C">Cat</option>
<option value="H">Horse</option>
</select> </p>
</card>
</wml>
```

La baraja genera el siguiente interfaz de usuario en el terminal (como se muestra en un teléfono modelo 6150):

Figura 2.6. Una carta con una lista de elección múltiple.

Fuente: [www.wapeton.com**]**

2.4.5.3.- Elemento option

El elemento ***option*** especifica una opción de elección sencilla en un elemento *select*.

- Elemento contenidos
  - onevent *
- Sintaxis

  Los atributos del elemento ***option*** se explican en la tabla 2.28:

Tabla: 2.28: Atributos del elemento option

Atributo	Explicación
value=vdata	Este atributo especifica el valor a usar cuando está fijada la variable *key*. Cuando el usuario selecciona esta opción, el valor resultante especificado en el atributo *value* se utiliza para colocar la variable *name* del elemento *select*. El atributo value puede contener referencias a variables, las cuales son evaluadas antes de que se coloque la variable *name*
title=vdata	Este atributo especifica un título para el elemento *option*, el cual puede usarse en la presentación de este objeto.
Onpick=href	El suceso *onpick* se produce cuando el usuario selecciona o deshace la selección de esta opción. Una lista de opciones de selección múltiple genera un suceso *onpick* siempre que el usuario seleccione o deseleccione esta opción. Una lista de opción de selección simple genera un suceso *onpick* cuando el usuario selecciona esta opción, es decir, no se genera ningún suceso en el caso de deseleccionar cualquier opción previamente seleccionada.

Ejemplo 2.36:

El siguiente ejemplo especifica una lista sencilla de elección simple. Si el usuario eligiera la opción "*dog*", la variable "*X*" tomaría el valor "*D*".

```
<wml>
<card>
<p>
Please choose your favorite animal:
```

```
<select name="X">
<option value="D">Dog</option>
<option value="C">Cat</option>
</select>
</p>
</card>
</wml>
```

La baraja genera el siguiente interfaz usuario en el terminal de usuario (como se muestra en un teléfono modelo 6150):

Figura 2.7. Una carta con una lista de elección simple.

Fuente: [www.wapeton.com**]**

El siguiente ejemplo especifica una lista de elección simple con un valor por defecto.

Hay que señalar lo siguiente:

La option "*dog*" sería preselecciónala si la variable "I" no ha sido colocada previamente (es decir no se le ha asignado ningún valor).
Si el usuario eligiera la opción "*cat*", la variable "I" tomaría e valor de "2".

```
<wml>
<card>
<p>
Please choose your favorite animal:
<select iname="I" ivalue="1">
<option value="D">Dog</option>
<option value="C">Cat</option>
</select>
</p>
</card>
</wml>
```

2.4.5.4.- Elemento optgroup

El elemento ***optgroup*** (‘option group’, grupo de opciones) permite agrupar elementos ***option*** relacionados jerárquicamente. El terminal del usuario utiliza la jerarquía para facilitar la disposición y la presentación.

Se debe incluir uno de los siguientes elementos al menos una vez dentro de un ***optgroup*****:**

optgroup (elemento anidado)
option

- Elementos contenidos
  - (optgroup | option) +
- Sintaxis

Los atributos del elemento ***optgroup*** se explican en la tabla 2.29:

Tabla: 2.29: Atributos del elemento optgroup

Atributo	Explicación
title=vdata	Este atributo especifica un título para el elemento *optgroup*, el cual puede usarse en la presentación de este objeto.

Ejemplo 2.37:

El siguiente ejemplo muestra el uso de grupos de opciones. La carta contiene dos grupos geográficos: Escandinavia y Europa.

```
<wml>
<card id="card1" title="Country">
<p>
Select a country:
<select name="country" multiple="true" tabindex="2">
<optgroup title="Scandinavia">
<option value="den">Denmark</option>
<option value="fin">Finland</option>
<option value="nor">Norway </option>
<option value="swe">Sweden </option>
</optgroup>
<optgroup title="Europe">
<option value ="fra">France </option>
                <option value ="ger">Germany</option>
<option value ="ita">Italy </option>
<option value ="spa">Spain </option>
</optgroup>
</select>
</p>
</card>
</wml>
```

La baraja genera la siguiente interfaz de usuario en el terminal (como se muestra en un teléfono modelo 6110):

Figura 2.8: Una baraja con grupos de opciones.

Fuente: [www.wapeton.com**]**

2.4.5.5.- Elemento fieldset

El elemento ***fieldset*** permite agrupar campos relacionados y texto. El agrupamiento proporciona información al terminal para optimizar la disposición y la navegación. Los elementos ***fieldset*** pueden ser almacenados, proporcionando al usuario medios de especificar el comportamiento de una amplia variedad de dispositivos.

- Elementos contenidos
  - input *
  - select *
  - fieldset * (elemento anidado)
  - a *
  - img *
  - tab *
  - br *
  - (em | strong | b | i | u | big | small) *

- Sintaxis

  Los atributos del elemento ***fieldset*** se explican en la tabla 2.30:

Tabla: 2.30: Atributos del elemento fieldset

Atributo	Explicación
title=vdata	Este atributo especifica un título para el elemento *fieldset*, el cual puede usarse en la presentación de este objeto.

Ejemplo 2.38:

El siguiente ejemplo especifica una baraja de WML que realiza la petición de una identidad básica y de información personal del usuario. Se separa en múltiples sets de campo, mostrando el agrupamiento de campos preferido por el terminal de usuario.

```
<wml>
<card id="info" title="Personal Info">
        <do type="accept" label="Submit">
                <go href="/submit?f=$(fname)&amp;amp;l=$(lname)&amp;amp;
                s=$(sex)&amp;amp;a=$(age)"/>
        </do>
        <p>
                <fieldset title="Name">
                First name: <input type ="text" name="fname"
                maxlength="32"/><br/>
                Last name: <input type="text" name="lname"
                maxlength="32"/><br/>
                </fieldset>

                <fieldset title="Info">
                <select name="sex">
                        <option value="F">Female</option>
                        <option value="M">Male</option>
```

```
                </select>
                <br/>
                Age: <input type="text" name="age" format="*N"/>
                </fieldset>
        </p>
</card>
</wml>
```

2.4.6.- ANCLAJES, IMÁGENES Y TEMPORIZADORES (TIMER)

Las secciones siguientes proporcionan información sobre los elementos ***anchor, image y timer*** de WML.

2.4.6.1.- Elemento anchor

El elemento ***anchor*** (anclaje) especifica la cabeza de un vínculo o vínculo. La cola del vínculo se especifica como parte de otros elementos (por ej. un atributo de nombre de una carta). Es un error englobar vínculos de anclaje.

Se puede utilizar anclajes en cualquier lugar del texto formateado que sea legítimo, excepto para elementos option.

Un vínculo de anclaje debe tener una tarea asociada que especifique el comportamiento cuando se selecciona el elemento anchor. Se debe relacionar uno de los siguientes elementos de tarea a un vínculo:

go
prev
refresh

Hay que señalar que se considera un error sintáctico en WML el especificar m s de una tarea en un elemento anchor.

- Elementos contenidos
  - (go | prev | refresh | br | img) *
- Sintaxis

  Los atributos del elemento ***anchor*** se explican en la tabla 2.31:

Tabla: 2.31: Atributo del elemento anchor

Atributo	Explicación
title=vdata	Este atributo especifica una cadena de texto breve que identifica el vínculo. Para que funcione correctamente con un rango amplio de terminales, se deben mantener los rótulos con seis caracteres.

Ejemplo 2.39:

```
<wml>
<card id="links" title="Links">
        <p>
                This is normal text, but here is a
                <anchor title="LINK">link!
                        <go href="dir/file.wml">
                                <setvar name="var_name" value="var_value"/>
                        </go>
                </anchor>
        </p>
</card>
</wml>
```

La baraja genera el siguiente interfaz de usuario en el terminal:

Figura 2. 9: Una carta que contiene un vínculo.

Fuente: [www.wapeton.com**]**

2.4.6.2.- Elemento a

El elemento ***a*** es una forma corta de un elemento ***anchor*** y es enlazado a una tarea ***go*** sin variables. Por ejemplo, la etiqueta siguiente:

```
<anchor>follow me
<go href="destination.wml"/>
</anchor>
```

Es lo mismo que:

```
<a href="destination.wml"> follow me</a>
```

Es un error sintáctico en WML anidar elementos *a*.

- Elementos contenidos
  - (br | img) *
- Sintaxis

  Los atributos del elemento ***a*** se explican en la tabla 2.32:

Tabla: 2.32: Atributos del elemento a

Atributo	Explicación
title=vdata	Este atributo especifica una cadena de texto breve que identifica el vínculo. Para que funcione correctamente con un rango amplio de terminales, se debe mantener los rótulos con seis caracteres.
Ref.	Este atributo especifica el URI de destino, por ejemplo, el URI de la carta a mostrar.

2.4.6.3.- Elemento img

Se pueden colocar imágenes dentro del texto usando el elemento ***img*** . Las imágenes utilizan la misma disposición que un texto normal.

- Elementos contenidos

  Ninguno.

- Sintaxis

  Los atributos del elemento ***img*** se explican en la tabla 2.33:

Tabla: 2.33: Atributos del elemento img

Atributo	Explicación
alt=vdata	Este atributo especifica una representación textual alternativa para la imagen utilizada cuando la imagen no puede ser mostrada utilizando cualquier otro método, es decir, los contenidos de la imagen no se pueden encontrar, o el terminal no ofrece la imagen por pantalla.
src=ref.	Este atributo especifica el URI para la imagen. El Browser descarga la imagen a partir del URI especificado y lo restituye (lo formatea y lo presenta) cuando el texto está siendo mostrado por pantalla.
localsrc=vdata	Este atributo especifica una representación interna alternativa para la imagen. Esta representación se utiliza si existe.; por otra parte la imagen es cargada desde el URI especificado en el atributo *SRC*, es decir, cualquier atributo *localsrc* especifico tiene prioridad sobre la imagen especificada en el atributo *src*.
Vspace=length	Estos atributos especifican la cantidad de espacio en blanco que se inserta a la izquierda y la derecha (*hspace*) y por encima y por debajo (*vspace*) en una imagen u objeto. El valor por defecto para este atributo no está especificado, pero generalmente es de un tamaño pequeño distinto de cero. Hay que señalar que si se especifica un tamaño como un valor de porcentaje, el tamaño resultante se basa en el espacio horizontal o vertical disponible, no en el tamaño natural de la imagen.
align= (top | middle | bottom)	Este atributo especifica el alineamiento de la imagen dentro del texto con respecto al punto de inserción actual. Puede tener tres posibles valores: **bottom** El extremo inferior (*bottom*) de la imagen está alineado verticalmente con la línea de base actual. Este es el valor por defecto. **middle** El centro (*middle*) de la imagen está verticalmente alineado con el centro de la línea de texto actual. **top** El extremo superior (*top*) de la imagen está

	verticalmente alineado con el extremo superior de la línea de texto actual.
Height=length	Estos atributos especifican el tamaño de una imagen u objeto. El agente usuario puede poner a escala objetos e imágenes emparejando estos valores si lo considera apropiado. Hay que señalar que si se especifica el tamaño como un valor de porcentaje, el tamaño resultante está basado en el espacio horizontal o vertical disponible, no en el tamaño natural de la imagen.

Ejemplo 2.40:

El siguiente ejemplo ilustra el uso de el elemento *img*. Hay que señalar que: *src="bitmaps / moon.wbmp"* se refiere al archivo *moon.wbmp* que está localizado en el directorio bitmaps del mismo dominio que la baraja en la que se incluye el elemento *img*.
Una unidad de espacio en blanco es insertada a la izquierda y derecha (*hspace*) y en la parte superior e inferior (*vspace*) de la imagen.

```
<img src="bitmaps/moon.wbmp" alt="Moon" src="default.wbmp"
hspace="1"
vspace="1"/>
```

2.4.6.4.- Elemento timer

El elemento ***timer*** (temporizador) instrumentaliza un temporizador en la carta WML, el cual se utiliza para procesar el tiempo útil o de inactividad. El ***timer*** se inicializa y empieza a la entrada de la carta y se para cuando la carta es sacada. La entrada de la carta es cualquier acción de usuario o de tarea que conduce a que la carta sea activada, por ejemplo, navegando hacia a la carta. El sacar la carta es la ejecución de alguna tarea. El valor de un ***timer*** disminuye desde el valor inicial, disparando el envío de un suceso intrínseco ***ontimer*** en una transición de un valor de uno a cero. Si el usuario no ha sacado la carta en el tiempo de expiración del ***timer*****,** se envía un suceso intrínseco *ontimer* a la carta.

Hay que señalar que es un error sintáctico en WML tener más de un *timer* en una carta.

El valor de *tiempo de descuento* (tiempo fuera) del elemento ***timer*** (contador de tiempo) se especifica en unidades de una décima parte (1/10) de segundo. Sin embargo, no se debería esperar una resolución del ***timer*** particular. Axial se recomienda que en las aplicaciones donde se requiera una resolución del ***timer*** exacta, se provea al terminal de otros medios para invocar la tarea de *timer*. Si el valor de tiempo de descuento no es un numero entero positivo, el terminal ignora el elemento timer. Un valor de cero deshabilita el timer.

- Elementos contenidos o ninguno.
- Sintaxis

  Los atributos del elemento ***timer*** se explican en la tabla 2.34:

Tabla: 2.34: Atributos del elemento timer

Atributo	Explicación
name=nmtoken	Este atributo especifica el nombre de la variable a colocar con el valor de *timer*. El valor de variable *name* se utiliza para colocar un periodo de tiempo fuera sobre la inicialización del timer. La variable llamada por el atributo *name* será fijada con el valor timer actual cuando la carta sea sacada o cuando el timer expire. Por ejemplo, si el timer expira, la variable *name* toma el valor "0".
value=vdata	Este atributo indica el valor por defecto de la variable denominada en el atributo name. Cuando el timer se inicializa y la variable denominada en el atributo name no está fijada a ningún valor, a la variable name se le asigna el valor especificado en el atributo value. Si la variable name ya contiene un valor, el atributo value se ignora. Si el atributo name no se especifica, El tiempo de descuento se inicializa siempre con el valor especificado en el atributo value. Este atributo es requerido.

Ejemplo 2.41:

La siguiente baraja mostrará un mensaje de texto durante aproximadamente 10 segundos y luego irá al siguiente URI (*next*).

```
<wml>
<card ontimer="/next">
        <timer value="100"/>
        <p>
                Hello World!
        </p>
</card>
</wml>
```

El mismo ejemplo podría ser implementado como:

```
<wml>
<card>
        <onevent type="ontimer">
                <go href="/next"/>
        </onevent>
        <timer value="100"/>
        <p>
                Hello World!
        </p>
</card>
</wml>
```

El siguiente ejemplo ilustra como un *timer* puede inicializar y rehusar un contador. cada vez que la carta es introducida, el timer se reinicializa tomando el valor de la variable *t*. Si no, el *timer* toma el valor de 5 segundos.

```
<wml>
<card ontimer="/next">
        <timer name="t" value="50"/>
        <p>
```

```
            Hello World!
        </p>
    </card>
    </wml>
```

2.4.7.- FORMATEADO DE TEXTO

Esta sección introduce los elementos y construcciones relacionadas con el formateado de texto.

2.4.7.1.- Espacio en blanco

La forma en que WML maneja el espacio en blanco y el cambio de línea se basa en el lenguaje XML y asume por defecto las normas de manejo del espacio en blanco. Esto implica que el terminal de usuario en WML convierte los espacios contiguos múltiples, "intros" y líneas en espacios simples entre palabras.

2.4.7.2.- Elementos de énfasis

Los elementos de ***énfasis*** especifican información de marcas para resaltar el texto. Las etiquetas de ***énfasis*** se explican en la tabla 2.35:

Tabla: 2.35: Etiquetas de énfasis

Etiqueta	Explicación
Em	Restituye (es decir, formatea y presenta) con énfasis
Strong	Da el formato y muestra el texto con fuerte énfasis.
I	Formatea y muestra el texto con fuente itálica.
B	Formatea y muestra el texto en negrita.
U	Formatea y muestra el texto subrayado.
Big	Formatea y presenta el texto en una fuente de tamaño grande.
Small	Formatea y presenta el texto con una fuente pequeña.

Se utilizan las etiquetas ***strong*** y ***em*** donde sea posible. No se recomienda utilizar etiquetas ***b*** **,** ***i*** y ***u*** excepto donde se requiera control explícito sobre la presentación del texto.

Ejemplo 2.42:

A continuación WML ilustra el uso de las etiquetas de Énfasis de texto.

