Universidad
Carlos III de Madrid
www.uc3m.es

TRABAJO FIN DE GRADO

ACCESO WEB CONTROLADO AL EMULADOR DEL MAEMO

Autor: César Zurita Díaz

Titulación: Grado en Ingeniería Telemática

Tutor: Pablo Basanta Val

Co-tutora: Marisol García Valls

Fecha: 15 / 05 / 2013

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor Pablo la dedicación que ha puesto a lo largo de todo el proyecto, sus consejos y orientación. Además quisiera agradecer su disposición conmigo tanto en la rápida respuesta de correos, como en la disponibilidad de concertar tutorías.

En segundo lugar, quisiera agradecer a mis padres y a Sandra por haberse preocupado tanto por mí y haberme ayudado no solo durante la duración del proyecto, sino en los cuatro años de carrera. Os quiero mucho a los tres.

También quisiera agradecer a la gran cantidad de amigos que han compartido esta experiencia conmigo porque sin ellos yo no estaría aquí.

Finalmente, una mención especial para un amigo que nos dejó el año pasado y hoy debería estar haciendo su propio proyecto. D.E.P. Ignacio Álvarez Picasso.

Índice de contenidos

Índice de figuras.

Índice de tablas.

Capítulo 1. Introducción.

1.1 Objetivos y motivación.

El objetivo principal de este proyecto es desarrollar un conjunto de mecanismos para conseguir que el emulador del Maemo se convierta en una aplicación web, con el fin educativo de crear una aplicación para la asignatura Arquitectura de Sistemas [9], impartida en los distintos grados de Ingeniería en Telecomunicaciones: Grado en Ingeniería Telemática, Grado en Ingeniería de Sistemas de Comunicaciones y Grado en Ingeniería de Sistemas Audiovisuales.

La idea principal consiste en cambiar el paradigma de ejecución en local de una imagen virtual de gran dimensión por una arquitectura multinivel (*multi-tier*), donde se ofrece una interfaz ligera en los clientes (interfaz web) que realiza las peticiones en remoto a los servicios que disponen de la lógica que se desea ejecutar.

Es decir, en la ejecución local la realización de la práctica requeriría que cada alumno tuviera una imagen propia instalada en su PC del laboratorio. Dicha imagen ocupa alrededor de 3GB, por lo que cada alumno debería descargarla previamente a la realización de la práctica. Con esta motivación, se propuso la idea de realizar una aplicación que pudiera acceder vía web a la máquina virtual.

Conforme se ha ido desarrollando el proyecto, se han establecido una serie de objetivos, los cuales se detallan a continuación:

- Introducción e investigación de la plataforma de desarrollo de software para dispositivos personales. Estudio de la arquitectura cliente/servidor multinivel para diseñar una aplicación en la que se comuniquen, mediante el uso de de servlets y JSPs.
- Desglosar por fases las distintas actividades del proyecto, para seguir un orden previamente estipulado y planificar distintas soluciones para afrontar problemas surgidos durante el desarrollo.
- Migración de un entorno de compilación cruzado de consola a otro web e implementación de una aplicación real y útil, enfocada a la docencia con el fin de desarrollar la solución en un entorno real.
- Evaluación de la solución desarrollada, estudiando su viabilidad para su implementación en la universidad y explicando el porqué en la toma de decisiones frente a las soluciones descartadas.

1.2 Estructura del documento.

Para el desarrollo de los objetivos parciales el documento tiene la siguiente estructura:

- **Capítulo 1. Introducción.**

En este primer capítulo se ofrece una visión global de las motivaciones y objetivos que se han perseguido a lo largo del desarrollo.

- **Capítulo 2. Análisis del estado del arte.**

Este capítulo enuncia las distintas tecnologías utilizadas, las cuales debe conocer el lector antes de adentrase en la solución técnica.

- **Capítulo 3. Diseño de la solución técnica.**

El capítulo 3 aborda el desarrollo de la aplicación, incluyendo las decisiones tomadas en los distintos casos y un desglose de los elementos utilizados.

- **Capítulo 4. Resultados e implementación.**

En este capítulo se muestran los resultados que deberían obtenerse mediante un correcto funcionamiento de la aplicación, además de la implementación de la solución técnica.

- **Capítulo 5. Planificación del trabajo y presupuesto.**

En el capítulo 5 se planifican las labores de análisis y desarrollo de la aplicación si se implementara en un entorno real por último se da un presupuesto global.

- **Capítulo 6. Conclusiones.**

Este capítulo consiste en un análisis de los objetivos cumplidos, las conclusiones y las líneas de trabajo futuras de la aplicación.

- **Capítulo 7. Anexos.**

El capítulo 7 contiene tres anexos. El primero es un manual para la configuración de Tomcat en la máquina virtual. En el segundo se detalla un glosario de la terminología utilizada a lo largo del documento. Por último, el tercero detalla como importar y exportar una imagen en VirtualBox.

- **Capítulo 8. Referencias.**

Por último este capítulo recoge las bibliografías utilizadas para la realización de la memoria.

Capítulo 2. Análisis del estado del arte.

Para poder utilizar la aplicación web del Maemo [10], es necesario tener un conocimiento más profundo acerca de las distintas tecnologías que la forman. Por ello, en este capítulo se hace hincapié en las herramientas de las que está compuesta.

2.1 Maemo SDK.

Es el entorno de desarrollo para Maemo, el cual se puede instalar en ordenadores Linux y también se puede descargar como máquina virtual.

El SDK crea un entorno Maemo en el ordenador Linux, denominado ScratchBox [11]. El lenguaje de programación soportado es C/C++, habiendo también buen soporte de Python.

ScratchBox permite realizar la compilación cruzada para el dispositivo.

Maemo SDK+ [1] proporciona un entorno de desarrollo cruzado para dispositivos basados en Maemo de Nokia.

El desarrollo cruzado difiere del desarrollo de la aplicación ordinaria en que las aplicaciones están desarrolladas en un máquina concreta y posteriormente son ejecutadas en una máquina diferente, posiblemente con una arquitectura hardware también distinta. En el desarrollo de aplicaciones ordinario (Ver Figura 1) el proceso de desarrollo sigue los siguientes pasos:

- Comienza con la instalación del entorno de desarrollo.
- A continuación, una solicitud inicial se desarrolla con las herramientas de edición.
- Cuando el desarrollo de aplicaciones ha llegado al punto de ser compilada, se compila usando las herramientas necesarias para su compilación.
- Si la compilación se ha realizado correctamente, la aplicación se instala por primera vez o se ejecuta directamente a ver qué ocurre.
- A continuación, la aplicación se desarrolla y el proceso continúa con una nueva compilación.

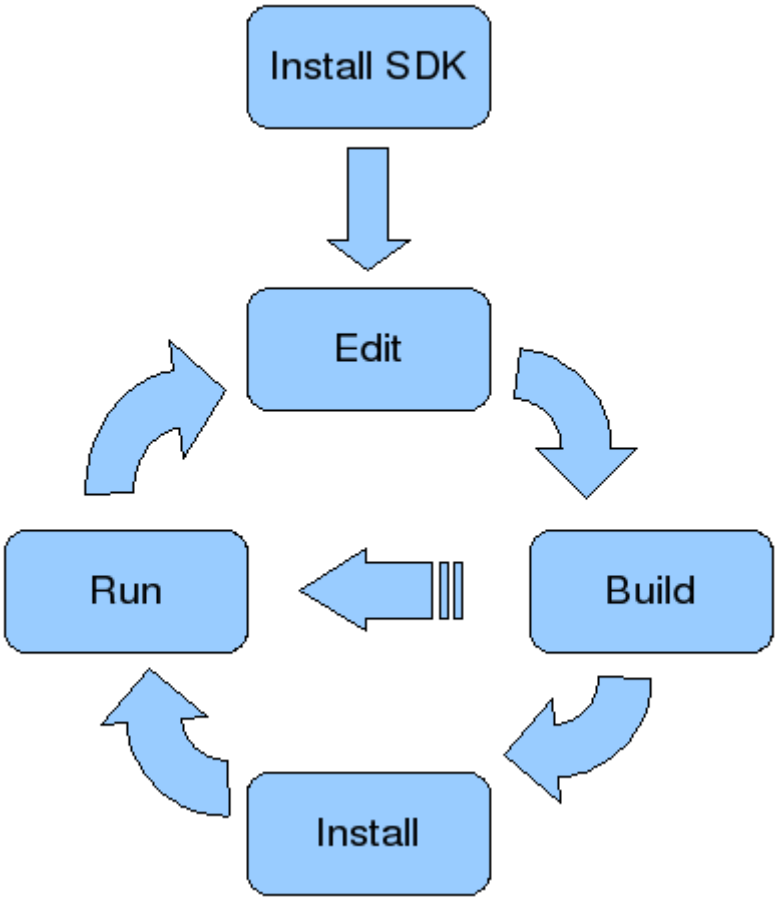


Figura 1. Ciclo de desarrollo de la aplicación en Maemo.

El desarrollo cruzado, sin embargo, es más complicado. Se trata de una estación de trabajo de desarrollo (huésped) donde se compila el software y un dispositivo de destino específico como Nokia N810 en el que finalmente se ejecuta la aplicación final.

El dispositivo de destino ejecuta una versión específica del sistema operativo que es soportado por el entorno de compilación. Además, un entorno de ejecución simulada puede ser ejecutado en el huésped para acelerar la depuración, permitiendo a la aplicación ser ejecutada sin necesidad de instalarla en un dispositivo real.

Así, el desarrollo cruzado implica múltiples universos (Ver Figura 2).

- Maemo SDK + sí está instalado en un universo de acogida que consiste en todo lo que existe en la estación de trabajo del desarrollador.

- El universo de compilación cruzada que consiste en librerías del dispositivo de destino y archivos de cabecera y herramientas capaces de compilar software para el dispositivo de destino.
- El tercer universo es el universo de ejecución que proporciona los medios para instalar y ejecutar software de destino.

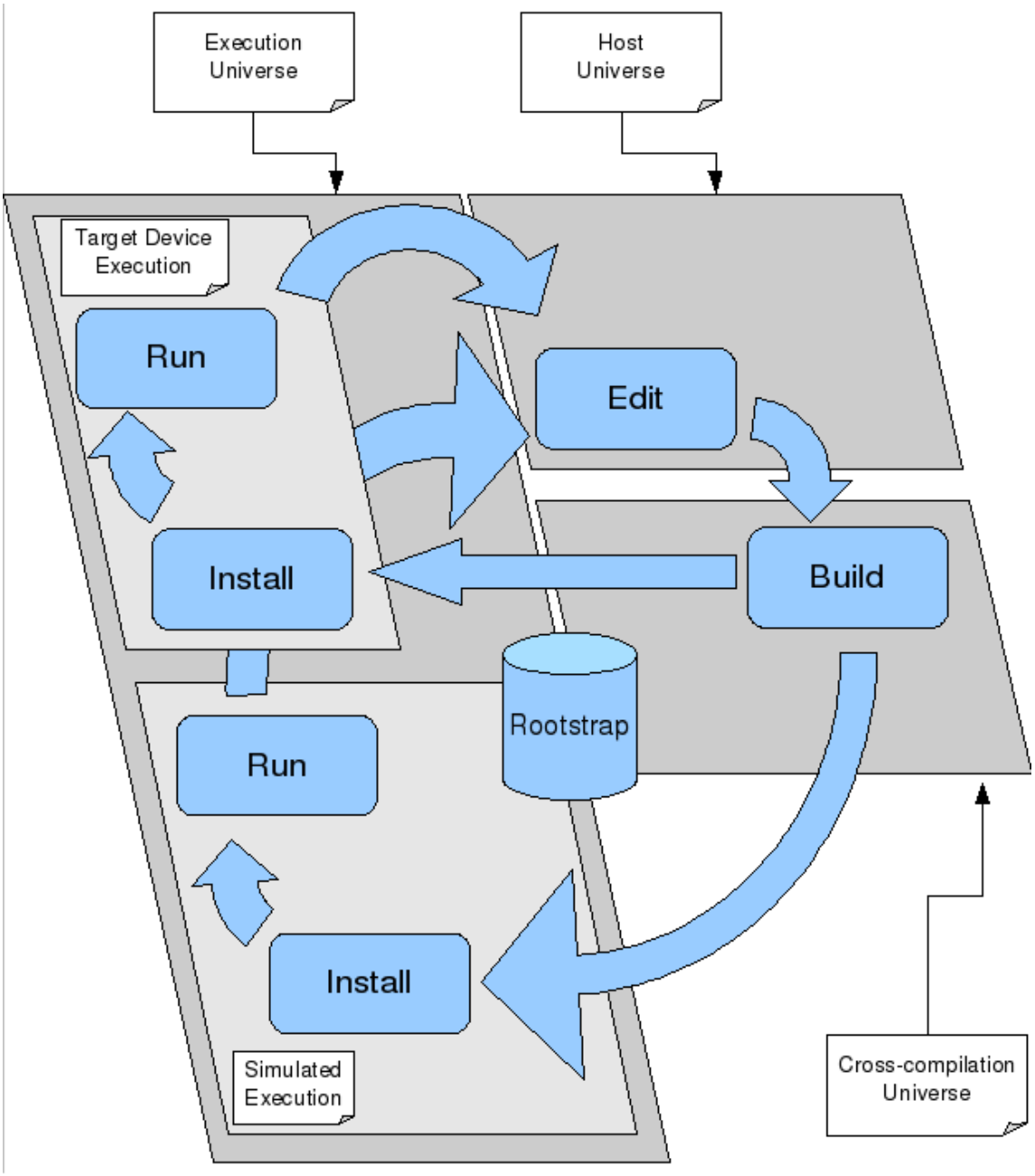


Figura 2. Esquema de desarrollo-cruzado en Maemo.

Maemo SDK + se basa en ScratchBox 2. ScratchBox2 (sbox2 o sb2) es un conjunto de herramientas de compilación cruzada diseñada para hacer el desarrollo de aplicaciones más fácil.

También proporciona un conjunto completo de herramientas para integrar y compilar en forma cruzada a toda la distribución de Linux.

Sin Scratchbox2, se debería configurar manualmente muchos parámetros e intervenir en el proceso "configure" para ser capaz de generar código.

Sin embargo, Scratchbox2 permite configurar un entorno "virtual" que emule a las herramientas y ejecutables haciéndolas pensar que se están ejecutando directamente sobre el objetivo integrado con su configuración. Se puede ver esta configuración con el siguiente comando:

$ sb2 –h

Algunas de las opciones de sb2 para la línea de comandos son:

- -v Mostrar la versión.
- -L level Habilitar el registro.
- -d Modo *debug*.
- -h Imprimir esta ayuda.
- -e Modo emulación.
- -R Simula permisos de *root*.