```
<wml>
<card id="card1">
        <p>
                <em>
                A
                <u>
                   Demonstration
                </u>
                of Nokia's
                   <i>
                     <strong> Wireless Application Protocol<br/> </strong>
                   </i>
                <b> Toolkit</b>
                </em>
        </p>
</card>
</wml>
```

La baraja genera el siguiente interfaz usuario en el terminal:

Figura 2.10: Carta con formato de texto.

Fuente: [www.wapeton.com**]**

Hay que señalar que el terminal sólo puede mostrar unas pocas líneas o renglones cada vez, así se debe ir avanzando para ver las últimas líneas de la carta.

2.4.7.3.- Elemento br

El elemento ***br*** establece el comienzo de una nueva línea. El terminal del usuario debe terminar la línea actual y continuar en la línea siguiente. Los terminales se esfuerzan en proporcionar el elemento ***br*** en tablas. (Ver "el elemento tabla" un poco más adelante).

- Elementos contenidos
  - Ninguno.

2.4.7.4.- Elemento p

El elemento ***p*** establece los parámetros de alineación y de "wrap" (embalaje o ajuste en la presentación) de línea o renglón de texto para un párrafo. Si no se especifica la alineación del texto, por defecto es a la izquierda. Si no se especifica el modo de ajuste de línea para su presentación (embalaje), es el mismo que el del párrafo anterior de la carta actual. Los párrafos vacíos, como un elemento vacío o un elemento con un espacio vacío sin utilidad, deben ser considerados como inútiles e ignorados por las pantallas de los terminales de usuario. Los párrafos inútiles no afectan al de embalaje o presentación de línea. Si el primer elemento ***p*** de una carta no especifica el modo de alineación ni ajuste de líneas, se establece por defecto el modo inicial de la carta. El agente usuario debe insertar un cambio de línea dentro del texto entre los elementos ***p*** inútiles.

Se pueden eliminar los párrafos inútiles antes del envío del documento al terminal.

WML tiene dos modos de ajuste de líneas para los terminales: cambiando de línea (o embalando) o sin cambiar de línea. El tratamiento de una línea demasiado larga para que se ajuste a la pantalla se especifica por los siguientes modos de ajuste de línea.

Si el modo =**"*wrap*"** está especificado, la línea es "embalada" o ajustada para su presentación en múltiples líneas. En este caso, los cambios de línea deben ser insertados dentro del texto de forma apropiada para la presentación en un dispositivo individual.

Si el modo=**"*unwrap*"** está especificado, la línea no es ajustada automáticamente. En este caso, el agente usuario debe proporcionar un mecanismo para visualizar las líneas que faltan por mostrar, tal como el que se vayan desplegando horizontalmente o algún otro mecanismo especifico del terminal.

Cualquier espacio entre palabras es un legítimo punto de cambio de renglón. La entidad de espacio de no cambio de línea (o) indica que el terminal no debe tratar el espacio como un espacio entre palabras. Se recomienda que se utilice   para prevenir cambios de línea no deseados. La entidad de carácter de guión opcional (o) indica una posición que puede ser usada por el terminal para cambiar de línea. Si el cambio de línea se produce en un guión opcional, el terminal inserta un carácter de guión (-) al final de la línea. En todas las demás operaciones, la entidad de guión opcional es ignorada. Hay que señalar también que el terminal puede ignorar los guiones opcionales cuando se formatean líneas de texto.

- Elementos contenidos
  - a
  - anchor
  - br
  - do

- em | strong | b | i | u | big |small
- fieldset
- img
- input
- select
- table

- Sintaxis

  Los atributos que soporta el elemento ***p*** se indican en la tabla 2.36:

Tabla: 2.36: Atributos del elemento p

Atributo	Explicación
align= (left | right | center)	Este atributo especifica el modo de alineación del texto en el párrafo. Se puede alinear de forma centrada, a la izquierda o a la derecha. La alineación por defecto es a la izquierda. Si no se especifica explícitamente, la alineación de texto se toma la alineación por defecto.
mode= (wrap | nowrap)	Este atributo especifica el modo de ajuste de línea para los párrafos siguientes. *Wrap* especifica el modo de texto con cambio de línea. *Nowrap* el modo sin cambio de línea.

Ejemplo 2.43

El siguiente ejemplo muestra un texto centrado y un texto justificado a la izquierda y la diferencia entre un texto con ajuste de línea y otro sin él.

```
<p align="center">
        centered text
</p>
<p align="left">
        left-justified
</p>
<p mode="wrap">
        This text is wrapped to the next line.
```

```
</p>
<p mode="nowrap">
        This text is truncated.
</p>
```

2.4.7.5.- Elemento table

El elemento ***table*** (tabla) se utiliza junto con los elementos *tr* y *td* para crear juegos de columnas de texto e imágenes en una carta. Se pueden anidar los elementos ***table*****.** Los elementos ***table*** determinan la estructura de las columnas. Estos elementos separan el contenido en columnas, pero no especifican la anchura de columna o el espacio entre las mismas. El terminal deber realizar la presentación de la información de la tabla de una manera apropiada al dispositivo.

La alineación del texto y las imágenes dentro de una columna se especifica con el atributo ***align*****.** Se puede alinear el contenido de una columna centrado, a la izquierda o a la derecha. El valor del atributo ***align*** es interpretado como una lista de diseños de alineación, uno por cada columna. Se puede especificar la alineación centrada con el valor "*C*", la izquierda con el valor "*L*" y la derecha con el valor "*R*". El primer diseño de la lista se aplica a la primera columna, el segundo a la segunda y así sucesivamente. La alineación por defecto se aplica para cualquier columna para la cual se haya omitido el diseño de alineación. Para idiomas que se escriben de izquierda a derecha, la alineación por defecto es a la izquierda. Para los idiomas que se escriben de derecha a izquierda, la alineación por defecto es a la derecha.

Se debe usar el atributo ***columns*** (columnas) para especificar el numero de columnas por fila. El terminal crea una fila con el numero exacto de columnas especificadas en el valor del atributo ***columns***. Si el numero actual de columnas en una fila es menor que el valor especificado en el atributo ***columns*****,** la fila debe ser rellenada efectivamente con columnas vacías. La orientación de la tabla

depende del idioma: para idiomas escritos de izquierda a derecha, la columna izquierda es la primera columna de la tabla. Las columnas se añaden en el lado derecho de una fila para rellenar las tablas de izquierda a derecha. Las columnas se añaden al lado izquierdo de una fila para rellenar tablas de derecha a izquierda.

Si el número real de columnas de una fila es mayor que el valor especificado por el atributo columns, las columnas extra de la fila deben agregarse en la ultima columna, de forma que la fila contenga exactamente el numero de columnas especificadas. Un espacio entre palabras simple debe ser insertado entre dos celdas que están siendo agregadas.

Dependiendo de las características disponibles para mostrar la información, el terminal puede crear columnas alineadas para cada tabla, o puede usar un juego simple de columnas alineadas para todas las tablas de la carta. Para asegurar la anchura más pequeña al mostrar la información, el terminal debe determinar la anchura de cada columna desde la máxima anchura de texto e imágenes en esa columna. Un canalón de anchura no nula debe usarse para separar cada columna no vacía.

- Elementos contenidos

  o tr +

- Sintaxis

  Los atributos que soporta el elemento ***table*** se muestran en la tabla 2.37:

Tabla: 2.37: Atributos del elemento table

Atributo	Explicación
title=vdata	Este atributo especifica un título para el elemento *table*, el cual puede utilizarse en la presentación de este objeto.
align=cdata	Este atributo especifica la disposición del texto y las imágenes dentro de las columnas de un conjunto de filas. Se puede alinear los contenidos de una columna centrados, alineados a la izquierda, o a la derecha. El valor

	del atributo se representa por una lista de diseños, uno para cada columna. Se puede centrar con el valor "C", alinear a la izquierda con el valor "L" y a la derecha con el valor "R".
columns=number	Este atributo especifica el número de columnas para cada conjunto de filas. El terminal debe crear un conjunto de filas con el número de columna especificado por el valor del atributo. Es un error sintáctico en WML especificar el valor de cero ("0").

Ejemplo 2.44:

El siguiente ejemplo contiene una carta con una tabla que tiene seis filas y tres celdas de datos en cada fila. De las filas dos a la 6, ambas incluidas, la segunda celda contiene un mapeo de bit inalámbrico (wireless bitmap).

```
<wml>
<card id="card1" title="Weather Forecast">
    <p>
        <table columns="3">
        <tr>
            <td>Day</td><td>Wthr</td><td>Temp</td>
        </tr>
        <tr>
            <td>M 6/7</td><td><img src="rainy.wbmp"
        alt="rain"/></td>
            <td>25' C</td>
        </tr>
        <tr>
            <td>T 6/8</td><td><img src="partcldy.wbmp" alt="part
            cldy"/></td>
            <td>27' C</td>
        </tr>
        <tr>  <td>W 6/9</td><td><img src="cloudy.wbmp"
            alt="cloudy"/></td>
```

```
                <td>24' C</td>
            </tr>
            <tr> <td>T 6/10</td><td><img src="rainy.wbmp"
                alt="rainy"/></td>
                <td>28' C</td>
            </tr>
            <tr>
                <td>F 6/11</td><td><img src="sunny.wbmp"
                alt="sunny"/></td>
                <td>29' C</td>
            </tr>
            </table>
        </p>
</card>
</wml>
```

Esta baraja genera el siguiente interfaz usuario en la pantalla del terminal (mostrado en un teléfono modelo 6110).

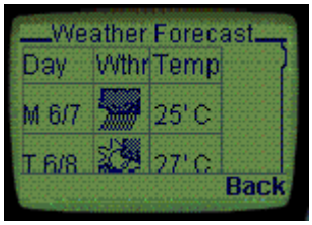


Figura 2. 11: Carta con formato de tabla

Fuente: [www.wapeton.com**]**

2.4.7.6.- Elemento tr

El elemento ***tr*** se utilizada como un contenedor para mantener una fila sencilla en una tabla. Las filas de la tabla puede estar vacías (por ejemplo, todas

las celdas de la fila están vacías). Las filas vacías son significativas y no deben de ser ignoradas.

- Elemento contenidos
  - td +

2.4.7.7.- Elemento td

El elemento ***td*** se utiliza para contener los datos de una celda en una tabla sencilla dentro de una fila de la tabla. Las celdas de los datos de la tabla pueden estar vacías. Las celdas vacías son significativas, y no deben ser ignoradas. El terminal debería ser capaz de tratar con celdas de datos de múltiples líneas que pueden resultar de la utilización de imágenes o cambios de líneas.

- Elementos contenidos
  - a
  - anchor
  - br
  - em | strong | b | i | u | big | small
  - img

2.5.- CONFIGURACION DEL SERVIDOR

Cuando arriva una solicitud al "Servidor de Páginas", éste inspecciona los "Headers" HTTP de la solicitud , en estos "Headers" va incluido un parámetro llamado Accept el cual indica *el tipo información que puede recibir el navegador* que esta realizando la solicitud, **TODO** aparato inalámbrico esta diseñado para enviar: text/**vnd.wap.wml**, este parámetro es una (de las pocas) *manera de distinguir si el navegador es inalámbrico (WML)* o el clásico Navegador (HTML) "Netscape" o "Explorer". Ya conocido el tipo de Navegador el servidor debe enviar el contenido pertinente, esto es, si la requisición es de un "Navegador

Inalámbrico" enviar información en WML, de no ser así, enviar el documento típico de HTML.

Para que el servidor de HTTP sepa que debe servir las páginas *.wml como páginas WAP debemos indicárselo, de lo contrario el servidor responderá como que el tipo de fichero es de texto.

Cualquier servidor web puede adaptarse para mostrar páginas WML. Tan sólo es necesario configurar el servidor para que asocie las extensiones wml y wmls a los correspondientes tipos MIME, de manera equivalente a como están configuradas las extensiones htm y html. Si definimos que el archivo índice se llame index.wml, entonces será suficiente con marcar www.nombre-de-mi-portal.com para llegar a la página principal, que previamente la habremos nombrado como index.wml. Esta página habitualmente tendrá un menú para llegar a otras páginas WML dentro del mismo web.

Algunos servidores como el de Google o el de PortalWap están configurados para detectar el origen del visitante. Si se utiliza un navegador WAP, se mostrará la correspondiente página WML. En cambio, si se accede a estos sitios web desde un navegador estándar (como Explorer o Netscape), aparecerá una página HTML.

¿Se puede ver una página web HTML desde un WAP? En principio no se puede, ya que sólo se muestran las páginas específicamente preparadas para WAP. Sin embargo, existen conversores que transforman estas páginas a formato WML. Por ejemplo, el navegador UP Phone, que está implementado en los teléfonos Motorola Timeport, incluye un conversor automático. Si nuestro navegador no dispone de conversor, también es posible utilizar conversores online, como el de Google (desde la página WAP de Google.com se pueden ver páginas HTML).

Para decirle al servidor que las páginas de extensión wml con páginas WAP existen los tipos MIME, con estos tipos indicamos al servidor como se debe comunicar con el cliente cuando le solicitan una página wml.

La configuración de tipos MIMES, si su servidor no está configurado ya, es la siguiente:

Tabla: 2.38: Tipos MIME para WML

Contenido	**Tipo MIME**	**Extensión**
WML Source	text/vnd.wap.wml	wml
Compiled WML	Application/vnd.wap.wmlc	wmlc
WMLScript source	text/vnd.wap.wmlscript	wmls
Compiled WMLScript	Application/vnd.wap.wmlscriptc	wmlsc
Wireless Bitmap	image/vnd.wap.wbmp	wbmp

Los servidores WAP son los encargados de conectar las terminales móviles con los servidores de contenidos, que pueden ser servidores de Internet o de alguna Intranet privada. Estos verifican los encabezamientos http y el contenido WML, y los codifican en formato binario, para luego conformar la respuesta WAP que se envía a la terminal móvil.

Las terminales se conectan con el servidor a través de un servicio de mensajes cortos (SMS) o de transferencia de datos por circuitos conmutados (CSD). El servidor WAP contiene adaptadores de portadoras para los diferentes servicios y un adaptador UDP que lo conecta con redes IP. La mayoría de los servidores WAP pueden funcionar como servidores de contenido pues poseen una interfaz abierta para que se ejecuten aplicaciones como Serverlets, CGI, entre otras.

2.5.1.- CONFIGURACIÓN DEL SERVIDOR APACHE.

Para añadir esos tipos al servidor Apache. Hay que hacer lo siguiente:

1. Editar el archivo de configuración “httpd.conf” y añadir las siguientes líneas en la sección AddType:
   AddType text/vnd.wap.wml .wml
   AddType text/vnd.wap.wmlscript .wmls
   AddType aplication/vnd.wap.wmlc .wmlc
   AddType aplication/vnd.wap.wmlscriptc .wmlsc
   AddType image/vnd.wap.wbmp .wbmp
2. Guardar los cambios en el fichero de configuración y re-lanzar el dominio httpd.

2.5.2.- CONFIGURACIÓN DEL MICROSOFT IIS

Para añadir esos tipos al IIS, hay que hacer lo siguiente:

1. En la consola del servidor, accede a Management console.
2. Para añadir un tipo MIME a un directorio: clic con el botón derecho del ratón sobre el directorio.
3. Selecciona la etiqueta http headers.
4. clic sobre el botón de file types, abajo.
5. clic sobre New type. Indica la extensión y el tipo (listados más arriba)

2.5.3.- CONFIGURACIÓN DEL PHP.INI

Una vez contemplando el acceso inalámbrico a su sitio, seguramente ya utiliza aplicación de servidor_como PHP para generar contenido dinámico, sin embargo, estas aplicaciones de Servidor fueron diseñadas para regresar contenido en HTML.

Para configurar que PHP reporte el contenido wml que requerimos hay que hacer lo siguiente:

1. Editar el archivo de configuración “php.ini”
2. Buscar la línea default_mimetype = "text/html" y comentarla anteponiendo un “;” de la siguiente manera:

;default_mimetype = "text/html"

3. Guardar los cambios en el fichero de configuración y re-lanzar el demonio httpd

CAPITULO III

3. METODOLOGÍA XPOOHD-WAP

3.1.- IMPORTANCIA DE LA METODOLOGÍA XPOOHD-WAP

Las aplicaciones hipermedia han evolucionado en los últimos años y se han concentrado mayormente en la WEB. Las antiguas aplicaciones distribuidas en cd's dieron lugar a aplicaciones dinámicas, de constante actualización e incluso personalizables, capaces de adaptarse a los tipos de usuarios y en casos avanzados, a cada usuario en particular. Estas características encuentran el

medio ideal en las aplicaciones WAP, ya que de otra forma sería costoso su mantenimiento y evolución.

La complejidad del desarrollo ocurre a diferentes niveles: dominios de aplicación sofisticados (financieros, médicos, geográficos, etc.); la necesidad de proveer acceso de navegación simple a grandes cantidades de datos multimediales y por último la aparición de nuevos dispositivos para los cuales se deben construir interfaces WAP fáciles de usar. Esta complejidad en los desarrollos de software sólo puede ser alcanzada mediante la separación de los asuntos de modelización en forma clara y modular.

La metodología XPOOHD-WAP ha sido utilizada para diseñar diferentes tipos de aplicaciones hipermedia como galerías interactivas, presentaciones multimedia y como veremos en este capitulo, aplicaciones WAP. El éxito de esta metodología es la clara identificación de los tres diferentes niveles de diseño en forma independiente de la implementación.

La justificación de tanto trabajo puede encontrarse en cualquier aplicación que requiera navegación: en términos de programación orientada a objetos, si los elementos por los que se navega son los del diseño conceptual se estaría mezclando la funcionalidad hipermedia con el comportamiento propio del objeto. Por otro lado, si los nodos de la red de navegación tienen la capacidad de definir su apariencia, se estaría limitando la extensión de la aplicación para ofrecer nuevas presentaciones del mismo elemento y eventualmente se estaría dificultando la personalización de la interfaz.

Es necesario, entonces, mantener separadas las distintas decisiones de diseño según su naturaleza (conceptual, navegacional, de interfaz) y aplicar las tecnologías adecuadas a cada capa en el proceso de implementación.

Esta idea de desarrollo será observada en detalle en el siguiente capitulo, donde con ayuda de un ejemplo concreto se podrá apreciar con claridad la real importancia del uso de una metodología de diseño.

3.2.- INTRODUCCIÓN A XPOOHD-WAP

La siguiente metodología es una adaptación a una propuesta metodológica que desarrollaron Luis León y Desireé Romero 2003. Se trata de XPOOHD, una propuesta que ofrece las ventajas y beneficios de la metodología XP (extreme Programing) para llevar a cabo el desarrollo de un sistema computacional y la metodología OOHDM (Object Oriented Hipermedia Development Model), que considera el diseño de la aplicación hipermedia la fase fundamental de desarrollo, esta propuesta es denominada XPOOHD-WAP (*eXtreme Programing Object Oriented Hypermedia Development for Applications WAP)*.

3.3.- FASES DE LA METODOLOGÍA XPOOHD-WAP

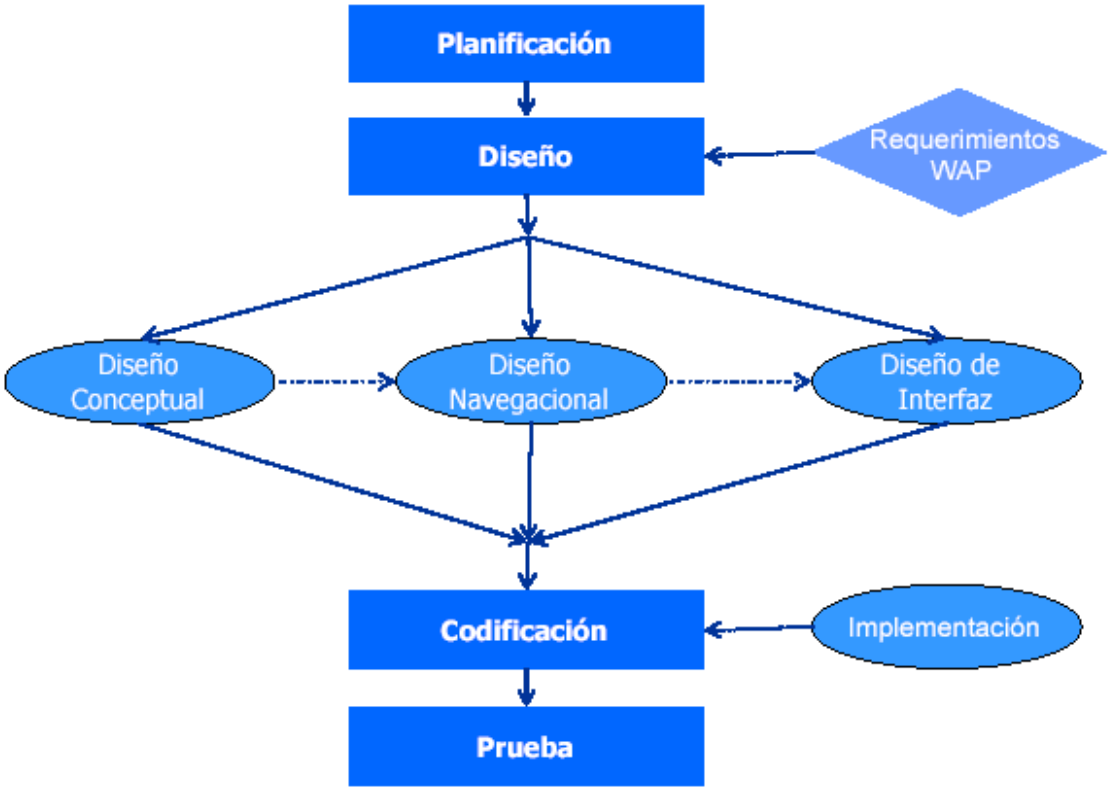


Figura 3.1: Grafico de la metodología XPOOHD-WAP

Fuente: [www.wapeton.com**]**

3.3.1.- PLANIFICACION

En la fase de planificación se realizan las siguientes actividades:

- Elicitación de requerimientos a través de las historias de usuarios
    - Versionando historias de usuarios
- Estudio de factibilidad

- Determinación de la factibilidad
    - Estudio de factibilidad
    - Objetivo de un estudio de factibilidad
    - Factibilidad técnica
    - Factibilidad económica
    - factibilidad operativa.
    - Definición de objetivos
- Recursos de los estudios de factibilidad
    - Factibilidad operativa
    - Factibilidad técnica
    - Factibilidad económica
- Presentación de un estudio de factibilidad

3.3.1.1.- Elicitación de requerimientos a través de las historias de usuarios.

Las historias de usuario tienen el mismo propósito que los casos de uso las escriben los propios clientes tal y como ven ellos las necesidades del sistema.

Las historias de usuario son similares al empleo de escenarios con la excepción de que no se limitan a la descripción de la interfaz de usuario también conducirán el proceso de creación de los test de aceptación (empleados para verificar que las historias de usuario han sido implementadas correctamente).

Existen diferencias entre estas y la tradicional especificación de requisitos. La principal diferencia es el nivel de detalle. Las historias de usuario solamente proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario.

El primer problema que encontramos al intentar trasladar las historias de usuario desde las fichas en papel a un sistema informático radica en la necesidad de definir que información deben contener las mismas y en que forma se va a representar dicha información.

Respecto al primer punto apenas existen indicaciones que orienten a la hora de crear las historias de usuario. Beck afirma que la única información que se debe recoger es el nombre de la historia y una descripción de la misma. Aun así en el mismo libro se incluyen plantillas para las historias de usuarios cuyo uso se ha desaconsejado. Otros por su parte mantienen que se debería registrar la cantidad mínima de información que sea útil para el proyecto de desarrollo.

Creemos que la sola descripción de la historia de usuario no es suficiente pero manteniendo la simplicidad de XPOOHD-WAP cualquier otra información asociada debe estar justificada con respecto al beneficio que aporte. Es mas la información de una historia de usuario podría variar y ajustarse a las características específicas del proyecto. Por otro lado, el usuario no se siente cómodo ante el trabajo adicional que supone decidir cuales son los datos relevantes para conformar sus historias de usuario prefiriendo que se le ofrezca una plantilla a rellenar.

Una vez conocida la información que compondrá las historias de usuario debemos elegir el medio más adecuado para representar estos datos. Dado que se va a realizar control de versiones sobre las historias de usuario es recomendable utilizar un formato en el que la información de las historias se almacene de forma textual. Esta representación permitirá al sistema de control de

versiones mostrar las diferencias entre una versión y otra de forma inteligible, los datos que se han considerado para las historias de usuarios son los siguientes.

- story id, identificador univoco para la historia.
- name, nombre de la historia de usuario.
- date, fecha en la que la historia de usuario fue creada.
- customer, cliente que ha confeccionado la historia de usuario.