2.2 Máquina Virtual: VirtualBox.

2.2.1 Introducción.

El modelo de máquina virtual está basado en la arquitectura cliente/servidor, donde cada cliente funciona como una imagen virtual de la capa hardware. Este modelo permite que el sistema operativo cliente funcione sin modificaciones. Además permite al administrador crear diferentes sistemas clientes con sistemas operativos independientes entre sí.

La ventaja principal de este modelo radica en el desconocimiento por parte de los sistemas huésped del sistema hardware real sobre el que está instalado. Sin embargo, realmente todos los sistemas virtuales hacen uso de recursos hardware físicos. Estos recursos son administrados por un sistema denominado hypervisor que coordina las instrucciones CPU.

El hypervisor es denominado comúnmente monitor de máquina virtual (VMM) y es el encargado de validar todas las peticiones e instrucciones de los sistemas virtuales a la CPU, supervisando todas las ejecuciones que requieran cambios de privilegios. Dos sistemas típicos de servidores virtuales son Vmware y VirtualBox.

2.2.2 VirtualBox.

VirtualBox [2] es un software de virtualización (tipo *full virtualizer*) para hardware x86 y AMD64/Intel64 dirigido tanto a empresas como usuarios personales. Actualmente es desarrollado por Oracle Corporation como parte de su familia de productos de virtualización.

Por medio de esta aplicación es posible instalar sistemas operativos adicionales, conocidos como “sistemas invitados”, dentro de otro sistema operativo “anfitrión”, cada uno con su propio ambiente virtual. Por ejemplo, se podrían instalar diferentes distribuciones de GNU/Linux en VirtualBox instalado en Windows XP o viceversa.

Entre los sistemas operativos soportados (en modo anfitrión) se encuentran GNU/Linux, Mac OS X, OS/2 Warp, Windows, y Solaris/OpenSolaris, y dentro de éstos es posible virtualizar los sistemas operativos FreeBSD, GNU/Linux, OpenBSD, OS/2 Warp, Windows, Solaris, MS-DOS y muchos otros.

La Figura 3 muestra el logo de Oracle VirtualBox.

Figura 3. Logo de Oracle VM VirtualBox.

Algunas de las principales características son:

- **Modularidad:** VirtualBox tiene un diseño extremadamente modular con interfaces bien definidas de programación interna.
- **Descripciones de Virtual Machine en XML:** Las opciones de configuración de las Virtual Machine se almacenan enteramente en ficheros XML y son independientes del hardware local, por lo que pueden ser portadas fácilmente a otros equipos.
- **Software adicional para los Sistemas operativos Guest:** Virtualbox tiene software especial que puede ser instalado dentro de Windows, Linux y Solaris (virtuales) para lograr una integración y un rendimiento mayor. Entre las características proporcionadas por este software adicional encontramos la integración puntero del ratón, soluciones arbitrarias de pantalla (por ejemplo cambiar el tamaño de la ventana del sistema operativo *guest*).
- **Carpetas compartidas:** VirtualBox permite declarar ciertos directorios del sistema operativo host como "*shared folders*" para el intercambio de datos entre el sistema operativo *host* y *guest* o entre los *guest*.
- **Soporte integrado iSCSI:** Esta característica única permite conectar una máquina virtual directamente a un servidor de almacenamiento iSCSI sin pasar por el sistema Host.
- **Multigeneración de Snapshot jerarquizados:** Virtualbox puede guardar vistas del estado de la máquina virtual. Se puede ir hacia atrás en el tiempo y recuperar la máquina virtual al estado anterior.

- **Arranque por red PXE:** Las tarjetas de red virtuales de VirtualBox soportan totalmente el arranque remoto a través de la Preboot Execution Environment (PXE).
- **Controladores virtuales USB:** VirtualBox implementa un controlador USB virtual y permite conectar dispositivos USB de manera arbitraria a los sistemas operativos *guest* sin tener que instalar controladores específicos en el host.
- **Protocolo de Escritorio Remoto (RDP):** A diferencia de cualquier otro software de virtualización, VirtualBox es altamente compatible con el estándar Remote Desk Protocol. Una máquina virtual puede actuar como un servidor RDP, lo que permite "ejecutar" la máquina virtual de manera remota en un cliente ligero que solo muestra los datos de RDP.
- **USB sobre RDP:** Otra característica única de VirtualBox es que la máquina servidor de RDP puede acceder a los dispositivos USB conectados del cliente RDP.

La Figura 4 muestra los detalles de la máquina virtual seleccionada en VirtualBox.

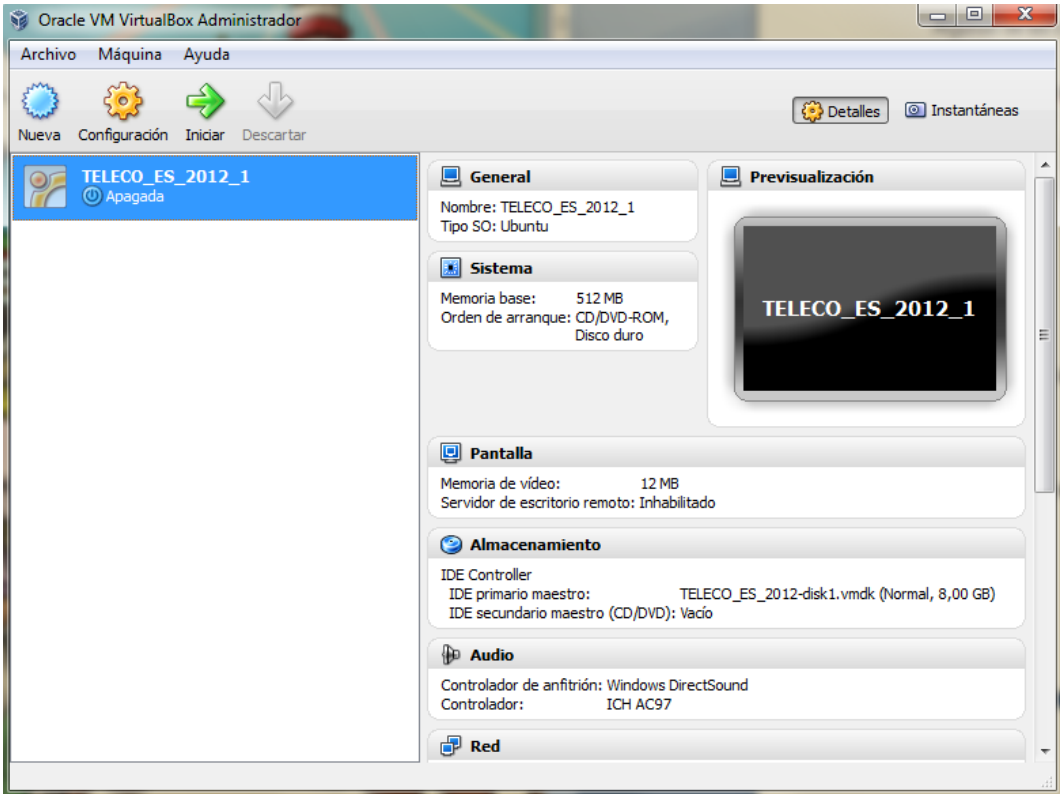


Figura 4. Detalles de la VM Teleco.

2.3 Tomcat: Servidor web.

2.3.1 Introducción.

Tomcat [3] (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de Java Server Pages (JSP) de Sun Microsystems.

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache.

Tomcat puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

La Figura 5 muestra el logo de Apache Tomcat.

Figura 5. Logo de Apache Tomcat.

2.3.2 Estado de desarrollo.

Tomcat es mantenido y desarrollado por miembros de la compañía Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la licencia Apache Software Licence [14].

Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 7.x que implementan las especificaciones de Servlet 3.0 y de JSP 2.2. Después de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

2.3.3 Estructura de los directorios.

La jerarquía de directorios de instalación de Tomcat (ver Figura 6) incluye:

- **bin** - arranque, cierre, y otros scripts y ejecutables.
- **conf** - ficheros XML y los correspondientes DTD para la configuración de Tomcat.
- **lib** – librerías que utilizará Tomcat en su ejecución.
- **logs** - logs de Catalina y de las aplicaciones.
- **temp** – carpeta temporal.
- **webapps** - directorio que contiene las aplicaciones Web.
- **work** - almacenamiento temporal de ficheros y directorios.

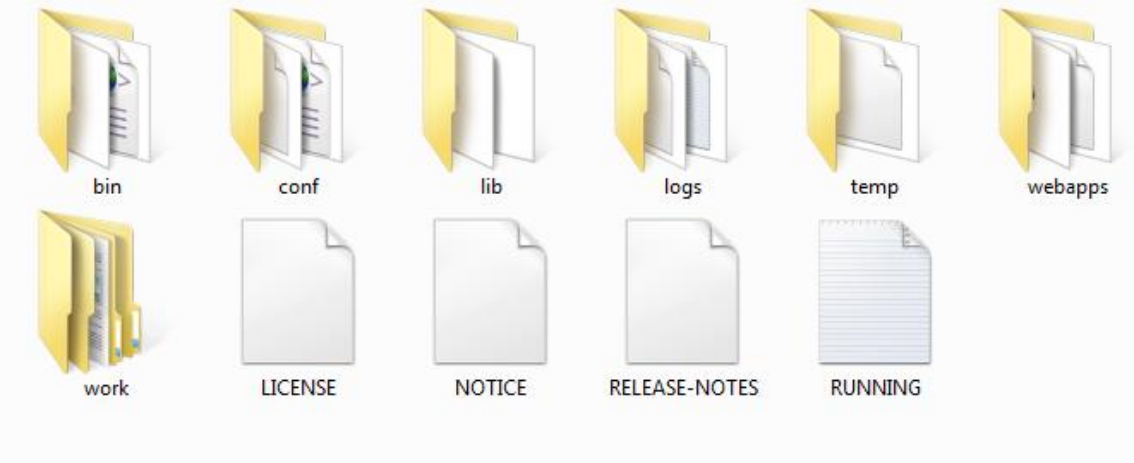


Figura 6. Estructura de los directorios en Tomcat.

2.4 Servlets.

2.4.1 Definición.

Los servlets [4] son componentes de una aplicación web que se ejecutan en el servidor.

- Permiten extender la funcionalidad del servidor (tanto de servidores HTTP como de otro tipo de servidores como por ejemplo ftp).
- Son una alternativa a CGI.
- Cada petición se ejecuta en un hilo diferente. Los servlets quedan residentes en memoria cuando la petición termina.

Un servlet es un pequeño código Java que el servidor Web carga para manejar peticiones del cliente.

- Estas clases Java utilizan el API Servlet.
- Se cargan y ejecutan dentro de un servicio de red (por ejemplo, un servidor Web).
- Implementan determinados interfaces que le permiten:
  - Recibir una petición HTTP.
  - Generar una respuesta.

Los Servlets son a Servidores como Applets son a Navegadores. Con la diferencia de que los servlets no suelen usar GUI.

2.4.2 Funcionamiento.

Para entender el funcionamiento de un servlet (ver Figura 7), hay que pasar por una serie de pasos:

1. Se parte de un fichero escrito en HTML (es decir, *Fichero.html*) que tenga un formulario (<FORM>) accesible a través de Internet. El formulario deberá especificar las diversas formas de paso de parámetros.
2. El cliente accederá al fichero HTML a través de un navegador (Firefox, Chrome, etc.), rellenará el formulario y pulsará la tecla de Aceptar (Submit).
3. La petición del cliente es atendida por un Servlet, escrito en Java, que da la debida respuesta tras procesarse la petición.
4. El cliente visualizará la respuesta.

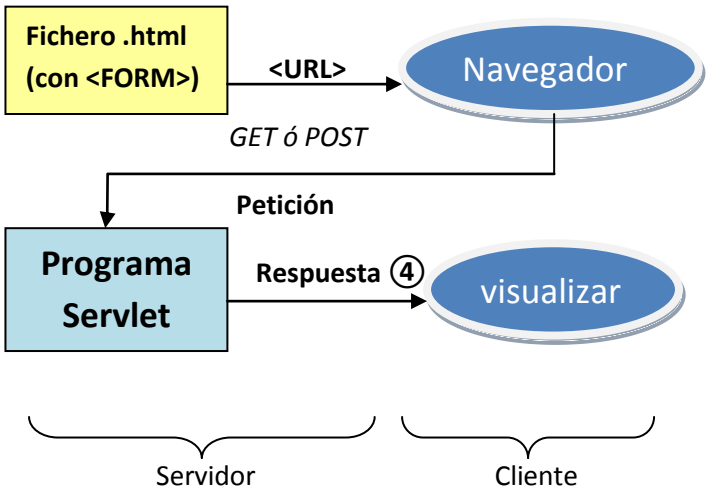


Figura 7. Funcionamiento de un Servlet.

2.4.3 Ciclo de vida de un Servlet.

Todos los Servlets tienen el mismo ciclo de vida que viene dado por sus métodos, con las siguientes fases:

1. Instalación e inicialización del servlet. Si aún no existe ninguna instancia del servlet (es la primera llamada), el contenedor web:
   - Primero carga la clase del servlet.
   - Después crea una instancia y la inicializa invocando su método *init*.
2. El servidor puede servir continuamente peticiones. Para cada llamada:
   - El contenedor crea un nuevo thread (hilo).
   - Se invoca al método *service* de dicho thread.
   - Los servlets residen típicamente en servidores multithread pero el programador es el responsable de sincronizar el acceso a los recursos compartidos.
   - También se puede restringir el acceso a un único thread.
3. El método *service* determina tipo de petición que ha llegado y llama al método correspondiente *doGet*, *doPost*, etc.

Un servlet se desactiva cuando recibe una llamada al método *destroy* o cuando su servidor se muere.