- priority, prioridad asignada por el cliente a la historia de usuario.
- dependence, dependencia con otras historias de usuario establecida por el personal de desarrollo.
- estimation, estimación realizada por el grupo de desarrollo del esfuerzo necesario para implementar la historia de usuario.
- risk, riesgo asociado a la implementación de la historia de usuario de acuerdo a la experiencia del equipo de desarrollo.
- release, release que incorpora la funcionalidad especificada por la historia de usuario.
- iteration, iteración en la que el cliente desea que se implemente la historia de usuario.
- upgrade, identificador de la historia de usuario que actúa como mejora o corrección de la actual.
- base, identificador de la historia de usuario modificada o ampliada por la actual.
- description, texto explicativo de la historia de usuario.

Aunque algunos campos pueden parecer complejos a la vista del cliente sobre todo los campos *upgrade* y *base*, cabe reseñar que el cliente solo debe rellenar los campos *name, priority, release, iteration* y *description*. El equipo de desarrollo se encargará de los campos *dependence, risk* y *estimation.*

3.3.1.1.1.- Versionando historias de usuario

Cuando el usuario se enfrenta a un nuevo requisito debe registrarlo como una nueva historia de usuario, pero cuando se trata de modificar un requisito existente apenas se encuentra información en la bibliografía sobre como abordar el problema. Con un sistema de control de versiones lo más sencillo parece ser modificar directamente la historia de usuario y registrar el cambio en el repositorio. Pero actuar de esta forma tiene una repercusión negativa en la estimación de esa historia de usuario que debe ser realizada de nuevo. Siguiendo la filosofía XPOOHDM-WAP un cambio en los requisitos se podría registrar como una nueva historia de usuario, si se adopta esta opción y queremos mantener la trazabilidad entre los requisitos nos vemos obligados a almacenar referencias entre la antigua historia de usuario y su correspondiente modificación.

Para enfrentar con éxito esta falta de información referente al método de desarrollo creamos una pequeña guía que sirva de orientación al usuario cuando se enfrente con este problema. Tras estudiar el conjunto de historias de usuario decidimos no limitarnos a una sola forma de modificar las historias sino seguir las siguientes pautas:

- Si la modificación que deseamos realizar no implica cambios en el contenido de la historia entonces crearemos una nueva tarea de programación para cubrir esa funcionalidad. De esta forma registramos las mejoras y modificaciones detectadas por el equipo de desarrollo pero que no afectan a los requisitos iniciales registrados en la historia de usuario
- Si la modificación implica cambios en la información de la historia tomaremos la decisión de acuerdo a la granularidad del cambio.
- Si la modificación es de una granularidad similar a la propia historia de usuario implica que nos encontramos ante una nueva historia de

usuario. En este caso se necesitan mantener referencias entre las dos historias para disponer de trazabilidad.

- Por otra parte si el cambio no alcanza el nivel de una nueva historia se modificara la historia de usuario afectada y será el control de versiones el encargado de registrar ese cambio.

3.3.1.2.- Estudio de factibilidad

3.3.1.2.1.- Determinación de la Factibilidad

Factibilidad se refiere a la disponibilidad de los recursos necesarios para llevar a cabo los objetivos o metas señalados, la factibilidad se apoya en 3 aspectos básicos:

- Operativo.
- Técnico.
- Económico.

El éxito de un proyecto esta determinado por el grado de factibilidad que se presente en cada una de los tres aspectos anteriores.

3.3.1.2.1.1.- Estudio de Factibilidad.

Sirve para recopilar datos relevantes sobre el desarrollo de un proyecto y en base a ello tomar la mejor decisión, si procede su estudio, desarrollo o implementación.

3.3.1.2.1.2.- Objetivo de un Estudio de Factibilidad.

- Auxiliar a una organización a lograr sus objetivos.
- Cubrir las metas con los recursos actuales en las siguientes áreas.

3.3.1.2.1.3.- *Factibilidad Técnica.*

- Mejora del sistema actual.
- Disponibilidad de tecnología que satisfaga las necesidades.

3.3.1.2.1.4.- *Factibilidad Económica.*

- Tiempo del analista.
- Costo de estudio.
- Costo del tiempo del personal.
- Costo del tiempo.
- Costo del desarrollo / adquisición.

3.3.1.2.1.5.- *Factibilidad Operativa.*

- Operación garantizada.
- Uso garantizado.

3.3.1.2.1.6.- *Definición de objetivos.*

La investigación de factibilidad en un proyecto consiste en descubrir cuales son los objetivos de la organización, luego determinar si el proyecto es útil para que la empresa logre sus objetivos. La búsqueda de estos objetivos debe contemplar los recursos disponibles o aquellos que la empresa puede proporcionar nunca deben definirse con recursos que la empresa no es capaz de dar.

En las empresas se cuenta con una serie de objetivos que determinan la posibilidad de factibilidad de un proyecto sin ser limitativos. Estos objetivos son los siguientes:

- Reducción de errores y mayor precisión en los procesos.
- Reducción de costos mediante la optimización o eliminación de recursos no necesarios.
- Integración de todas las áreas y subsistemas de la empresa.

- Actualización y mejoramiento de los servicios a clientes o usuarios.
- Aceleración en la recopilación de datos.
- Reducción en el tiempo de procesamiento y ejecución de tareas.
- Automatización optima de procedimientos manuales.

3.3.1.2.2.- Recursos de los estudios de Factibilidad

La determinación de los recursos para un estudio de factibilidad sigue el mismo patrón considerado por los objetivos vistos anteriormente, el cual deberá revisarse y evaluarse si se llega a realizar un proyecto.

Estos recursos se analizan en función de tres aspectos:

- Operativos.
- Técnicos.
- Económicos.

3.3.1.2.2.1.- Factibilidad Operativa.

Se refiere a todos aquellos recursos donde interviene algún tipo de actividad (Procesos), depende de los recursos humanos que participen durante la operación del proyecto. Durante esta etapa se identifican todas aquellas actividades que son necesarias para lograr el objetivo y se evalúa y determina todo lo necesario para llevarla a cabo.

3.3.1.2.2.2.- Factibilidad Técnica.

Se refiere a los recursos necesarios como herramientas, conocimientos, habilidades, experiencia, etc., que son necesarios para efectuar las actividades o procesos que requiere el proyecto.

Generalmente nos referimos a elementos tangibles (medibles). El proyecto debe considerar si los recursos técnicos actuales son suficientes o deben complementarse.

3.3.1.2.2.3.- Factibilidad Económica.

Se refiere a los recursos económicos y financieros necesarios para desarrollar o llevar a cabo las actividades o procesos y/o para obtener los recursos básicos que deben considerarse son el costo del tiempo, el costo de la realización y el costo de adquirir nuevos recursos.

Generalmente la factibilidad económica es el elemento más importante ya que a través de el se solventan las demás carencias de otros recursos, es lo más difícil de conseguir y requiere de actividades adicionales cuando no se posee.

3.3.1.2.3.- Presentación de un estudio de Factibilidad.

Un estudio de factibilidad requiere ser presentado con todas las posibles ventajas para la empresa u organización, pero sin descuidar ninguno de los elementos necesarios para que el proyecto funcione. Para esto dentro de los estudios de factibilidad se complementan dos pasos en la presentación del estudio:

- Requisitos Óptimos.
- Requisitos Mínimos.

El primer paso se refiere a presentar un estudio con los requisitos óptimos que el proyecto requiera, estos elementos deberán ser los necesarios para que las actividades y resultados del proyecto sean obtenidos con la máxima eficacia.

El segundo paso consiste en un estudio de requisitos mínimos, el cual cubre los requisitos mínimos necesarios que el proyecto debe ocupar para

obtener las metas y objetivos, este paso trata de hacer uso de los recursos disponibles de la empresa para minimizar cualquier gasto o adquisición adicional.

Un estudio de factibilidad debe representar gráficamente los gastos y los beneficios que acarreará la puesta en marcha del sistema, para tal efecto se hace uso de la curva costo-beneficio.

3.3.1.3.- Elaboración del modelo general de casos de uso

3.3.1.3.1.- Diagramas de Casos de Uso

Un diagrama de casos de uso (Use Case Diagram) es una representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones. Todo sistema tiene como mínimo un diagrama Main Use Case, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso).

Un diagrama de casos de uso muestra, por tanto, los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones). Se muestra como ejemplo los casos de uso de una máquina de café (figura 3.2).

Un caso de uso, denotando un requisito funcional exigido al sistema, se representa en el diagrama por una elipse y un nombre significativo. En el caso del ejemplo se tienen como casos de uso de la máquina de café *RecibirDinero*, *PedirAzucar*, *PedirProducto*, *DarVueltas* y *Cancelar*.

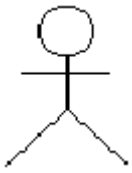


Figura 3.2: Representación de un actor: *stickman*
Fuente: [http://www-gris.det.uvigo.es/~avilas/UML/node25.html**]**

Un actor es una entidad que utiliza alguno de los casos de uso del sistema. Se representa mediante el símbolo de la figura 3.2 acompañado de un nombre significativo, si es necesario. En el ejemplo observamos un único actor representando al usuario de la máquina de café, aunque en un modelo de casos de uso más detallado se podría incluir otro actor para responsable del mantenimiento de la máquina. Un actor en un caso de uso representa un rol que alguien o algo podría desempeñar y no un alguien o algo específico.

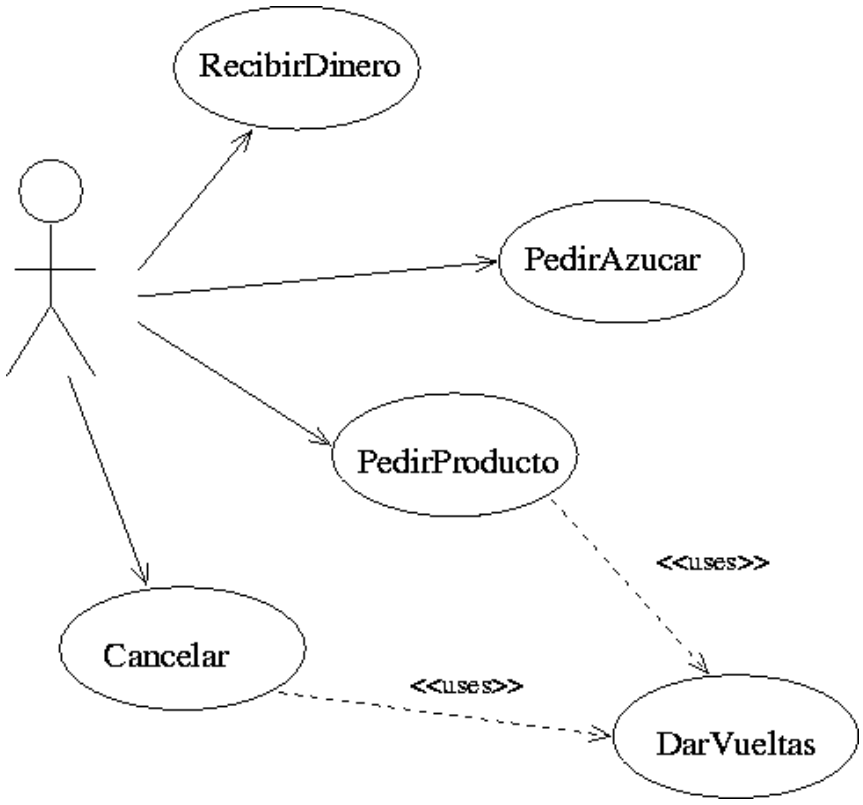


Figura 3.3: Representación de un actor: *stickman*
Fuente: [http://www-gris.det.uvigo.es/~avilas/UML/node23.html**]**

Entre los elementos de un diagrama de casos de uso se pueden presentar tres tipos de relaciones, representadas por líneas dirigidas o no entre ellos:

‘’comunica’’ (<<communicates>>): Relación (asociación) entre un actor y un caso de uso que denota la participación del actor en dicho caso de uso. En el

diagrama del ejemplo de la figura 3.3 todas las líneas que salen del actor denotan este tipo de asociación (en realidad estereotipada como <<comunicates>>).

''usa'' (<<uses>>) (o <<include>> en la nueva versión de UML): Relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento de un escenario en otro. En el caso del ejemplo el caso de uso *Cancelar* incluye en su comportamiento al de *DarVueltas* y *PedirProducto* incluye también *DarVueltas*.

''extiende'' (<< extends>>): Relación de dependencia entre dos casos de uso que denota que un caso de uso es una especialización de otro. Por ejemplo, podría tenerse un caso de uso que extienda la forma de pedir azúcar, para que permita escoger el tipo de azúcar (normal, dietético o moreno) y además la cantidad en las unidades adecuadas (cucharadas o bolsas). Un posible diagrama se muestra en la figura 3.4.

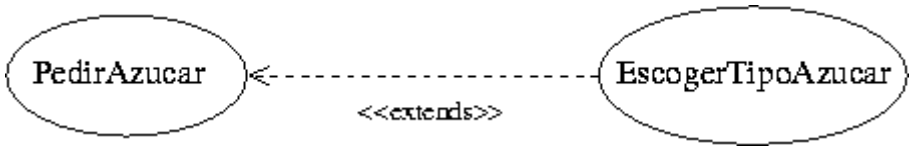


Figura 3.4: Representación de un actor: *stickman*
Fuente: [http://www-gris.det.uvigo.es/~avilas/UML/node23.html**]**

Se utiliza una relación de tipo **<<extends>>** entre casos de uso cuando nos encontramos con un caso de uso similar a otro pero que hace algo más que éste (variante). Por contra, utilizaremos una relación tipo **<<uses>>** cuando nos encontramos con una parte de comportamiento similar en dos casos de uso y no queremos repetir la descripción de dicho comportamiento común.

En una relación **<< extends>>**, un actor que lleve a cabo el caso de uso base puede realizar o no sus extensiones. Mientras, en una relación **<<include>>** el actor que realiza el caso de uso base también realiza el caso de uso incluido.

En general utilizaremos **<<extends>>** cuando se presenta una variación del comportamiento normal, y **<<include>>** cuando se repite un comportamiento en dos casos de uso y queremos evitar dicha repetición.

Por último en un diagrama de casos de uso, además de las relaciones entre casos de uso y actor (asociaciones) y las dependencias entre casos de uso (**<<include>>** y **<<extends>>**), pueden existir relaciones de herencia ya sea entre casos de uso o entre actores.

Llamamos modelo de casos de uso a la combinación de casos de uso y sus correspondientes diagramas. Los modelos de casos de uso se suelen acompañar por un glosario que describe la terminología utilizada. El glosario y el modelo de casos de uso son importantes puntos de partida para el desarrollo de los diagramas de clases.

Por último se debe tener en cuenta, que aunque cada caso de uso puede llevar a diferentes realizaciones, es importante reflejar en cada representación el motivo que nos ha llevado a descartarla, si es el caso.

3.3.1.4.- Descripción de los usuarios que interactúan con el sistema

Descripción de Usuarios: Se definen perfiles de usuario, permisos, diagramas de casos de uso y descripción de módulos del sistema. Se identifica además, la información que manipula cada usuario de acuerdo a su perfil, las restricciones, las interfaces involucradas, etc.

3.3.1.5.- Creación de un prototipo del sistema.