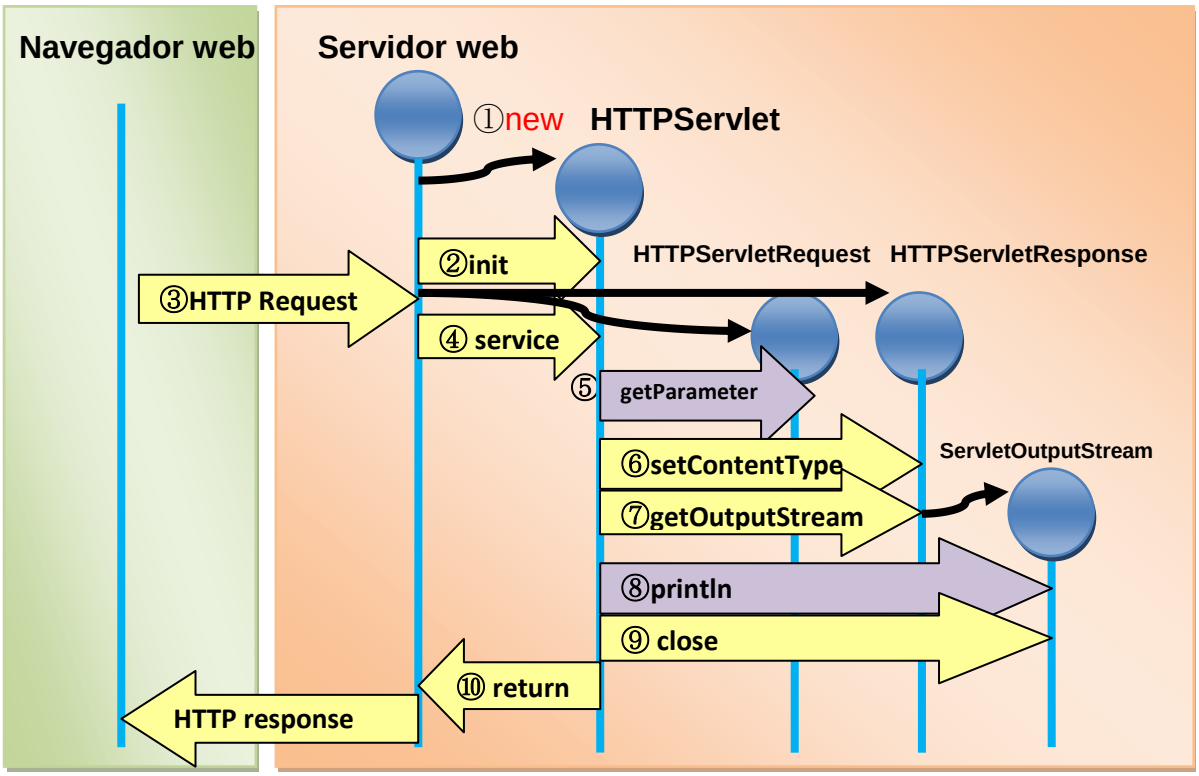


Figura 8. Escenario básico de un servlet.

2.5 Java Server Pages (JSPs).

JSP [5] es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Las JSPs permiten la utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de bibliotecas de etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas. Se puede considerar una alternativa a los servlets o se pueden utilizar de forma complementaria.

Los JSPs son en realidad servlets: un JSP se compila a un programa en Java la primera vez que se invoca y del programa en Java se crea una clase que se empieza a ejecutar en el servidor como un servlet. La principal diferencia entre los servlets y los JSPs es el enfoque de la programación: un JSP es una página web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa Java puro que recibe peticiones y genera a partir de ellas una página web.

2.6 Hojas de estilo en cascada (CSS).

El modo de funcionamiento de las CSS [6] consiste en definir, mediante una sintaxis especial, la forma de presentación que le aplicaremos a:

- Una web entera, de modo que se puede definir la forma de toda la web de una sola vez.
- Un documento HTML o página, se puede definir desde la forma, en un pequeño trozo de código en la cabecera, a toda la página.
- Una porción del documento, aplicando estilos visibles en un trozo de la página.
- Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta.

2.7 JavaScript.

JavaScript [7] es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, en bases de datos locales al navegador. Existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

2.8 Acceso con seguridad LDAP.

LDAP [8] son las siglas de Lightweight Directory Access Protocol (en español Protocolo Ligero de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también se considera una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

Un directorio es un conjunto de objetos con atributos organizados en una manera lógica y jerárquica. El ejemplo más común es el directorio telefónico, que consiste en una serie de nombres (personas u organizaciones) que están ordenados alfabéticamente, con cada nombre teniendo una dirección y un número de teléfono adjuntos. Para entender mejor, es un libro o carpeta, en la cual se escriben nombres de personas, teléfonos y direcciones, y se ordena alfabéticamente.

Habitualmente, almacena la información de autenticación (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados, etc). A manera de síntesis, LDAP es un protocolo de acceso unificado a un conjunto de información sobre una red.

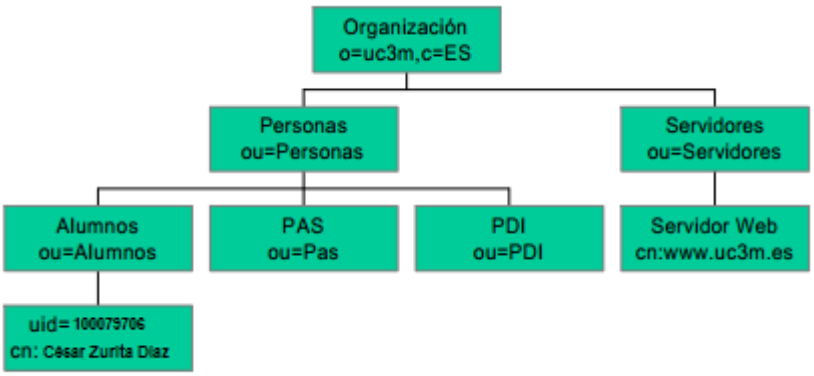


Figura 9. Estructura LDAP de la UC3M.

En el caso de este proyecto, almacenará el NIA de los alumnos de la Universidad Carlos III (ver figura 9) y su contraseña para poder acceder a la aplicación. LDAP es totalmente seguro, ya que, ni el cliente, ni el servidor llegan a ver nunca las contraseñas de los alumnos, simplemente reciben un parámetro si es correcta y otro distinto si no lo es.

Para realizar la búsqueda, se deben especificar los siguientes parámetros:

- **Base,** DN que indica el punto de partida para la búsqueda.
- **Scope,** ámbito de la búsqueda, puede ser:
  - **Base,** solo se busca en la entrada base.
  - **One,** se busca en el nivel inmediatamente inferior a la entrada base.
  - **Subtree,** se busca en todo el subárbol bajo la entrada base.
- **Filtro de búsqueda,** indica el criterio de búsqueda.
- **Atributos a devolver,** se puede indicar que atributos se devuelven y si se devuelve el valor del atributo o el tipo de dato contenido
- **Alias derreferencing,** indica si el servidor debe seguir las entradas referral o por el contrario debe enviarse la petición al servidor referenciado.
- **Límite,** indica el número máximo de entradas que serán devueltas o el tiempo empleado para realizar dicha búsqueda. Los servidores pueden imponer límites más estrictos que los indicados por los clientes.

2.9 FileUpload.

Descargar ficheros es sencillo desde el servidor, el problema surge cuando queremos subirlos, los JSP y Servlets de Java no tienen mecanismos para poder manejar la subida de ficheros desde formularios, siendo un problema tratar de extraer el contenido desde *HTTP request*. Una mejor opción que nos puede ayudar es utilizar librerías de terceros. Para este propósito existe diseñada una robusta librería llamada FileUpload [13] del paquete Apache Jakarta Commons FileUpload.

El paquete Commons FileUpload hace que sea más fácil subir, de forma robusta y con alto rendimiento, archivos a los servlets y aplicaciones web. FileUpload analiza peticiones HTTP que se ajusten a la RFC 1867. Es decir, si una solicitud HTTP se envía mediante un formulario con el método **"POST"** y con un tipo de contenido **"multipart / form-data"** entonces FileUpload puede analizar esa petición y poner los resultados a disposición del servidor.

La Figura 10 muestra el formulario tipo para subir cualquier archivo.

```
<form method="POST" enctype="multipart/form-data" action="fup.cgi">
  File to upload: <input type="file" name="upfile"><br/>
  Notes about the file: <input type="text" name="note"><br/>
  <br/>
  <input type="submit" value="Press"> to upload the file!
</form>
```

Figura 10. Formulario tipo para subir archivos.

Capítulo 3. Diseño de la solución técnica

Una vez analizadas las distintas tecnologías a utilizar en el proyecto, se describe en este capítulo el diseño de la aplicación web que está dividida claramente en dos grandes bloques; estos se desglosarán en las sucesivas secciones del capítulo.

Dichos bloques a analizar son el cliente web y el servidor web.

3.1 Cliente web.

El cliente web es la parte desde la que el usuario interactúa con la aplicación e incluye campos que contienen la información necesaria para la configuración de la posterior validación. Está compuesto por una página JSP, que incluye la presentación con código HTML, una hoja de estilo CSS que asocia los distintos componentes de manera ordenada y un servlet que gestiona las llamadas a la página y los datos que se envían desde ella.

En la Figura 11 y Figura 12 se pueden ver la estructura del cliente web en el modo autenticación y en el modo consola.

Figura 11. Cliente Web en modo autenticación.

Figura 12. Cliente Web en modo consola.

En esta aplicación existen dos vistas desde el punto de vista del usuario. Es necesario autenticarse correctamente para poder acceder desde el modo autenticación al modo consola.

- **Modo autenticación.**

El cual consta únicamente del bloque autenticación, donde el usuario debe rellenar los campos para autenticarse. Si la autenticación se realiza correctamente se accede al modo consola.

- **Modo consola.**

Existen varios bloques dentro del modo consola: la consola de comandos, el *pwd*, el bloque de subida de archivos y el bloque de usuario y *logout*.

Las entidades principales de la aplicación, las cuales interactúan entre sí, son:

- **Portada:** Es la vista que tiene el cliente de la aplicación, en ella se encuentran los dos modos: autenticación y consola.
- **Autenticación:** Se ayuda de la entidad *LDAPHandler* para comprobar si la autenticación ha sido correcta o no.
- **Cerrar:** Hace *logout* y vuelve al modo autenticación.
- **Buscar:** Procesa comandos ayudándose de la entidad *Base*.
- **Fichero:** Sube el fichero al directorio donde se encuentre el modo consola.

La Figura 13 muestra la forma en la que interactúan las distintas entidades y la Figura 14 muestra cómo interactúan el cliente y el servidor.

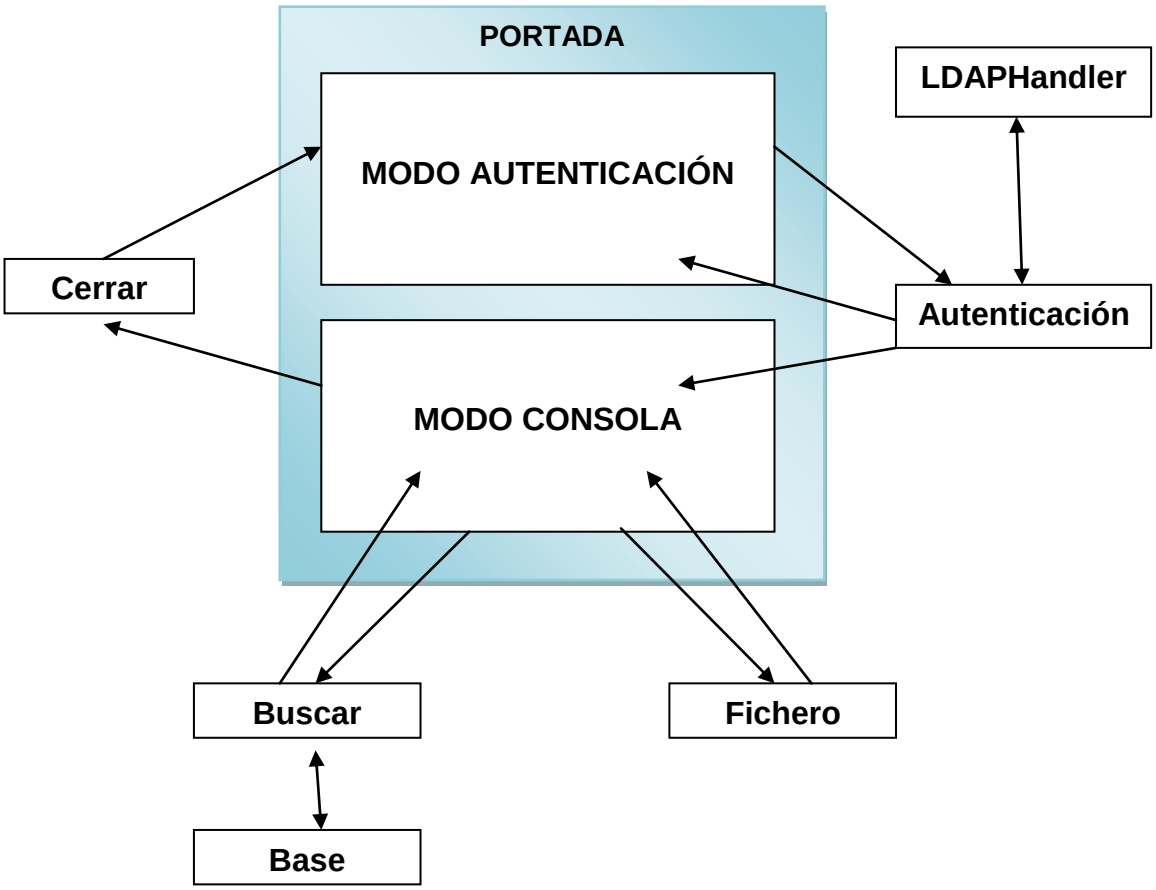


Figura 13. Interactuación entre las distintas entidades.

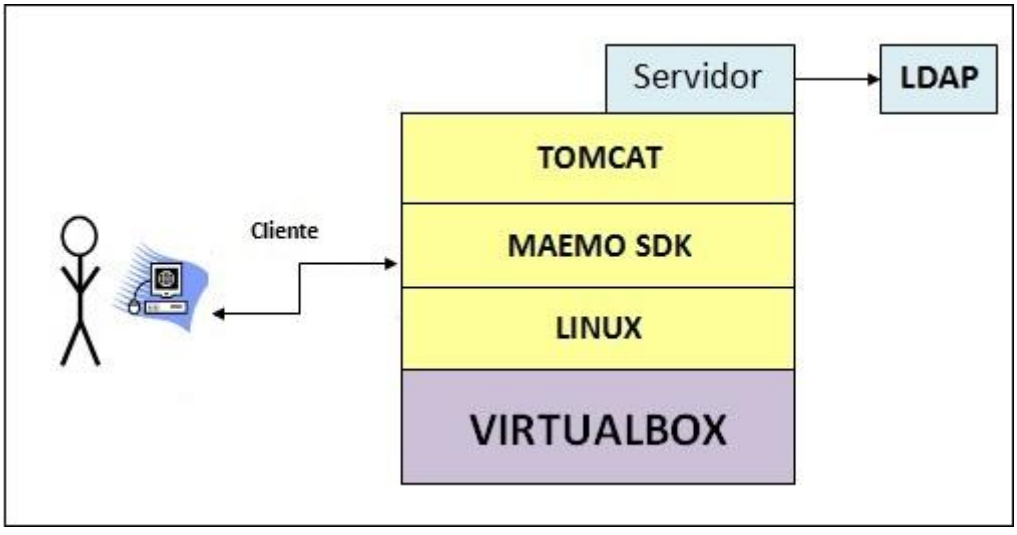


Figura 14. Interactuación entre cliente/servidor.

3.1.1 Autenticación con LDAP.

El alumno accede directamente al modo autenticación, en el que se puede encontrar el formulario de autenticación que contiene dos campos que el usuario debe rellenar: el campo *Usuario* y el campo *Password*, para rellenar con el NIA y la contraseña respectivamente.

El acceso a la aplicación de los usuarios, está regulada mediante la autenticación LDAP personalizada para alumnos de la Universidad Carlos III de Madrid mediante la clase *LDAPHandler.java*, la cual es utilizada como auxiliar por el servlet de recepción de los datos de autenticación, la clase *Autenticacion.java*.

Para poder autenticarse en la aplicación el alumno debe escribir el NIA y la contraseña que utilice para acceder a Campus Global. Una vez introducidas debe pulsar el botón “Validar” para poder salir del modo autenticación y entrar al modo consola. En la Figura 15 se muestra el formulario de autenticación.

Figura 15. Formulario de autenticación.

3.1.2 Restricciones en autenticación.

Existen una serie de restricciones en el formulario de autenticación controladas mediante JavaScript para comprobar de forma local si existen errores en formularios antes de que éstos sean enviados.

- Si el alumno deja algún campo del formulario vacío y valida el envío de dicho formulario, esto es capturado en el lado del cliente y un mensaje será mostrado informando del error al alumno. En cualquier caso, desde el servidor, se comprueba que ninguno de los campos es vacío (*null*). La Figura 16 y la Figura 17 muestran los dos mensajes lanzados por la función de JavaScript en caso de dejar algún campo del formulario vacío.

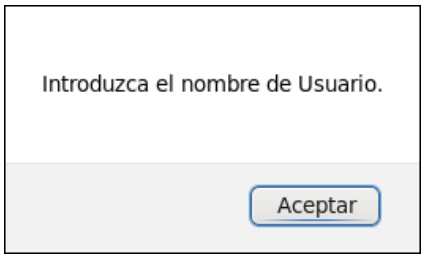


Figura 16. Mensaje de NIA vacío.

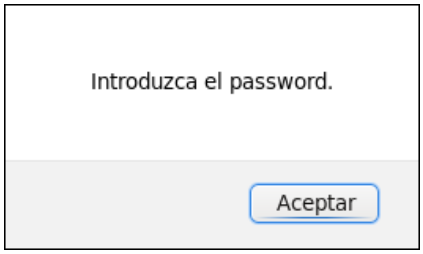


Figura 17. Mensaje de contraseña vacía.

- La otra restricción en el formulario de autenticación, es la longitud del NIA, el cual se sabe que debe tener 9 caracteres. Con lo cual, si el alumno introdujera un NIA con una longitud incorrecta se lanzaría un mensaje de error, indicándole que no es un NIA válido. Con esto se evita la espera de que el servidor compruebe mediante LDAP si la validación ha sido correcta, cuando desde un principio se sabe que no será así. La Figura 18 muestra el mensaje lanzado por la función de JavaScript en caso de que la longitud del NIA no sea la adecuada.

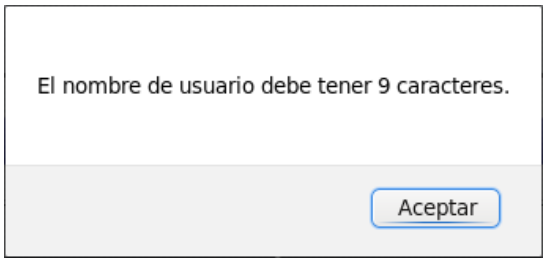


Figura 18. Mensaje de longitud de NIA incorrecto.

3.1.3 Consola de comandos.

Se ha emulado un terminal de comandos para que el usuario pueda enviar comandos al servidor y pueda realizar la práctica. La respuesta al comando aparecerá en su totalidad en la pantalla de salida de comandos.

El comando se enviará en un formulario a un servlet de recepción el cual lo gestionará de manera oportuna.

La pantalla de salida de comandos mostrará la respuesta al comando, tal y como debería aparecer en el terminal BASH. En un apartado posterior se comentará la especial gestión del comando "*cd*", pero desde el punto de vista del cliente no se tendrá en cuenta.

Con respecto al diseño del terminal, para que parezca exactamente al utilizado dentro de la máquina virtual, con la hoja de estilo CSS se le da la apariencia necesaria como muestra la Figura 19.

Figura 19. Consola de comandos.

3.1.4 Restricciones en enviar comando.

Existen unas restricciones para impedir al usuario enviar comandos vacíos o comandos con un solo carácter. Con esto evitamos comandos que ya sabemos de antemano van a devolver un error.

Mediante una función JavaScript se comprueba que el comando a enviar no esté vacío o no tenga únicamente un carácter, lanzando los mensajes apropiados en caso contrario. La Figura 20 y Figura 21 muestran dichos mensajes.

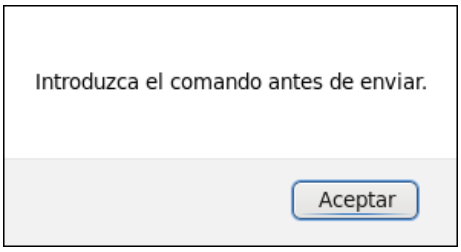


Figura 20. Mensaje de comando vacío.

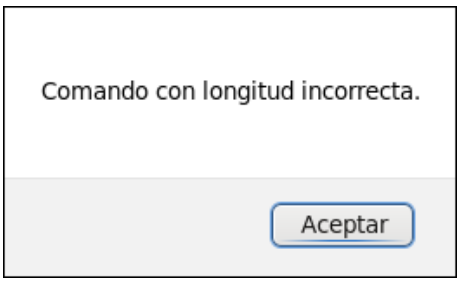


Figura 21. Mensaje de comando de un único carácter.

3.1.5 El directorio actual de trabajo.

El comando *pwd* (Print Working Directory) se utiliza para imprimir el nombre del directorio actual en una sesión de comandos bajo un sistema operativo Unix.

Para que el usuario sepa en qué directorio se encuentra en cada momento, se ha añadido en la esquina superior izquierda del terminal de comandos dicho directorio de trabajo. En la Figura 22 se puede observar dicho cuadro en el directorio inicial.

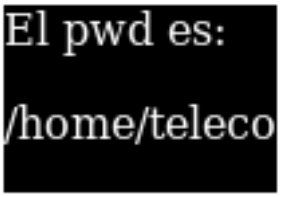


Figura 22. Pwd inicial.

3.1.6 Subida de archivos.

La aplicación permite un método de subida de archivos mediante la librería FileUpload. El alumno podrá subir sus propios archivos ya sean en formato C o ZIP (conteniendo en su interior únicamente archivos *.c*).

El usuario será quien decida donde subir sus ficheros y como gestionarlos ya que la subida de archivos se realizará en el directorio en el cual nos encontremos, es decir, en el *pwd*.

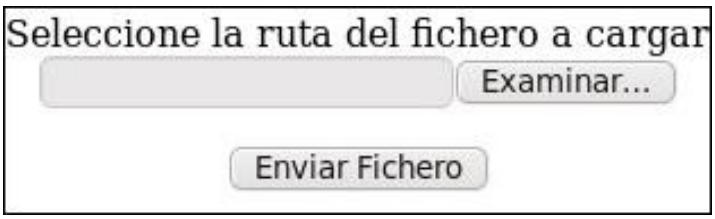


Figura 23. Subida de archivos.

La clase encargada de procesar el fichero es *Fichero.java* la cual utiliza el pwd para guardar el fichero en dicho directorio.

3.1.7 Restricciones en subida de archivos.

Existen unas restricciones para impedir subir al usuario a la aplicación cualquier tipo de archivos. La función de JavaScript comprueba que la extensión de los archivos sea "*.c*" o "*.zip*" y en caso contrario lanza un mensaje de error indicando que el formato del archivo no es el correcto, tal y como muestra la Figura 24.

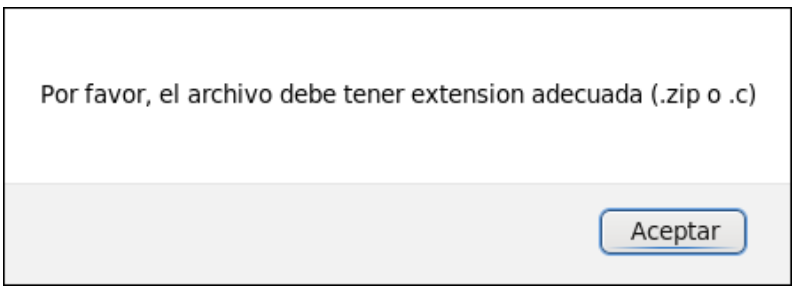


Figura 24. Mensaje de formato de archivo no aceptado.

3.1.8 Usuario y salida.

Ha sido definida la clase Usuario.java que guarda un *string* NIA y un *string* contraseña. La necesidad de esta clase es capital, no solo para la autenticación, sino para mostrar al usuario su NIA mientras esté autenticado.

Para cerrar sesión se ha definido la clase Cerrar.java que hace *logout* cuando el usuario pulsa el botón de "*Cerrar Sesion*".

En la Figura 25 se muestra una imagen del bloque que muestra el nombre de usuario y *logout*.

Figura 25. Bloque usuario y logout.

3.2 Servidor Web.

El servidor web es el encargado de gestionar los comandos enviados por el cliente. Su función es, básicamente, desde un punto de vista menos técnico, recibir un comando, procesarlo, copiar la respuesta que generaría el terminal de bash y enviársela al cliente para que la imprima por pantalla.

3.2.1 Servlet de recepción de comandos.

El servlet de recepción es la clase *Buscar.java* que utiliza como clase auxiliar la clase *Base.java*.

La metodología de la clase *Buscar.java* es:

- Recibir el comando enviado por el cliente.
- Comprobar si el comando es "*cd*", ya que éste debe ser tratado de forma especial, como ya veremos más adelante.
- Obtener el pwd actual en el cual se encuentra el cliente.

- Procesar el comando mediante la función *terminal(String pwd, String comando, HttpSession sesion)*, la cual está alojada dentro de la clase auxiliar Base.java y será la encargada de copiar la salida del comando en la sesión para que el usuario pueda acceder a ella desde la aplicación.

En la Figura 26 se incluye un diagrama de estados de la metodología de la clase *Buscar.java*.

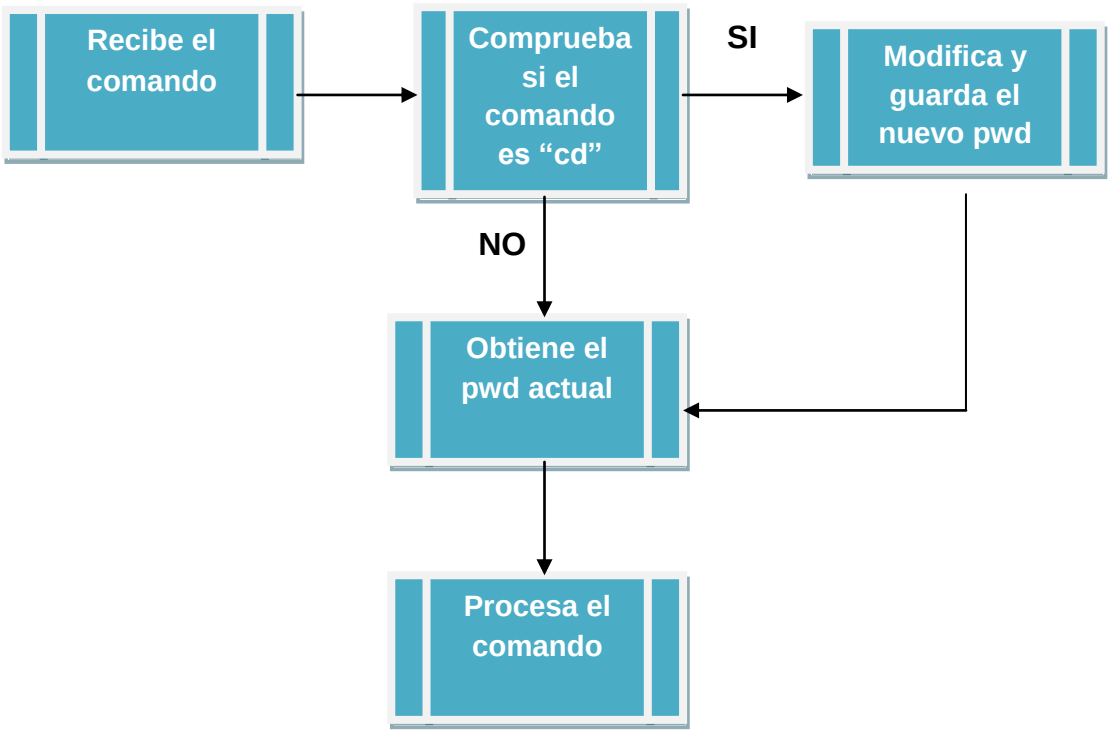


Figura 26. Diagrama de estados de la clase Buscar.java.

3.2.2 Procesamiento del comando.

La clase auxiliar Base.java es la encargada de procesar el comando y obtener la salida por pantalla. Su método *terminal(String pwd, String comando, HttpSession sesion)* recibe el directorio de trabajo actual, el comando a procesar y la sesión, la cual necesitamos para guardar la salida y que el cliente tenga acceso a ella desde la vista en modo consola.

En la Figura 27 se muestran los métodos que contiene la clase auxiliar Base.java.

Base.java
int validarLdap(String login, String password)
int validarAuth(String dn,String password)
String getpwd(HttpSession sesion)
boolean esCd(String pwd, String comando, HttpSession sesion)
terminal(String pwd, String comando, HttpSession sesion)
String noReturn(String s)
Process cambiamosPwd(String c, String pwd)
Process comando(String c, String pwd)
BufferedReader salidaComando(Process p)
BufferedReader errorComando(Process p)
boolean validaCD(String comando)

Figura 27. Clase *Base.java*

3.2.3 Comando "*cd*".

El comando "*cd*" es en UNIX el encargado de cambiar de directorio. Debe ser tratado de forma diferente, ya que ha dado problemas a la hora de ser llamado desde la clase java en Bash.

Las distintas posibilidades que tiene el usuario de utilizar correctamente el comando "*cd*" son las siguientes:

- **Caso A.** Utilizarlo de manera única, es decir, sin añadir ningún comando más con "&&". Ejemplos posibles: "*cd home/teleco*", "*cd ..*", "*cd /home*".
  Excepción del Caso A: La utilización de "~" dentro del comando. Un ejemplo de un comando incorrecto en la aplicación sería "*cd ~directorio*", el cual equivaldría a "*cd ../directorio*".