A menos que el proyecto de sistemas sea lo más tradicional o muy básico, los usuarios no siempre podrán definir sus requerimientos en forma adecuada y precisa o simplemente no pueden especificar los requerimientos de manera previa, sino que se deben descubrirlo.

3.3.1.5.1.- Prototipo

Es un modelo a escala o facsímil de lo real, pero no tan funcional para que equivalga a un producto final, ya que no lleva a cabo la totalidad de las funciones necesarias del sistema final. Proporcionando una retroalimentación temprana por parte de los usuarios acerca del Sistema.

3.3.1.5.2.- Importancia de Definir su Objetivo

Siempre se debe establecer cual es su objetivo, ya que un prototipo puede ser útil en diferentes fases del proyecto, por ello su objetivo debe ser claro. Durante la fase de análisis se usa para obtener los requerimientos del usuario. En la fase de diseño se usa para ayudar a evaluar muchos aspectos de la implementación seleccionada.

3.3.1.5.3.- Propósitos del Prototipo

En la fase de Análisis de un proyecto, su principal propósito es obtener y validar los requerimientos esenciales, manteniendo abiertas, las opciones de implementación. Esto implica que se debe tomar los comentarios de los usuarios, pero debemos regresar a sus objetivos para no perder la atención.

En la fase de Diseño, su propósito, basándose en los requerimientos previamente obtenidos, es mostrar las ventanas, su navegación, interacción, controles y botones al usuario y obtener una retroalimentación que nos permite mejorar el Diseño de Interfaz.

3.3.1.5.4.- Características de los Prototipos

El proceso de desarrollo y empleo de prototipos tiene las siguientes características:

- El prototipo es una aplicación que funciona
- Los prototipos se crean con rapidez
- Los prototipos evolucionan a través de un proceso iterativo
- Los prototipos tienen un costo bajo de desarrollo

3.3.1.5.5.- Información Obtenida con el uso del Prototipo

3.3.1.5.5.1.- Reacciones Iniciales del Usuario

El profesional de Sistema por medio de la observación, evaluación y la retroalimentación, obtendrá como reaccionan los usuarios al trabajar con el prototipo, y que tan conveniente es el acoplamiento entre las necesidades y las características modeladas en el sistema. A través de la recopilación de tales reacciones, el profesional, irá descubriendo nuevas perspectivas del prototipo, incluso si los usuarios se encuentran satisfechos con él, o si habrá dificultades para vender o implantar el sistema.

3.3.1.5.5.2.- Sugerencias

Las sugerencias son el fruto de la relación de los usuarios con el prototipo, las sugerencias aportadas por el usuario indican al profesional porque caminos dirigirse para refinar el prototipo, modificarlo o depurarlo, de forma que satisfaga mejor las necesidades de los usuarios.

3.3.1.5.5.3.- Innovaciones

Las innovaciones son aquellas características nuevas del sistema que no fueron contempladas previamente a la interacción con el prototipo.

3.3.1.5.5.4.- *Prioridades*

La información que se obtiene con el uso de prototipos permite al profesional establecer prioridades y reorientar sus planes de una manera menos costosas y con un mínimo de contratiempo.

Una de las peores cosas que le puede pasar a un profesional es diseñar e implantar un sistema que el usuario no necesita, ni desean.

3.3.1.5.6.- Desarrollo de Prototipo

3.3.1.5.6.1.- *Problemas Candidatos*

Para decidir si el prototipo debe incluirse o no Ciclo de Desarrollo de Sistema de Información, el profesional considera los siguientes factores:

- Problemas no estructurado, novedosos y complejos, de información personalizada del usuario, ya que sus salidas no son predecibles y definidas
- Problemas de ambiente Inestable, el profesional también debe evaluar el contexto del sistema
- Experiencia en diseños similares
- No se conocen los requerimientos, la naturaleza del sistema es tal que existe poca información con respecto a las características que debe tener el nuevo sistema para satisfacer las necesidades del usuario
- Los requerimientos deben evaluarse, se conocen los requerimientos aparentes de información pero es necesario verificarlos y evaluarlos
- Costos altos, donde la inversión involucra gran cantidad de recursos financieros y humanos.
- Altos riesgo, la evaluación inexacta de los requerimientos o el desarrollo incorrecto ponen en peligro a la organización

- El usuario, donde no está dispuesta examinar modelos en papel, o no sabe lo que quiere pero lo reconocerá cuando lo vea.
- Tecnologías Nuevas, la falta de experiencia en el uso de dichas tecnologías, junto con el deseo de instalar nuevas tecnología hace que sea propicio el uso del prototipo.

3.3.1.5.6.2.- *Etapas del Prototipo*

El desarrollo de un prototipo se lleva a cabo en forma ordenada a través de las siguientes etapas:

3.3.1.5.6.3.- *Identificación de Requerimientos Conocidos*

El profesional de sistema identifica los requerimientos conocidos, generales, o características esenciales y determina el propósito del prototipo de la aplicación.

3.3.1.5.6.4.- *Desarrollo de un Modelo*

En esta etapa se explica el método iterativo y las responsabilidades a los usuarios ya que el usuario participa directamente en todo el proceso. La rapidez con la que se genera el sistema es esencial para que no se pierda el estado de ánimo sobre el proyecto y los usuarios puedan comenzar a evaluar la aplicación con la mayor brevedad posible. El profesional de sistema para construcción inicial del prototipo emplea cualquier herramienta, como Lenguajes de Cuarta Generación, Generadores de Reportes, Generadores de Pantallas.

En el desarrollo de un prototipo se preparan los siguientes componentes:

- El lenguaje para el diálogo o conversación entre el usuario y el sistema
- Pantallas y formatos para la entrada de datos
- Módulos esenciales de procesamientos

- Salida del sistema

La incorporación en la interfaz de entrada/salida de características representativas de las que serán incluidas en el sistema final permite una mayor exactitud en el proceso de evaluación.

3.3.1.5.6.5.- *Revisión del Prototipo*

Es responsabilidad del usuario trabajar con el prototipo y evaluar sus características y operación. La experiencia con el sistema bajo condiciones reales permite la familiaridad indispensable para determinar los cambios o mejoras que sean necesarios, o también la eliminación de características innecesarias.

El profesional de sistemas captura la información sobre lo que le gusta y lo que le desagrada a los usuarios. Esta información tiene influencia en la siguiente versión del prototipo, la cual se presenta modificada y refinada.

3.3.1.5.6.6.- *Iteración*

Las dos últimas etapas escritas anteriormente se repiten varias veces hasta que estén usuarios y profesionales de sistema de acuerdo en que el prototipo ha evolucionado lo suficiente o que una iteración más no traerá beneficios adicionales.

3.3.1.5.6.7.- *Prototipo Terminado*

Cuando el prototipo está terminado, es decir, tenemos la información que buscamos, seguimos en el punto donde habíamos quedado dentro del Ciclo de Desarrollo de Sistema.

3.3.1.5.7.- Estrategias para el Desarrollo de Prototipos

Se puede desarrollar un prototipo para cada uno de los componentes de la aplicación

3.3.1.5.7.1.- Prototipos por Pantallas

La interfase entre el sistema y el usuario es la pantalla de visualización, esta es el vehiculo para presentar la información tal como ésta es proporcionada al sistema o como es recuperada de éste.

Los prototipos de pantalla permiten evaluar la posición de información sobre la pantalla, los encabezados, los botones, mensajes. También permite la reacción de los usuarios por la cantidad de información sobre la pantalla. La creación de un prototipo de pantalla conduce a:

- Que debe presentarse como información sobre la pantalla principal
- Cuál pertenece a una pantalla de detalle

3.3.1.5.7.2.- Prototipos para Procedimientos de Procesamientos

Las funciones de procesamiento incluye entradas, cálculos, recuperar información y actividades de salidas. Como los datos pocas veces son ingresados de la forma correcta o en la secuencia válida, es por ello que la aplicación se diseña para asegurar la detección de errores.

El objetivo es determinar si los procedimientos de aplicación fueron desarrollados adecuadamente.

La evaluación de los procedimientos y la observación de errores y equivocaciones cometidas por los individuos cuando emplean el prototipo, pueden

sugerir la adición de características de manejo de errores que no se habían anticipado.

3.3.1.5.7.3.- Prototipos de Funciones Básicas

Para determinar los requerimientos de una aplicación no es necesario desarrollar todos los módulos del sistema, sino los básicos, son aquellos que forman el núcleo de la aplicación. Incluye las funciones primarias de la aplicación como edición y validación, y excluye las secundarias como el manejo de archivos que no forman parte del procesamiento esencial. Por ejemplo:

Una aplicación de Reclamos de una venta, tendrá módulos de:

- Recepción de la información de la venta que se reclama
- Validación del número de factura
- Recuperación de la venta
- Generación de Nota de Crédito
- Y pueden omitirse por ejemplo:
- La impresión de la Nota de Crédito
- Registro de esta operación

3.3.1.5.8.- Roles

3.3.1.5.8.1.- Rol del Usuario

El papel del usuario con el prototipo puede resumirse en compromiso y honestidad. Si carece de compromiso pocos son los motivos para desarrollar un prototipo, ya que el usuario es el pivote del proceso de desarrollo y evaluación. Los usuarios interactúan con el prototipo teniendo las siguientes responsabilidades:

- Utilizar y evaluar el prototipo las veces que sea necesario

- Identificar mejoras
- Sugerir las característica no deseadas
- Describir los requerimientos de datos
- Describir la salida deseada

3.3.1.5.8.2.- Rol del Profesional de Sistema

El papel del profesional de sistema no solo debe construir el prototipo sino también que debe:

- Crear el clima adecuado al usuario para que este se exprese sin temor alguno.
- Familiarizar al usuario con el prototipo.
- Crear el plan para el desarrollo del prototipo.
- Construir la versión inicial.

Evaluar las reacciones del usuario y plasmar las modificaciones en una nueva versión.

3.3.2.- DISEÑO

En la fase de Diseño se elaboran las tarjetas CRCs (Clase, Responsabilidad y Colaboración), las cuales son creadas por los miembros del equipo de desarrollo, con la finalidad de simular las relaciones existentes entre los objetos del sistema haciendo uso de los diagramas de colaboración, de igual manera se establece la arquitectura del software. Esta etapa se subdivide en:

- Diseño Conceptual
- Diseño Navegacional
- Diseño de Interfaz Abstracta

3.3.2.1.- Diseño Conceptual

Construcción del esquema conceptual representado a través de los objetos del dominio, las relaciones y colaboraciones establecidas entre ellas.

Durante esta actividad se construye un esquema conceptual representado por los objetos del dominio, las relaciones y colaboraciones existentes establecidas entre ellos. En las aplicaciones hipermedia convencionales, cuyos componentes de hipermedia no son modificados durante la ejecución, se podría usar un modelo de datos semántico estructural (como el modelo de entidades y relaciones).

De este modo, en los casos en que la información base pueda cambiar dinámicamente o se intenten ejecutar cálculos complejos, se necesitará enriquecer el comportamiento del modelo de objetos.

En XPOOHD-WAP, el esquema conceptual está construido por clases, relaciones y subsistemas. Las clases son descritas como en los modelos orientados a objetos tradicionales. Sin embargo, los atributos pueden ser de múltiples tipos para representar perspectivas diferentes de las mismas entidades del mundo real.

Se usa notación similar a UML (Lenguaje de Modelado Unificado) y tarjetas de clases y relaciones similares a las tarjetas CRC (Clase Responsabilidad Colaboración). El esquema de las clases consiste en un conjunto de clases conectadas por relaciones. Los objetos son instancias de las clases. Las clases son usadas durante el diseño navegacional para derivar nodos, y las relaciones que son usadas para construir enlaces.

3.3.2.2.- Diseño Navegacional

Elaboración del modelo navegacional como una vista del Diseño Conceptual, proporcionando los recursos para la realización de los distintos modelos de acuerdo a los perfiles de los usuarios. Los esquemas de clases navegacionales y contexto navegacional son utilizados con el fin de expresar los resultados obtenidos en esta etapa del diseño.

La primera generación de aplicaciones web fue pensada para realizar navegación a través del espacio de información, utilizando un simple modelo de datos de hipermedia. En XPOOHD-WAP, la navegación es considerada un paso crítico en el diseño aplicaciones. Un modelo navegacional es construido como una vista sobre un diseño conceptual, admitiendo la construcción de modelos diferentes de acuerdo con los diferentes perfiles de usuarios. Cada modelo navegacional provee una vista subjetiva del diseño conceptual.

El diseño de navegación es expresado en dos esquemas: el esquema de clases navegacionales y el esquema de contextos navegacionales. En XPOOHD-WAP existe un conjunto de tipos predefinidos de clases navegacionales: nodos, enlaces y estructuras de acceso. La semántica de los nodos y los enlaces son las tradicionales de las aplicaciones hipermedia, y las estructuras de acceso, tales como índices o recorridos guiados, representan los posibles caminos de acceso a los nodos.