- **Caso B.** Utilizarlo de manera conjunta, es decir, unidos mediante "*&&*". El resultado será el esperado pero no nos moveremos del directorio en el cual nos encontrábamos antes de realizar el comando.
  Ejemplo posible: "*cd teleco && ls*". En el ejemplo nos moveremos al directorio *teleco*, mostraremos sus ficheros, pero no nos habremos movido del directorio en el que nos encontrábamos al principio.

Conclusión.

La única manera de moverse de directorio, desde el punto de vista del cliente, es utilizar el comando *cd* de manera única.

Esto es posible porque mediante un método situado en Base.java, emulamos el comando "*cd*", cuando este llega, cambiando de directorio y guardándolo en el pwd si este existe.

CAPÍTULO 4. Resultados y Evaluación.

Una vez diseñada, se muestran ciertos aspectos de la aplicación ya implementada, para dar una idea de cómo funciona, su estructura y los diferentes componentes que la forman.

4.1 Introducción.

La práctica sobre la cual se ha desarrollado este proyecto es **"El emulador de Maemo"** de la asignatura Arquitectura de Sistemas [12].

Con la realización de dicha práctica el alumno aprende a:

- Compilar una aplicación con un compilador de un entorno de desarrollo cruzado.
- Ejecutar dicha aplicación en el emulador.
- Copiar permanentemente la aplicación dentro de un entorno de ejecución simulado (en su imagen), de tal manera que permanezca en el emulador hasta que se borre.
- Instalar dicha aplicación utilizando el gestor de aplicaciones, mediante la construcción de un paquete Debian.

4.2 Objetivo de la actividad.

Es importante diferenciar los dos tipos de objetivos, uno es el objetivo principal de la práctica y otro el objetivo principal del proyecto.

- El objetivo principal de la práctica de Arquitectura de Sistemas es que los alumnos creen utilizando herramientas sencillas un paquete que se puede instalar en el N810.
- El objetivo del proyecto es que el alumno pueda realizar la práctica, a través de una aplicación web.

4.3 Plan de trabajo vía web.

4.3.1 Introducción

Una vez autenticados en la aplicación los alumnos, como se ha dicho anteriormente, deben realizar la práctica de la asignatura vía web.

A continuación, se muestra cómo realizarla y las distintas especificaciones de cada paso.

4.3.2 Edición en el universo huésped.

La aplicación que va a realizarse es un *Holamundo* y se almacena en un directorio llamado *project*.

1. Crear la carpeta *project* con *mkdir* en la cuenta del usuario *teleco*. (Ver Figura 28).

Figura 28. Realizar la práctica vía web. Paso 1.

(En caso de que ya exista la carpeta este comando muestra un error por pantalla).

2. El siguiente paso es moverse al directorio de la carpeta recién creada. (Ver Figura 29).

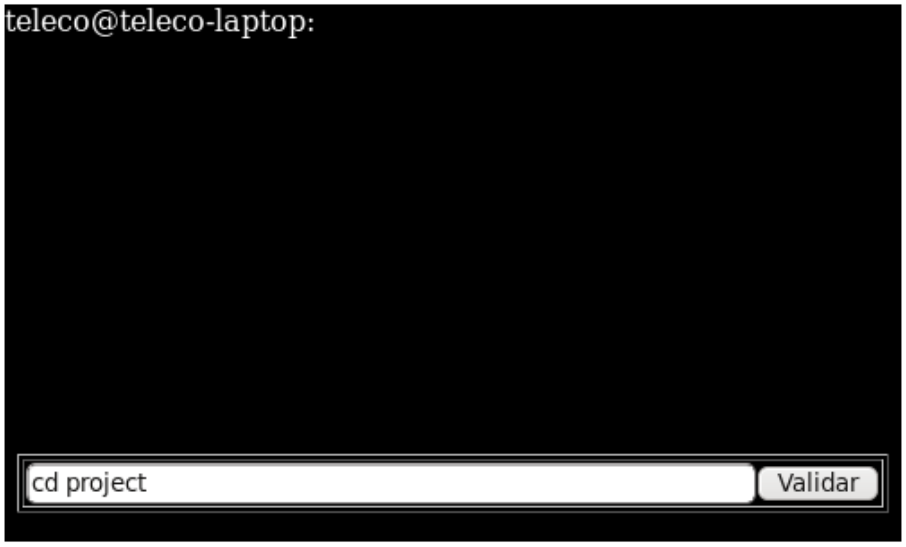


Figura 29. Realizar la práctica vía web. Paso 2.

NOTA: Para moverse desde la aplicación web a un directorio es necesario introducir únicamente el comando *cd* en el terminal (sin ningún otro comando).

Para comprobar que estamos dentro de la carpeta *project* se mira el pwd. (Ver Figura 30).

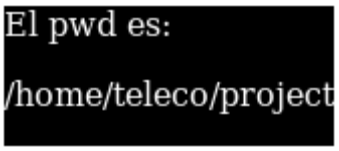


Figura 30. Pwd en el directorio project.

3. Crear el fichero *helloworld* (con *kate* u otro editor) para que muestre por pantalla el mensaje: “Hola Mundo! Soy …”, y el nombre del alumno. (Ver Figura 31).

```c
#include <stdio.h>

main(){

    printf("Hola Mundo! Soy César\n");

}
```

Figura 31. Ejemplo *helloworld.c*

4.3.3 Compilación en el universo de compilación cruzada.

Para crear una aplicación ejecutable para el dispositivo Nokia a partir del código es sencillo, tan solo hay que hacer que el comando de compilación *gcc* se invoque desde el entorno o universo de compilación cruzada del emulador **ScratchBox** mediante el comando *sb2 gcc*.

1. Se compila con *sb2 gcc* en el universo de compilación cruzada del emulador ScratchBox. (Ver Figura 32).

Figura 32. Realizar la práctica vía web. Paso 3.

NOTA: Se puede introducir más de un comando a la vez, utilizando && entre los distintos comandos. El comando ls nos permite ver que realmente se ha creado el ejecutable helloworld.

En el universo de compilación cruzada hay herramientas especiales (como el compilador para la plataforma ARM que corre en el NOKIA) que permiten compilar la aplicación.

2. Es importante fijarse en que el ejecutable generado es para un *ARM* y no para la *arquitectura I386* (la de la máquina virtual o los PCs del laboratorio). (Ver Figura 33).

Figura 33. Realizar la práctica vía web. Paso 4.

NOTA: El comando *file* indica el tipo de fichero que se ha generado. En este caso es un ejecutable para ARM).

En general para ejecutar un comando en los universos del emulador, solo hay que preceder el comando elegido por la orden sb2 si es para el universo de compilación cruzada, o sb2 –e si es para el universo de ejecución simulada. Por ejemplo para hacer ls en el entorno de compilación ejecutaríamos sb2 ls.

4.3.4 Universo de ejecución: Entorno simulado de ejecución.

1. Para probar la aplicación en el entorno de ejecución simulado del ScratchBox, puede hacerse con el comando sb2 -e ./helloworld. (Ver Figura 34).

Figura 34. Realizar la práctica vía web. Paso 5.

El directorio donde estamos trabajando *(project)* se comparte entre el universo del huésped y los universos del emulador ScratchBox, por lo que no es necesario copiar nada al emulador para poder ejecutarlo.

2. Cabe destacar que si se ejecutara ese archivo desde el universo huésped, el código ya no funciona, pues está compilado para la arquitectura de un ARM. (Ver Figura 35).

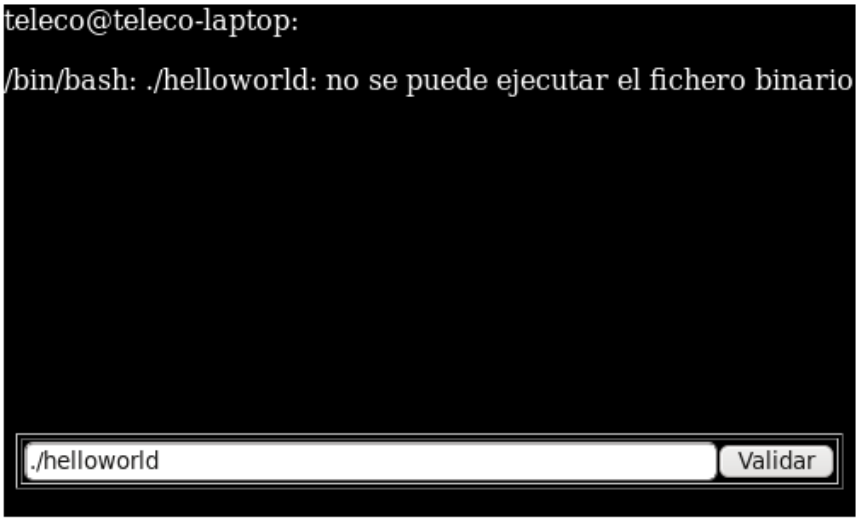


Figura 35. Realizar la práctica vía web. Paso 6.

3. La aplicación también funciona en el entorno o universo de compilación cruzada. Esto es así porque este entorno corre también sobre un ARM. (Ver Figura 36).

Figura 36. Realizar la práctica vía web. Paso 7.

Para copiar la aplicación en el directorio de programas del emulador *(/usr/bin)* para que se quede permanentemente, los pasos a seguir son los siguientes:

1. Antes de nada se comprueba que no existe el fichero en el directorio */usr/bin* del emulador con el comando *ls*, ejecutándolo en el universo de ejecución. (Ver Figura 37).

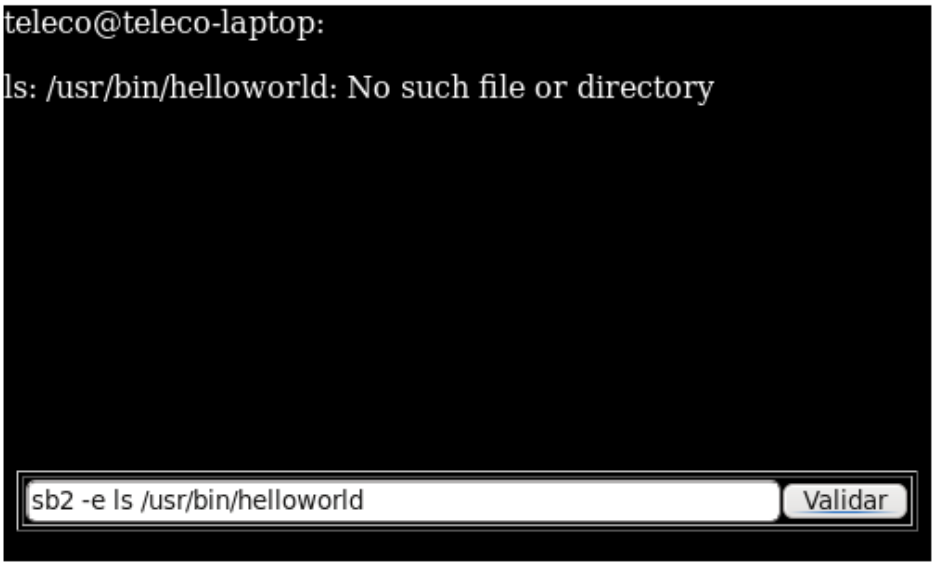


Figura 37. Realizar la práctica vía web. Paso 8.

2. Después, se copia el ejecutable desde el directorio *project* a */usr/bin* con el comando *cp*. El comando debe ir precedido de *sb2 -eR* porque se quiere trabajar dentro del entorno de ejecución. (Ver Figura 38).

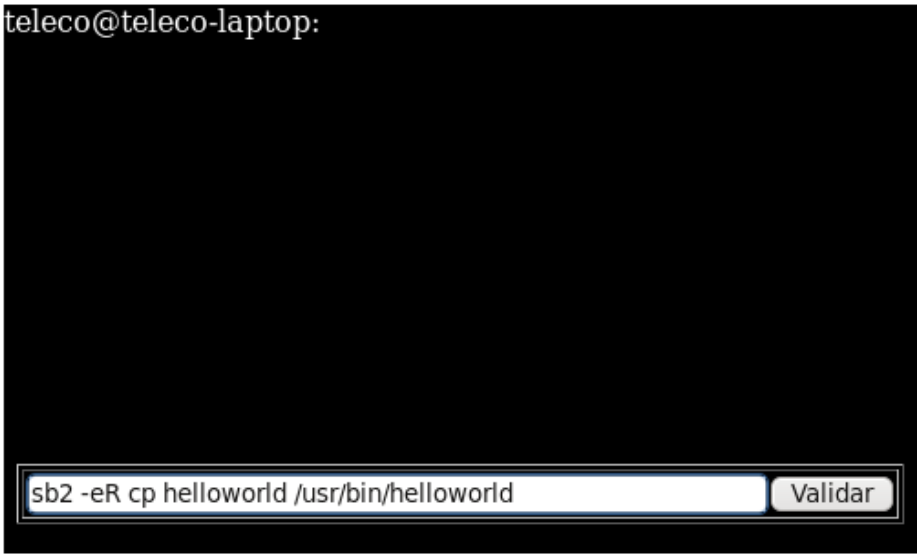


Figura 38. Realizar la práctica vía web. Paso 9.

NOTA: La opción R ejecuta el comando como *root* (esto es, con privilegios máximos). Esta opción es necesaria en este caso para poder copiar el fichero en el directorio */usr/bin* en el cual sólo puede crear ficheros el *root*.

3. Ejecutando el programa directamente en el universo de ejecución del ScratchBox, ya no es necesaria la opción *-R*. Esto es así porque no hacen falta privilegios de *root* para ejecutar una aplicación. (Ver Figura 39).

Figura 39. Realizar la práctica vía web. Paso 10.

4. Ahora el fichero se encuentra instalado en el directorio de programas del entorno de ejecución simulada. (Ver Figura 40).

Figura 40. Realizar la práctica vía web. Paso 11.

5. Los archivos se comparten entre los distintos universos del emulador, así que también se podría comprobar que el programa está instalado en el entorno de compilación cruzada, aunque no es necesario.

6. Si intentara ejecutar en el universo huésped (sin sb2) el comando falla y ni siquiera encuentra el ejecutable. (Ver Figura 41).

Figura 41. Realizar la práctica vía web. Paso 12.

7. Para borrar el fichero una vez instalado se puede realizar esta operación con "*rm*" (precedido claro está del comando *sb2 -eR*). (Ver Figura 42).

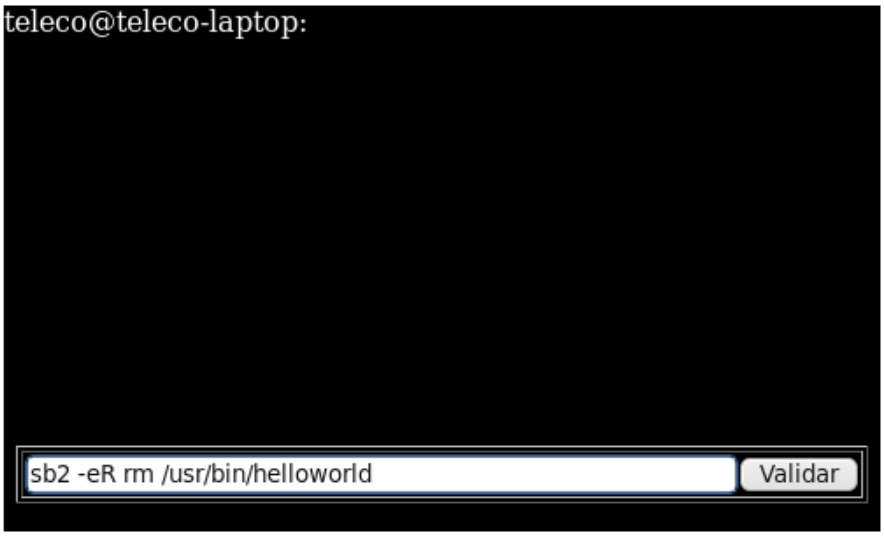


Figura 42. Realizar la práctica vía web. Paso 13.

Una vez más se necesitan privilegios de *root* para poder realizar la ejecución.

Existe la posibilidad de que se deseara instalar la aplicación (no copiándola directamente en el directorio */usr/bin*) haciendo uso de un paquete.

La ventaja que tiene es que para gestionarlos se puede utilizar el gestor de aplicaciones.
Para ello haciendo uso de alien se puede empaquetar el código que ya tenemos en un paquete llamado "helloworld".

1. Primero se crea una jerarquía de directorios similar a la del entorno final. Esto es, con subdirectorios idénticos a los que tiene el universo de ejecución del dispositivo (o del simulador), en este caso "usr/" y "usr/bin". (Ver Figura 43).

Figura 43. Realizar la práctica vía web. Paso 14.

NOTA: El punto y coma también nos permite introducir dos comandos (uno tras otro) en un mismo paso (al igual que &&).

2. Se copia el fichero *helloworld* en esta jerarquía de ficheros. (Ver Figura 44).

Figura 44. Realizar la práctica vía web. Paso 15.

3. Se comprime todo en un fichero "*.tar.gz*" marcando el número de la versión del paquete. (Ver Figura 45).

Figura 45. Realizar la práctica vía web. Paso 16.

Varios detalles:

- Las opciones de *tar* para comprimir son *cvfz*.
- El nombre del paquete ha de coincidir con el nombre dado al fichero "*tar*" que se ha creado.
- La versión del paquete también aparece reflejada en el nombre del fichero (*helloworld-1.0.0.tar.gz*).

4. Una vez comprimido, se puede utilizar la herramienta "*alien*" para pasar de "*.tar.gz*" a "*.deb*". Esta herramienta consigue pasar un fichero "*.tar.gz*" a un paquete Debian sencillo (no comprueba las dependencias, pero permite que sea reconocido por el gestor de paquetes). (Ver Figura 46).

Figura 46. Realizar la práctica vía web. Paso 17.

NOTA: Si se intentara ejecutar *alien* sin utilizar *fakeroot* nos dirá que necesitamos ser root para generar el paquete. La forma de solucionarlo es llamar al comando utilizando *fakeroot*, un comando que permite simular que se poseen privilegios de root.

5. El resultado final queda disponible en el directorio actual. (Ver Figura 47).

Figura 47. Realizar la práctica vía web. Paso 18.

Instalar y desinstalar el paquete "helloworld" es sencillo. Para ello se utilizan las herramientas "apt-get" y "dpkg" en el universo de ejecución del emulador.

Previo a la instalación del paquete, hay que comprobar que la aplicación no está instalada haciendo uso de ls.

A continuación, para instalar se utiliza dpkg -i nombre-del-paquete.deb precedido de sb2 -eR. (Ver Figura 48).

Figura 48. Realizar la práctica vía web. Paso 19.

Tras instalar, ya se puede invocar helloworld en el entorno de ejecución simulado. (Ver Figura 49).

Figura 49. Realizar la práctica vía web. Paso 20.

Se puede borrar el paquete llamando al comando apt-get remove en el universo de ejecución en modo root con el comando “sb2 –eR apt-get remove helloworld”.

NOTA: Al introducir el comando anterior vía web el tiempo de respuesta en la aplicación supera los 4 minutos, por lo que se desaconseja totalmente utilizar el borrado de esta manera.

4.4 Tiempos de Carga

El tiempo de validación de la aplicación depende de los comandos utilizados. Es necesario comprobar los tiempos de los distintos comandos, para estudiar la viabilidad de la aplicación en esta práctica.

A continuación en las tablas 1 y 2, se pueden observar los distintos comandos de la práctica, el tiempo real, de usuario y de sistema, comprobados introduciendo el comando "time" antes de cada uno de ellos:

Comando	Tiempo real	Tiempo de usuario	Tiempo de sistema
mkdir project	0.052s	0.004s	0.012s
sb2 gcc helloworld.c –o helloworld	3.373s	0.596s	1.468s
ls	0.012s	0.004s	0.012s
file helloworld	0.075s	0.004s	0.012s
sb2 –e ./helloworld	1.747s	0.388s	1.052s
./helloworld	0.129s	0.004s	0.004s
sb2 ./helloworld	1.753s	0.532s	1.096s
sb2 –e ls /usr/bin/helloworld	1.715s	0.452s	1.040s
sb2 -eR cp ./helloworld /usr/bin/helloworld	1.893s	0.396s	1.240s
sb2 –e /usr/bin/helloworld	1.615s	0.488s	0.988s

Tabla 1. Tiempos de carga 1/2.

Comando	Tiempo real	Tiempo de usuario	Tiempo de sistema
sb2 –e ls /usr/bin/helloworld	2.015s	0.412s	1.108s
/usr/bin/helloworld	0.005s	0.008s	0.000s
sb2 –eR rm /usr/bin/helloworld	1.739s	0.416s	1.136s
mkdir usr && mkdir usr/bin	0.013s	0.004s	0.012s
cp helloworld usr/bin/	0.027s	0.004s	0.012s
tar cvfz helloworld-1.0.0.tar.gz usr/	0.065s	0.004s	0.028s
fakeroot alien helloworld-1.0.0.tar.gz	6.017s	1.400s	2.660s
sb2 –eR dpkg –i ./helloworld_1.0.0-2_all.deb	10.299s	1.756s	2.336s
sb2 –e /usr/bin/helloworld	1.696s	0.460s	1.044s
sb2 –eR apt-get remove helloworld	>4minutos (Comando no viable vía web)		

Tabla 2. Tiempos de carga 2/2.

Capítulo 5. Planificación del trabajo y presupuesto.

En este capítulo se analizará la planificación del trabajo desarrollado y el presupuesto del que se requeriría en unos escenarios de pruebas.

5.1 Planificación del trabajo.

En los siguientes apartados, se va a desglosar el proyecto en las distintas fases por las que ha pasado, las horas que se han empleado en cada actividad y un diagrama de Gantt.

5.1.1 Desglose por fases del proyecto.

A continuación, se especifican las actividades que se han llevado a cabo para la realización del proyecto, incluyendo una pequeña explicación sobre lo que ha sido necesario para desarrollar cada una de ellas.

1. **Lectura de documentación sobre VirtualBox.**

Preparación para la utilización de Oracle VirtualBox leyendo la guía de usuario http://download.virtualbox.org/virtualbox/UserManual.pdf.

2. **Lectura de documentación sobre Maemo SDK.**

Preparación para la realización de la práctica de la asignatura Arquitectura de Sistemas leyendo la guía de usuario de Maemo SDK http://maemo-sdk.garage.maemo.org/user-guide.html.

3. **Instalación de Apache Tomcat dentro de la máquina virtual.**

Instalación del servidor web Apache Tomcat dentro de la máquina virtual. Preparar el entorno de trabajo no fue trivial. En las máquinas del laboratorio siempre se ha trabajado para preparar el entorno con los comandos “setenv” pero dentro de la máquina virtual dichos comandos no son aceptados, por tanto hubo que encontrar otra forma y los comandos “export” fueron la solución para preparar el entorno.

4. **Diseño de la aplicación.**

En un principio se diseñó un boceto con una serie de puntos a ir completando a medida que avanzaba el proyecto. Una vez diseñada la aplicación se comenzó a implementar código para la realización del proyecto.

5. **Familiarización con la plataforma de desarrollo.**

Familiarizarse desde cero con la plataforma de desarrollo como trabajo previo a empezar a escribir código.

6. **Implementación de autenticación MySQL.**

Diseño e implementación de un servlet portada y un servlet autenticación, al que se envía un formulario y este comprueba si está en la base de datos MySQL.

7. **Implementación de un escenario de comunicaciones sencillo entre cliente/servidor.**

Una vez autenticado en la aplicación, la siguiente parte fue mediante un formulario desde la página de portada, enviarlo a un servlet recepción y que este únicamente redirigiera a la página de portada. Así se comprueba la correcta comunicación entre los dos servlets.

8. **Emulación de un terminal mediante hojas de estilo (CSS).**

Mediante las hojas de estilo emular una consola de comandos negra, desde la cual se envia el comando que deberá ser procesado por el servidor.

9. **Implementación de métodos en el servidor para el procesamiento de los comandos.**

Se implementan los métodos que necesitará el servlet receptor del comando para poder enviar la respuesta al cliente y que este la imprima por pantalla.

10. Problemas con el comando cd (especial procesamiento).

Detectando la imposibilidad de la aplicación de procesar el comando cd, implementamos métodos para detectar que el comando que viene es cd y procesarlo adecuadamente cambiando el pwd en el cual se encontrará la aplicación.

11. Implementación del pwd.

Dada la necesidad de que el usuario vea en el directorio en el que se encuentra, se implementa el pwd. En portada se trata como un “div” aparte y mediante las hojas de estilo se pone del mismo color que la consola de comandos.

12. Subida de archivos (FileUpload).

Para que el usuario pueda subir sus propios ficheros a la aplicación se implementa un servlet que deje sus archivos en el pwd en el cual se encuentre. Todo esto con la ayuda de la librería FileUpload, sin la cual no se podría realizar la subida de los ficheros.

13. Implementación de autenticación LDAP.

Al ser la aplicación para los alumnos de la universidad se decidió cambiar el modo de autenticación de la base de datos MySQL, para implantar autenticación LDAP con la base de datos de la universidad. En la cual los alumnos pueden meter sus NIAs y contraseñas que tienen asignados para Campus Global y podrán autenticarse en la aplicación.

14. Pruebas.

Una vez todo funcionando, se prosiguió con la realización de pruebas para comprobar los tiempos de carga de cada uno de los comandos, la validación o no de ciertos comandos y la facilidad de importación a otra máquina virtual.

15. Depuración de código.

Tras acabar todos los servlets, se continuó creando una clase auxiliar Base.java para que contuviera todos los métodos necesarios de forma ordenada y agrupada.

16. **Diseño final de la apariencia de la web.**

Un último paso antes de la documentación fue el diseño de la aplicación de forma definitiva, creando un logo, ordenando los distintos “div” de la portada y modificando los tamaños para que tuviera un diseño satisfactorio.

Además se han añadido funciones JavaScript para el correcto funcionamiento de la web, tales como comprobar si el fichero subido tiene la extensión correcta y sino saltar una excepción, comprobar si envía un comando vacío o en la autenticación se ha dejado algún campo vacío del formulario, etc.

17. **Documentación.**

Por último la realización de la memoria. Aunque esté situada en último lugar en todo momento en el proyecto ha habido una documentación previa y se han estado apuntando todos los datos necesarios para la memoria final.

5.1.2 Desglose de las actividades en horas.

En la siguiente tabla se especifica el tiempo gastado en la suma de actividades y para ser más genéricos se considerará:

- **Lectura de documentación:** Como lectura de documentación sobre VirtualBox y lectura de documentación sobre Maemo SDK.
- **Instalación y preparación del entorno:** Como instalación de Apache Tomcat dentro de la máquina virtual.
- **Familiarización con la apliación:** Como familiarización con la plataforma de desarrollo.
- **Análisis y diseño:** Como diseño de la aplicación y diseño final de la web.
- **Implementación:** Como implementación de autenticación MySQL, implementación de un escenario sencillo, emulación del terminal, implementación de métodos en el servidor, problemas con el comando “cd”, implementación del pwd, subida de archivos y autenticación LDAP.
- **Pruebas:** Como pruebas.
- **Depuración:** Como depuración de código.
- **Documentación:** Como realización de la memoria.

El tiempo total empleado ha sido 720 horas aproximadamente.

Actividades	Horas
Lectura de documentación.	40 horas
Instalación y preparación del entorno.	50 horas
Familiarización con la aplicación.	20 horas
Análisis y diseño.	80 horas
Implementación.	360 horas
Pruebas.	80 horas
Depuración.	10 horas
Documentación.	80 horas

Tabla 3. Tiempos de cada actividad.

5.1.3 Diagrama de Gantt

Un diagrama de Gantt sitúa de forma secuencial las diferentes fases del proyecto a lo largo de una línea temporal que representa el tiempo total del proyecto.