La principal estructura primitiva del espacio navegacional es la noción de contexto navegacional. Un contexto navegacional es un conjunto de nodos, enlaces, clases de contextos, y otros contextos navegacionales (contextos anidados). Pueden ser definidos por comprensión o extensión, o por enumeración de sus miembros.

Los contextos navegacionales juegan un rol similar a las colecciones y fueron inspirados sobre el concepto de contextos anidados. Organizan el espacio

navegacional en conjuntos convenientes que pueden ser recorridos en un orden particular y que deberían ser definidos como caminos para ayudar al usuario a lograr la tarea deseada.

Los nodos son enriquecidos con un conjunto de clases especiales que permiten de un nodo observar y presentar atributos (incluidos las anclas), así como métodos (comportamiento) cuando se navega en un particular contexto.

3.3.2.3.- Diseño de Interfaz Abstracta

Se emplea para expresar de forma detallada la interfaz del usuario de la aplicación, de acuerdo a las pautas establecidas para la creación de aplicaciones bajo tecnología WAP.

Una vez que las estructuras navegacionales son definidas, se deben especificar los aspectos de interfaz. Esto significa definir la forma en la cual los objetos navegacionales pueden aparecer, cómo los objetos de interfaz activarán la navegación y el resto de la funcionalidad de la aplicación, qué transformaciones de la interfaz son pertinentes y cuándo es necesario realizarlas.

Una clara separación entre diseño navegacional y diseño de interfaz abstracta permite construir diferentes interfaces para el mismo modelo navegacional, dejando un alto grado de independencia de la tecnología de interfaz de usuario.

El aspecto de la interfaz de usuario de aplicaciones interactivas (en particular las aplicaciones web) es un punto crítico en el desarrollo que las modernas metodologías tienden a descuidar. En XPOOHD-WAP se utiliza el diseño de interfaz abstracta para describir la interfaz del usuario de la aplicación de hipermedia.

El modelo de interfaz ADVs (Vista de Datos Abstracta) especifica la organización y comportamiento de la interfaz, pero la apariencia física real o de los atributos, y la disposición de las propiedades de las ADVs en la pantalla real son hechas en la fase de implementación.

Es importante resaltar que durante la fase de diseño se consideraron como base las pautas establecidas para el desarrollo de aplicaciones bajo tecnología WAP; entre las mas importantes: 20% de funcionalidad, tasa de actividades de usuarios, diseño de una estructura de árbol, minimización de la entrada de datos, personalización, uso de textos concisos, implementación de una funcionalidad de retorno, aplicación de pruebas. Cabe destacar que es en el diseño navegacional donde estas reglas se visualizan fácilmente, ya que es allí donde se especifica la organización y comportamiento de la interfaz del usuario.

3.3.3.- CODIFICACIÓN

La fase de Codificación formaliza la implementación de la aplicación, con la meta de lograr un diseño amigable y libre de redundancia, perfeccionando la calidad de la programación, originando un código reusable e ideal para efectuar el mantenimiento del software.

En esta fase, el diseñador debe implementar el diseño. Hasta ahora, todos los modelos fueron construidos en forma independiente de la plataforma de implementación; en esta fase es tenido en cuenta el entorno particular en el cual se va a correr la aplicación.

Al llegar a esta fase, el primer paso que debe realizar el diseñador es definir los ítems de información que son parte del dominio del problema. Debe identificar también, cómo son organizados los ítems de acuerdo con el perfil del usuario y su tarea; decidir qué interfaz debería ver y cómo debería comportarse. A fin de implementar todo en un entorno WAP, el diseñador debe decidir además qué información debe ser almacenada.

3.3.4.- PRUEBAS

Por ultimo en la fase de Prueba de la aplicación desarrollada, se diseñan y aplican pruebas al software, tales como pruebas unitarias a cada historia de usuario posterior a su implementación. Asimismo, es necesario garantizar la satisfacción del cliente para lo cual se practicaron pruebas de aceptación al sistema.

El objetivo fundamental es conseguir la aceptación final del sistema por parte de los usuarios del mismo, para ello:

- Se combinarán por primera vez todo el equipo lógico y los procedimientos para un trabajo del sistema real.
- Se realizarán las pruebas de aceptación, las cuáles constituyen un procedimiento formal ejecutado por los usuarios que permite verificar que el sistema producido es totalmente funcional y satisface los requisitos iniciales, como un paso previo a su implantación.
- Se realizarán los procedimientos necesarios para la implantación y puesta en producción del sistema.

CAPITULO IV

4. DESARROLLO E IMPLEMENTACION DE LA METODOLOGÍA XPOOHD-WAP

4.1.- PLANIFICACIÓN

4.1.1.- ELICITACION DE REQUISITOS A TRAVES DE LAS HISTORIAS DE USUARIOS

Anexo Historia de Usuarios

4.1.2.- ESTUDIO DE FACTIBILIDAD

Después de definir la problemática presente y establecer las causas que ameritan de un nuevo sistema, es pertinente realizar un estudio de factibilidad para determinar la infraestructura tecnológica y la capacidad técnica que implica la implementación del sistema en cuestión, así como los costos, beneficios y el grado de aceptación que la propuesta general en la Institución, este análisis permitió determinar las posibilidades de diseñar el sistema propuesto y su puesta en marcha, los aspectos tomados en cuenta para este estudio fueron clasificados en tres áreas, las cuales se describen a continuación:

4.1.2.1.- Factibilidad Técnica

La falibilidad técnica consistió en realizar una evaluación de la tecnología existente en la institución, este estudio estuvo destinado a recolectar información sobre los componentes técnicos que posee la organización y la posibilidad de hacer uso de los mismos en el desarrollo e implementación del sistema propuesto y de ser necesario, los requerimientos tecnológicos que deben ser adquiridos para el desarrollo y puesta en marcha del sistema en cuestión.

De acuerdo a la tecnología necesaria para la implantación del portal WAP-ESPEL de la Escuela Politécnica del Ejército Sede Latacunga, se evaluó bajo dos enfoques: Hardware y Software.

4.1.2.1.1.- Hardware

Evaluando el hardware existente y tomando en cuenta la configuración mínima necesaria, la institución no requiere inversión inicial para la adquisición de nuevos equipos, ni tampoco repotenciar o actualizar los equipos existentes, ya que los mismos satisfacen los requerimientos establecidos tanto para el desarrollo y puesta en funcionamiento del sistema propuesto.

4.1.2.1.2.- Software

En cuanto al software, la Institución cuenta con todas las aplicaciones que se emplearon para el desarrollo del proyecto como es el servidor html, servidor de aplicaciones y servidores de bases de datos, lo cuál no amerita inversión alguna para la adquisición de los mismos. En cuanto a los navegadores WAP utilizados como terminales, la mayoría de celulares cuentan con estas aplicaciones, además existen emuladores para PCS como WinWap y deckit que se pueden bajar gratuitamente de Internet, como en estas direcciones citadas.

http://www.freedownloadmanager.org/es/downloads/winwap_para_windows.22750_p/free.htm

http://www.wap-proof.com/wap_browser_download.php

http://www.yospace.com/spede_html

Como resultado de este estudio técnico se determino que en los actuales momentos, la institución posee la infraestructura tecnológica (Hardware y

Software) necesaria para el desarrollo y puesta en funcionamiento del sistema propuesto.

4.1.2.2.- Factibilidad Económica

A continuación se presenta un estudio que dio como resultado la factibilidad económica del desarrollo del nuevo sistema de información. Se determinaron los recursos para desarrollar, implantar, y mantener en operación el sistema programado, haciendo una evaluación donde se puso de manifiesto el equilibrio existente entre los costos intrínsecos del sistema y los beneficios que se derivaron de éste, lo cuál permitió observar de una manera más precisa las bondades del sistema propuesto.

4.1.2.2.1.- Análisis Costos-Beneficios

Este análisis permitió hacer una comparación entre la relación costos del sistema actual, y los costos que tendría un nuevo sistema, conociendo de antemano los beneficios que la ciencia de la informática ofrece.

Como se menciono anteriormente en el estudio de la factibilidad técnica, la Institución contaba con las herramientas necesarias para la puesta en marcha del sistema, por lo cuál el desarrollo de la propuesta no requirió de una inversión inicial.

A continuación se presenta un resumen de los costos intrínsecos del sistema propuesto y una lista de los costos que conlleva implantar el mismo, y los costos de operación. Luego a través de un análisis de valor se determinaron los beneficios que no necesariamente para el nuevo sistema son monetarios o cuantificables.

El resumen del análisis costos – beneficios se definieron a través de una comparación de los costos implícitos, tanto del sistema actual como del propuesto con los beneficios expresados en forma tangible.

4.1.2.3.- Elaboración del modelo general de casos de uso

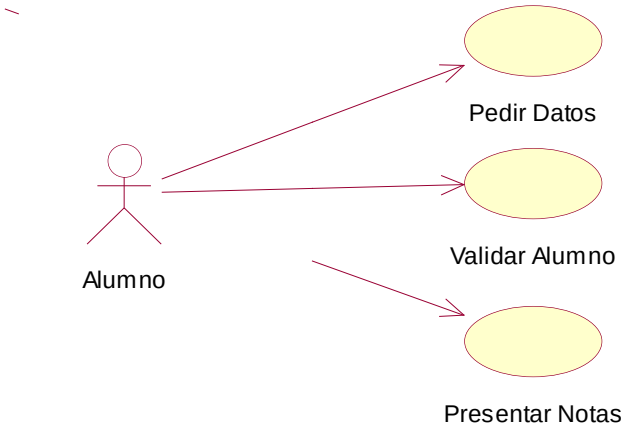
4.1.2.3.1.- Diagramas de casos de uso

Caso: Consulta de Notas

Autor: Alumno

Propósito: Primario Escencial

Descripción: El alumno ingresa su cedula y password, estos datos son validados en caso de ser correctos presenta las notas respectivas, en caso de ser incorrectos emite un mensaje de error.

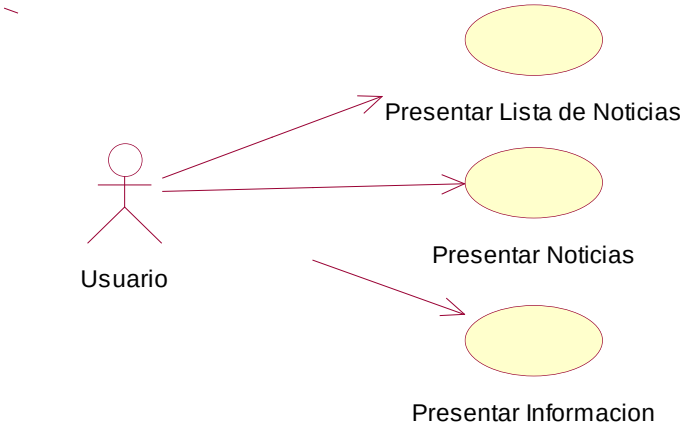


Caso: Noticias

Autor: Usuario

Propósito: Escencial

Descripción: El usuario selecciona noticias, se presenta una lista de noticias, selecciona la noticia deseada a leer, visualiza la información página por página.

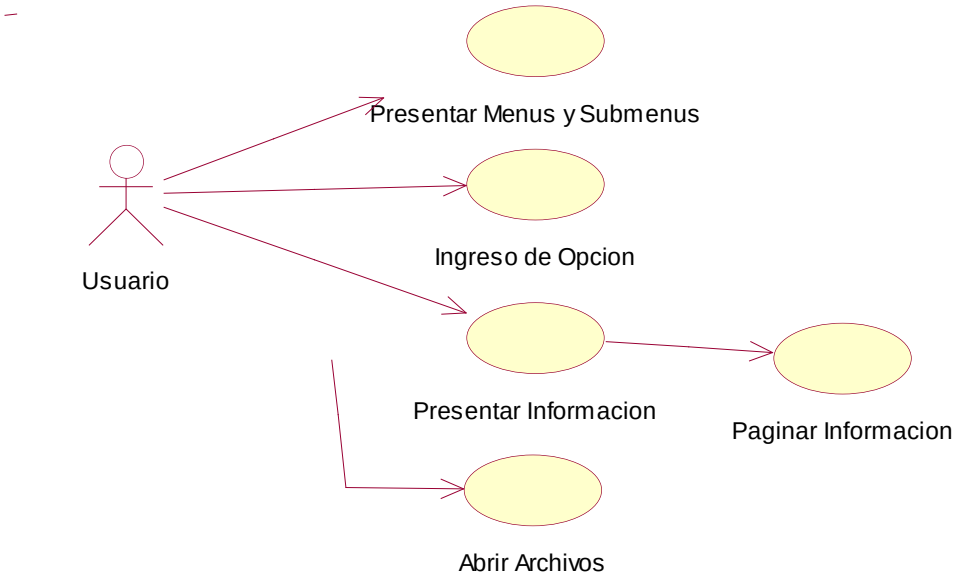


Caso: Consulta de Información General

Autor: Usuario

Propósito: Primario Escencial

Descripción: El usuario visualiza las opciones existentes, selecciona una de ellas, presenta la información correspondiente, seguidamente lee el contenido página por página.



4.1.3.- DESCRIPCIÓN DE USUARIOS QUE INTERACTUAN CON EL SISTEMA

El futuro sistema facilitará a los usuarios una mejor manera de poder obtener información de la institución por medio de su celular, tendrá pantallas no tan vistosas a las que nos hemos acostumbrados pero serán amigables con el usuario.

El sistema será independiente a otros sistemas externos, es una página WAP, para poder acceder a este deberán estar conectados en Internet. Se deberá considerarse como mínimo las siguientes especificaciones:

- Control de usuario (contraseña) en el caso de consulta de notas este caso considerará el número de cédula.
- Implementación de funciones de los módulos como:
    - Departamentos de:
        - Ciencia exactas
        - Lenguas

        - Energía y mecánica
        - Eléctrica y electrónica
        - Ciencias administrativas
    - Campus
        - ¿Quiénes somos?
        - Mensaje del director
        - Filosofía
        - Historia
        - Autoridades
        - Convenios
        - Ubicación geográfica
        - Directorio
    - Información académica
        - Proceso de admisión
        - Programas de financiamiento
        - Aranceles
        - Becas
    - Carreras
        - Tecnología en computación
        - Tecnología en electrónica
        - Electrónica e instrumentación
        - Sistemas e informática
        - Electromecánica
        - Mecatrónica
    - Servicios
        - Biblioteca
        - Policlínicos
        - Centro de producción
        - Clubes
        - Bar estudiantil
    - Calificaciones

También se incluirá opciones como:

- Noticias

4.1.4.- DISEÑO

4.1.4.1.- Diseño Conceptual

Para el sistema se definió 5 conceptos que interactúan entre ellos para el funcionamiento del sistema

4.1.4.1.1.- PRESENTACION

Todo sistema requiere una presentación o bienvenida para que el usuario sepa a que a punta la realización de este.

4.1.4.1.2.- HOME

Es la parte principal que contiene todas las opciones del sistema y permite el acceso a estas.

4.1.4.1.3.- OPCIONES

Son las diferentes opciones que del cual el sistema permite consultar la información.

4.1.4.1.4.- NOTICIAS

Es un ejemplo de opción del sistema la cual interactúa con una base de datos que contiene la información de las noticias de la Escuela

4.1.4.1.5.- CONTENIDO

Es la información presentada en estilo wap a la cual se accede mediante las opciones del sistema.

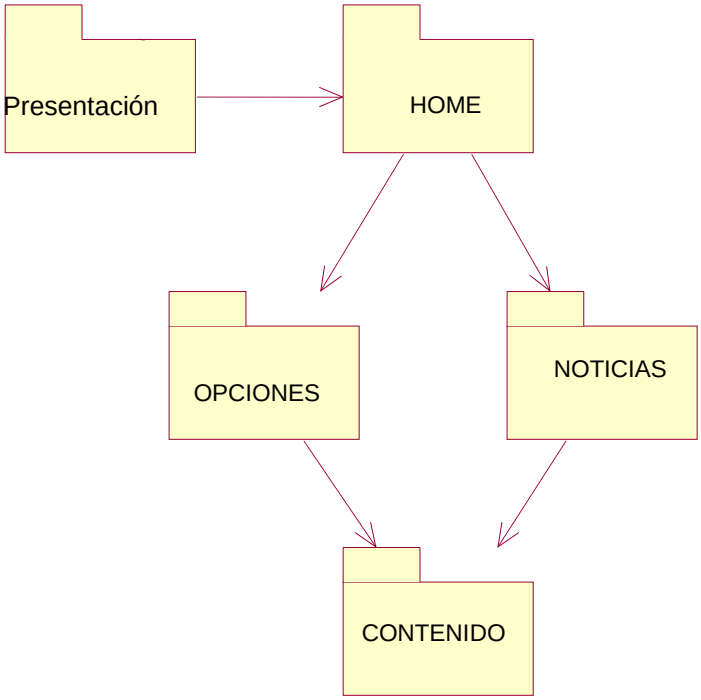


Figura 4.1: Gráfico de Diseño Conceptual.

4.1.4.2.- Diseño Navegacional

El diseño navegacional es muy importante ya que de este depende la agilidad de navegar y llegar a la información de una manera sencilla y rápida, cada pagina tiene un retroceso a la anterior o menú principal con un solo link a través de un menú común de acceso rápido y muestra la información con máximo 3 links lo que permite facilidad de navegación a los usuarios y acceso rápido a las diferentes opciones.

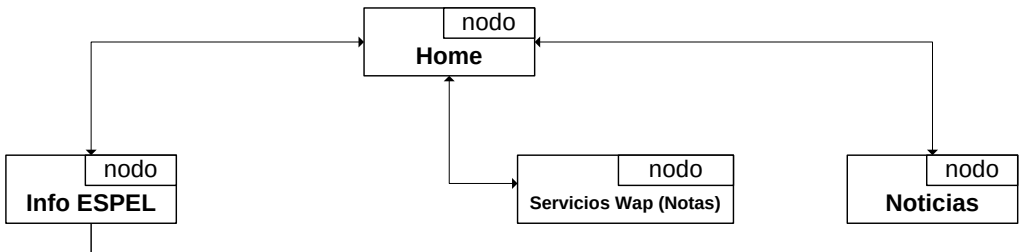


Figura 4.2: Gráfico de Diseño Navegacional.

4.1.4.3.- Diseño de Interfaz