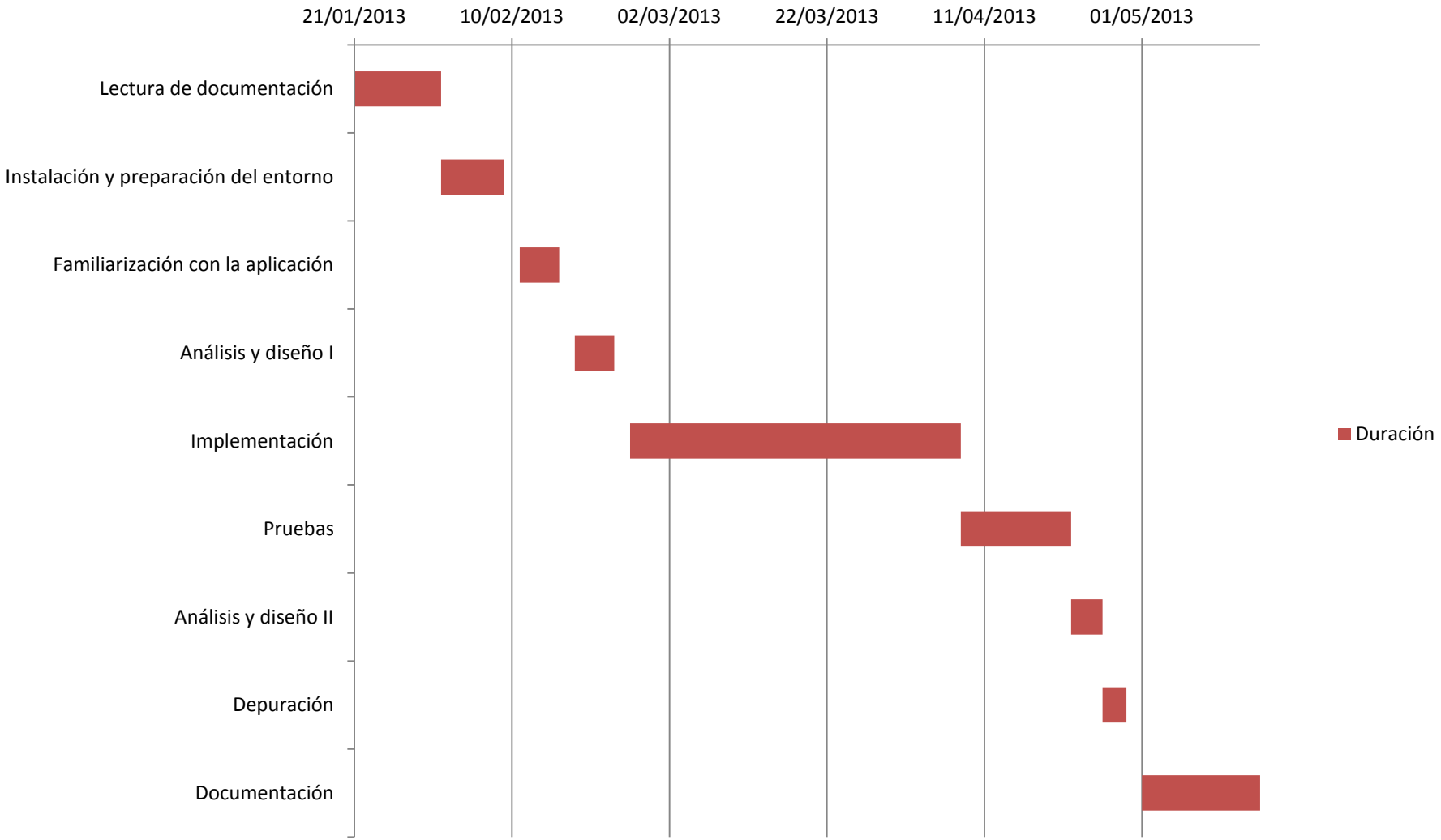


Tabla 4. Diagrama de Gantt.

5.2 Presupuesto.

5.2.1 Gastos de trabajador imputable al proyecto.

Para la realización del proyecto se contrata a tres personas encargadas cada una de un ámbito: un analista, un programador y una persona encargada de la documentación. El sueldo medio por hora de los tres sería 20€/hora.

Con estos datos se va a realizar una tabla para ver el gasto que supondría la contratación de esos tres trabajadores para realizar el proyecto.

Cargo	Horas	Gasto
Analista • Lectura de documentación. • Análisis y diseño. • Pruebas.	**140 h.** 40 h. 80 h. 20 h.	**2.800 €** 800 € 1.600 € 400 €
Programador • Instalación y preparación del entorno. • Familiarización con la aplicación. • Implementación. • Pruebas. • Depuración.	**480 h.** 50 h. 20 h. 360 h. 60 h. 10 h.	**10.000 €** 1.000 € 400 € 7.200 € 1.200 € 200 €
Documentación • Documentación.	**80 h.** 80 h.	**1.600 €** 1.600 €
TOTAL	**700 h.**	**14.400 €**

Tabla 5. Presupuesto asociado al proyecto.

Entre los tres trabajadores habrán consumido 720 horas de trabajo para realizar el proyecto, por lo que a una media de 20 €/hora, habría costado a la empresa encargada 14.400 € de gasto.

5.2.2 Recursos materiales.

Para la realización del proyecto se han utilizado los siguientes recursos materiales por persona, que en el hipotético caso de contratar a tres trabajadores sería la adquisición de tres unidades de cada recurso material.

Recurso	Unidades	Coste
Portatil emachines e725	3	1.200 €
Paquete Office 365	3	300 €
Licencia Windows 7	3	261 €
TOTAL		**1.761 €**

Tabla 6. Gastos materiales del proyecto.

El resto del software utilizado es software libre (freeware).

- El entorno de desarrollo Eclipse, para la realización de los servlets y las clases auxiliares.
- El servidor web Apache Tomcat, como contenedor de servlets y JSPs.
- La máquina virtual VirtualBox y el sistema operativo instalado en ella (Ubuntu).
- Las distintas librerías utilizadas, como FileUpload para la subida de archivos.

En caso de implementar esta solución en todos los ordenadores de las aulas Telemáticas de la Universidad Carlos III, habría que contar con más elementos a incluir en el presupuesto como, un servidor donde cada alumno tuviera su imagen y un programador que instale la aplicación en todos los ordenadores del laboratorio.

5.2.3 Resumen.

La siguiente tabla resume todos los costes de los que ha requerido el proyecto:

Recurso	Coste
Gastos de personal	14.400 €
Recursos materiales	1.761 €
TOTAL	**16.161 €**

Tabla 7. Presupuesto total.

El coste total habría sido de **16.161 €** de haberlo realizado en un entorno empresarial, con tres trabajadores a cargo de la realización del proyecto.

Capítulo 6. Conclusiones y líneas futuras de trabajo.

En este último capítulo del documento, tras el desarrollo de todo el trabajo previo, se va a llevar a cabo un análisis de los resultados obtenidos para evaluar la aplicación y definir posibles líneas de trabajo futuro.

6.1 Conclusiones.

El objetivo del proyecto era la creación de una aplicación web para la asignatura de Arquitectura de Sistemas. Para ello se trabajó con el fin de conseguir una maqueta que cumpliera las necesidades básicas principales de la práctica “El emulador de Maemo”.

En este aspecto, se ha conseguido la maqueta de una aplicación funcional y real con muchas posibilidades en el campo de la docencia.

El desarrollo del trabajo ha sido similar al planteamiento inicial del proyecto aunque en ocasiones se han tenido que estudiar otras soluciones ante problemas imprevistos, tales como, descartar la viabilidad de un terminal web, descartar el uso de la base de datos MySQL y los problemas del comando *cd* en Bash.

Finalmente y teniendo en cuenta los resultados de las pruebas realizadas, se ha llegado a la conclusión de que la aplicación es viable y que se adapta completamente al entorno docente para la cual ha sido ideada.

6.2 Líneas de trabajo futuro.

Como se ha comentado a lo largo de este documento, esta aplicación es una maqueta con la finalidad de que en algún momento pueda ser implantada en un escenario real.

Bien es cierto, que existen una serie de ampliaciones y posibilidades que no están en la versión final de la aplicación pero que, en un futuro, sería muy interesante poder estudiar incorporarlas:

- **Dar a la aplicación la función multiusuario.**
- **Posibilidad de subir la aplicación a la nube.**
- **Exportar el cliente fuera de la máquina virtual.**

- **Incorporar tecnología AJAX en el lado del cliente (*client-side*).**

Estas son las cuatro principales mejoras por las que sería interesante continuar un estudio acerca de la posibilidad de incorporarlas.

Capítulo 7. Anexos.

ANEXO I: Manual de configuración de Tomcat en la VM.

1. **Instalar el JDK.**

En este caso ya está instalada la versión 1.6.0_37 del jdk y está colocada en */usr/dist/*.

En el caso de no tenerlo instalado, se puede descargar la última versión en el siguiente enlace:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. **Instalar Apache Tomcat.**

Como en el caso anterior, ya está instalada la versión 7.0.25 de Apache Tomcat y está colocado en el directorio */home/teleco/.* El directorio donde esté colocado es importante a la hora de la asignación de variables.

Descargar la última versión de Apache Tomcat del siguiente enlace y tener la aplicación de “maemo” dentro del directorio webapps.

http://tomcat.apache.org/download-70.cgi

NOTA: Se recomienda seguir el mismo orden de directorios, para asegurar el correcto funcionamiento de la aplicación.

3. Asignación de variables.

Se asignan valores a las variables *CATALINA_HOME*, *JAVA_HOME* y *PATH*. Dependiendo del intérprete *setenv* ó *export*.

```
setenv CATALINA_HOME /home/teleco/apache-tomcat-7.0.25
setenv JAVA_HOME /usr/dist/jdk1.6.0_37
setenv PATH ${JAVA_HOME}/bin:${PATH}
```

```
export CATALINA_HOME=/home/teleco/apache-tomcat-7.0.25
export JAVA_HOME=/usr/local/jdk1.6.0_37
export PATH=${JAVA_HOME}/bin:${PATH}
```

NOTA: Si se quisiera hacer alguna modificación a la aplicación, únicamente habría que compilar. Para compilar habría que introducir el siguiente comando en el directorio dónde estén situados los archivos **.java*.

```
javac –classpath ${CATALINA_HOME}/lib/servlet-
api.jar:/home/teleco/apache-tomcat-
7.0.25/webapps/maemo/WEB-INF/lib/commons-fileupload-
1.2.2.jar:. -d ../../classes *.java
```

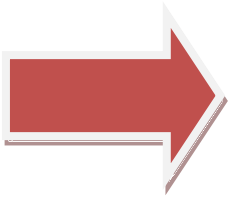
4. **“Encender” Tomcat.**

Lanzamos Tomcat entrando en el directorio *bin* dentro de *apache-tomcat* e introducimos el siguiente comando.

./startup.sh

5. **Acceder a la portada.**

Para acceder a la portada abrir el navegador y sabiendo que la aplicación trabaja en el puerto 9090, introducir la siguiente URL en la barra de direcciones.



localhost:9090/maemo/portada

¡¡FELICIDADES!!

Ya has configurado correctamente Tomcat para el uso de la aplicación.

ANEXO II: Glosario de términos.

- **Máquina virtual.**

Es un software que simula a una computadora y puede ejecutar programas como si fuese una computadora real.

- **Handheld.**

Es un anglicismo que significa en castellano “palmar” y describe a una computadora portátil que se puede llevar en una mano a cualquier parte mientras se utiliza.

- **VirtualBox.**

Es un software de virtualización, actualmente desarrollado por Oracle.

- **Oracle.**

Una de las mayores compañías de software del mundo.

- **Servlet.**

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

- **JSP (Java Server Pages).**

Los JSPs son otra forma de crear servlets ya que el código JSP se traduce a código servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta.

- **Tomcat.**

Funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta.

- **Maemo.**

Es una plataforma de desarrollo para dispositivos handheld basado en Debian GNU/Linux.

- **Debian GNU/Linux.**

Es un sistema operativo libre, desarrollado por más de mil voluntarios alrededor del mundo, que colaboran a través de internet.

- **Ubuntu.**

Es un sistema operativo mantenido por unas reglas y la comunidad informática. Utiliza un núcleo Linux y su origen está basado en Debian.

- **C/C++, Java, Phyton.**

Distintos lenguajes de programación.

- **Hypervisor.**

El hipervisor o monitor de máquina virtual es una plataforma que permite aplicar diversas técnicas de control de virtualización para utilizar al mismo tiempo, diferentes sistemas operativos en una misma computadora.

- **VMware.**

Es una filial de EMC Corporation que proporciona software de virtualización disponible para ordenadores compatibles x86.

- **x86.**

Es un conjunto de instrucciones utilizada en la microarquitectura de CPU, siendo también una denominación genérica dada a ciertos microprocesadores.

- **Mac OS X.**

Es una serie de sistemas operativos basados en Unix desarrollado, comercializado y vendido por Apple Inc. que ha sido incluido en su gama de computadoras Macintosh desde 2002.

- **OS/2 Warp.**

Es un sistema operativo de IBM que intentó suceder a DOS como sistema operativo de las computadoras personales.

- **MS-DOS.**

Es un sistema operativo para computadoras basadas en x86.

- **Solaris.**

Es un sistema operativo de tipo Unix desarrollado desde 1992 inicialmente por Sun Microsystems y actualmente por Oracle Corporation.

- **FreeBSD.**

Es un sistema operativo libre basado en las CPU de arquitectura Intel, incluyendo procesadores Intel 80386, Intel 80486 y Pentium.

- **OpenBSD.**

Es un sistema operativo libre tipo Unix multiplataforma, basado en 4.4BSD. Es un descendiente de NetBSD, con un foco especial en la seguridad y criptografía.

- **PXE (Preboot Execution Environment).**

Es un entorno para arrancar e instalar el sistema operativo en ordenadores a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles (como discos duros) o de los sistemas operativos instalados.

- **RDP (Remote Desk Protocol).**

Es un protocolo propietario desarrollado por Microsoft que permite la comunicación en la ejecución de una aplicación entre un terminal y un servidor Windows.

- **JBoss.**

Es un servidor de aplicaciones Java EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java.

- **JOnAS.**

Es un servidor de aplicaciones Java EE de código abierto implementado en Java.

- **CGI (Common Gateway Interface).**

Es una tecnología que permite al cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web.

- **GUI (Graphical User Interface).**

La interfaz gráfica de usuario (GUI) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

- **CSS (Cascading Style Sheets).**

Hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas webs escritas en lenguaje HTML y XHTML, pero también puede ser aplicado a cualquier tipo de documento XML.

- **ECMAScript.**

Define un lenguaje de tipos dinámicos ligeramente inspirado en Java y otros lenguajes del estilo C. Soporta algunas características de la programación orientada a objetos mediante objetos basados en prototipos y pseudoclases.

- **DOM (Document Object Model).**

Es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

- **SSJS o JavaScript.**

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

- **MySQL.**

Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

- **LDAP (Lightweight Directory Access Protocol).**

Es un protocolo a nivel de aplicación que permite el acceso de un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

- **Pwd (Print Working Directory).**

El comando pwd se utiliza para imprimir el nombre del directorio actual en una sesión de comandos bajo un sistema operativo Unix o derivado.

- **Bash.**

Es un programa informático cuya función consiste en interpretar órdenes. Está basado en la Shell de Unix y es compatible con POSIX.

- **Arquitectura i386 y arquitectura ARM.**

Distintas arquitecturas de microprocesadores.

ANEXO III: Importar y exportar imagen en VirtualBox.

Importar imagen.

Para importar una imagen a VirtualBox y poder trabajar con ella hay que seguir una serie de pasos:

- Abrir la pestaña “Archivo” situada en la esquina superior izquierda.

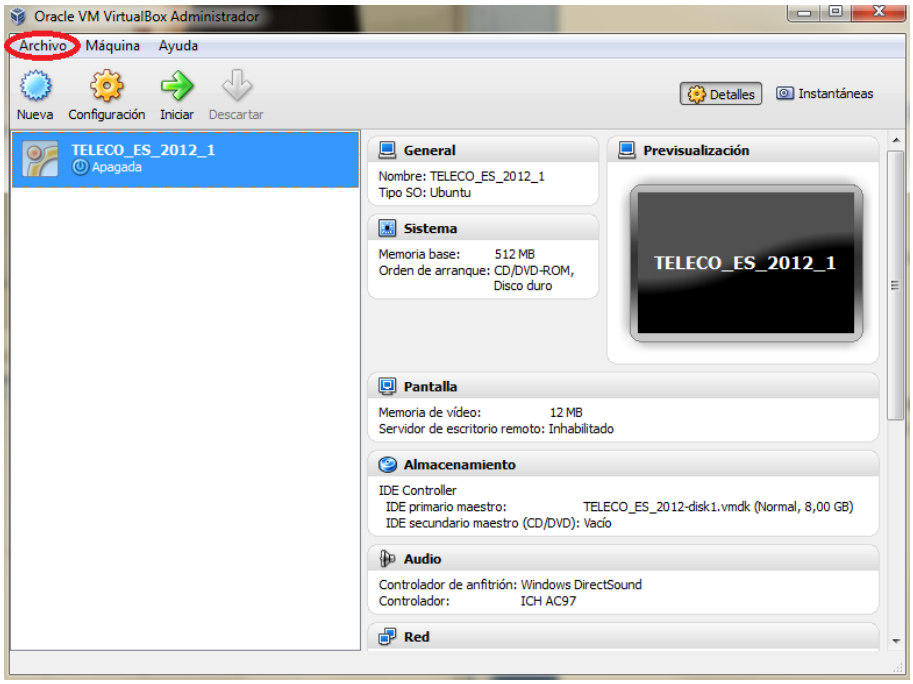


Figura 50. Importar imagen. Paso 1.

- Dentro de “Archivo”, seleccionar la opción “Importar servicio virtualizado…”.

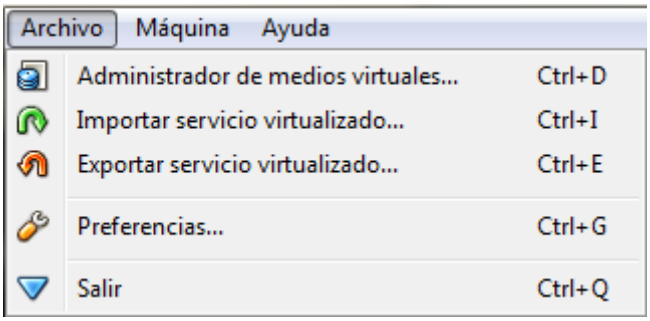


Figura 51. Importar imagen. Paso 2.

- Pulsar la opción “Seleccionar…”, elegir la imagen que se quiera importar y pulsar “Next”.

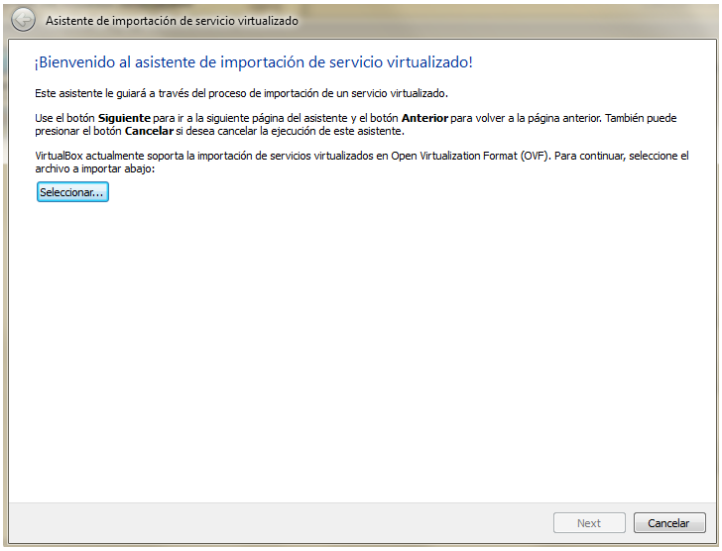


Figura 52. Importar imagen. Paso 3.

- Aparecerá una descripción de la imagen seleccionada, dando la opción de cambiar algunos parámetros antes de la importación e incluso restaurar los valores por defecto. Por último seleccionar “Importar” para tener la imagen disponible en VirtualBox.

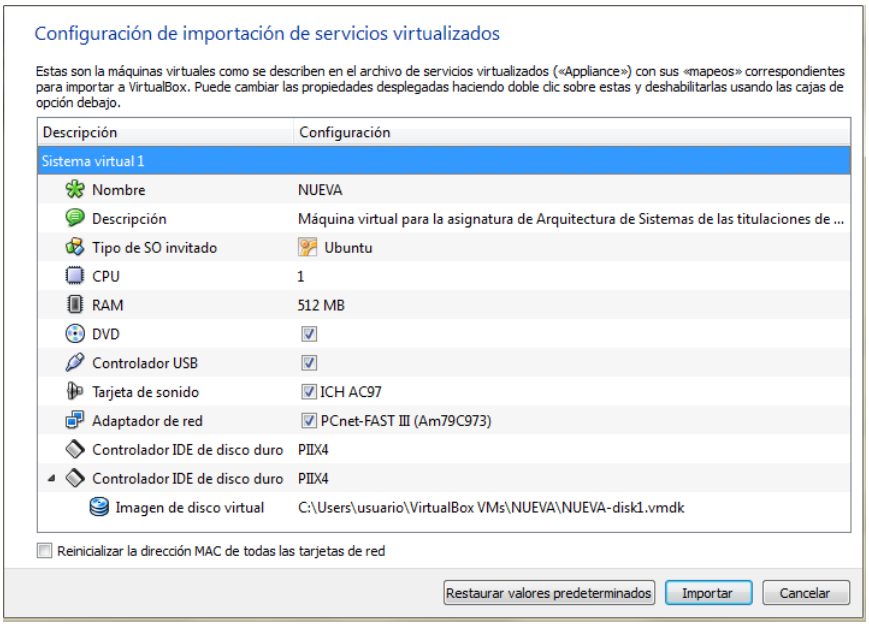


Figura 53. Importar imagen. Paso 4.

Exportar imagen.

Para exportar una imagen que se quiera utilizar en otro ordenador o simplemente para guardar una copia de seguridad de la imagen, hay que seguir una serie de pasos:

- Al igual que para importar, pulsar “Archivo” y se desplegará una pestaña. En este caso, hay que seleccionar la opción “Exportar servicio virtualizado…”.

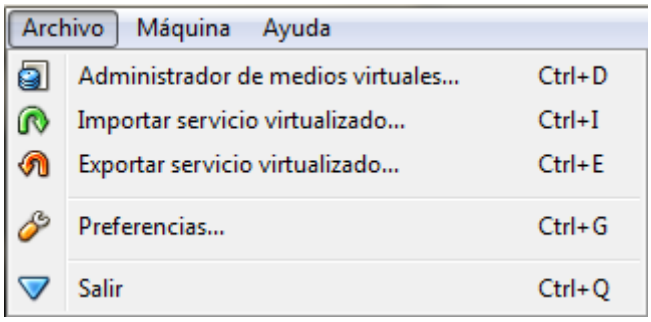


Figura 54. Exportar imagen. Paso 1.

- Aparecerá una lista de las imágenes en las cuales se está trabajando. Elegir la imagen deseada y pulsar “Next”.

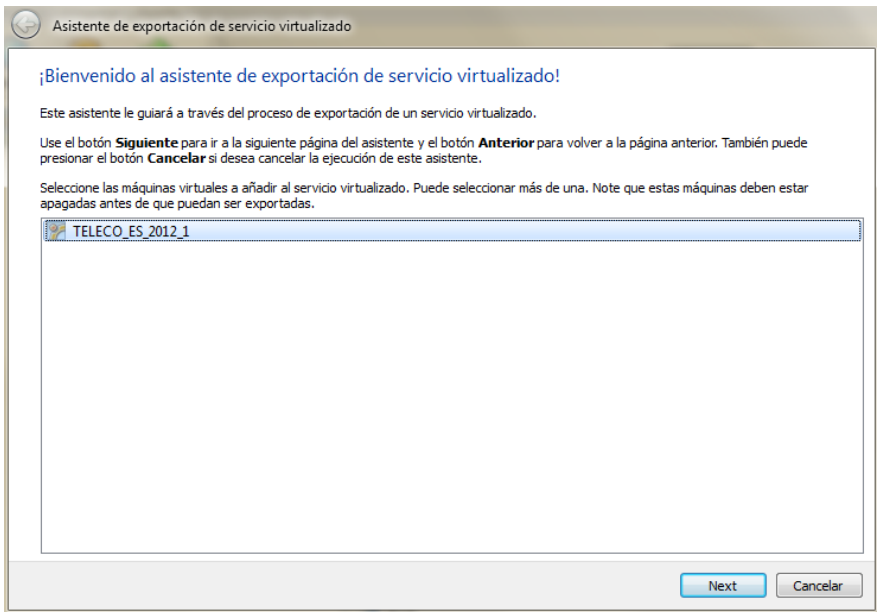


Figura 55. Exportar imagen. Paso 2.

- Se puede elegir el destino donde quedará almacenada la imagen y el tipo de formato deseado *ova* o *ovf*, por último también se puede elegir si guardar el archivo *Manifest*. Seleccionar los parámetros deseados y hacer clic en “Next”.

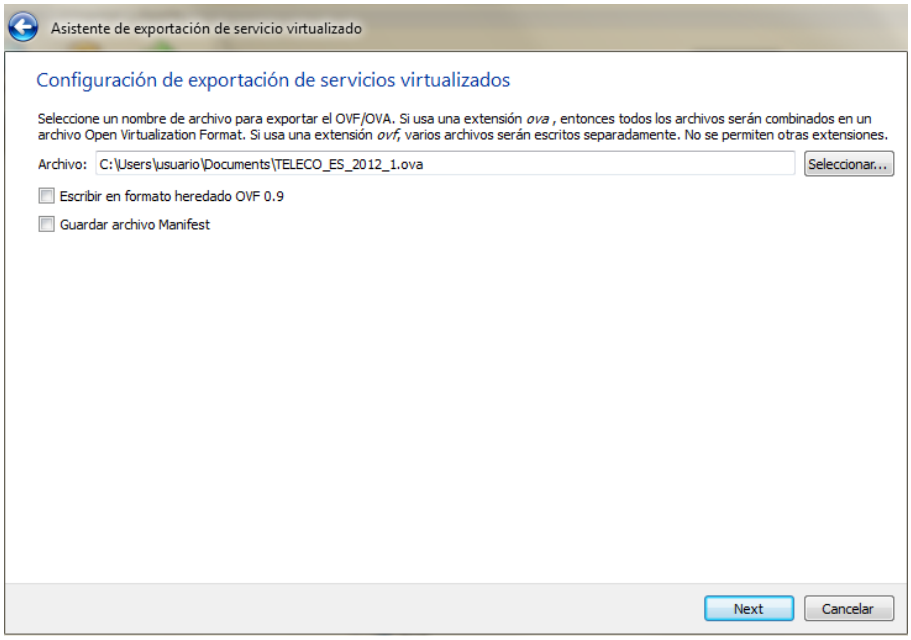


Figura 56. Exportar imagen. Paso 3.

- Por último aquí se pueden cambiar los valores de configuración adicionales para la máquina virtual seleccionada. Puede modificar la mayoría de las propiedades mostradas haciendo doble clic en ellas. Hacer clic en exportar para finalizar con éxito.

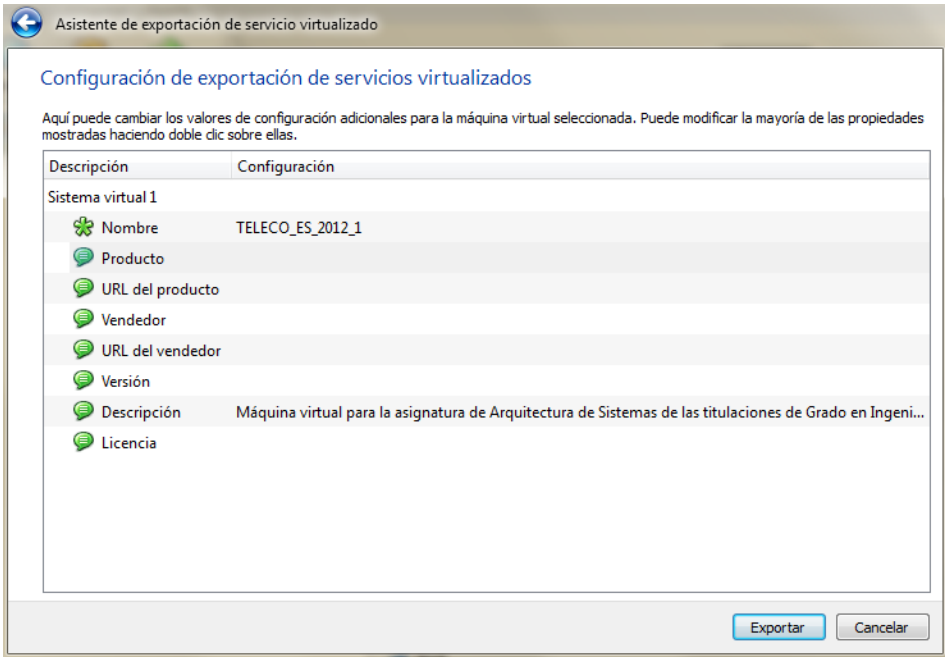


Figura 57. Exportar imagen. Paso 4.

Capítulo 8. Referencias.

[1] Maemo SDK User Guide:

http://maemo-sdk.garage.maemo.org/user-guide.html

[2] Oracle VM VirtualBox User Manual:

http://download.virtualbox.org/virtualbox/UserManual.pdf

[3] Resumen de Apache Tomcat:

http://es.wikipedia.org/wiki/Tomcat

[4] Funcionamiento de los servlets:

http://www.it.uc3m.es/mcfp/docencia/si/material/4_servlets_mcfp.pdf

[5] Introduction to Java Server Pages:

http://es.wikipedia.org/wiki/JavaServer_Pages

[6] Manual CSS Hojas de Estilo:

http://www.xsvc.com.ve/tutoriales/ManualCssHojasDeEstilos.pdf

[7] Definición de JavaScript:

http://es.wikipedia.org/wiki/JavaScript

[8] Instalación y configuración de LDAP:

http://rm-rf.es/ldap-openldap-instalacion-y-configuracion-i/

[9] Web de la asignatura Arquitectura de Sistemas:

http://www.it.uc3m.es/labas/

[10] Maemo Community.

http://maemo.org/

[11] ScratchBox website.

http://scratchbox.org/

[12] The Maemo development environment. The emulation.

http://www.it.uc3m.es/abel/as/MSA/L1/inclass_en.html

[13] Commons FileUpload.

http://commons.apache.org/proper/commons-fileupload/

[14] Apache Software License.

http://www.apache.org/licenses/