Para este sistema se diseño cuatro tipo de pantallas basados en los requerimientos WAP y considerando las limitaciones de tiempo costos capacidad y sobre todo pantalla para la presentación de la información, cada pantalla tiene un orden jerárquico y todas presentan la opción de de volver a la pantalla anterior o la principal, los diseños de pantallas son: Presentación, Home, Opción y Contenido.

4.1.4.3.1.- PRESENTACION

Esta pantalla se presentara al inicio del portal WAP como bienvenida, esta compuesta por el escudo de la ESPE al inicio de la pantalla, seguido del texto de bienvenida y la opción de ingreso.

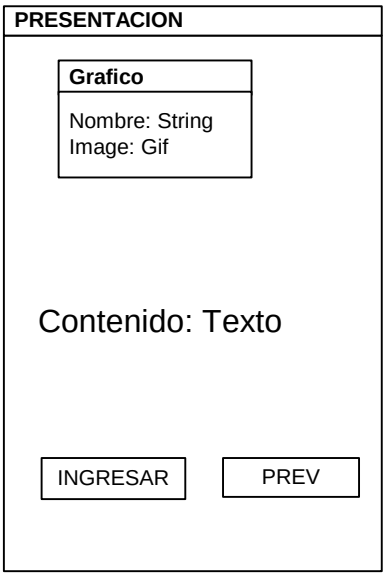


Figura 4.3: Gráfico de Diseño de Interfáz (Presentación)

4.1.4.3.2.- HOME

Es la pantalla principal, en la parte superior se encuentra un logo de la Escuela seguida del menú de acceso a todas las opciones, y un menú de opciones adicionales común para todas paginas para facilitar el acceso a opciones anteriores.

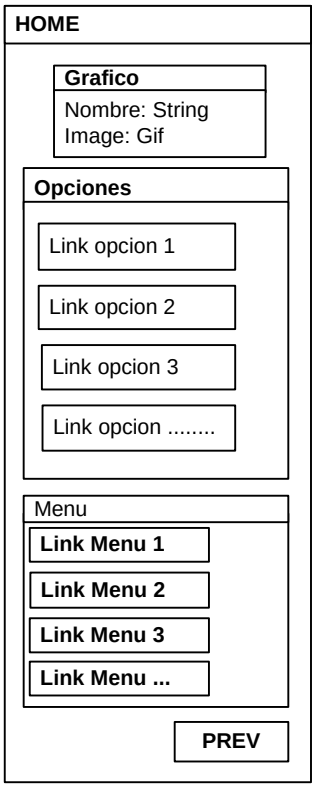


Figura 4.4: Gráfico de Diseño de Interfáz (Home)

4.1.4.3.3.- OPCION Y CONTENIDO

En esta opción se presenta la información paginada automáticamente para que se pueda leer de mejor manera en el dispositivo móvil y se puede navegar con dos links de siguiente y anterior, si esta información tiene contenido relacionado se utiliza la pantalla de OPCION que contiene links para presentar el

resto de contenido y si no se tiene información relacionada se utiliza la pantalla de CONTENIDO.

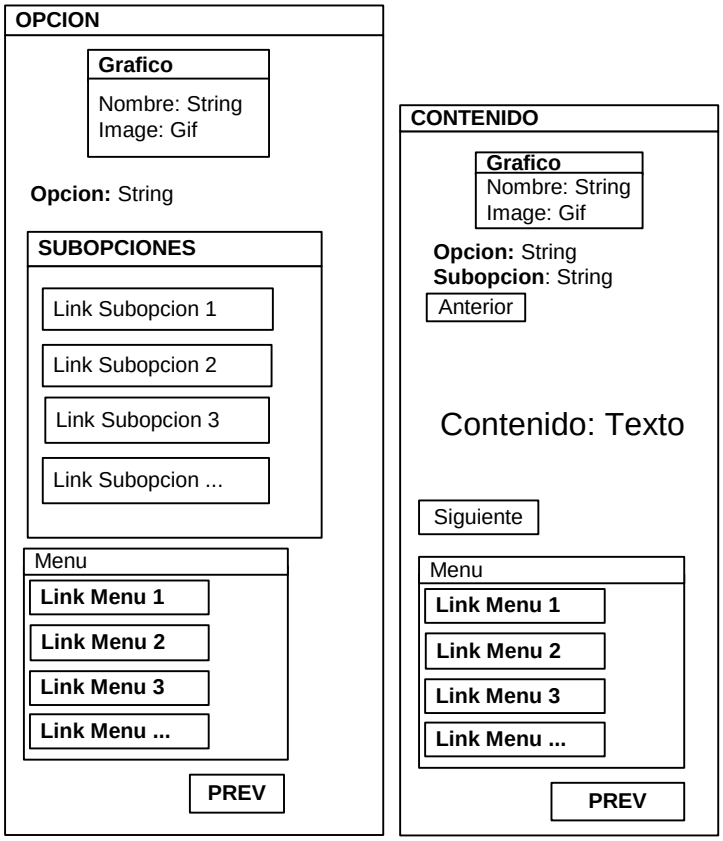


Figura 4.5: Gráfico de Diseño de Interfáz (Opción y Contenido)

4.1.5.- CODIFICACIÓN

Anexo en Cd ” Codificación”

4.1.6.- PRUEBA

Anexo en Cd “Pruebas”

CAPITULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1.- CONCLUSIONES

- La metodología XPOOHD-WAP es una buena opción para el desarrollo de aplicaciones wap por su facilidad de aplicación ya que esta basada en la metodología XP y aprovecha característica de OOHD para incluir en el diseño los aspectos de WAP.
- Siguiendo los pasos de esta metodología se ha obtenido un producto muy bueno, el cual con el respaldo de los documentos: Plan de Aseguramiento para la Calidad del Software, Estimación del Software, Análisis y Gestión del Riesgo y Gestión de la Configuración del Software concluimos que es un producto de calidad (Anexos).
- El portal WAP-ESPEL ha sido concluido en todas las fases cumpliendo con los requerimientos de información propuestos por la institución.
- El portal WAP-ESPEL permite acceder a toda la información que se tiene en la página Web.
- Dividir la información de la búsqueda permite que el usuario pueda ver los datos más importantes y a la vista de los mismos decidir si quiere realizar toda la consulta.
- Las URLs fáciles de teclear y de recordar permiten al usuario una navegación mas rápida y simple.
- Al seguir las recomendaciones del desarrollo de la interfaz WAP logramos que las aplicaciones se vean bien en la mayoría de los celulares facilitando la navegación del usuario.

5.2.- RECOMENDACIONES

- Se recomienda el estudio de la metodología XP, el lenguaje WML, características y limitaciones de la tecnología WAP para el mejor entendimiento de las fases de la metodología XPOOHD-WAP.
- Para crear nuevas páginas WML se recomienda copiar una ya existente y modificarla para conservar los formatos y presentación ya definida.
- Se recomienda mantener la validación del tipo de terminal para presentar el portal requerido y hacer más fácil su acceso.
- Para futuras modificaciones es recomendable no sobrepasar los siguientes tamaños:
    a. Tamaño general del terminal: 1.4 K
    b. Longitud de texto: entre 16 y 20 líneas como máximo
    c. Longitud de la página: no más de 15 caracteres
    d. Longitud de texto entre los anclajes: 1 o 2 palabras como máximo
    e. Tamaño de las etiquetas aceptar y "prev": no más de 15 caracteres entre las dos (recomendable no más de 5 caracteres cada una)
    f. Tamaño de filas de tablas: no más de 12 caracteres cada una
    g. Longitud de lista: no más de 9 elementos

5.3.- REFERENCIAS BIBLIOGRAFICAS

- http://www.osmosislatina.com/aplicaciones/wap.htm
- http://www.wmlclub.com
- http://www.wapforum.com/
- http://www.wapforum.org/
- http://www.cellular.co.za/wap.htm

5.4.- ANEXOS

Gestión de configuración de software se encuentra en el cd

Historias de Usuarios

Pruebas

I. GESTION DE LA CONFIGURACIÓN DEL SOFTWARE

INTRODUCCIÓN

El presente documento contiene la información obtenida durante el proceso de la Gestión de Configuración de Software(GCS) aplicada al portal WAP-ESPEL(Portal WAP de la Escuela Politécnica del Ejército sede Latacunga). la información que se detalla en este documento esta basado en resultados obtenidos de las diferentes actividades de la GCS, esta basada en el Std, 828.

PROPÓSITO

- Continuar con el proceso técnico de la Ingeniería de Software.
- Crear el documento correspondiente a la GCS.
- Documentar las técnicas formales de desarrollo para realizar los cambios en los documentos de referencia ERS.
- Identificar los elementos de configuración de software(ECS).
- Documentar los elementos de configuración de software identificados en los correspondientes reportes durante todo el ciclo de vida del software.

ALCANCE

El documento GCS tiene como fin la gestión de los elementos de software que a continuación se detallan.

DOCUMENTOS:

- Documento ERS

AREAS INVOLUCRADAS

- Unidad de Organización y Sistemas de la ESPEL
- Carrera de Sistemas e Informática de la ESPEL

ACTIVIDADES DEL CICLO DE VIDA

- Planificación
- Especificación de Requisitos
- Diseño Conceptual
- Diseño Navegacional
- Diseño de Interfaz
- Codificación
- Pruebas

DEFINICIONES Y ACRONIMOS

GCS.- Gestión de configuración de Software

GGC.- Grupo de Gestión de Configuración.

ECS.- Elementos de configuración de Software

WAP_ESPEL.- Portal WAP de la Escuela Politécnica del Ejército Sede Latacunga.

ESPEL.- Escuela Politécnica del Ejército Sede Latacunga.

Std.- Estándar.

REFERENCIAS

Para la elaboración de este documento se ha considerado las siguientes fuentes de información:

- ERS

  Tipo: Especificación de Requisitos de Software(ERS)

  Fecha de realización: Febrero del 2007

  Versión: 1.0

  Desarrollador: Darwin Vinicio Herrera Veintimilla.

BREVE DESCRIPCIÓN DEL DOCUMENTO

El documento de GCS esta dividido en varias secciones, las mismas que permiten organizar de acuerdo al estándar establecido en este caso Std, 828

Sección 1. Introducción.- Se describe los objetivos, alcance, los documentos de sustento de este material, definiciones, acrónimos, referencias.

Sección 2. Gestión.- Se describe aspecto como la organización, responsabilidades, control de la interfase, la implementación de la GCS, políticas, directivas y procedimientos.

Sección 3. Actividades de la GCS.-Esta sección trata aspectos como la identificación de la Configuración, documentos de reuniones formales, resultados de auditorias, reportes GCS,. Centrándose en la identificación de la línea base, identificación del proyecto y el control de la configuración.

Sección 4. Herramientas, técnicas y metodologías.- Se describe las metodologías utilizadas para el desarrollo de este documento y adicionalmente las herramientas Case utilizadas en este proceso.

Sección 5. Control de la Promoción.- Describe las técnicas sugeridas como mejores prácticas para el adecuado desarrollo del proyecto de software.

Sección 6. Archivo y Retención.- Se citan los métodos de almacenamiento y las normativas establecidas para este efecto.

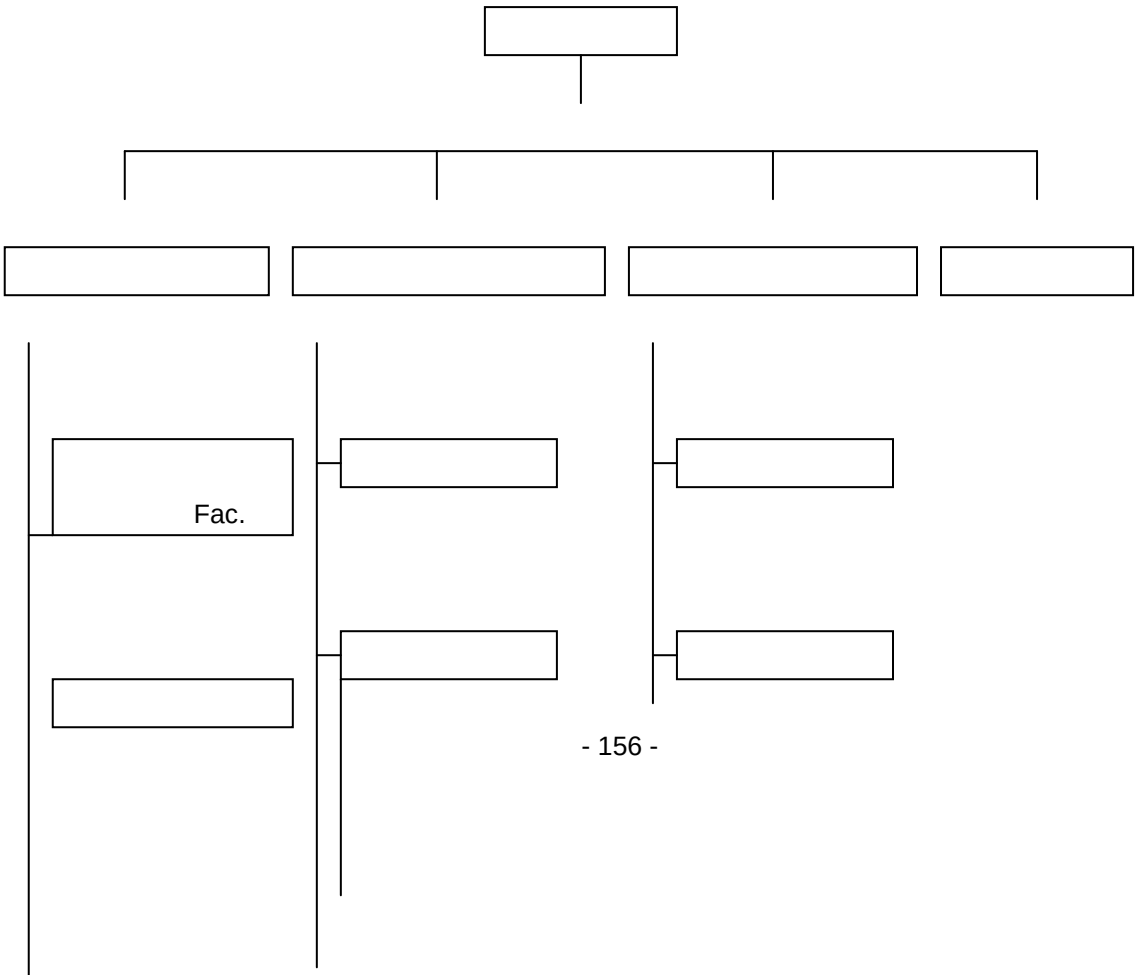
Sección 7. Anexos.- Se incluye documentos de soporte así como todo lo citado en forma referencia en este trabajo.

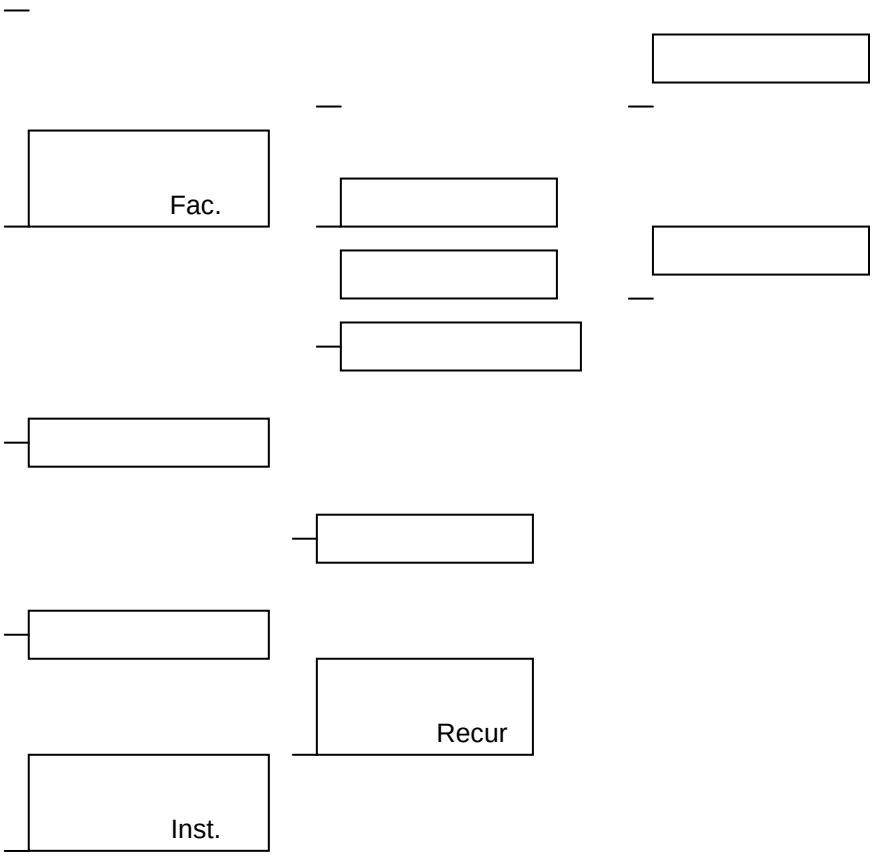
GESTIÓN

ORGANIZACIÓN

Para la elaboración del presente documento lo va realizar la Sr Darwin Vinicio Herrera Veintimilla, siendo Jefe de GCS, Jefe de Auditoría y Jefe de RTF.

Organigrama de la ESPEL




Fac.
Recur
Inst.
Departa

RESPONSABILIDADES

A continuación se describe las responsabilidades para cada tarea de la Gestión de la Configuración del software(GCS).

Sr. Darwin Herrera	1. Coordinar el procedo del GCS.
Jefe del GCS	2. Elaborar el documento del GCS.
	3. Organizar reuniones para determinar requerimientos.
	4. Difundir lo objetivos del estudio.
	5. Informar a los niveles de decisión de los resultados.
	6. Proponer y ejecutar metodologías de Ingeniería.
	7. Estandarizar los procesos de desarrollo.
	8. Gestión de la calidad del Software.
	9. Verificación y validación

Sr. Darwin Herrera	1. Realizar los documentos de RTF.

Jefe de Auditoría	2. Realizar los documentos de Auditoría.
Jefe RTF	3. Coordinar con los usuarios cambios de las historias.
	4. Revisar el estado de las historias de usuarios.
	5. Proponer mejoras en el proceso.
	6. Informar al nivel inmediato superior el avance del proyecto.
Alumnado	1. Proporcionar historias de usuarios.
	2. Participar en la toma de información.
	3. Colaborar en casos de estudio.
Sr. Darwin Herrera	1. Planificación
Ciclo de Vida	2. Diseño
	3. Implementación

CONTROL DE LA INTERFASE

El control de las interfaces esta determinado por los reportes RTF en los cuales se realizará un análisis de los Requerimientos de Software para determinar si cumplen con lo requerido.

1. Control de los ERS, utilizando un RTF (REP-RTF-WAP_ESPEL)
2. documentar las reuniones (REP-REU- WAP_ESPEL)
3. Control de la auditoría (REP-AUD- WAP_ESPEL)

IMPLEMENTACIÓN DEL GCS

IDENTIFICACIÓN

Se ha identificado los siguientes elementos de configuración de software:

Análisis(Planificación)

ERS

POLÍTICAS, DIRECTIVAS, PROCEDIMIENTOS

Para la correcta aplicabilidad del documento GCS se han previsto la ejecución de las siguientes políticas como parte de la norma de desarrollo del portal propuesto ”portal WAP-ESPEL(Portal WAP de la Escuela Politécnica del Ejército sede Latacunga)”

- Documentar la GCS en tiempo real.

- Utilizar estándares para todas las fases de ciclo de vida del software.
- La información hacia los niveles de decisión se realizará a través del jefe del GCS.

ACTIVIDADES DE LA GCS

Se describen los resultados de la aplicación del GCS

IDENTIFICACIÓN DE LA CONFIGURACIÓN

Se detalla a continuación los productos obtenidos como parte del proceso de GCS, durante la fase inicial de aplicación del mismo.

CONTROL

RTF

Se ha realizado revisión de los ERS los cuales se encuentran resumidos en el formato RTF.

Auditorias

Como parte del proceso de revisión que involucra el proceso del GCS se ha realizado el documento de auditoria.

Reporte GCS

Una vez finalizado el proceso del GCS se resume el proceso seguido.

IDENTIFICACIÓN DE LA LÍNEA BASE

Se constituye el punto de partida para

Items de las Líneas Base

- Documento de Planificación
- Historias de Usuarios
- Documento de Diseño Conceptual
- Documento de Diseño Navegacional
- Documento de Diseño de Interfaz
- Documento de Codificación
- Documento de Pruebas

Proceso de revisión de las Líneas Base

Se ha revisado los documentos que conforma la Línea Base de este proyecto actualmente, habiéndose encontrado ajustes, los mismos que son citados en los documentos de reporte correspondientes.

Tipos de usuarios y desarrolladores

Dentro del ámbito del sistema se determinan los siguientes tipos de usuarios y administradores.

Usuarios:

Alumnos

Docentes

Personal

Público en general

Desarrolladores:

Darwin Vinicio Herrera Veintimilla.

IDENTIFICACIÓN DEL PROYECTO

Fechas de compilación

El sistema se encuentra en la etapa de implementación.

Numeración de líneas de código fuente

El sistema tiene 3580 líneas de código en la etapa de implementación.

CONTROL DE LA CONFIGURACIÓN

NIVEL DE AUTORIDAD

El grupo de trabajo esta claramente definido, estableciendo adecuadamente los niveles de autoridad considerando que la ESPEL esta regido por niveles jerárquicos bien definidos. Los niveles de autoridad para el proyecto están representados de la siguiente forma:

Director

Jefe Administrativo

Jefe de la Unidad de Organización y Sistemas

Jefe de Proyecto (Darwin Vinicio Herrera Veintimilla.)

Jefe de RTF (Darwin Vinicio Herrera Veintimilla.)

Jefe de Auditoria (Darwin Vinicio Herrera Veintimilla.)

Esta organización esta sujeta a cambios de mando dependiendo del personal que se encuentre designado a cada una de las unidades de gerencia.

DEFINICIÓN DE MÉTODOS

Información para cada Grupo de Gestión de la configuración(GGC)

En este caso la gestión de todo el proyecto esta a cargo de una sola persona que a su vez culminado el proyecto será entregado a la unidad de Organización y Sistemas. Considerando esto se puede decir que se tiene un dominio completo de la información generada y requerida para la ejecución del proyecto.

Métodos e interfaces

Para la gestión de la información se usarán los reportes diseñados y estandarizados para efectos de este control, se propiciará la realización de reuniones de trabajo y verificación de requisitos.

Software especial de apoyo

Para la ejecución de este documento no se utilizará ningún tipo de herramienta case simplemente se utilizará Office.

CONTABILIDAD DE LA CONFIGURACIÓN

Para la ejecución de esta configuración se considera un solo involucrado en este proyecto, debido a lo cuál no se incurrirá en ningún tipo de gasto adicional para este proyecto.

AUDITORÍAS Y REVISIONES

Las revisiones de auditoria se realizaron para comprobar la ejecución de las recomendaciones citadas en los documentos base, así como para verificar la correcta escritura e interpretación de los requerimientos. Los resultados que

actualmente se han logrado se encuentran resumidos en los reportes correspondientes a Auditorias.

HERRAMIENTAS, TÉCNICAS Y METODOLOGÍAS

No se a utilizado ninguna herramienta Case, para el desarrollo del trabajo. Se han observado todas las normas y estándares de la IEEE definidos para wl proceso de GCS.

CONTROL DE LA PROMOCIÓN

El proyecto del portal WAP_ESPEL, considerando que al Escuela Politécnica del Ejército sede Latacunga se constituye como una unidad educativa de excelencia, será difundido a las áreas de sistemas y además estará a disponibilidad de los estudiantes y docentes de la institución.

ARCHIVO Y RETENCIÓN

Toda la información generada será ubicada en el Servidor de Archivos en una sección específica para el almacenamiento de la información de proyectos.

Adicionalmente se obtendrá respaldo externos periódicamente en CD-RW a fin de tener un respaldo externo de seguridad. Los documentos impresos residirán bajo seguridad en el anaquel de archivos de los sistemas informáticos.

ANEXOS

Estándar para la generación de los documentos.

- El documento será realizado en papel tipo A4 de 74grs.
- Los márgenes serán superior 3cm; inferior 3cm; derecho 2cm; izquierdo 3,5cm.
- El encabezado será de 1,25cm y pie de página será de 2cm del margen.
- Se usará letra Arial para el texto del documento de tamaño 12.

<table>
<tr><td rowspan="2">REP-REU-WAP_ESPEL</td><td>REPORTE REUNION</td><td rowspan="2">Fecha:

15/03/2007</td><td rowspan="2">Pág. 1/1</td></tr>
<tr><td>PROYECTO: Portal WAP de la Escuela Politécnica del Ejército sede Latacunga(-WAP_ESPEL)</td></tr>
<tr><td>Document</td><td colspan="3">(Cuales son los documentos utilizados)</td></tr>
</table>

<table>
<tr><td>o Base</td><td>Código:

• Planificación
• Historias de Usuarios</td><td>Nombre:

• Planificación
• Historias de Usuarios</td><td>Versión/Fecha

1.0/ Enero/07

1.0/Enero/07</td><td>Responsable:

Darwin Herrera</td></tr>
<tr><td rowspan="2">Identificación de la Reunión</td><td colspan="4">(Daros concretos de la Reunión y quienes participan)</td></tr>
<tr><td colspan="2">Técnicas Utilizadas:

• Reunión de Verificación</td><td>Personal:

• Ing. Edgar Montaluisa
• Sr. Darwin Herrera.</td><td>Fecha:

04/Feb/07</td></tr>
<tr><td rowspan="2">Descripción de las acciones realizadas</td><td colspan="4">(Descripción de la técnica y resultados esperados)</td></tr>
<tr><td colspan="2">Acción:

• Revisión de la documentación
• Revisión de requisitos
• Determinación de realización del proyecto</td><td colspan="2">Resultados:

• Decisión de seguir con el desarrollo del proyecto</td></tr>
<tr><td>Aprobació</td><td colspan="4">(Cajas para marcar el tipo de aceptación que merece este documento)</td></tr>
</table>

n del Documento	Aceptado (x) Sin cambios	Aceptado ( ) Con cambios	No Aceptado ( )	Pendiente ( )
Autentificación	(Firmas de cada una de las personas que participaron en la reunión)			
	Nombre: Ing. Edgar Montaluisa Sr. Darwin Herrera	Firma:	Fecha:	
	Realizado por: Sr. Darwin Herrera.		Reporte No 1	

REP-RTF-WAP_ESPEL	**REPORTE RTF**	Fecha:	Pág. 1/1
	PROYECTO: Portal WAP de la Escuela Politécnica del Ejército sede Latacunga(-WAP_ESPEL)	15/03/2007	

<table>
<tr><td>Documento Base</td><td colspan="4">(Cuales son los documentos utilizados para RTF)</td></tr>
<tr><td></td><td>Código:

• Historias de Usuarios</td><td>Nombre:

• Historias de Usuarios</td><td>Versión/Fecha

1.0/ Enero/07</td><td>Responsable:

Darwin Herrera</td></tr>
<tr><td>Identificación de RTF</td><td colspan="4">(Daros concretos de RTF y quienes participan)</td></tr>
<tr><td></td><td>Técnicas RTF Utilizadas:

• Reunión de Formal</td><td>ódulo:

.1.16</td><td>Personal:

Ing. Edgar Montaluisa</td><td>Fecha:

10/Feb/07</td></tr>
<tr><td>Descripción de la acción RTF</td><td colspan="4">(Descripción de la técnica y resultados esperados)</td></tr>
<tr><td></td><td colspan="2">Acción:

• Revisión de STORY NUMBER: 0007</td><td colspan="2">Resultados:

• Facilidad de edición de las noticias.</td></tr>
<tr><td>Aprobaci</td><td colspan="4">(Cajas para marcar el tipo de aceptación que merece el RTF)</td></tr>
</table>

ón del Producto	Aceptado (x) Sin cambios	Aceptado ( ) Con cambios	No Aceptado ( )	Pendiente ( )
Autentificación	(Firmas de cada una de las personas que participaron en el RTF)			
	Nombre: Ing. Edgar Montaluisa Sr. Darwin Herrera	Firma:	Fecha:	
	Realizado por: Sr. Darwin Herrera.		Reporte No 1	

<table>
<tr><td rowspan="2">REP-RTF-WAP_ESPEL</td><td>REPORTE RTF</td><td rowspan="2">Fecha:

15/03/2007</td><td rowspan="2">Pág. 1/1</td></tr>
<tr><td>PROYECTO: Portal WAP de la Escuela Politécnica del Ejército sede Latacunga(-WAP_ESPEL)</td></tr>
<tr><td rowspan="3">Documento Base</td><td colspan="3">(Cuales son los documentos utilizados para RTF)</td></tr>
<tr><td>Código:

• Historias de Usuarios</td><td>Nombre:

• Historias de Usuarios</td><td>Versión/Fecha

2.0/ Enero/07</td><td>Responsable:

Darwin Herrera</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>Identifica</td><td colspan="3">(Daros concretos de RTF y quienes participan)</td></tr>
</table>

<table>
<tr><td>ción de RTF</td><td>Técnicas RTF Utilizadas:

• Revisión del documento</td><td>ódulo:

.1.17</td><td>Personal:

Ing. Edgar Montaluisa</td><td>Fecha:

10/Feb/07</td></tr>
<tr><td rowspan="2">Descripción de la acción RTF</td><td colspan="4">(Descripción de la técnica y resultados esperados)</td></tr>
<tr><td colspan="2">Acción:

• Revisión de STORY NUMBER: 0010</td><td colspan="2">Resultados:

• Obtener la misma información que presenta la pagina Web</td></tr>
<tr><td rowspan="2">Aprobación del Producto</td><td colspan="4">(Cajas para marcar el tipo de aceptación que merece el RTF)</td></tr>
<tr><td>Aceptado (x)

Sin cambios</td><td>Aceptado ( )

Con cambios</td><td>No Aceptado ( )</td><td>Pendiente ( )</td></tr>
<tr><td rowspan="2">Autentificación</td><td colspan="4">(Firmas de cada una de las personas que participaron en el RTF)</td></tr>
<tr><td>Nombre:

Ing. Edgar Montaluisa

Sr. Darwin Herrera</td><td>Firma:</td><td colspan="2">Fecha:</td></tr>
</table>

	Realizado por: Sr. Darwin Herrera	Reporte No 2

REP-AUD-	**REPORTE AUDITORIA**	Fech	Pág. 1/1

<table>
<tr><td>WAP_ESPEL</td><td colspan="2">PROYECTO: Portal WAP de la Escuela Politécnica del Ejército sede Latacunga(-WAP_ESPEL)</td><td>a:
15/03/2007</td><td></td></tr>
<tr><td rowspan="2">Documento Base</td><td colspan="4">(Cuales son los documentos utilizados para la Auditoria)</td></tr>
<tr><td>Código:
• Historias de Usuarios</td><td>Nombre:
• Historias de Usuarios</td><td>Versión/Fecha
3.0/ Enero/07</td><td>Responsable:
Darwin Herrera</td></tr>
<tr><td rowspan="2">Identificación de la Auditoria</td><td colspan="4">(Daros concretos de la Auditoria y quienes participan)</td></tr>
<tr><td>Información original
• Es sistema será multiusuario</td><td>Cambios propuestos:
Vía WEB</td><td>Auditor:
Ing. Edgar Montaluisa</td><td>Fecha:
10/Feb/07</td></tr>
<tr><td rowspan="2">Cuestionario</td><td colspan="4">(Resultado obtenido)</td></tr>
<tr><td colspan="2">Pregunta:
• Se requiere que el tiempo de respuesta sea oportuno</td><td colspan="2">Resultado:
• Para que los usuarios no se cansen en esperar el tiempo de respuesta.</td></tr>
</table>

<table>
<tr><td rowspan="2">Aprobación de la Auditoria</td><td colspan="4">(Cajas para marcar el tipo de aceptación que merece la Auditoria)</td></tr>
<tr><td>Aceptado (x)
Sin cambios</td><td>Aceptado ( )
Con cambios</td><td>No ( )
Aceptado</td><td>Pendiente ( )</td></tr>
<tr><td rowspan="2">Autentificación</td><td colspan="4">(Firmas de cada una de las personas que participaron en la Auditoria)</td></tr>
<tr><td colspan="2">Nombre:
Ing. Edgar Montaluisa
Sr. Darwin Herrera</td><td>Firma:</td><td>Fecha:</td></tr>
<tr><td></td><td colspan="2">Realizado por:
Sr. Darwin Herrera</td><td>Reporte No 1</td></tr>
</table>

REP-GCS-WAP_ESPEL	**REPORTE FINAL DE LA GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE**	Fecha : 15/03/2007	Pág. 1/1
	PROYECTO: Portal WAP de la Escuela Politécnica del Ejército sede Latacunga(-WAP_ESPEL)		
Documentos de	(Listado de los documentos utilizados)		

referencia	Código: • Planificación • Historias de Usuarios • REP-REU-WAP_ESPEL • REP-RTF-SIGEDOC • REP-AUD-SIGEDOC	Nombre: • Planificación • Historias de Usuarios • Reporte de reuniones • Reporte de RTF • Reporte de Auditoria	Versión/Fecha 1.0/ Enero/07 1.0/ Enero/07 10/ Febre/07 10/ Febre/07 10/ Febre/07	Responsable: Darwin Herrera
Resulta	(Información general sobre RTF´s)			

<table>
<tr><td>dos de la RTF</td><td>Acciones:

• STORY NUMBER: 0008

• STORY NUMBER: 0004</td><td>Resultados:

• Facilidad de edición de las noticias.
• Validad contraseña encriptada</td><td>Responsables:

• Ing. Edgar Montaluisa
• Darwin Herrera</td></tr>
<tr><td rowspan="2">Auditorias</td><td colspan="3">(Breve descripción de la Auditoria y resultados esperados)</td></tr>
<tr><td colspan="2">Pregunta:

• Se requiere tiempo de respuesta sea oportuno</td><td colspan="2">Resultado:

• Para que no sea lento el tiempo de respuesta.</td></tr>
<tr><td rowspan="2">Aprobación de la GCS</td><td colspan="4">(Cajas para marcar el tipo de aceptación que merece la GCS)</td></tr>
<tr><td>Aceptado (x)

Sin cambios</td><td>Aceptado ( )

Con cambios</td><td>No ( )

Aceptado</td><td>Pendiente ( )</td></tr>
<tr><td>Autentifi</td><td colspan="4">(Firmas de cada una de las personas que participaron todo el proceso de GCS)</td></tr>
</table>

cación	Nombre: Ing. Edgar Montaluisa Sr. Darwin Herrera	Firma:	Fecha:
	Realizado por: Sr. Darwin Herrera		Reporte No 1

Historia de Usuario	
Número: 1	**Usuario:** Todos
Nombre historia: Interfase	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 4	**Iteración asignada: 1**
Programador responsable: Darwin Herrera	
Descripción: El portal Wap ofrecerá una interfaz de usuario intuitivo, fácil de aprender, sencillo de manejar y sobretodo amigable, teniendo en cuenta las limitaciones de los celulares como el tamaño de su pantalla, la capacidad de memoria, las velocidades de transmisión de datos. El sistema deberá presentar un alto grado de usabilidad. Lo deseable sería que un usuario nuevo se familiarizase con el portal en una o dos horas.	
Observaciones: **-** Los alumnos deben ingresar de manera más rápida a la consulta de notas.	

Historia de Usuario	
Número: 2	**Usuario:** Darwin Herrera
Nombre historia: Limitaciones	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Darwin Herrera	
Descripción: La información debe ser paginada para una mejor navegación del portal debido a las limitaciones de pantalla, el sistema de navegación debe ser accesible en todas las páginas para poder acceder a la información rápidamente, la información no debe tener muchos gráficos ya que algunos terminales no van ser capaces de visualizar ese tipo de información.	
Observaciones:	

Historia de Usuario	
Número: 3	**Usuario:** Metodología
Nombre historia: Consideraciones de Tecnología	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3.5	**Iteración asignada:** 1
Programador responsable: Darwin Herrera	
Descripción: Se deberá mantener los tamaños por debajo del límite máximo, con ello aseguras que los dispositivos de baja potencia accedan al contenido y se compense la lentitud de la conexión inalámbrica. Es recomendable no sobrepasar los siguientes tamaños: Tamaño general del terminal: 1.4 K. Longitud de texto: entre 16 y 20 líneas como máximo.	

Longitud de la página: no más de 15 caracteres. Longitud de texto entre los anclajes: 1 o 2 palabras como máximo. Tamaño de las etiquetas aceptar y "prev": no más de 15 caracteres entre las dos (recomendable no más de 5 caracteres cada una). Tamaño de filas de tablas: no más de 12 caracteres cada una. Longitud de lista: no más de 9 elementos. Tamaño de imagen: anchura: menor de 91 pixeles altura: menor de 47 pixeles
Observaciones:

	Historia de Usuario
Número: 4	**Usuario:** Objetivos

Nombre historia: Dependencias	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Darwin Herrera	
Descripción: El portal Wap_Espel se va a realizar con el paradigma de Wap - Wml, por lo que el sistema en si va a interactuar con terminales móviles (celulares), debido que es una aplicación orientado a Internet se va depender de un servidor Apache	
Observaciones:	

Historia de Usuario	
Número: 5	**Usuario:** Área de Organización y Sistemas
Nombre historia: Consulta de notas	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 5	**Iteración asignada:** 2
Programador responsable: Darwin Herrera	

Descripción:

Cuando el estudiante intente consultar sus notas a través del portal Wap deberá introducir su identificación (Cédula) y una contraseña, y el sistema deberá comprobar que se trata de un usuario autorizado.

La aplicación a implementar podrá interactuar con el Sistema Académico de la ESPE, en lo relacionado a consulta de notas, mediante este servicio un alumno podrá consultar a través de su celular sus notas, sin necesidad de acercarse a secretaría Académica.

Observaciones: Si el identificador introducido no corresponde a un usuario autorizado o la clave no coincide con la almacenada, se dará una indicación de error.

Historia de Usuario	
Número: 6	**Usuario:** Estudiante
Nombre historia: Noticias	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Darwin Herrera	

Descripción:

La aplicación a implementar podrá generar noticias de las actividades o eventos importantes que se llevaran a cabo en la ESPE.

Observaciones:

- Las noticias se tomaran de la base de datos del portal Web de la ESPEL

INDICE